Adapting Deep Neural Information Retrieval Models to Long Documents and New Domains

Minghan Li

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES

École doctorale : MSTII - Mathématiques, Sciences et technologies de l'information, Informatique
Spécialité : Informatique
Unité de recherche : Laboratoire d'Informatique de Grenoble

Adapter des modèles de recherche d'information basés sur les réseaux neuronaux profonds pour les documents longs et les nouveaux domaines

Adapting Deep Neural Information Retrieval Models to Long Documents and New Domains

Présentée par :

Minghan LI

Direction de thèse :

Eric GAUSSIER — Directeur de thèse
Professeur des Universités, Université Grenoble Alpes

Rapporteurs :

BENJAMIN PIWOWARSKI
Chargé de recherche HDR, CNRS DELEGATION PARIS CENTRE
LYNDA TAMINE LECHANI
Professeur des Universités, UNIVERSITE TOULOUSE 3 - PAUL SABATIER

Thèse soutenue publiquement le **7 juillet 2023**, devant le jury composé de :

ERIC GAUSSIER Professeur des Universités, UNIVERSITE GRENOBLE ALPES	Directeur de thèse
BENJAMIN PIWOWARSKI Chargé de recherche HDR, CNRS DELEGATION PARIS CENTRE	Rapporteur
LYNDA TAMINE LECHANI Professeur des Universités, UNIVERSITE TOULOUSE 3 - PAUL SABATIER	Rapporteure
DIDIER SCHWAB Professeur des Universités, UNIVERSITE GRENOBLE ALPES	Président
JAAP KAMPS Professeur associé, Universiteit van Amsterdam	Examinateur
SOPHIE ROSSET Directeur de recherche, CNRS DELEGATION ILE-DE-FRANCE SUD	Examinatrice

Abstract

In the era of big data, information retrieval (IR) plays a pivotal role in our daily lives. Deep neural networks, specifically Transformer-based models, have shown remarkable enhancements in neural IR. However, their effectiveness is constrained by limitations. This thesis aims to advance neural IR by addressing three key topics: long document retrieval for Transformer-based models, domain adaptation for dense retrieval and conversational search, and a novel differentiable approximation of listwise loss functions.

The first topic addresses the challenge of retrieving relevant information from long documents. The self-attention mechanism has the quadratic complexity, making Transformer-based models difficult to process long documents. This thesis proposes a framework that pre-ranks passages within a long document based on the query, and then combines or processes the filtered top-ranking passages to obtain the document relevance score. Experiments on IR collections with both interaction and late interaction based models demonstrate state-of-the-art level effectiveness.

The second topic explores domain adaptation for dense retrieval and conversational search. Dense retrieval models' generalization ability on target domains is limited. This thesis proposes a self-supervision approach that generates pseudo-relevance labels for queries and documents on the target domain, using an interaction-based model T5-3B from a BM25 list. Different negative mining strategies are investigated to improve the proposed

approach. Conversational search is challenging as the system needs to understand ambiguous user intent in each query turn, and obtaining labels for target datasets is difficult. Existing approaches for training conversational dense retrieval models can be further improved to tackle the domain shift issue. This thesis uses a T5-Large model to generate rewritten queries for target datasets and applies a similar approach as in dense retrieval to generate pseudo-relevance data. Experiment results show that the pseudo-relevance labeling approach improves the dense retrieval and conversational dense retrieval models on the target domain when fine-tuned on the generated data.

The third topic focuses on the use of listwise loss functions for learning to rank in IR. Popular IR metrics are not differentiable, limiting the potential of training better IR models. This thesis proposes a softmax-based approximation of the rank indicator function, a key component in the design of IR metrics. Experiments on learning to rank and text-based IR tasks demonstrate the good quality of the proposed approximations of IR metrics.

Overall, this thesis contributes novel approaches to address important challenges in IR. The proposed approaches demonstrate improvements and provide valuable insights into the development of effective IR systems.

Résumé

À l'ère du big data, la recherche d'information (RI) joue un rôle central dans notre vie quotidienne. Les réseaux neuronaux profonds, plus précisément les modèles basés sur les Transformers, ont montré des améliorations remarquables dans la RI neuronale. Cependant, leur efficacité est limitée par certaines contraintes. Cette thèse vise à faire avancer la RI neuronale en abordant trois sujets clés : la recherche de documents longs pour les modèles basés sur les Transformers, l'adaptation de domaine pour la recherche dense et conversationnelle, ainsi qu'une nouvelle approximation différentiable des fonctions de perte listwise.

Le premier sujet aborde le défi de la récupération d'informations pertinentes à partir de documents longs. Le mécanisme d'auto-attention a une complexité quadratique, ce qui rend difficile le traitement de documents longs par les modèles basés sur les Transformers. Cette thèse propose un cadre qui pré-classe les passages d'un document long en fonction de la requête, puis combine ou traite les passages les mieux classés pour obtenir le score de pertinence du document. Des expériences sur des collections de RI avec des modèles basés sur l'interaction et des modèles basés sur l'interaction tardive démontrent l'efficacité de l'état de l'art.

Le deuxième sujet explore l'adaptation de domaine pour la recherche dense et la recherche conversationnelle. La capacité de généralisation des modèles de recherche dense sur les domaines cibles est limitée. Cette thèse propose une approche d'auto-supervision qui génère des étiquettes de pseudo-pertinence

pour les requêtes et les documents du domaine cible, en utilisant un modèle T5-3B à partir d'une liste BM25. Différentes stratégies d'extraction de données négatives sont étudiées pour améliorer cette approche. La recherche conversationnelle est un défi car le système doit comprendre l'intention ambiguë de l'utilisateur à chaque tour de requête, et l'obtention d'étiquettes pour les ensembles de données cibles est difficile. Les approches existantes pour l'entraînement des modèles de recherche dense conversationnelle peuvent être améliorées pour résoudre le problème du décalage de domaine. Cette thèse utilise un modèle T5-Large pour générer des requêtes réécrites pour les ensembles de données cibles et applique une approche similaire à celle de la recherche dense pour générer des données de pseudo-pertinence. Les résultats des expériences montrent que l'approche d'étiquetage de pseudo-pertinence améliore les modèles de recherche dense et conversationnelle sur le domaine cible lorsqu'ils sont entraînés sur les données générées.

Le troisième sujet se concentre sur l'utilisation de fonctions de perte listwise pour l'apprentissage du classement en RI. Les métriques populaires en RI ne sont pas différentiables, ce qui limite le potentiel d'entraînement de modèles de RI plus performants. Cette thèse propose une approximation basée sur le softmax de la fonction indicatrice de rang, un composant clé dans la conception des métriques de RI. Les expériences sur l'apprentissage du classement et les tâches basées sur le texte en RI démontrent la bonne qualité des approximations proposées des métriques de RI.

Dans l'ensemble, cette thèse propose des approches novatrices pour relever les défis importants de la RI. Les approches proposées montrent des améliorations et fournissent des perspectives précieuses pour le développement de systèmes de RI efficaces.

Acknowledgement

It has been nearly three years since starting this PhD program, and almost four years since departing China to pursue my studies abroad. During this time, we have encountered the serious disruptions caused by COVID-19 and I have also met several other difficulties. Before this PhD program, I was a in a difficult situation. Fortunately, with the unwavering assistance of my thesis advisor, Eric Gaussier, who gave me the opportunity and encouragement, I was able to complete my thesis and possibly, my entire PhD journey. I am deeply grateful to him for his patience, expertise, kindness, and unwavering support.

I would also like to extend my gratitude to my friends in Europe and China for their companionship and encouragement, as well as my collaborators for their contributions to our joint projects and papers. I am grateful to Tat-Seng Chua for agreeing to serve as my visiting advisor during my stay at the National University of Singapore, which afforded me the opportunity to explore new research environments and learn from others. Additionally, I would like to thank my Master thesis advisor Jingxuan Wei and all other teachers who provided me with guidance and support.

I am also appreciative of the external expert of my PhD committee, Benjamin Piwowarski, for taking the time to meet with me annually and offer suggestions, as well as serving as a reviewer for my thesis. I am also grateful to the other reviewer Lynda Tamine-Lechani and the other PhD defense jury

members: Didier Schwab, Jaap Kamps and Sophie Rosset for their valuable time and efforts.

Lastly, I want to express my heartfelt appreciation to my parents, who raised me and provided me with support throughout my life: thank you for your patience in listening to me during difficult times, providing me with suggestions and encouragement.

Contents

1 Introduction

1.1 Information Retrieval

Definition Information retrieval (IR) is an important research domain that involves the process of extracting relevant information from vast collections of data. With the exponential growth of diverse forms of digital information, such as text, images, videos, and more, the need for effective and efficient methods to retrieve and organize this information has become increasingly crucial. Among these forms, text-based information represents the most prevalent scenario in daily usage, including web search and digital libraries. Query-document information retrieval is a critical research field that specifically focuses on retrieving relevant information from text-based documents. In this thesis, it is referred to as information retrieval. It involves employing techniques and methodologies to match user queries with document content, rank the documents based on relevance, and present the most relevant documents to users.

Introduction Ranking models are fundamental to the field of IR research [50]. Over the past decades, numerous ranking models have been proposed, from vector space methods [151] and probabilistic methods [147] to learning to rank (LTR) methods [95, 87]. LTR models have demonstrated considerable success in various IR applications. However, these models heavily rely on hand-crafted features, which are labor-intensive and often overly specific in

their definitions [50]. With the resurgence of interest in neural networks, particularly deep neural networks or deep learning, significant advancements have been made in computer vision and natural language processing (NLP) tasks. Consequently, the field of neural information retrieval (Neural IR), which involves utilizing (deep) neural networks to directly construct the ranking function for IR, has been the subject of many studies [60, 154, 124, 49, 126, 125, 62, 182, 36, 82] which have led to the development of several interesting IR models for learning representations of documents and queries. These models employ deep neural networks, such as convolutional neural network (CNN), recurrent neural network (RNN), and other similar architectures.

The Transformer model, proposed by Vaswani et al. [166], is a novel architecture that relies exclusively on multi-head attention mechanisms. This model has demonstrated superior performance compared to recurrent neural networks, as it exhibits higher quality results, improved parallelizability, and reduced training time [166]. The self-attention mechanism employed in the Transformer model allows for more effective capture of long-range dependencies and contextual information. As a result, the Transformer model has significantly impacted various NLP tasks, such as machine translation, text generation, and sentiment analysis, leading to substantial advancements in the field of NLP. Using a multi-layer bidirectional transformer encoder, the authors of [30] have proposed Bidirectional Encoder Representations from Transformers (BERT), a method for pre-training deep bidirectional representations from unlabeled text by conditioning all layers on both left and right context. Fine-tuning of pre-trained BERT models has led to cutting-edge performance in various applications. Inspired by the success of BERT in natural language processing, several studies proposed models based on BERT [30] like transformer architectures for IR tasks [118, 102, 26, 83], resulting in some of the current state-of-the-art models in *ad hoc* IR [102, 83].

Neural IR models, especially models based on BERT like architectures, can be categorized into three types. The first is interaction-based models[118, 83], the second is dense retrieval models and the third is late interaction based [72]. The first method, similar to a vanilla BERT model [118], concatenates query tokens and document tokens as BERT inputs and applies full self-attention, which is considered highly effective [51], but suffers from high computational complexity. On the other hand, dense retrieval (DR) methods [70, 183, 181] generate two representations [144] for a query and a document, respectively. When the document representations can be pre-stored, this approach allows for efficient fast retrieval, albeit at the expense of effectiveness. To leverage the advantages of both approaches, late-interaction based methods like ColBERT [72] have been proposed. In ColBERT, token-level passage embeddings are pre-stored and late-interacted with query embeddings to produce a relevance score. This method is slightly less efficient than representation-based methods, but significantly more effective.

Limitations Neural information retrieval (IR) models have demonstrated significant success but also exhibit limitations. The first is about the limitation of length to process long documents. One main advantages of the self-attention mechanism is that it allows to capture dependencies between tokens in a sequence regardless of their distance. However, despite its excellent results, the self-attention mechanism has difficulty to process long sequences due to its quadratic complexity in the number of tokens, which also limits the application of transformer-based models to long document information retrieval, where each document could contain thousands of tokens. Existing approaches for addressing the challenge of long documents in information retrieval include truncating the documents, which results in potential loss of relevant information [118], segmenting them into several

passages [83], which may lead to miss some information and high computational complexity when the number of passages is large, or modifying the self-attention mechanism [66] to make it sparser as in sparse-attention models, at the risk again of missing some information. These approaches still have limitations.

Another limitation is observed in recent studies, such as BEIR [157], which highlight that dense retrieval models trained on a source domain exhibit diminished generalization compared to traditional models, such as BM25 and interaction-based models, when applied to out-of-distribution (OOD) datasets. Although training on target datasets with annotated gold labels is a standard approach, it can be time-consuming and costly, so that this approach may have limitation on many real world usages. It is thus important to address the issue of OOD scenarios for dense retrieval. To address this issue, researchers have resorted to adversarial learning [181] and query generation approaches [100], however both approaches resulted in limited improvements. Conversational search poses additional challenges and complexity, as the query in each turn of a conversation may contains ambiguity. Besides, one key issue is the scarcity of labeled data, as obtaining human rewritten queries and relevance labels can be labor-intensive and time-consuming. To address this challenge, researchers have proposed promising approaches that leverage source domain data resources to mitigate the data scarcity issue, such as few-shot learning techniques, for conversational dense retrieval model [190, 93]. However, there is still potential for further improvement when adapting these approaches to target domain data.

Furthermore, in standard information retrieval systems, the primary objective is to maximize metrics based on rankings, such as precision or NDCG [175]. However, the non-differentiable nature of the ranking operation poses challenges for directly optimizing these metrics in state-of-the-art neural IR

models, which heavily rely on the computation of meaningful gradients for optimization.

Contributions In this thesis, we propose three parts to improve neural IR models, and addressed the limitations discussed above.

Firstly, in order to address the limitation of the Transformer architecture, which can only handle only limited input length documents, we propose a novel framework for long document retrieval. Our approach involves a two-step process: first, local query-block pre-ranking is used to select key blocks from the long document, and then a subset of these blocks are aggregated to form a shorter document that can be effectively processed by models such as BERT, or the blocks are used as input to an IR model. This framework enables improved efficiency and effectiveness in handling long documents by filtering out noisy blocks and selecting the most relevant information for downstream processing. The framework we have proposed has been implemented and evaluated on interaction-based models, such as BERT and PARADE [83]. Furthermore, we have extended our approach to incorporate a late interaction mechanism for long document retrieval, where the late interaction mechanism is similar to ColBERT [72]. The proposed model is trained with multi-task learning [92, 71] to have the abilities of generating token level representations and passage level representations simultaneously. Our experimental results on both models demonstrate the effectiveness and efficiency of our proposed framework.

Secondly, we address the limited generation ability of dense retrieval (DR) models when applied to new domain data by proposing a self-supervision approach. We automatically generate pseudo-relevance labels for the target domain by utilizing the BM25 model to obtain an initial document ranking, and then employing the T5-3B [141] interaction-based model to re-rank

the documents for pseudo-positive labeling. Various strategies for selecting hard negative examples are explored to improve the data generation quality. Our experimental results demonstrate that the use of T5-3B for pseudo-relevance labeling leads to improved performance of the DR model and the state-of-the-art query generation approach GPL [172] when fine-tuned on the pseudo-relevance labeled data. Additionally, for conversational search, we propose to use T5-Large [141] to generate rewritten queries for target datasets, followed by a similar approach as in dense retrieval to generate pseudo-relevance data. Our experiments reveal that by further training on the generated training data on target dataset, conversational dense retrieval models [190, 93] can achieve better effectiveness.

Thirdly, we propose a smooth approximation of the rank indicator function, which acts as a basic building block for the development of differentiable approximations of IR metrics. This is achieved through a softmax-based approximation. By employing these approximate metrics, we are able to optimize end-to-end neural approaches using listwise loss functions. Through experiments conducted on both learning to rank and text-based information retrieval tasks, we provide evidence of the effectiveness of the proposed approach.

1.2 Thesis Outline

Chapter 1 is a introduction about the background, limitations and contributions proposed. Then the proposed approaches are present in three parts.

In the first part, we propose the first contribution: Adapting Transformer based IR models to long document retrieval.

- Chapter 2 presents the proposed framework for interaction based models, including BERT and PARADE.
- Chapter 3 presents a BERT-based late interaction approach for long document retrieval.

In the second part, the second contribution is proposed: Adapting dense retrieval (DR) and conversational dense retrieval (CDR) models to new domains. This is achieved by proposing a self-supervision approach that generates pseudo-relevance labels for queries and documents on the target domain.

- Chapter 4 presents the pseudo-relevance labeling and the data generation approach for dense retrieval model.
- Chapter 5 further presents to use T5-Large model to generate rewritten queries and use the same pseudo-relevance labeling and data generation approach for conversational search.

The third part focuses on the use of listwise loss functions for learning to rank in IR. Chapter 6 roposes a softmax-based approxi- mation of the rank indicator function for designing of popular IR metrics.

At last, Chapter 7 concludes the thesis.

Part I

Long Document Information Retrieval for Transformer-Based Models

2

Improve Interaction-Based Models based on Transformers for Long Document Retrieval

2.1 Introduction

The field of query-document information retrieval (IR) has seen increasingly rapid advance in the past decades. Learning-to-rank (LTR) models [95, 87] have already achieved great success in many IR applications. However, LTR models mainly rely on hand-crafted features which are time-consuming and often over-specific in definition [50]. With the resurgence of interest in neural networks, especially deep neural networks or deep learning, we have witnessed dramatic improvements in computer vision and natural language processing (NLP) tasks. Neural Information Retrieval (Neural IR), which refers to the use of (deep) neural networks to directly construct the ranking function for IR, has been the subject of many studies [60, 154, 124, 49, 126, 125, 62, 182, 36, 82] which have led to the development of several interesting IR models for learning representations of documents and queries.

The transformer model [166], which is exclusively based on multi-head attention mechanism, has shown to be higher in quality while being more parallelizable and requiring substantially less training time than models based on recurrent neural networks [166]. Using a multi-layer bidirectional transformer encoder, the authors of [30] have proposed Bidirectional Encoder Representations from Transformers (BERT), a method for pre-training deep bidirectional representations from unlabeled text by conditioning all layers on both left and right context. Pre-trained BERT models can be fine-tuned to produce cutting-edge models for a variety of applications. In particular, following their success in NLP, several works have focused on transformers [166] and derived models based on BERT [30] for solving IR tasks [118, 102, 26, 83], leading to some of the current state-of-the-art models in *ad hoc* IR [102, 83].

One main advantages of the self-attention mechanism is that it allows to capture dependencies between tokens in a sequence regardless of their distance. However, despite its excellent results, the self-attention mechanism has difficulty to process long sequences due to its quadratic complexity in the number of tokens, which also limits the application of transformer-based models to long document information retrieval, where each document could contain thousands of tokens.

Three standard strategies based on BERT have been adopted to circumvent this problem. The first one consists in truncating long documents (*e.g.*, [118]), the second in segmenting long documents into shorter passages (e.g., [26]) and the last one in replacing the complex self-attention module with sparse-attention ones (e.g., [66]). In the first case, important information may be lost and the relevance judgement is damaged. In the second case, a hierarchical architecture can be further adopted to build a document-level representation on top of the representations of each passage [83]. This said,

despite the state-of-the-art results this strategy may lead to, there remain issues concerning the time, memory and energy consumption associated to it. Furthermore, the consideration of passages that may not be relevant to the query may introduce noise in the final representation and limit the identification of long-distance dependencies between relevant tokens [33]. In the third case, sparsity constraints may lead to miss important dependencies which can lead to under-optimal results.

The approach we propose is slightly different with these strategies, aiming at capturing, in long documents, the blocks which are the most important to decide on the relevance status of the whole document. Besides, it can be integrated to the second strategy. It is based on three main steps: (a) selecting key (i.e., likely relevant) blocks with local pre-ranking using either classical IR models or a learning module reminiscent of the judge module used in [33], (b) learning a joint representation of queries and key blocks using a standard BERT model, and (c) computing a final relevance score which can be regarded as an aggregation of local relevance information [177].

Our contributions are two-fold. We first conduct an analysis which reveals that relevance signals can appear at different locations in documents and that such signals can be better captured by semantic relations than by exact matches. We then investigate two methods to select blocks, one based on standard IR functions and the other on a learned function operating on semantic representations, and show how to integrate these methods in state-of-the-art IR models. In this approach, as well as in previous approaches based on passages as PARADE [83], blocks occurring at different positions of a document are concatenated or selected in the order they occur in the document and can be seen as a digest of the elements necessary to assess the relevance of the whole document to the query. Although the blocks selected

are not coherent physically, they still are coherent in that they are all relevant to the same query.

The remainder of the chapter is organized as follows: Section 2.2 describes related work. Section 2.3 investigates the relation between the potential relevance and the position of a block in a document as well as the importance of fuzzy vs exact matching when selecting blocks. Section 2.4 presents the block selection approach based on standard IR functions whereas Section 2.5 describes the block selection approach based on a learned function. Sections 2.6 and 2.7 show the benefits of selecting key blocks on different collections. Finally, Section 2.8 summarizes our findings and concludes the chapter.

2.2 Related Work

Let q denote a query and d a document. Without loss of generality, the ranking function f of an IR system takes the form [50]:

$$f(q, d) = g(\psi(q), \phi(d), \eta(q, d)), \tag{2.1}$$

where ψ and ϕ are representation functions that extract features from q and d respectively, η is the interaction function that models query-document representation from (q, d) pairs, and g is the evaluation function that calculates the relevance score based on the extracted features or interaction. According to the choices made on the representation and interaction functions, neural information retrieval models can be grouped into two categories [50]: representation-based and interaction-based architectures. Besides these two categories, some neural information retrieval models adopt a hybrid approach.

The Deep Structured Semantic Model (DSSM) [60] is one of the earliest representation-based models for document ranking which uses a fully-connected network for the functions ψ and ϕ. To map the query and the documents to a shared semantic space, a non-linear projection is used. The relevance of each document given the query is then calculated with the cosine similarity between their vectors in that semantic space. Clickthrough data is then used to discriminatively train the model by maximizing the conditional likelihood of the clicked documents. Other studies in this category proposed to exploit distributed representations via DSSM variations, or relied on different representation functions [121]. For example, ARC-I [59] and CLSM [154] use convolutional neural networks (CNN) for ψ and ϕ while [124] uses a recurrent neural networks.

One of the first neural IR models which outperformed traditional IR models is the interaction-based model referred to as Deep Relevance Matching Model (DRMM) [49]. The interaction function η is defined as the matching histogram mapping between each query term and the document. A feed-forward network for term-level relevance and a gating network for score aggregation in the evaluation function g are further used. In this work, the term vectors are fixed to Word2Vec word embeddings [110]. Similarly, Xiong et al. [182] proposed KNRM which employs a translation matrix that utilizes word embeddings to represent word-level similarities, a unique kernel-pooling technique for extracting multi-level soft match features, and a learning-to-rank layer that combines those features into the final ranking score. The entire model is trained end-to-end, and the word embeddings are tuned to produce the desired soft matches [182]. Inspired by the way humans assess the relevance of a document, Pang et al. [125] proposed DeepRank, a model which splits documents into term-centric contexts according to each query term. A tensor containing both the word representations of query/query-centric context as well as their interactions is first built. It is then passed through a measure

network, based on CNN [76] or 2D-GRU [169], to produce a representation of local relevance. Finally, the global relevance score is calculated using an aggregation network. Hui et al. [62] proposed PACRR, a model inspired by the neural models used in image recognition [50]. PACRR takes a similarity matrix between a query and a document as input. Then multiple CNN kernels capture the query-document interactions. Following this work, Hui et al. [61] provided a lightweight contextualization model called CO-PACRR which averages word vectors within a sliding window and appends the similarities to those of the PACRR [62] model [58].

Since it is sometimes difficult to produce good high-level representations of long texts, the representation-based architecture is better suited to short input texts. Models in this category are good for online computing since they allow one to pre-calculate text representations. Interaction-based models, on the other hand, tend to yield better results as they can tune document representations towards a given query. Unfortunately, since the interaction function η cannot be pre-calculated until the input pair (q, d) is seen, models in this category are not as efficient for online computation as representation-focused models [50].

Models Based on transformers Benefiting from pre-trained language models based on transformers [166], especially BERT [30], different research teams have developed state-of-the-art neural IR models, significantly outperforming traditional and previous neural IR models. Nogueira and Cho [118] proposed to use BERT as a re-ranker for the passage re-ranking task by fine-tuning it and achieved state-of-the-art results. The passages are truncated if too long (typically over 512 tokens). This work proved the effectiveness of fine-tuning BERT for IR problems. MacAvaney et al. [102], through a model called CEDR which combined BERT with other neural IR

models, as PACRR [62], KNRM [182] and DRMM [49], and showed the benefits of this combination. Dai and Callan [26] proposed to first segment documents into short, overlapping passages, and then used BERT to define the relevance score of the document, using either the first passage, the best passage or the sum of all passages. Hofstätter et al. [58] proposed a reranking model called Transformer Kernel, in short TK, which uses a hybrid approach based on a small number of transformer layers to contextualize query and document word embeddings separately. Then RBF-kernels [182] are used for interaction scoring, where each kernel focuses on a specific similarity range. Experimental results show that although the effectiveness is not as good as BERT reranker, TK has strong efficiency. In a similar vein, Li et al. [83] explored strategies for aggregating relevance signals from a document's passages into a final ranking score, leading to a model called PARADE. A hierarchical layer, in the form of a max-pooling, attention, CNN or transformer aggregator is used to aggregate the passage representations so as to obtain a joint query-document representation for long documents. They showed that passage representation aggregation strategies can outperform techniques proposed previously. In particular, PARADE can improve results significantly on collections with broad information needs where relevance signals can be disseminated throughout the document. Grail et al. [46] also proposed a hierarchical model in which each transformer layer, used to learn a representation for each sentence of a document, is followed by an RNN which captures dependencies between the CLS tokens representing the different sentences of a document. As shown in the experiments, this model is particularly well adapted for long-document summarization.

In the above models, as transformers are limited in their input length due to their quadratic complexity, researchers have either truncated long documents or segmented them into passages. There have however been different attempts to use transformers on long documents. For example, Dai et al.

[28] introduced a model with left-to-right recurrence between transformer windows, consisting of a segment-level recurrence mechanism and a novel positional encoding scheme. The left-to-right approach processes the document in chunks moving from left-to-right and thus not adapted to tasks which benefit from bidirectional contexts [3]. Child et al. [21] introduced several sparse factorizations of the attention matrix which reduce the quadratic complexity to $O(n\sqrt{n})$. Hofstätter et al. [56] proposed a local self-attention which considers a sliding window over the document and restricts the attention to that window in order to deal with long documents. Their model, called TKL, adapts TK [58] with this mechanism. Beltagy et al. [3] introduced the Longformer with an attention mechanism which scales linearly with sequence length, combining windowed local-context self-attention with task-motivated global attention to encode inductive bias about the task. Longformer achieves state-of-the-art results on the character-level language modeling tasks, and when pretrained from the RoBERTa [96] checkpoint, it consistently outperforms RoBERTa on long document tasks. Zhao et al. [194] proposed Transformer-XH which enables to represent structured texts. It shares similar motivation with Dai et al. [28] and Child et al. [21], and is particularly well adapted to multi-hop QA tasks [186] and fact verification tasks [158]. Ainslie et al. [1] introduced the Extended Transformer Construction (ETC) model to address two key challenges of standard transformers: scaling input length and encoding structured inputs. A novel global-local attention mechanism is introduced where the local sparsity reduces the quadratic scaling of the attention mechanism. They further show that by including a pre-training Contrastive Predictive Coding (CPC) task [123], the performance for tasks where structure matters improves even further. Zaheer et al. [191] proposeed BigBird, which combines local and global attention with random sparse attention. Kitaev et al. [74] only computed self-attention between similar tokens, as defined through locality-sensitive hashing.

Despite such models' described effectiveness, there remain problems. Firstly, as described in [191], coalesced memory operations, which load blocks of contiguous bytes at once, are where hardware accelerators like GPUs and TPUs really shine. As a result, small sporadic look-ups caused by a sliding window or random element queries are not very efficient. This is addressed by "blockifying" the lookups. It is generally known [47, 187] that GPUs cannot efficiently execute sparse multiplications, which are commonly employed by models with tailored attention mechanisms. Naively using for-loops or masking the matrix may result in even worse efficiency than the full self attention [66]. Thus, such models with customized attention mechanisms need specifically designed tricks or customized CUDA kernels [3], which are inconvenient or require expertise in low-level GPU operations [66]. Jiang et al. [66] proposed Query-Directed Sparse Transformer (QDS-Transformer), which also induce sparsity in self-attention mechanism, containing local windowed attention and query-directed global attention. Experiments demonstrate consistent and robust advantages of QDS-Transformer over previous approaches. However, this model still needs customized TVM implementation [19]. Secondly, as these customized attention models only rely on partial attention, their accuracy does not match, in general, the one of full attention models.

Selecting key blocks The approach we advocate here to solve the above-mentioned problems is to select key, important blocks from a document and base the relevant score of the full document on just these blocks. Note that this differs from the approach followed in PARADE [83] in which the passages retained are arbitrary. It is however reminiscent of both [125] and [33], which are partly inspired by cognitive theory and reckon that, in order to assess the relevance of a document, humans first scan the whole document to detect relevant locations where query terms occur and then aggregate

local relevance information to decide on the relevance of a document [177]. Compared to [125], our approach is simpler conceptually and can consider blocks which do not contain query words but are nevertheless relevant to the query. We show in Sections 2.3 and 2.6 that such a *fuzzy matching* can help improve the block selection process and the overall IR system built upon it. The study described in [33] focuses on reading comprehension, multi-hop question-answering and text classification. We show here how this approach can be simplified for IR purposes by using the same BERT model for the *reasoner* and *judge*. In addition, we investigate the use of standard models to select key blocks, which leads to an entirely different and simpler architecture.

Furthermore, we want to mention the study presented in Muntean et al. [114] which, in order to assign a relevance score to a document, weighs each passage differently by identifying salient terms using TF-IDF and KL divergence scores [12] are used to identify salient terms and to derive the weights. Although this model treats passages in a long document differently, the weights do not reflect the relevance to the query (salient terms are identified independently of the query). This is a major difference with our approach which aims to select blocks according to their relevance to the query. Furthermore, we focus here on neural IR models which have difficulties in dealing with long documents.

Lastly, in parallel to our work, Hofstätter et al. [53] also introduce a model called IDCM which also learns to select top scoring blocks which are then used to score documents with respect to a given query. IDCM first trains a block ranking model based on BERT (and called ETM for Effective Teacher Model) on MS-MARCO[1], prior to fine-tuning this model on each collection for document ranking. Then, a block selection model (called ESM for Efficient

[1]MS-MARCO is a passage retrieval collection available at: `https://microsoft.github.io/msmarco/`

Student Model) is learned via knowledge distillation from ETM. Both ESM and ETM are then used to score new documents, ESM allowing the selection of the most important blocks and ETM being used to score documents on the basis of the selected blocks. This contrasts with the way we learn to select blocks: first of all, for all proposed models except the last one KeyB(PARADE5)$_{BinB2}$, we do not require any pre-training on additional collections; second, we use the same model for selecting blocks and scoring documents, the rationale being that in both cases one computes a relevance score with respect to the same query. The last model that we propose can also re-use an additional BERT ranker from the KeyB(vBERT) models previously proposed to select blocks. Our approach is thus simpler (see Section 2.5) and finally leads to better results as shown in Section 2.7. Furthermore, it is interesting to note that IDCM scores each block separately and then aggregates the block scores, while our approach for improving Vanilla BERT scores a document using concatenated selected blocks. As shown in our experiments, the two strategies are effective, with a slight advantage for the latter on the TREC 2019 DL collection. We also want to mention that we recently proposed Li and Gaussier [88] to select key blocks for late-interaction models by training a BERT-based model with multi-task learning. This study however has a different focus that the current one, which only concerns interaction-based models.

2.3 A Finer-Grained Look at Documents

We take in this section a closer look at documents and the blocks they contain by addressing two questions:

1. Are relevant signals spread over the entire document, and thus can appear in any block, or are they concentrated in particular regions, as the beginning of documents?

2. Should one rely only on exact matching of query words to select important blocks or is additional information contained in related words, as synonyms, important?

Our analyses thus aim to reveal which blocks to select with respect to their positions and how to select them.

2.3.1 Preliminaries

To conduct our investigation of the above points, we use four standard IR datasets, namely MQ2007, MQ2008, GOV2[2] (also referred to as Trec-terabyte 2004/2005/2006) and Robust04. MQ2007 and MQ2008 are standard LETOR [138] benchmark datasets for learning to rank. GOV2 contains documents resulting from a crawl of .gov websites made in early 2004. Robust04[3] contains news article from the Financial Times, the Federal Register 94, the LA Times, and FBIS. In each dataset, the title of the topics have been used as queries and the content of the documents have been extracted using Anserini [185]. Relevance judgements can take three values: 0 (irrelevant), 1 (relevant) or 2 (very relevant). A document-query pair with an associated relevant judgement will be referred to as a *labeled* document-query pair. All our analyses are based on labeled document-query pairs so as to avoid assumptions on the relevance status of non-labeled pairs. Furthermore, as it has become common to first filter documents with a standard IR system prior to deploy deep IR models, we first select, for each query, the top 200

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

documents using BM25 and only retain the labeled document-query pairs associated with these documents. Note that the above filtering is not run on MQ2007 and MQ2008 which already rely on a small subset of documents for each query. Table 2.1 displays, for each collection, the number of queries and documents as well as the number of unique labeled document-query pairs (in the original dataset as well as the one filtered with BM25). The proportions of irrelevant, relevant and very relevant pairs are computed on the filtered version for GOV2 and Robust04, and on the original version for MQ2007 and MQ2008.

Dataset	MQ2007	MQ2008	GOV2	Robust04
Nb of queries	1,692	784	150	250
Original				
Nb of documents	65,302	14,381	ca. 25M	ca. 0.5M
Nb of labeled document-query pairs	69,599	15,208	135,352	311,410
BM25 filtered				
Nb of unique documents	-	-	29,769	42,156
Nb of labeled document-query pairs	-	-	26,155	95,336
Proportion of irrelevant pairs	0.74	0.81	0.80	0.94
Proportion of relevant pairs	0.20	0.13	0.17	0.05
Proportion of very relevant pair	0.06	0.6	0.03	0.01

Tab. 2.1.: Statistics of the datasets used.

To divide documents into blocks, we use the recent CogLTX [33] block decomposition method, which is based on a dynamic programming method, each block having a maximum of 63 tokens. This method, which was used with success on several NLP tasks, sets different costs for different punctuation marks and aims at segmenting in priority on strong punctuation marks such as "." and "!", making it close to a sentence segmentation procedure.

Lastly, to measure to which extent a block is relevant or not to a given query, we use the Retrieval Status Value (RSV) of the block which simply amounts here to the score provided by an IR model, in our case either BM25 or the cosine similarity computed on semantic representations of queries and

blocks. The higher the score obtained, the likelier the corresponding block is relevant.

We now turn to answer the two questions we asked before.

2.3.2 Relevance Signals Appear at Different Positions in Documents

We first analyze the length of documents with respect to the number of blocks they contain. To do so, we plot in Figure 2.1 the proportion of documents containing exactly n blocks, n varying from 1 to more than 125[4]. In addition, the median as well as the first and third quartiles of the distribution of the number of blocks for each dataset are provided. Even though the shape of the curve for each dataset varies, one can notice that more than 25% of the documents in MQ2007, MQ2008 and GOV2 contain more than 80 blocks, way above the size limitation of current transformer-based IR models. For Robust04, 25% of the documents have more than 30 blocks, a size that can also exceed the limitation of current transformer-based IR models[5].

As the number of blocks contained in a document varies a lot from one document to another, for assessing where relevance signals appear in different documents, we consider p positions $(1, 2, \cdots, p)$ and allocate each block of a document to one of the p positions with the constraint that the first block of the document should be allocated to the first position, and the last block to the last position. This can be easily done with the following function

[4]The maximum number of blocks for MQ2007, MQ2008, Robust04 and GOV2 respectively is 3225, 3225, 2959, 3311. We do not display the entire distribution for reading purposes.
[5]The average number of tokens per block for MQ2007, MQ2008, Robust04 and GOV2 respectively is 54.39, 54.44, 53.16, 54.33.

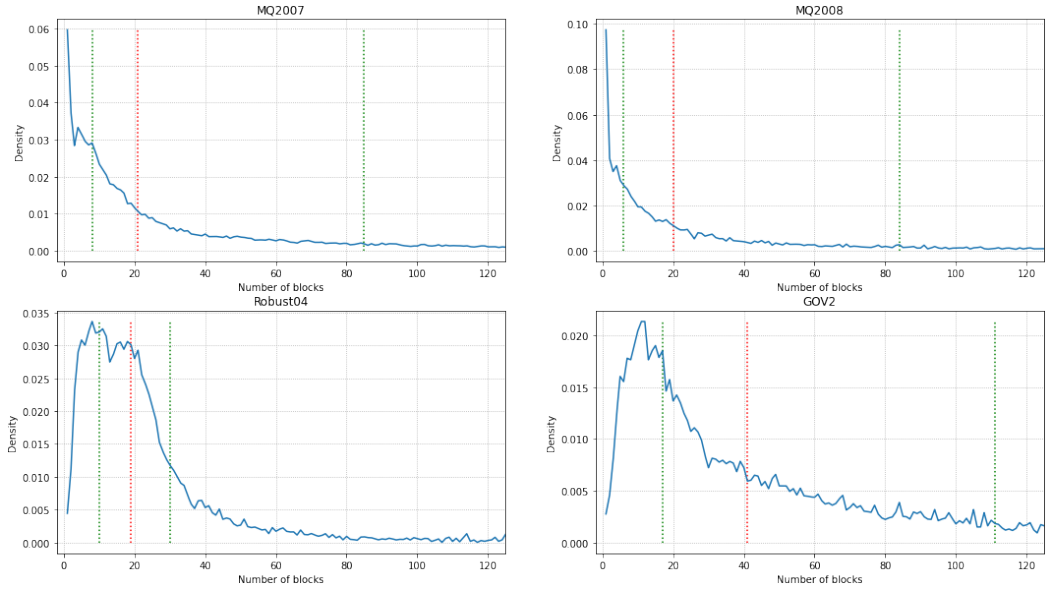


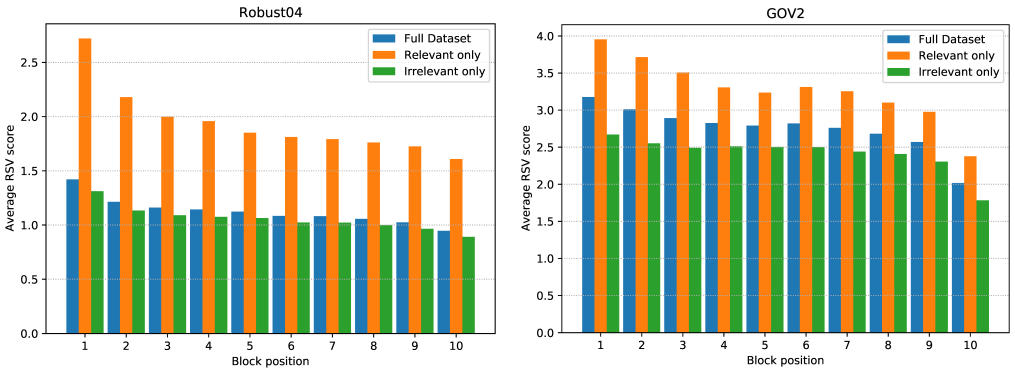
Fig. 2.1.: Interpolated curve of the density of the number of blocks per document for the different datasets. The vertical line in red denotes the position of the median, whereas the position of the 1st and 3rd quartile are displayed in green. For the sake of readability, only the first 125 blocks are shown here.

which provides the position of the i^{th} block of a document containing b blocks ($1 \leq i \leq b$):

$$pos(i) = \left\lceil \frac{10 \times i}{b} \right\rceil,$$

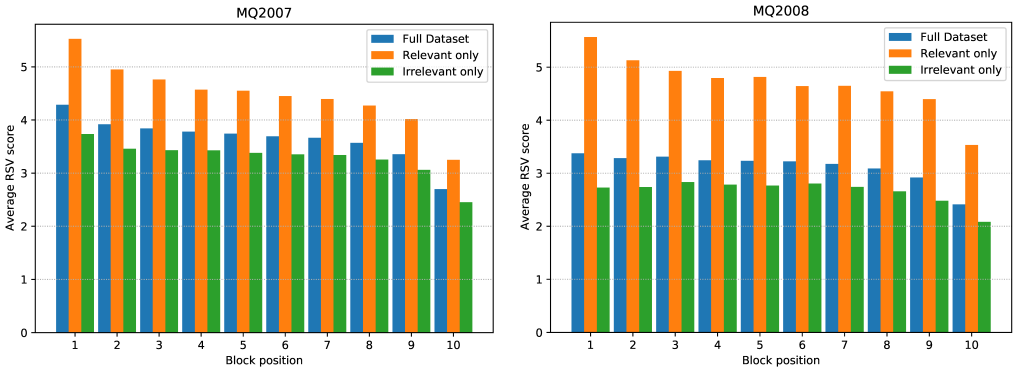
where $\lceil x \rceil$ is the ceiling function which returns the integer greater than or equal to x. For example, the fifth block in a document containing 100 blocks is in the first position. Only documents that have at least 15 blocks are considered for analysis, preventing missing positions.

Figure 2.2 displays the distribution of the average BM25 RSV score on each position with $p = 10$ for all labeled query-document pairs (referred to as full dataset), for very relevant and relevant pairs (referred to as relevant only) and for irrelevant pairs (referred to as irrelevant only). On all collections, the average RSV score decreases when the position increases. However, the average RSV scores are still important in the middle positions and non-



(a) BM25 RSV of Robust04

(b) BM25 RSV of GOV2



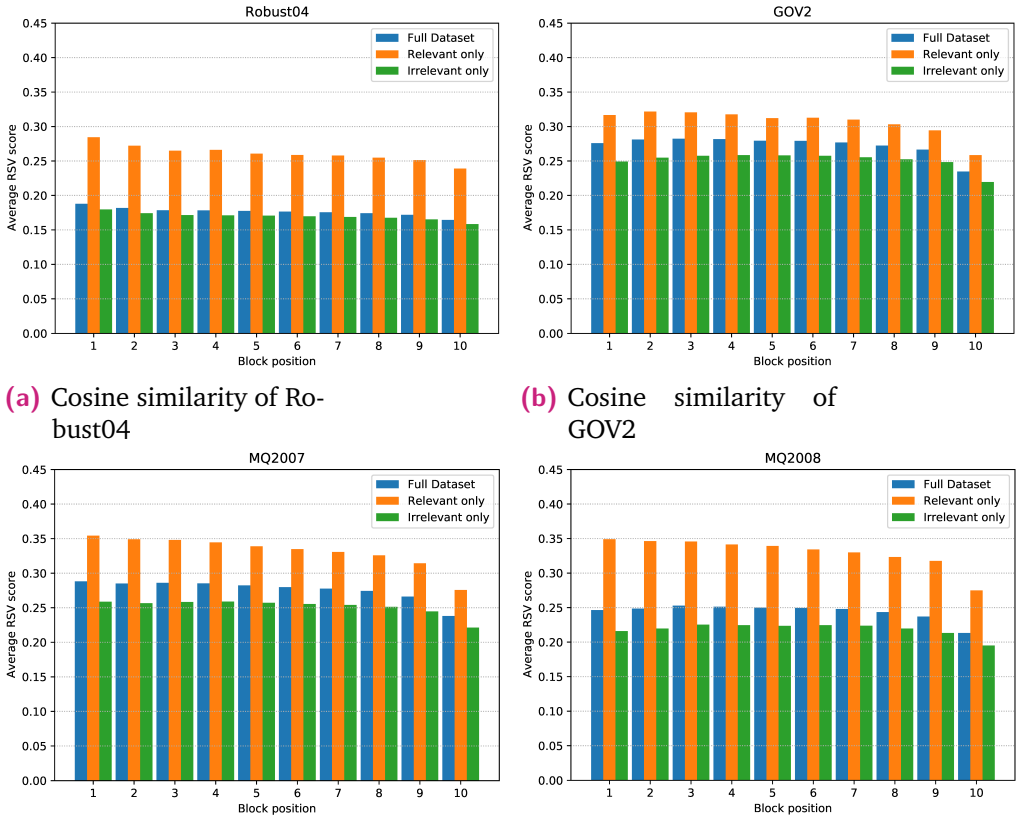
(c) BM25 RSV of MQ2007

(d) BM25 RSV of MQ2008

Fig. 2.2.: Average BM25 RSV scores per position of a block in a document using the original query q.

negligible in the last positions. Furthermore, if one assumes that all blocks of an irrelevant document are irrelevant[6], then the difference, for a given position, between the average RSV scores of relevant and irrelevant pairs may serve as an additional indicator of whether or not relevance signals are found in that particular position. When there is no difference between the average RSV scores of relevant and irrelevant pairs at a given position, then the only conclusion one can draw is that all blocks from relevant and irrelevant documents behave in the same way with respect to the RSV score, and there is no indication that relevance signals are present at that position. On the contrary, when the average RSV score for relevant pairs is significantly higher than the one for irrelevant pairs at a given position, then there is a

[6]We believe this is a reasonable assumption, at least when the blocks are not too short.



(a) Cosine similarity of Robust04

(b) Cosine similarity of GOV2

(c) Cosine similarity of MQ2007

(d) Cosine similarity of MQ2008

Fig. 2.3.: Average RSV scores (cosine similarity) per position of a block in a document using the original query q.

clearer indication that there are relevant signals at that position. This is the case for all positions of the four collections.

To complement the above analysis, we used a different RSV score, namely the cosine similarity between the semantic representations of blocks and queries obtained with the pre-trained Sentence-BERT [144] model[7] (we simply input each query-block pair to Sentence-BERT which outputs query and block representations on which a cosine similarity is computed). Contrary to the BM25 score which is mainly 'lexical', this score aims to capture additional semantic relationships between queries and blocks. Figure 2.3 displays the distribution

[7]The all-MiniLM-L12-v2 version from `https://www.sbert.net/docs/pretrained_models.html`

Tab. 2.2.: Example query and extensions

Original query q	"minimum wage increase"
Synset ("minimum") Synset ("wage") Synset ("increase")	minimal earnings, pay, remuneration, salary growth, gain, addition
Expanded query q^{exp}	minimum wage increase minimal earnings pay remuneration salary growth gain addition
Random expanded query q^{rand_exp}	minimum wage increase cadent gravely stuffiness puller complaisant sunlight profusely asterism
Original query boolean representation q_{bool}	("minimum" OR "wage" OR "increase")
Extended query boolean representation q_{bool}^{syn}	(("minimal" OR "earnings" OR "pay" OR "remuneration" OR "salary" OR "growth" OR "gain" OR "addition") AND NOT ("minimum" OR "wage" OR "increase"))
Random extended query boolean representation q_{bool}^{rand}	(("cadent" OR "gravely" OR "stuffiness" OR "puller" OR "complaisant" OR "sunlight" OR "profusely" or "asterism") AND NOT ("minimum" OR "wage" OR "increase"))

of the average cosine similarity RSV score on each position. This analysis confirms that all positions can contain relevance signals. Furthermore, the decrease in RSV scores when the position of the block increases is less marked so that the difference between blocks at the beginning, the middle and the end of a document (except the last block which gets in general significantly smaller scores) is not really important.

Overall, our analysis reveals that, if terms at the beginning of a document may be more important that terms at the end, all positions in a document can contain important relevance signals and should *a priori* be explored for IR purposes. This conclusion is in agreement with the verbosity hypothesis [149].

2.3.3 Fuzzy Matching May Help Select Better Blocks on Some Collections

The second question we address is whether one should solely rely on an exact matching of the words present in the query to select a given block or whether fuzzy matching including words related to the query words may help

retrieve better blocks. We consider here that a word related to a query word is any synonym, as provided by WordNet [39], of that query word. Other semantic relations can of course be used; we chose synonymy because it has the advantage of being a simple relation which is at least partly captured in modern word embeddings and for which useful resources such as WordNet are available.

Our goal here is to assess whether using synonyms can help select useful blocks. If this is the case, then one can conclude that it may be useful to use matching strategies that go beyond an exact matching of query words. It is important to note here that if many studies have been devoted to the utility of synonyms in IR systems [161, 84], our study differs from them in that it focuses on the use of synonyms for selecting blocks and does not aim to assess different query expansion strategies. In particular, we are not interested in assigning different weights to expanded terms even though such a strategy may lead to better query expansion results [38, 37].

We first try to answer the question: *Can synonyms identify blocks that would not have been identified without them?* To do so, from the original query, we construct three boolean queries. The first one consists of the disjunction of all query words and will be referred to as q_{bool}. The second one consists of the disjunction of all the synonyms from WordNet of all query words and excludes the original query words. It will be referred to as q_{bool}^{syn}. Lastly, the third one, referred to as q_{bool}^{rand}, which has the same length as the second one, consists of the disjunction of words randomly selected from WordNet. This last query helps assess to which extent the phenomena observed depend solely on the query length[8]. Table 2.2 provides an example of these three

[8]Adding terms to a boolean query through a disjunction is likely to increase the number of blocks retrieved by the query. This said, please bear in mind that here the words added to the original query come from an external resource and may not be present in the collections queried.

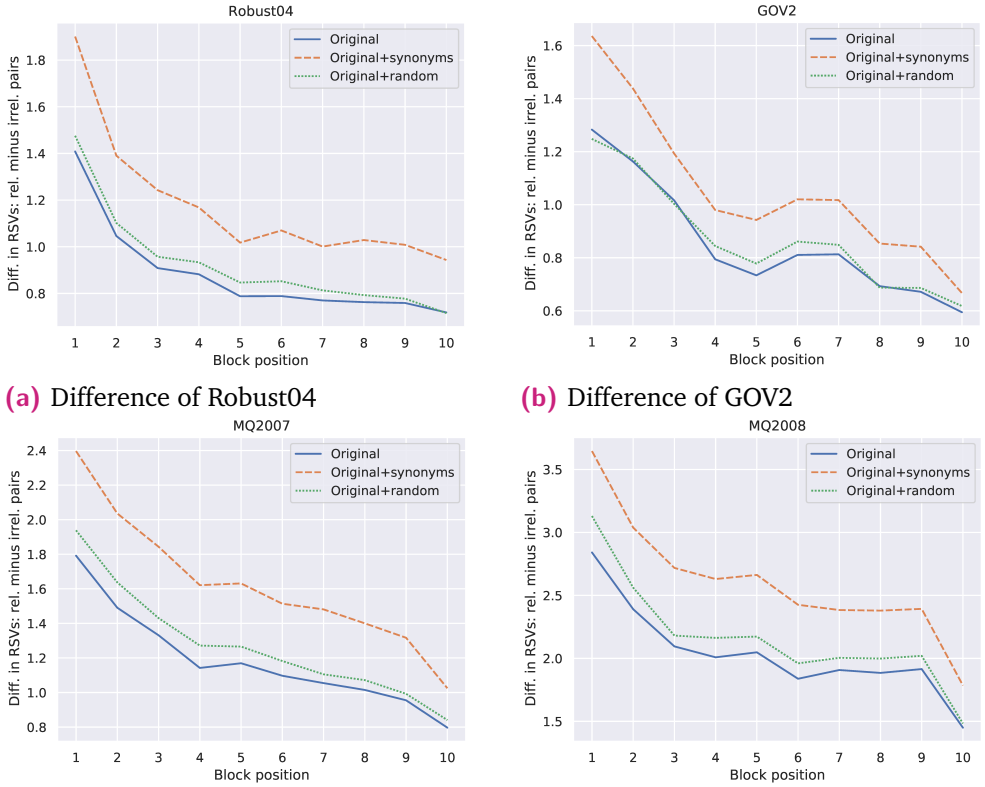
Tab. 2.3.: Statistics: number of blocks selected.

Dataset	MQ2007	MQ2008	Robust04	GOV2
# of blocks matching q_{bool}	914,901	217,035	45,566	399,549
# of blocks matching $q_{bool} + q_{bool}^{syn}$	994,657 (+8.72%)	228,451 (+5.26%)	55,976 (+22.85%)	458,277 (+14.70%)
# of blocks matching $q_{bool} + q_{bool}^{rand}$	917,026 (+0.23%)	217,378 (+0.16%)	45,717 (+0.33%)	400,570 (+0.26%)

types of boolean queries. Using the the *eldar* package[9], we then computed, for very relevant and relevant documents, the number of blocks matching q_{bool}, those matching either q_{bool} or q_{bool}^{syn}, and matching either q_{bool} or q_{bool}^{rand}. The results are given in Table 2.3 with the percentage increase with respects to the number of blocks matched by q_{bool}. One can observe that leveraging the knowledge about synonyms enables to match more blocks: up to 22.85% increase in the number of blocks matched in the case of the Robust04 dataset (against 0.33% for the random query), 14.70% increase for GOV2 (against 0.26% for the random query, 8.72% in the case of MQ2007 (against 0.23% for the random query) and 5.26% increase for MQ2008 (against 0.16% for the random query).

To complement the above analysis, we also assessed whether synonyms can leverage relevance signals in blocks at different positions. To do so, we used a standard expansion of the original query by simply adding all synonyms of all query terms without duplicates. For comparison purposes, we did the same with the words randomly selected in WordNet. The obtained queries, an example of which is given in Table 2.2, will be respectively referred to as $Original + synonyms$ and $Original + random$. We then computed the difference in average BM25 scores between relevant and irrelevant documents across all positions for the three types of queries: Original, Original + synonyms, Original + random. The results obtained are reported in Figure 2.4. As one can note, blocks in relevant documents score higher than the blocks in irrelevant documents, the gap being consistently and significantly higher when using the query with synonyms than when using the original

[9] `https://github.com/kerighan/eldar`



(a) Difference of Robust04

(b) Difference of GOV2

(c) Difference of MQ2007

(d) Difference of MQ2008

Fig. 2.4.: Difference in RSV scores between relevant and irrelevant documents for the original query q, the expanded one q^{exp} and the random expanded one q^{rand_exp} across block positions.

query or the original query with additional random words. Furthermore, the two curves, Original and Original + random, are very close to each other and almost identical on GOV2. This shows that the increase in the BM25 scores when using synonyms is not due to the length of the query, and that synonyms help identify relevant signals in blocks. It thus may be useful to use matching strategies that go beyond an exact matching of query words.

In the next sections, we will present two different ways to select blocks in documents. The first one is based on standard IR functions, namely TF-IDF and BM25, to compute relevance scores between queries and blocks; it is thus based on an exact matching of words present in the query. The second one aims at learning a scoring function that exploits the semantic similarities

between query words and document words. In both cases, the top scoring blocks are then used to compute the relevance score of the document.

2.4 Selecting Blocks with Standard IR Functions: TF-IDF and BM25

TF-IDF is a popular IR model which amounts to score a document through the product of the term frequency (TF) and inverse document frequency (IDF) scores of the query words present in that document. Applied at the block level, this yields the following retrieval status value (RSV):

$$RSV(q,b)_{TF-IDF} = \sum_{w \in q \cap b} \underbrace{(\ln tf_w^b + 1)}_{TF} \cdot IDF(w).$$

in which tf_w^b corresponds to the number of occurrences of word w in block b and $IDF(w)$ to the inverse document frequency of word w. $IDF(w)$ is defined here according to scikit-learn [130] by:

$$IDF(w) = \ln \frac{N+1}{df_w + 1},$$

where N is the number of documents in the collection and df_w corresponds to the number of documents containing w.

For BM25 [150, 147], the RSV score is defined by:

$$RSV(q,b)_{BM25} = \sum_{w \in q \cap b} IDF(w) \cdot \frac{tf_w^b}{k_1 \cdot (1 - b + b \cdot \frac{l_b}{l_{avg}}) + tf_w^b},$$

where l_b is the length of block b, l_{avg} the average length of the blocks in d, and k_1 and b two hyperparameters. As standard in this setting, we use the

IDF formulation of Lucene version as shown in Kamphuis et al. [67] which slightly differs from the previous one:

$$IDF(w) = \ln \frac{N+1}{df_w + 0.5}.$$

According to [120], RSVs in different IR systems have different scales. In the boolean model, RSVs are either 0 or 1. Vector space model can generate RSV in $[-1, 1]$ by cosine similarity or scalar product $\mathbb{R}$. In this section, we rely on TF-IDF and BM25, which can be viewed as an model that rely on term matching and the scale of the RSVs is $\mathbb{R}$. The blocks are ranked according to RSVs by descending order.

Block or document level IDF As the reader may have noticed, the IDF is based on documents and not on blocks. There are two main reasons for this. First of all, considering blocks instead of documents for the IDF may artificially increase the df score of a word since important words of a document are likely to occur in many blocks of that document. The second reason is that one can directly re-use existing IDF scores computed at the document level. Note that all words are lowercased prior to compute TF-IDF and BM25 scores.

The overall architecture of an IR neural system relying on the above standard IR models to select blocks is presented in Figure 2.5. As one can note, the query-block scoring part is used to select relevant blocks across the whole document, which can be viewed as a pre-ranking strategy. The Deep Neural IR Network can represent any neural IR network which generates relevance scores for query-document pairs, scores which can in turn be used as input to a learning-to-rank (LTR) loss, be it a pairwise or listwise loss. We focus in this study on two state-of-the-art neural IR Models, namely Vanilla BERT,

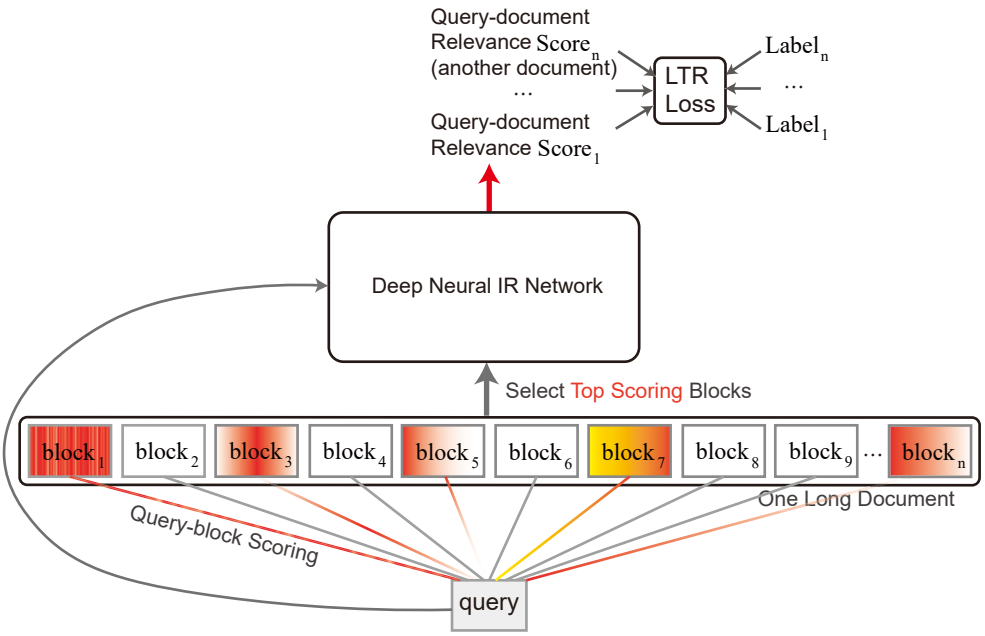
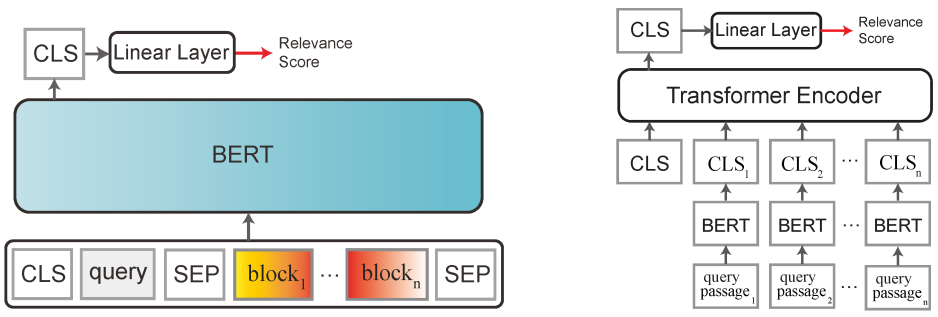


Fig. 2.5.: An illustration of the architecture of KeyB (e.g., TF-IDF or BM25).

in which case we refer to the models obtained as *KeyB(vBERT)*$_{TF-IDF}$ and *KeyB(vBERT)*$_{BM25}$, and PARADE, in which case we refer to the models obtained as *KeyB(PARADEk)*$_{TF-IDF}$ and *KeyB(PARADEk)*$_{BM25}$, k representing in that case the number of passages retained. Figure 2.6 provides an illustration of Vanilla BERT and PARADE.



(a) Vanilla BERT neural IR network

(b) PARADE neural IR network

Fig. 2.6.: Different deep neural IR networks.

2.4.1 KeyB(vBERT)

The KeyB(vBERT) models rely on four steps:

1. *Block segmentation* We adopt here the dynamic programming method proposed in [33] to segment documents into blocks, each block having a maximum of 63 tokens. This method sets different costs for different punctuation marks and aims at segmenting in priority on strong punctuation marks such as "." and "!". It is thus close to a sentence segmentation procedure.

2. *Block selection* Each block is assigned a relevance score provided by either TF-IDF or BM25, as described above.

3. *Query-blocks representation* The most relevant blocks are then concatenated together in their order of appearance in the document and with the query to form the input of BERT (see Figure 2.6). As the input number of tokens for BERT is limited to 512, the last block is truncated if necessary. The number n of selected blocks depends on the capacity of BERT and is defined by:

$$3 + l_q + \sum_{i=1}^{n-1} l_{b_i} + \text{trunc}(l_{b_n}) = 512,$$

   where l_q denotes the length of the query and l_{b_i} the length of a block, potentially truncated for the last selected block. The final query-representation corresponds to the [CLS] embedding of the learned BERT.

4. **Document ranking** We rely here on a one-layer dense, linear network to generate the final relevance score, using a learning to rank loss computed on a mini-batch (see Section 2.4.3).

2.4.2 KeyB(PARADEk)

As mentioned in Section 2.2, PARADE is a state-of-the-art model which computes a query-document representation on the basis of the query-passage representations. A document is first segmented into passages. Each passage is then fed, together with the query, to a BERT model. The CLS embedding thus obtained corresponds to the query-passage representation. Denoting by p_i the i^{th} passage and p_i^{cls} the corresponding query-passage representation, one has:

$$p_i^{cls} = BERT(q, p_i).$$

The query-passage representations are then aggregated to obtain the query-document representation, that is finally fed to a feed-forward neural network to obtain the relevance score of the document.

Four types of aggregation methods have been proposed: PARADE–Max, PARADE–Attn, PARADE–CNN and PARADE–Transformer. PARADE–Max uses a max pooling operation on the passage relevance representations. PARADE–Attn assumes that each passage contributes differently to the relevance of a document to the query and passage weights are predicted by a feed-forward network. PARADE–CNN stacks Convolutional Neural Network (CNN) layers in a hierarchical way whereas PARADE–Transformer stacks a full attention model which enables query-passage representations to interact in a hierarchical manner through a transformer. Its architecture is depicted in Fig. 2.6b. Because of its good behavior [83], we have retained this last aggregation method here. We will refer to is as just PARADE in the remainder.

Let $x^{(\ell)}$ denote the input of the ℓ transformer layer. $x^{(0)}$ consists in the concatenation of the query-passage representations (p_i^{cls}). $x^{(\ell+1)}$ is then given by:

$$x^{(\ell+1)} = \text{LayerNorm}(h + \text{FFN}(h)),$$
$$\text{with } h = \text{LayerNorm}(x^{(\ell)} + \text{MultiHead}(x^{(\ell)})).$$

LayerNorm refers to the layer-wise normalization described in [2] and Multi-Head to the multi-head self-attention [166]. FFN is a two-layer feed-forward network with a ReLU activation in between. The [CLS] vector of the final output layer, which is denoted by d^{cls}, constitutes the query-document representation. A linear layer is then used to generate the query-document relevance score:

$$RSV(q, D) = W d^{cls},$$

where W is a learnable weight matrix.

It is important to note that the PARADE model described above can have a high complexity when the number of passages considered is large (for example, when using 10 passages, the model can not fit on a standard GPU with 11 GB memory even with mixed precision). Indeed, in that case, the input consists in a large tensor which can only be stored in a large CUDA memory [45] with high computational complexity. To avoid this, the number of passages is limited to 16. When a document contains more than 16 passages, then only the first passage, the last passage and 14 sampled passages are used. Whether one restricts the number of passages or not, one problem faced by PARADE lies in the fact that non relevant passages can bring noise in the query-document representation, and hamper the computation of a precise retrieval score. To address this problem, we propose here to select a fixed, small number of passages, denoted by k. The selected passages are

then concatenated and fed to a transformer as in the original PARADE model. When using standard IR functions as described above to select passages, the query-document representations denoted by d^{cls}_{TF-IDF} and d^{cls}_{BM25} are thus obtained. The final relevance score of the document is then obtained by:

$$RSV(q, D) = W d^{cls}_{TF-IDF} \text{ or } W d^{cls}_{BM25}.$$

2.4.3 Model Training

The block/passage selection process is applied in both the training and testing phases. The BERT models used in Vanilla BERT and PARADE are fine-tuned during training. The parameters of all the other components (final layer for Vanilla BERT, final layer and transformer layers for PARADE) are directly trained. For fine-tuning and training, the following pairwise hinge loss [49] is used:

$$\mathcal{L}(q, d^+, d^-; \Theta) = \max(0, 1 - s(q, d^+; \Theta) + s(q, d^-; \Theta)),$$

where q represents a query, (d^+_q, d^-_q) a positive and negative training document pair for q, Θ the parameters of the model considered and s the predicted relevance score for a query-document pair. Other choices are of course possible, including ones based on a listwise loss. We chose the pairwise hinge loss here as it is a very common choice, used in many IR methods [102, 89, 125].

2.5 Learning to Select Blocks

The analysis conducted in Section 2.3 suggests that a fuzzy matching procedure may be preferred over one based on an exact matching. We thus aim here to learn a scoring function that exploits the semantic similarities between words in queries and blocks using the same two neural IR model as before, Vanilla BERT and PARADE.

2.5.1 Improving Vanilla BERT

We focus here on the Vanilla BERT model to compute the relevance score of a block. That is, instead of just using the semantic representation of query-block pairs to compute the relevance score of a document as in the previous models, we also make use here of these representations to select the most appropriate blocks. Furthermore, we propose here to share the semantic representation for both purposes, selecting blocks and computing the overall relevance score, as both are based on the relevance information contained in each block. It is of course possible to make use of two different models, with however higher computational and training costs. The overall architecture of the model proposed is depicted in Figure 2.7, in which the same BERT model and linear layer are used at different time slices, first to compute a query-block representation, from which ([CLS] embedding) the relevance score of the block is derived, and then to compute the query-document representation ([CLS] embedding) based on the top ranked blocks, finally to obtain the score of the document. This query-document representation is identical to the one used in the KeyB(vBERT) model, the only difference lying in the way the blocks are selected. For the query-block represntation, both the BERT model and the linear layer are only used for scoring and are not updated

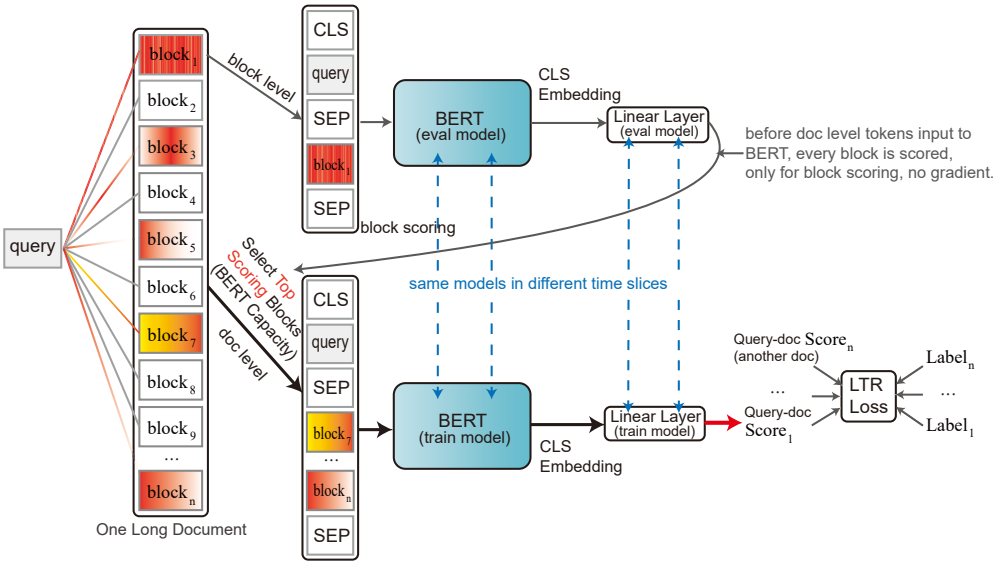


Fig. 2.7.: An illustration of the architecture of KeyB(vBERT)$_{BinB}$. Here, the BERT model and linear layer are used to select blocks too. While the neural model being trained with document level annotations, this model would become able to score blocks for a query.

via back-propagation (hence the term "eval model" used in Figure 2.7). The score of a block b for a given query q is defined by:

$$RSV(q,b)_{BERT} = W_d b^{cls},$$

where W_d is the weight of the dense linear layer on top of BERT, and b^{cls} is the query-block relevance representation:

$$b^{cls} = BERT(q,b).$$

For a given query, the BERT model is first used to generate a relevance score for each block in the document. Since the BERT model is not well fine-tuned at the beginning of the process, the block selection process acts as an almost random selector. However, with the query-document level relevance labels, after each back-propagation, the BERT model is improved and provides better relevance scores for each block in the next iterations. In return, the

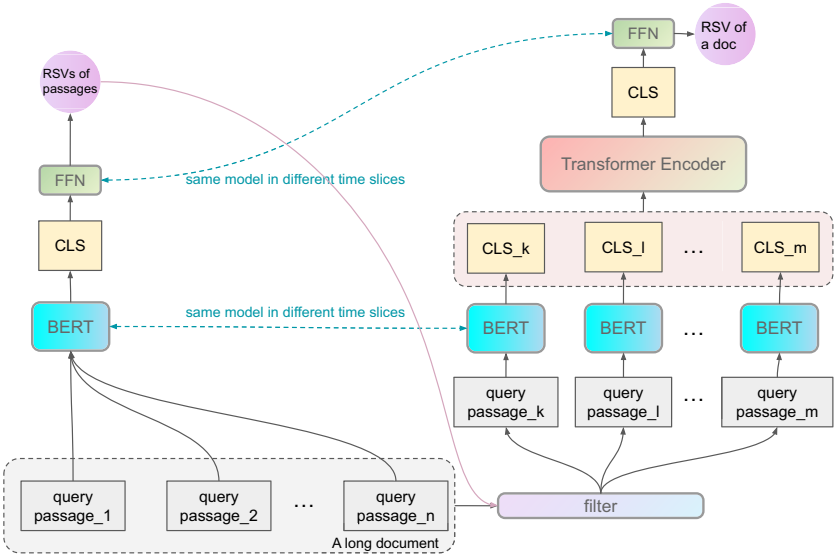


Fig. 2.8.: An illustration of the architecture of KeyB(PARADEk)$_{BinB}$. Here, the BERT model and linear layer of PARADE are used to select blocks too.

BERT model benefits from the block selection too. This can be viewed as a self-evolution process: the BERT model evolves to provide appropriate query-block representations to be able to select blocks, and meanwhile, appropriate query-document representations to be able to generate relevance scores for query-document pairs. Because of this self-evolution, we refer to this model as KeyB(vBERT)$_{BinB}$, where $BinB$ means "BERT in BERT".

2.5.2 Improving PARADE

We here propose to improve the PARADE model with the learning approach for selecting passages.

To do so, we first follow the same strategy as the one used for Vanilla BERT by trying to use the same modules for both selecting passages and scoring documents. A major difficulty for doing so here is that the Transformer encoder used in PARADE to score a complete document takes as input the representation of several query-block pairs. As such, it cannot be used to

score a single block. For this reason, we first propose to only share the initial BERT model and the final feed-forward neural network as illustrated in Figure 2.8. In this new architecture, one obtains the [CLS] query-passage representation, denoted by p^{cls} for passage p, using the base BERT model of PARADE which is shared in different time slices. The final feed-forward neural network is used to provide, from a CLS representation, a score for a query-passage pair (Fig. 2.8, left) and a score for a query-document pair, where a document is seen as the concatenation of passages (Fig. 2.8, right). The score of a passage p for a given query q is then defined by:

$$RSV(q,p)_{BERT} = W_d p^{cls},$$

where W_d is the weight of the feed-forward neural network after the Transformer encoder, and p^{cls} is the query-passage relevance representation:

$$p^{cls} = BERT(q,p).$$

We refer to this approach as KeyB(PARADEk)$_{BinB}$, where k is the number of selected passages.

If the previous attempt makes it possible to reuse modules for selecting passages and scoring documents, it may however suffer from the fact that the same feed-forward neural network is required to produce a relevance score from two different CLS representations: one restricted to a single query-passage pair for selecting passages, and one resulting from an encoding, through a Transformer, of several query-passage representations for scoring the document. We propose to fix this issue by decoupling the passage scoring module from the main model, as illustrated in Figure 2.9 which relies on a different BERT module and feed-forward neural network to select passages. But which BERT model and feed-forward layer to use? A simple answer to

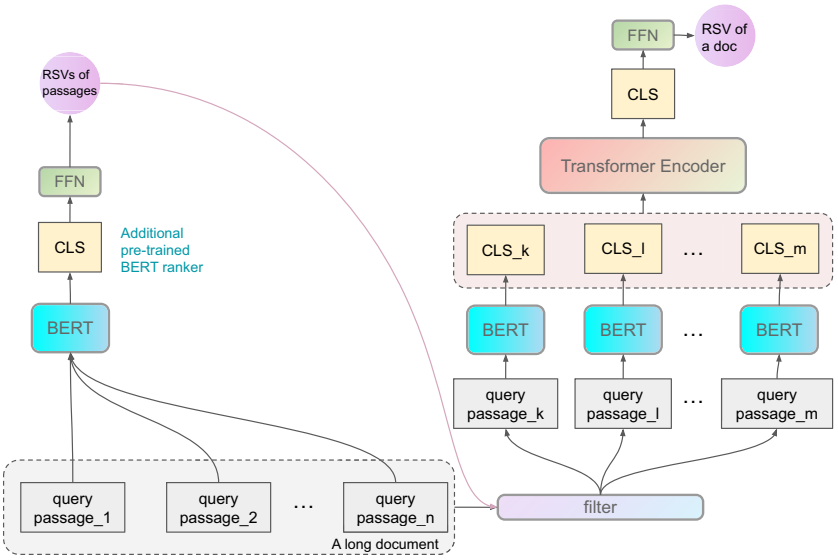


Fig. 2.9.: An illustration of the architecture of KeyB(PARADEk)$_{BinB2}$. Here, additional BERT model and additional linear layer are used to select blocks.

this question is to re-use the pretrained BERT ranker and final feed-forward neural network of one of the KeyB(vBERT) models previously proposed as these models are compatible with the input used here. The score of a passage p for a given query q is in this case defined by:

$$RSV(q,p)_{BERT2} = W_d^2 p^{cls_2},$$

where W_d^2 is the weight of the feed-forward neural network following the additional BERT module, and p^{cls_2} is the query-passage relevance representation after the additional BERT module:

$$p^{cls_2} = BERT2(q,p).$$

In practice, we use the BERT module and feed-forward neural network from the KeyB(vBERT)$_{BM25}$ model and refer to this approach as KeyB(PARADEk)$_{BinB2}$, where k is the number of selected passages.

2.6 Experiments on Standard IR Collections

We conducted a first series of experiments on the same collections as the ones used in the analysis presented in Section 2.3, namely MQ2007, MQ2008, Robust04 and GOV2. These experiments aim at assessing the validity of the models proposed, KeyB(vBERT) and KeyB(PARADEk)[10], and at comparing them to models that have provided state-of-the-art results on these collections. These latter models are:

- **DeepRank**: We compare with the DeepRank with CNN based measure network, which leads to better results than 2D-GRU based on [125]. We used the PyTorch implementation of DeepRank [11] with given hyperparameter for the architecture. Following the implementation, the number of words per document is set to 2000. Hence, by adopting the default parameters the documents longer than 2000 tokens are truncated.

- **PARADE**: We compare our model with the PARADE-Transfomer version as this version performs mostly better and we call this baseline PARADE in short. This method is a state-of-the-art IR method which first segments documents into passages that are fed to a BERT model. Transformer layers are then used to compute global attention scores over the [CLS] embeddings of different passages. A final linear layer is used for computing the document level relevance score. We have used the open-sourced PyTorch implementation[12].

- **CEDR-KNRM**: CEDR is also a reported state-of-the-art model that incorporates BERT's classification vector into existing neural models. We

[10] These models are developed on top of the Georgetown IR Lab implementation and are available at: `https://github.com/lmh0921/keyB`.
[11] `https://github.com/pl8787/DeepRank_PyTorch`
[12] `https://github.com/capreolus-ir/capreolus`

choose the CEDR-KNRM as the baseline for it is reported better than other variants. We have used the Georgetown IR Lab implementation[13] with the Hugging Face transformer module[14].

For assessing the various components of the models proposed, we also used the following baseline models:

- **BM25**: We use the BM25 implementation of Anserini [185], with default hyperparameters. This model is the one presented in Section 2.4 and is used both as a baseline on all collections. In addition, it is used as a first stage ranker, retaining only the top 200 documents, for the neural IR models on Robust04 and GOV2. For MQ2007 and MQ2008, we converted the original documents to JSON files which can be indexed by Anserini. We then used the BM25 model from Anserini as a baseline but not as a first stage ranker as in MQ2007 and MQ2008 each query only contains dozens of labeled documents for ranking. For all experiments, the hyper-parameters $k1$ and b of BM25 are set to Anserini's default values: 0.9 and 0.4.

- **Vanilla BERT**: This is a BERT baseline that truncates long documents to the first 512 tokens. Except for this difference in the input used for BERT, the architecture is the same as the one of KeyB(vBERT). This baseline thus allows one to evaluate the impact of key blocks. For implementation, we have used the same library as the one for CEDR-KNRM.

- **Random Select**: This architecture is the same as the one with KeyB(vBERT) except that it does not incorporate the block selection mechanism. To be specific, a long document is also firstly segmented into blocks, but

[13] `https://github.com/Georgetown-IR-Lab/cedr`
[14] `https://github.com/huggingface/transformers`

each block is given a random score. This is to say, without local block pre-ranking step, the blocks are selected randomly.

2.6.1 Experimental Design

It is common in IR to first filter documents with a classical IR system prior to re-rank them with a more complex (and usually more time consuming) system [118, 102, 26, 83]. We adopt here this approach and use BM25 as the first filtering system, retaining only, for each query, the first 200 documents on Robust04 and GOV2 as done in [102]. As queries in MQ2007 and MQ2008 are associated with far less than 200 documents, this filtering step is not necessary. Following [125], we merged the MQ2007 and MQ2008 training sets as the training set of MQ2008 is relatively small. The validation and testing sets remain unchanged. For all neural IR models, the pairwise hinge loss [125] is used for training the models.

Furthermore, for all experiments, 5-fold cross-validation is used with three folds for training, one fold for validation and one fold for testing. For Robust04 and GOV2, we used the keyword (title) version of queries [26]. For MQ2007 and MQ2008, the default queries are used. All neural IR models based on BERT use the "BERT-Base, Uncased, L=12, H=768"[15] pre-trained language model (but not further pre-trained with additional data) for fair comparison.

For DeepRank, we followed the experimental setup of [125] and used GloVe embeddings [131] of dimension 50, which are pre-trained on Wikipedia 2014 + Gigaword 5. For the preprocessing of document and query words, we applied lower-case, removed English stop words, stemmed with Krovetz

[15] https://github.com/google-research/bert

stemmer [77] and removed words occurring less than 5 times. The Adam optimizer is used for training the network and the learning rate is searched over values of $\{0.01, 0.005, 0.001\}$. We selected the model that leads to the best MAP score on the validation set.

For PARADE, the first and last passages are always selected and the other passages are randomly sampled[16]. The number of passages considered is a hyperparameter in PARADE that needs to be set. Following [83], the passages are obtained with 225 document tokens with stride size 200, the maximum passage sequence length is set as 256, and the number of passages is set to 16 for all collections. Note again that for a fair comparison, BERT is not further trained on MS-MARCO [117]. For all BERT based IR models, we use the BERT implementation of PyTorch Huggingface library [176].

For the variants of PARADE we have proposed in Section 2.4.2, we have used TF-IDF or BM25 to select the top 5 passages. The choice of 5 passages provides a good balance for effectiveness, as ca. 1000 tokens are considered, and efficiency, as a standard RTX 2080ti GPU with 11GB memory is not able to deal with 12 passages and automatic mixed precision [109] for example. The resulting models, based on PARADE and integrating the top 5 passages using TF-IDF or BM25 for local pre-ranking, are referred to as KeyB(PARADE5)$_{TF-IDF}$ and KeyB(PARADE5)$_{BM25}$. The variants of PARADE proposed in Section 2.5.2 and with top 5 passages are are referred to as KeyB(PARADE5)$_{BinB}$ and KeyB(PARADE5)$_{BinB2}$. We here use the "cross-encoder/ms-marco-MiniLM-L-12-v2"[17] version model as the standalone BERT ranker for KeyB(PARADE5)$_{BinB2}$. This standalone BERT ranker is firstly finetuned on each collection as the way of KeyB(PARADE5)$_{BM25}$ (trained with document level labels), then it can generate a RSV for each query-

[16] As done `https://github.com/canjiali/PARADE/blob/master/generate_data.py` in line 304.

[17] `https://www.sbert.net/docs/pretrained_cross-encoders.html`

passage pair (this means that a query and passage are concatenated and directly input to the BERT ranker). All other settings are the same as the ones for the PARADE model described above. For comparison purposes, we also used another variant of PARADE, called PARADE5, relying on the first, last and 3 randomly chosen passages.

Each model is trained for a maximum of 10 epochs. For Robust04 and GOV2, one epoch represents 1024 batches of two pairs, each pair being of the form $((q, d_q^+), (q, d_q^-))$ where d_q^+ is a positive document for query q and d_q^- a negative document. Since MQ2007 and MQ2008 have more queries than Robust04 and GOV2, each epoch of these two collections is composed of 2048 batches of two pairs identical to the ones above. The negative example in a pair is generated randomly for all models from the set of documents which are either labeled not relevant or not labeled for the query. Although different negative sampling mechanisms may impact final results [99], the above simple negative sampling strategy achieves very good performance and has been successfully used in previous studies [102, 89, 125]. Gradient accumulation is employed every 8 steps to fit on a single GPU with 11GB memory like a RTX 2080ti GPU, simulating a batchsize with 16 training pairs, as done in [102]. Automatic mixed precision [109] is used to speed up training. We adopt a validation mechanism with validation set to report each metric on the test set. That is to say, for each evaluation metric, we obtain the best performing model in the 10 epochs on the validation set, and use this model to obtain results on the test set for this metric. Each model is trained using Adam optimizer (the transformer layers are trained with a rate of $2 * 10^{-5}$ while the linear layer with a rate of 10^{-3}).

Results, for all models, are measured with P@1, P@5, P@10, P@20, MAP, NDCG@1, NDCG@5, NDCG@10, NDCG@20 and NDCG, NDCG and MAP being computed on all available documents for each query (the number of

documents per query is 200 on Robust04 and GOV2 and varies from one query to the other on MQ2007 and MQ2008). Each metric is calculated with `pytrec_eval`[18] [165], which is a wrapper of `trec_eval`[19]. Lastly, a paired t-test is used to assess whether differences are significant or not.

2.6.2 Experimental Results

The results obtained on all the four collections are displayed in Table 2.4 to Table 2.7[20]. We first analyze the results of the KeyB(vBERT) models, prior to analyze the ones of KeyB(PARADE5)$_{BM25}$ and compare them.

Improving Vanilla BERT model with selected key blocks We propose to analyze the experimental results by answering several research questions.

RQ1 How effective are KeyB(vBERT) models compared to baseline models (BM25, DeepRank)?

The first conclusion we draw from Tables 2.4 to 2.7 is that all KeyB(vBERT) models outperform both baseline models on all collections, for all metrics, by a large margin. Furthermore, on all collections, KeyB(vBERT) models significantly outperform most of metrics.

RQ2 How effective are KeyB(vBERT) models compared to standard BERT based models (Vanilla BERT, CEDR-KNRM)?

[18] `https://github.com/cvangysel/pytrec_eval`
[19] `https://trec.nist.gov/trec_eval`
[20] As pointed out in e.g. `https://github.com/Georgetown-IR-Lab/cedr/issues/22`, the results obtained for CEDR-KNRM with the code provided by the authors differ from the ones reported in the original paper. We have also observed this in our experiments. The same holds for DeepRank; in that case however the original paper provides results for most of the metrics we have retained on MQ2007 and MQ2008. We have thus reported the original results in Tables 2.6 and 2.7, under the name DeepRank*. These results do not change our conclusions.

Tab. 2.4.: Results on *Robust04* dataset. Best results are in **bold**. For KeyB(vBERT) models, a significant difference with BM25 is marked with a 'B', with DeepRank with a 'D', with Vanilla BERT with a 'V', with CEDR-KNRM with a 'C', and with Random select with an 'R'. For KeyB(PARADE) models, a significant difference with BM25 is marked with a 'B', with DeepRank with a 'D', with PARADE with a 'P', and with PARADE5 with a '5'. A paired t-test ($p - value \leq 0.05$) is used for measuring significance.

Model	P@1	P@5	P@10	P@20	MAP	NDCG@1	NDCG@5	NDCG@10	NDCG@20	NDCG
Baseline models										
BM25	0.5542	0.5004	0.4382	0.3631	0.2334	0.5080	0.4741	0.4485	0.4240	0.4402
DeepRank	0.5663	0.4538	0.3907	0.3331	0.2145	0.5081	0.4386	0.4051	0.3864	0.4272
BERT based models										
Vanilla BERT	0.6067	0.5478	0.4843	0.4088	0.2510	0.5706	0.5337	0.4945	0.4678	0.4553
CEDR-KNRM	0.6220	0.5542	0.4840	0.4097	0.2440	0.5878	0.5253	0.5093	0.4803	0.4600
Random Select	0.5983	0.5108	0.4730	0.4059	0.2453	0.5482	0.4856	0.4880	0.4688	0.4540
KeyB(vBERT)$_{TF-IDF}$	0.6146	0.5430^{BDR}	0.4963^{BDR}	0.4208^{BDR}	0.2628^{BDVCR}	0.5764	0.5275^{BDR}	0.5099^{BD}	0.4884^{BDVR}	0.4684^{BDVCR}
KeyB(vBERT)$_{BM25}$	0.6468^{BD}	0.5622^{BDR}	0.4976^{BDR}	**0.4241**^{BDVR}	0.2609^{BDCR}	0.6004^{BD}	0.5512^{BDR}	0.5166^{BDVR}	0.4941^{BDVR}	0.4687^{BDVCR}
KeyB(vBERT)$_{BinB}$	**0.6710**^{BDR}	**0.5661**^{BDR}	**0.5088**^{BDVCR}	**0.4241**^{BDVR}	**0.2722**^{BDVCR}	**0.6289**^{BDR}	**0.5554**^{BDR}	**0.5249**^{BDVR}	**0.4958**^{BDVR}	**0.4768**^{BDVCR}
PARADE based models										
PARADE	0.6869	0.5686	0.5080	0.4309	0.2739	0.6166	0.5510	0.5146	0.5017	0.4737
PARADE5	0.6388	0.5334	0.4868	0.4125	0.2477	0.5905	0.5215	0.5055	0.4761	0.4594
KeyB(PARADE5)$_{TF-IDF}$	0.6790^{BD}	0.5687^{BD5}	0.5093^{BD5}	0.4319^{BD5}	0.2714^{BD5}	**0.6348**^{BD}	0.5467^{BD}	0.5218^{BD}	0.4989^{BD5}	0.4710^{BD5}
KeyB(PARADE5)$_{BM25}$	**0.6871**^{BD}	**0.5768**^{BD5}	**0.5177**^{BD5}	0.4337^{BD5}	0.2757^{BD5}	0.6329^{BD}	**0.5636**^{BD5}	**0.5304**^{BD5}	0.5040^{BD5}	0.4735^{BD5}
KeyB(PARADE5)$_{BinB}$	0.6308^{B}	0.5479^{BD}	0.5057^{BD}	0.4200^{BD}	0.2629^{BDP5}	0.5885^{BD}	0.5329^{BD}	0.5283^{BD5}	0.4967^{BD5}	0.4687^{BD5}
KeyB(PARADE5)$_{BinB2}$	0.6427^{BD}	0.5758^{BD5}	0.5112^{BD5}	**0.4378**^{BD5}	**0.2779**^{BD5}	0.5965^{BD5}	0.5625^{BD5}	0.5247^{BD}	**0.5058**^{BD5}	**0.4778**^{BD5}

Tab. 2.5.: Results on *GOV2* dataset. Best results are in **bold**. Best results are in **bold**. For KeyB(vBERT) models, a significant difference with BM25 is marked with a 'B', with DeepRank with a 'D', with Vanilla BERT with a 'V', with CEDR-KNRM with a 'C', and with Random select with an 'R'. For KeyB(PARADE) models, a significant difference with BM25 is marked with a 'B', with DeepRank with a 'D', with PARADE with a 'P', and with PARADE5 with a '5'. A paired t-test ($p - value \leq 0.05$) is used for measuring significance.

Model	P@1	P@5	P@10	P@20	MAP	NDCG@1	NDCG@5	NDCG@10	NDCG@20	NDCG
Baseline models										
BM25	0.6510	0.6054	0.5792	0.5362	0.2331	0.5034	0.4904	0.4867	0.4774	0.4296
DeepRank	0.6453	0.5682	0.5143	0.4880	0.2151	0.4738	0.4363	0.4194	0.4170	0.4120
BERT based models										
Vanilla BERT	0.6241	0.6068	0.5672	0.5475	0.2321	0.4531	0.4954	0.4837	0.4764	0.4279
CEDR-KNRM	0.6239	0.6133	0.5886	0.5556	0.2375	0.4929	0.4891	0.4892	0.4769	0.4315
Random Select	0.6839	0.6169	0.5984	0.5640	0.2467	0.4995	0.4811	0.4955	0.4853	0.4358
KeyB(vBERT)$_{TF-IDF}$	0.7122	0.6735^{BDVCR}	0.6446^{BDVCR}	0.6123^{BDVCR}	0.2583^{BDVCR}	**0.5574**^{D}	0.5256^{DR}	0.5340^{BDVCR}	0.5269^{BDCR}	0.4413^{BDC}
KeyB(vBERT)$_{BM25}$	0.6634	0.6524^{D}	0.6303^{BDC}	0.5997^{BDVCR}	0.2643^{BDVCR}	0.5171	0.5341^{DVR}	0.5272^{BDVC}	0.5199^{BDCR}	0.4447^{BDVCR}
KeyB(vBERT)$_{BinB}$	**0.7651**^{BDVC}	**0.6937**^{BDVCR}	**0.6645**^{BDVCR}	**0.6125**^{BDVCR}	**0.2674**^{BDVCR}	**0.5574**^{D}	**0.5414**^{BDVCR}	**0.5356**^{BDVCR}	**0.5295**^{BDVCR}	**0.4453**^{BDVCR}
PARADE based models										
PARADE	0.7244	0.7016	0.6631	0.6133	0.2621	**0.5930**	0.5518	0.5562	0.5466	0.4484
PARADE5	0.6906	0.6429	0.6246	0.5707	0.2462	0.5463	0.5327	0.5161	0.5053	0.4386
KeyB(PARADE5)$_{TF-IDF}$	0.7386	0.6931^{BD5}	0.6605^{BD5}	0.6222^{BD5}	0.2728^{BDP5}	0.5569^{D}	0.5536^{BD}	0.5498^{BD}	0.5352^{BD5}	0.4537^{BD5}
KeyB(PARADE5)$_{BM25}$	**0.7720**^{BD}	0.6931^{BD5}	0.6528^{BD}	0.6397^{BDP5}	**0.2745**^{BDP5}	0.5806^{D}	**0.5783**^{BD5}	0.5529^{BD5}	**0.5624**^{BD5}	**0.4578**^{BDP5}
KeyB(PARADE5)$_{BinB}$	0.7055	**0.7196**^{BD5}	0.6563^{BD}	0.6212^{BD5}	0.2680^{BD5}	0.5640^{D}	0.5660^{BD}	0.5451^{BD}	0.5379^{BD}	0.4495^{BD}
KeyB(PARADE5)$_{BinB2}$	0.7253	0.7034^{BD5}	**0.6771**^{BD5}	**0.6407**^{BDP5}	0.2733^{BDP5}	0.5706^{D}	0.5512^{BD}	**0.5676**^{BD5}	0.5591^{BD5}	0.4554^{BD5}

Tab. 2.6.: Results on *MQ2007* dataset. DeepRank* represents the results from the original paper. Best results are in **bold**. For KeyB(vBERT) models, a significant difference with BM25 is marked with a 'B', with DeepRank with a 'D', with Vanilla BERT with a 'V', with CEDR-KNRM with a 'C', and with Random select with an 'R'. For KeyB(PARADE) models, a significant difference with BM25 is marked with a 'B', with DeepRank with a 'D', with PARADE with a 'P', and with PARADE5 with a '5'. A paired t-test ($p - value \leq 0.05$) is used for measuring significance.

Model	P@1	P@5	P@10	P@20	MAP	NDCG@1	NDCG@5	NDCG@10	NDCG@20	NDCG
Baseline models										
BM25	0.4186	0.3969	0.3757	0.3391	0.4527	0.3712	0.3954	0.4309	0.4962	0.5933
DeepRank	0.4444	0.4201	0.3898	0.3473	0.4596	0.3942	0.4168	0.4468	0.5088	0.6012
DeepRank*	0.508	0.452	0.412	-	0.497	0.441	0.457	0.482	-	-
BERT based models										
Vanilla BERT	0.5266	0.4741	0.4257	0.3606	0.5073	0.4708	0.4808	0.5070	0.5620	0.6379
CEDR-KNRM	0.5284	0.4768	0.4233	0.3601	0.5066	0.4814	0.4874	0.5084	0.5601	0.6380
Random Select	0.5343	0.4768	0.4347	0.3656	0.5207	0.4808	0.4980	0.5224	0.5775	0.6499
KeyB(vBERT)$_{TF-IDF}$	0.5425^{BD}	0.4926^{BDVCR}	0.4465^{BDVCR}	0.3702^{BDVCR}	0.5323^{BDVCR}	0.4917^{BD}	0.5043^{BDVC}	0.5342^{BDVCR}	0.5864^{BDVCR}	0.6551^{BDVC}
KeyB(vBERT)$_{BM25}$	0.5526^{BDVC}	0.4946^{BDVCR}	0.4408^{BDVC}	0.3705^{BDVCR}	0.5305^{BDVCR}	0.4933^{BDV}	0.5061^{BDVC}	0.5339^{BDVCR}	0.5824^{BDVC}	0.6528^{BDVC}
KeyB(vBERT)$_{BinB}$	**0.5597**^{BDVCR}	**0.4971**^{BDVCR}	**0.4503**^{BDVCR}	**0.3759**^{BDVCR}	**0.5457**^{BDVCR}	**0.5133**^{BDVCR}	**0.5134**^{BDVCR}	**0.5496**^{BDVCR}	**0.5969**^{BDVCR}	**0.6627**^{BDVCR}
PARADE based models										
PARADE	0.5474	0.5009	0.4486	0.3747	0.5418	0.5054	0.5255	0.5499	0.5950	0.6599
PARADE5	0.5686	0.4824	0.4370	0.3714	0.5291	0.5174	0.5142	0.5356	0.5851	0.6538
KeyB(PARADE5)$_{TF-IDF}$	0.5721^{BDP}	0.5034^{BD5}	**0.4491**^{BD5}	0.3737^{BD}	0.5477^{BD5}	0.5198^{BD}	0.5221^{BD}	0.5488^{BD5}	0.5998^{BD5}	0.6645^{BD5}
KeyB(PARADE5)$_{BM25}$	**0.5769**^{BDP}	0.5063^{BD5}	0.4486^{BD5}	0.3748^{BD5}	**0.5494**^{BDP5}	0.5151^{BD}	0.5261^{BD5}	**0.5530**^{BD5}	**0.6021**^{BD5}	**0.6664**^{BDP5}
KeyB(PARADE5)$_{BinB}$	0.5580^{BD}	**0.5079**^{BD5}	0.4487^{BD5}	0.3740^{BD}	0.5427^{BD5}	0.5054^{BD}	0.5200^{BD}	0.5493^{BD5}	0.5978^{BD5}	0.6623^{BD5}
KeyB(PARADE5)$_{BinB2}$	0.5709^{BDP}	0.5066^{BD5}	0.4488^{BD5}	**0.3766**^{BD5}	0.5461^{BD5}	**0.5213**^{BD}	**0.5266**^{BD5}	0.5513^{BD5}	0.6005^{BD5}	0.6650^{BD5}

Tab. 2.7.: Results on *MQ2008* dataset. DeepRank* represents the results from the original paper. Best results are in **bold**. For KeyB(vBERT) models, a significant difference with BM25 is marked with a 'B', with DeepRank with a 'D', with Vanilla BERT with a 'V', with CEDR-KNRM with a 'C', and with Random select with an 'R'. For KeyB(PARADE) models, a significant difference with BM25 is marked with a 'B', with DeepRank with a 'D', with PARADE with a 'P', and with PARADE5 with a '5'. A paired t-test ($p - value \leq 0.05$) is used for measuring significance.

Model	P@1	P@5	P@10	P@20	MAP	NDCG@1	NDCG@5	NDCG@10	NDCG@20	NDCG
Baseline models										
BM25	0.3816	0.3316	0.2411	0.1515	0.4538	0.3297	0.4376	0.4841	0.5086	0.5243
DeepRank	0.3992	0.2816	0.1920	0.1150	0.4356	0.3641	0.4373	0.4672	0.4878	0.4917
DeepRank*	0.482	0.359	0.252	-	0.498	0.406	0.496	-	-	-
BERT based models										
Vanilla BERT	0.5063	0.3650	0.2560	0.1566	0.5230	0.4508	0.5165	0.5489	0.5697	0.5810
CEDR-KNRM	0.5050	0.3678	0.2561	0.1569	0.5220	0.4515	0.5151	0.5488	0.5674	0.5794
Random Select	0.5000	0.3663	0.2574	0.1579	0.5196	0.4387	0.5096	0.5427	0.5611	0.5729
KeyB(vBERT)$_{TF-IDF}$	0.5166^{BD}	**0.3862**^{BDVCR}	0.2597^{D}	0.1580^{D}	0.5318^{BD}	0.4649^{BD}	0.5330^{BDVCR}	0.5596^{BDR}	0.5755^{BDR}	0.5869^{BDR}
KeyB(vBERT)$_{BM25}$	0.5165^{BD}	0.3760^{BDVR}	0.2579^{D}	0.1582^{D}	0.5350^{BDR}	0.4629^{BD}	0.5317^{BDVCR}	0.5609^{BDVCR}	**0.5788**^{BDCR}	**0.5891**^{BDR}
KeyB(vBERT)$_{BinB}$	**0.5254**^{BD}	0.3819^{BDVCR}	**0.2624**^{DVCR}	**0.1589**^{D}	**0.5425**^{BDVCR}	**0.4661**^{BDR}	**0.5382**^{BDVCR}	**0.5616**^{BDVCR}	**0.5788**^{BDCR}	**0.5891**^{BDR}
PARADE based models										
PARADE	0.5089	0.3811	0.2617	0.1590	0.5375	0.4502	0.5321	0.5656	0.5799	0.5867
PARADE5	0.4999	0.3763	0.2578	0.1574	0.5254	0.4514	0.5226	0.5523	0.5716	0.5821
KeyB(PARADE5)$_{TF-IDF}$	**0.5369**^{BDP5}	0.3829^{BD}	0.2621^{D5}	0.1585^{D}	**0.5436**^{BD5}	**0.4859**^{BDP5}	**0.5390**^{BD5}	**0.5728**^{BD5}	**0.5851**^{BD5}	0.5942^{BD5}
KeyB(PARADE5)$_{BM25}$	0.5267^{BD5}	0.3831^{BD}	0.2635^{D5}	**0.1592**^{D5}	0.5409^{BD5}	0.4744^{BDP}	0.5373^{BD5}	0.5646^{BD5}	0.5812^{BD}	0.5907^{BD5}
KeyB(PARADE5)$_{BinB}$	0.5229^{BD}	**0.3888**^{BDP5}	0.2634^{D5}	**0.1592**^{D}	0.5428^{BD5}	0.4668^{BD}	0.5354^{BD5}	0.5702^{BD5}	0.5835^{BD5}	0.5914^{BD}
KeyB(PARADE5)$_{BinB2}$	0.5242^{BD}	0.3847^{BD5}	**0.2639**^{D5}	0.1583^{D}	0.5431^{BD5}	0.4789^{BDP5}	0.5373^{BD5}	0.5689^{BD5}	0.5837^{BD5}	**0.5952**^{BD5}

As one can see, KeyB(vBERT)$_{BM25}$ and KeyB(vBERT)$_{BinB}$ models outperform standard BERT models (Vanilla BERT and CEDR-KNRM) on all collections and for all metrics. KeyB(vBERT)$_{TF-IDF}$ outperforms standard BERT models (Vanilla BERT and CEDR-KNRM) for all metrics on all collections except Robust04, for which it yields lower results than Vanilla BERT on P@5, NDCG@5, and lower result than CEDR-KNRM on P@1, P@5 and NDCG@1. In addition, the best KeyB(vBERT) model (on each metric respectively) significantly improves the Vanilla BERT model on 6 metrics out of 10 on Robust04, on 9 metrics out of 10 on GOV2, on all metrics on MQ2007 and on 5 metrics out of 10 on MQ2008; it is furthermore significantly better than CEDR-KNRM on 2 metrics on Robust04, on 9 metrics out of 10 on GOV2, on all metrics on MQ2007, and on 6 Metrics out of 10 on MQ2008.

RQ3 Is it important to accurately select blocks?

We are interested here in assessing whether it is important to accurately select blocks or not. For this, we compare the results obtained by the different KeyB(vBERT) models with the ones obtained by the Random Select strategy which amounts to randomly selecting blocks. As one can also note, all KeyB(vBERT) models outperform the Random Select strategy on all collections, for all metrics. Furthermore, the best KeyB(vBERT) model is significantly better than Random Select on 10 metrics out of 10 on Robust04, on 8 metrics out of 10 on GOV2, on 10 metrics out of 10 on MQ2007, and on 8 Metrics out of 10 on MQ2008.

The above analysis shows that the KeyB(vBERT) models should be preferred over all baseline and standard BERT-based IR models. We now turn to the comparison of KeyB(vBERT) models.

RQ4 What are the differences between the different KeyB(vBERT) models?

On Robust04, KeyB(vBERT)$_{BinB}$ is the best model on 10 metrics and KeyB(vBERT)$_{BM25}$ on 1 metric. We further conduct significant tests between KeyB(vBERT) models. KeyB(vBERT)$_{BinB}$ is significantly better than KeyB(vBERT)$_{BM25}$ on MAP and NDCG while shows no significant difference than KeyB(vBERT)$_{TF-IDF}$. On GOV2, KeyB(vBERT)$_{BinB}$ is the best model on 10 metrics and KeyB(vBERT)$_{TF-IDF}$ on 1 metric. KeyB(vBERT)$_{BinB}$ is significantly better than KeyB(vBERT)$_{BM25}$ on p@1 and p@10 while shows no significant difference than KeyB(vBERT)$_{TF-IDF}$. On MQ2007, KeyB(vBERT)$_{BinB}$ is the best model over all metrics and significantly outperforms KeyB(vBERT)$_{TF-IDF}$ on MAP, P@20, NDCG, NDCG@1, NDCG@10 and NDCG@20, significantly outperforms KeyB(vBERT)$_{BM25}$ on MAP, P@10, P@20, NDCG, NDCG@10 and NDCG@20. On MQ2008, KeyB(vBERT)$_{BinB}$ is the best model on all metrics but P@5, KeyB(vBERT)$_{TF-IDF}$ on 1 metric and KeyB(vBERT)$_{BM25}$ on 2 metrics. The difference between all three models is however not really significant as KeyB(vBERT)$_{BinB}$ significantly outperforms KeyB(vBERT)$_{TF-IDF}$ on only MAP and KeyB(vBERT)$_{BM25}$ on P@10.

From this analysis, one can see that the model KeyB(vBERT)$_{BinB}$ is either significantly better or on a par with KeyB(vBERT)$_{TF-IDF}$ and KeyB(vBERT)$_{BM25}$. This justifies the use of a learning mechanism to select blocks. This said, even a simple approach to select blocks as the one implemented in KeyB(vBERT)$_{BM25}$ can yield good results on collections such as Robust04 and MQ2008. We now turn to the PARADE models.

Improving PARADE with selected passages As mentioned before, PARADE is the original PARADE model with 16 passages corresponding to the first and last passages, and 14 randomly selected passages in between, PARADE5 is another variant with only 5 passages corresponding to the first and last passages, and 3 randomly selected passages in between, and KeyB(PARADE5)

models are the PARADE models with only 5 passages selected with BM25, TF-IDF or learning based approaches. We propose to analyze the experimental results by answering several research questions.

RQ5 How effective are KeyB(PARADE5) models compared to baseline models (BM25, DeepRank)?

From Tables 2.4 to 2.7, one can see that KeyB(PARADE5) models outperform both baselines on all collections, the difference being significant for all metrics and all on collections but P@1 on GOV2.

RQ6 How effective are KeyB(PARADE5) models compared to PARADE and PARADE5?

As one can note, on all collections, KeyB(PARADE5)$_{TF-IDF}$, KeyB(PARADE5)$_{BM25}$ and KeyB(PARADE5)$_{BinB2}$ obtain better average results or on a par with PARADE. For example, comparing with the original PARADE model, on Robust04, KeyB(PARADE5)$_{BM25}$ outperforms PARADE on 9 metrics out of 10, even though the difference is never significant. On GOV2, KeyB(PARADE5)$_{BinB2}$ outperforms PARADE on 8 metrics out of 10 and is significantly better on 2 metrics. On MQ2007, KeyB(PARADE5)$_{BM25}$ outperforms PARADE on 9 metrics out of 10 and is significantly better on 3 metrics. On MQ2008, KeyB(PARADE5)$_{TF-IDF}$ outperforms PARADE on 9 metrics out of 10 and is significantly better on 2 metrics.

Comparing with PARADE5, KeyB(PARADE5)$_{TF-IDF}$, KeyB(PARADE5)$_{BM25}$ and KeyB(PARADE5)$_{BinB2}$ obtain better average results on all collections and metrics, except KeyB(PARADE5)$_{BM25}$ on NDCG@1 on MQ2007 (0.5151 vs 0.5174). More precisely, on Robust04, KeyB(PARADE5)$_{BM25}$ and KeyB(PARADE5)$_{BinB2}$ are significantly better than PARADE5 on 8 metrics. On GOV2, KeyB(PARADE5)$_{BM25}$ and KeyB(PARADE5)$_{BinB2}$ are significantly better than PARADE5 on 7 metrics.

On MQ2007, KeyB(PARADE5)$_{TF-IDF}$ and KeyB(PARADE5)$_{BM25}$ are significantly better than PARADE5 on 6 and 8 metrics respectively. On MQ2008, KeyB(PARADE5)$_{TF-IDF}$ and KeyB(PARADE5)$_{BinB2}$ are significantly better than PARADE5 on 8 metrics.

The model KeyB(PARADE5)$_{BinB}$, which reuses the BERT and feed-forward neural networks in PARADE for selecting passages, is however sometimes less effective than the other KeyB(PARADE5) models. On Robust04, it is below the other three KeyB(PARADE5) models as well as below PARADE. On GOV2, KeyB(PARADE5)$_{BinB}$ is higher than PARADE on 5 metrics and lower on 5 metrics. On MQ2007 and MQ2007, KeyB(PARADE5)$_{BinB}$ is mostly better than PARADE. Comparing with PARADE5, KeyB(PARADE5)$_{BinB}$ obtains almost always better results, especially on MQ2007 and MQ2008, being significantly better on 6 and 5 metrics respectively. This shows that, despite its mitigated results on some metrics and collections, KeyB(PARADE5)$_{BinB}$ is still a powerful approach that obtains several best results on different metrics.

RQ7 What are the differences between the different KeyB(PARADE5) models?

As one can note, the best results on each metric is somehow distributed on the different KeyB(PARADE5) models. On Robust04, KeyB(PARADE5)$_{BM25}$ obtains 5 best results, KeyB(PARADE5)$_{BinB2}$ 4 and KeyB(PARADE5)$_{TF-IDF}$ 1. On GOV2, KeyB(PARADE5)$_{BM25}$ obtains 5 best results, KeyB(PARADE5)$_{BinB2}$ 3 and KeyB(PARADE5)$_{BinB}$ 1. On MQ2007, KeyB(PARADE5)$_{BM25}$ obtains 5 best results, KeyB(PARADE5)$_{BinB2}$ 3 and the other two models 1 each. On MQ2008, KeyB(PARADE5)$_{TF-IDF}$ obtains 6 best results, KeyB(PARADE5)$_{BM25}$ 1 and KeyB(PARADE5)$_{BinB}$ and KeyB(PARADE5)$_{BinB}$ 2 each. Besides, as discussed above, although KeyB(PARADE5)$_{BinB}$ does not perform well on

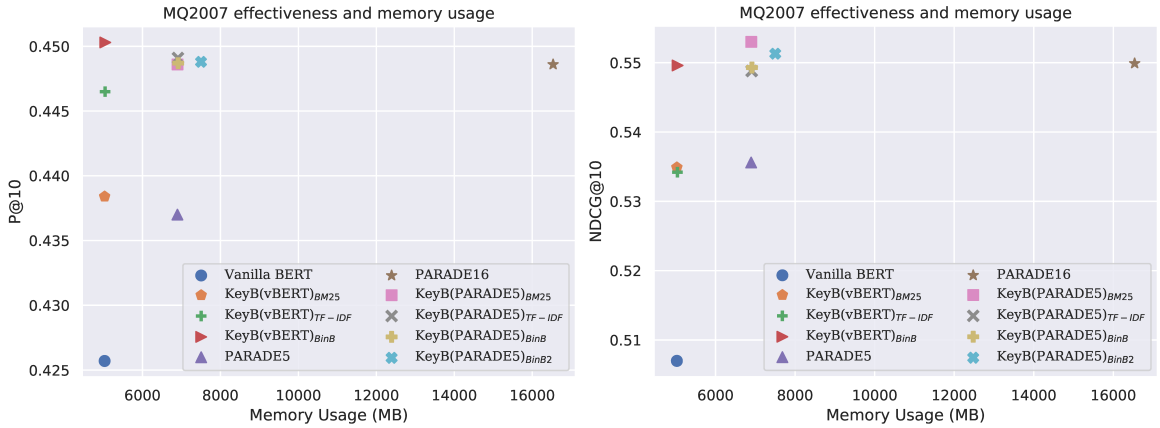
Robust04, it is still competitive with PARADE5 on this collection and performs well on the other collections. KeyB(PARADE5)$_{BinB2}$ tends however to be more stable across the collections and metrics.

Overall, the PARADE variants we have introduced in general significantly outperform the PARADE5 model and are either on a par or significantly outperform the original PARADE model. This is all the more remarkable that these models use three times less passages than the original PARADE model and require less memory while being faster, as illustrated below. Lastly, the best KeyB(PARADE5) model tends to be slightly better than the best KeyB(vBERT) model, on all collections and almost all metrics, even though the difference is in general small. Their latency is however not the same (see below).

2.6.3 Memory Usage

The memory usage of all models are similar across datasets. We thus only report here the memory usage of different models on MQ2007 as this dataset contains more queries and requires longer training than Robust04 and GOV2. The memory usage corresponds to the GPU consumption for training a given model. We remind the reader that a training batch contains two pairs consisting of four queries and four documents. The results obtained are shown in Figure 2.10 on two official LETOR metrics [138].

The best models in terms of accuracy (as measured by either P@10 or NDCG@10) and memory usage are located in the top left corner: they use less memory and achieve higher results. As one can note, KeyB(vBERT)$_{BinB}$ and KeyB(PARADE5) models are located in this area. They need less GPU memory while achieving similar or higher results than PARADE on the two



(a) P@10 performance

(b) NDCG@10 performance

Fig. 2.10.: GPU memory usage and effectiveness comparisons, automatic mixed precision is used for all models which would reduce memory usage. Top left models show better performance.

metrics. Furthermore, KeyB(PARADE5)$_{TF-IDF}$, KeyB(PARADE5)$_{BM25}$ and KeyB(PARADE5)$_{BinB}$ uses the same amount of memory as PARADE5 but are better on both metrics. KeyB(PARADE5)$_{BinB2}$ uses slightly more memory than PARADE5 but is also better on both metrics.

2.6.4 Ranking Speed

We measure here the speed of ranking of the different models on two sets of queries: all queries from one test fold of Robust04, each with 200 documents, and a randomly selected subset of 100 queries from MQ2007, again from one test fold, each with 40 documents. Note that the documents in MQ2007 are on average longer than the ones in Robust04. Latency results, as well as the average time for processing a query (in seconds) and a document (in milliseconds) on a RTX 6000 GPU are reported in Table 2.8 for Robust04 and Table 2.9 for MQ2007. The passage splitting time is not counted as this step can be performed offline.

As one can see on both tables, the three fastest models are Vanilla BERT, KeyB(vBERT)$_{TF-IDF}$ and KeyB(vBERT)$_{BM25}$, the latter two being only slightly slower than the former one. Furthermore, on both collections, the KeyB(PARADE5)$_{TF-IDF}$ and KeyB(PARADE5)$_{BM25}$ are faster than PARADE. They are also faster than PARADE5 on Robust04 and only slightly slower than PARADE5 on MQ2007. These two variants, TF-IDF and BM25, because of their performance, their memory usage and their speed, represent strong alternatives to the original Vanilla BERT and PARADE models.

Regarding the models based on learning the selection block method, if their performance is higher than the one of other models, their latency is also higher: KeyB(vBERT)$_{BinB}$, KeyB(PARADE5)$_{BinB}$ and KeyB(PARADE5)$_{BinB2}$ are, at best, 10 times slower than the KeyB(vBERT)$_{TF-IDF}$ and KeyB(vBERT)$_{BM25}$ models on both collections. Their current latency may prevent their use in a commercial system. This said, there are several paths that one can follow to make them faster, including a two-stage approach at the block level, using a fast model as BM25 for filtering out less relevant blocks and using the more complex models on the remaining blocks.

Tab. 2.8.: Reranking latencies (seconds) on *Robust04* test set for one folder (50 queries each with 200 documents).

Model	Latency	Seconds/query	Milliseconds/doc
Vanilla BERT	16.784	0.336	1.678
KeyB(vBERT)$_{TF-IDF}$	18.475	0.370	1.848
KeyB(vBERT)$_{BM25}$	19.679	0.394	1.970
KeyB(vBERT)$_{BinB}$	178.339	3.567	17.834
PARADE	75.601	1.512	7.560
PARADE5	55.661	1.113	5.566
KeyB(PARADE5)$_{TF-IDF}$	47.210	0.944	4.721
KeyB(PARADE5)$_{BM25}$	47.781	0.956	4.778
KeyB(PARADE5)$_{BinB}$	135.499	2.710	13.550
KeyB(PARADE5)$_{BinB2}$	175.562	3.511	17.556

Tab. 2.9.: Ranking latencies (seconds) on *MQ2007* test set for 100 queries each with 40 documents.

Model	Latency	Seconds/query	Milliseconds/doc
Vanilla BERT	6.962	0.070	1.741
KeyB(vBERT)$_{TF-IDF}$	11.598	0.116	2.900
KeyB(vBERT)$_{BM25}$	13.742	0.137	3.436
KeyB(vBERT)$_{BinB}$	211.648	2.12	52.912
PARADE	27.909	0.279	6.977
PARADE5	18.701	0.187	4.675
KeyB(PARADE5)$_{TF-IDF}$	26.894	0.269	6.724
KeyB(PARADE5)$_{BM25}$	24.320	0.243	6.080
KeyB(PARADE5)$_{BinB}$	242.309	2.423	60.577
KeyB(PARADE5)$_{BinB2}$	336.789	3.368	84.197

2.6.5 Analysis of the Position of Selected Blocks

We are finally interested here in analyzing the positions at which blocks are selected. To do so, we retained all documents containing at least 15 blocks and looked at which position the top eight scoring blocks occur for two models with different selection strategies, namely KeyB(vBERT)$_{BM25}$ and KeyB(vBERT)$_{BinB}$. The position is computed as in Section 2.3. The results obtained are displayed in Figure 2.11 for KeyB(vBERT)$_{BM25}$ and in Figure 2.12 for KeyB(vBERT)$_{BinB}$. In both Figures, a heat map is used to represent the probability the blocks selected occupy a particular position.

As one can observe, both models are more likely to select blocks at the beginning of a document, this tendency being more marked in KeyB(vBERT)$_{BM25}$. Interestingly, KeyB(vBERT)$_{BinB}$ is more likely to select blocks in the second position (and even in the third position) than in the first position. This said, both models also have a non null probability to select blocks at later positions. For example, in MQ2008, the possibility of selecting blocks in the last five positions, *i.e.*, in the second half of a document, amounts to 41.8% for KeyB(vBERT)$_{BM25}$ and to 42.7% for KeyB(vBERT)$_{BinB}$. The situation is similar on MQ2007 and GOV2, as well as on Robust04 for KeyB(vBERT)$_{BinB}$.

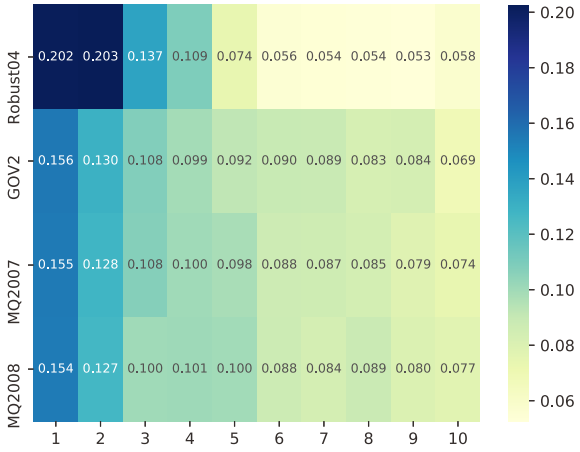


Fig. 2.11.: The probabilities of top 8 block appearing locations in KeyB(vBERT)$_{BM25}$.

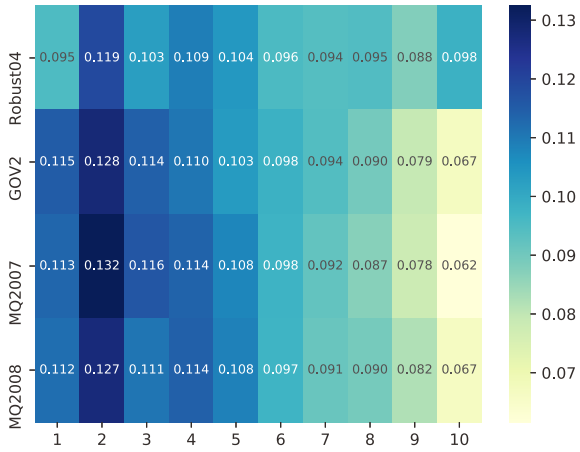


Fig. 2.12.: The probabilities of top 8 block appearing locations in KeyB(vBERT)$_{BinB}$.

These results, in line with the analysis conducted in Section 2.3, show the capacity of the proposed models to rely on blocks at different positions in the document; the good performance of these models further shows that the blocks selected tend to contain relevant information.

To conclude this series of experiments on standard IR collections and for illustration purposes, we display an example in Figure 2.13, about the top 8 blocks selected by the KeyB(vBERT)$_{BinB}$ model on MQ2007. As one can see, the different blocks are distributed across different positions at the beginning,

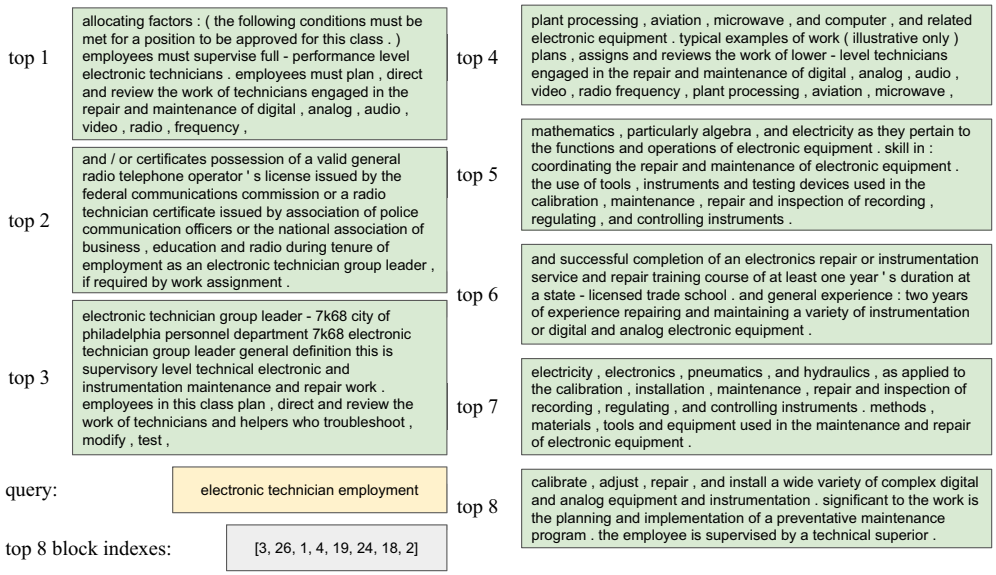


Fig. 2.13.: An example of top 8 blocks selected by the KeyB(vBERT)$_{BinB}$ model on MQ2007.

middle and end of the document. Furthermore, each block is related to the query.

2.7 Experiment on TREC 2019 DL and Comparison With Sparse Attention Based Models and IDCM

This section sheds additional light on the behaviour of the models proposed by comparing them to different baseline models, including sparse attention models and IDCM discussed in Section 2.2, on the relatively recent TREC DL dataset introduced to evaluate neural IR models. Table 2.10 summarizes the main characteristics of the the TREC 2019 Deep Learning Track collection. Following [66], we aim here to rerank the official top 100 retrieved documents and use the official evaluation metric NDCG@10 on the test set, which

we complement with MAP. Since this collection is larger than previously used collections, we train the models, again using the pairwise hinge loss [49], for longer steps, *i.e.* for 10 epochs each epoch being composed of 15000 batches of 2 pairs (four documents). As the qrels for the training and validation sets only contain one annotated document for each query (and it is relevant), each pair is composed of a query, its relevant document and another randomly sampled document which is viewed as irrelevant. For each metric and each model, we select the hyper-parameters leading to the lowest loss on the validation set and report its performance on the test set. The other experimental settings are the same as those in Section 2.6.1.

2.7.1 Comparison with Sparse Attention Based Models

Query-Directed Sparse Transformer (QDS-Transformer) [66] makes use of sparse local attention and global attention for long document information retrieval. In [66], the experiments conducted on the TREC 2019 Deep Learning Track collection [22] showed that QDS-Transformer improves the standard retrofitting BERT ranking baselines and outperforms more recent transformer architectures as Sparse Transformer [21], Longformer [3], and Transformer-XH [194]. We compare here our proposed approach with this QDS-Transformer and related baselines on this same collection.

Tab. 2.10.: Statistics of the TREC 2019 DL document ranking task.

Collection	# Documents	# Train queries	# Train qrels	# Dev queries	# Dev qrels	# Test queries	# Test qrels
TREC19 DL	3,213,835	367,013	384,597	5,193	5,478	43	16,258

Table 2.11 shows the results obtained. Note that for PARADE, the number of passages is set to 16 and the max query length to 30 (other settings are the same as in Section 2.6.1). For the other models, we report the results given in [66]. As one can see, the best results are obtained with KeyB(vBERT)$_{BinB}$

Tab. 2.11.: Experiment on TREC 2019 DL and comparison with sparse attention models and IDCM. Best results are in **bold**.

TREC Deep Learning Track Document Ranking		
Model	NDCG@10	MAP
Baseline models		
BM25	0.488	0.234
CO-PACRR [62]	0.550	0.231
TK [58]	0.594	0.252
TKL [56]	0.644	0.277
RoBERTa (FirstP) [96, 26]	0.588	0.233
RoBERTa (MaxP) [96, 26]	0.630	0.246
PARADE [83]	0.655	0.280
Sparse attention models		
Sparse-Transformer [21]	0.634	0.257
Longformer-QA [3]	0.627	0.255
Transformer-XH [194]	0.646	0.256
QDS-Transformer [66]	0.667	0.278
Select blocks models		
IDCM [53]	0.679	0.273
KeyB(vBERT)$_{BM25}$	0.678	0.277
KeyB(vBERT)$_{BinB}$	**0.707**	**0.281**
KeyB(PARADE5)$_{BM25}$	0.672	0.280
KeyB(PARADE5)$_{BinB}$	0.676	0.277
KeyB(PARADE5)$_{BinB2}$	0.678	0.279

which outperforms all baselines and sparse attention models, including QDS-Transformer, reaching 0.707 on NDCG@10 and 0.281 on MAP. It is closely followed by KeyB(vBERT)$_{BM25}$ which outperforms all baseline and sparse attention models on NDCG@10, reaching 0.678 compared with QDS-Transformer's 0.667. For MAP, KeyB(vBERT)$_{BM25}$ is slightly below QDS-Transformer and PARADE, and on a par with TKL.

The KeyB(PARADE5) models are very close to the KeyB(vBERT) models, both in terms of NDCG@10 and MAP. They also outperform baseline and sparse attention models on NDCG@10, and outperform all baseline and sparse attention models but PARADE and QDS-Transformers on MAP (they are on a par with these to models on MAP). One can note however that on this collection the KeyB(PARADE5) models not as effective as on the previous collections NDCG@10. Li et al. [83] also observed this for the PARADE model and attributed it to the fact that the effectiveness of PARADE across

collections is related to the number of relevant passages per document in these collections: TREC DL only has 1–2 relevant passages per document by construction; with such a low number of relevant passages, the benefit of utilizing complex passage aggregation methods such as PARADE is diminished. We see here however the advantage of the KeyB(PARADE5) models which rely on fewer d-passages, more likely to be relevant to the query.

2.7.2 Comparison with IDCM

As mentioned in Section 2.2, IDCM [53] is a recently proposed model that also learns how to select blocks. The motivation behind this model is to obtain an IR model more efficient as it would only rely on a few blocks. Our motivation slightly differs as we aim to improve the overall IR system by filtering out non relevant, likely noisy blocks. Furthermore, our approach can be directly used with different IR models by selecting blocks with standard IR systems. This is the basis of the models KeyB(vBERT)$_{BM25}$ and KeyB(PARADE5)$_{BM25}$ for example.

Table 2.11 shows a comparison of our approaches with IDCM (last four lines) where for each query the official top 100 documents are used (this setting is used for all models reported in Table 2.11). For IDCM, we have used the authors' notebook[21]. IDCM reaches 0.679 for NDCG@10, which is higher than all baseline and sparse attention models, and 0.273 for MAP, which is higher than all baseline and sparse attention models but TKL, PARADE and QDS-Transformer. In contrast, KeyB(vBERT)$_{BinB}$ outperforms all models, including IDCM, on both evaluation metrics. In addition, both KeyB(vBERT)$_{BM25}$ and KeyB(PARADE5) variants, even though they did not benefit from an additional pre-training on MS-MARCO and rely on a much simpler procedure

[21] https://github.com/sebastian-hofstaetter/intra-document-cascade

to select blocks, obtain results comparable with IDCM: KeyB(vBERT)$_{BM25}$ is 0.001 point below IDCM on NDCG@10 and 0.004 above on MAP whereas KeyB(PARADE5)$_{BM25}$ is 0.007 point below IDCM on NDCG@10 and 0.007 above on MAP, and KeyB(PARADE5)$_{BinB}$, KeyB(PARADE5)$_{BinB2}$ have higher MAP results.

Overall, the results obtained on TREC DL 2019 once again prove the effectiveness of the proposed selecting key blocks approaches, which obtain better results than baseline and sparse attention models without the need to customize CUDA kernels. Besides, when selecting key blocks with the trained BERT model, which corresponds to the model KeyB(vBERT)$_{BinB}$, one obtains the near state-of-the-art level performance [22] of 0.707 on this collection for NDCG@10. Lastly, the faster and less memory demanding variant KeyB(vBERT)$_{BM25}$ is a close competitor to KeyB(vBERT)$_{BinB}$ and should be preferred if time or memory constraints are important.

2.8 Conclusion

Benefiting from pre-trained BERT models, the field of information retrieval has seen remarkable progress in neural IR models, as exemplified by the success of Vanilla BERT which has become a strong, yet simple, baseline for neural IR models. To overcome the limitations of BERT-based models regarding long documents, we have proposed to divide documents into blocks and to select only the most important key blocks. This is reminiscent of the way humans assess the relevance of a document for a given query: one first identifies blocks relevant to the query, blocks which are then aggregated to obtain the overall assessment of the document. In order to select blocks, we have investigated two approaches: the first one is straightforward and

makes use of standard retrieval functions as TF-IDF or BM25; the second one learns a single BERT model used for both ranking blocks and documents. Both approaches have been shown to improve over standard baselines and previous BERT-based models. We have followed the same approach on another highly competitive neural IR model, namely PARADE, here again with improved results. All in all, selecting blocks is advantageous for the two models studied here, Vanilla BERT and PARADE. We conjecture that this selection is a way to remove passages in documents which are not relevant to the query and which are likely to bring noise when matching queries and documents.

Comparing our different proposals, if the selection strategy based on learned mechanisms performs in average better than the one based on standard IR models with similar GPU memory usage, its ranking latency is not as appealing. We thus recommend in practice to use the TF-IDF and BM25 versions of Vanilla BERT and PARADE if latency constraints are important (among these two variants, one may prefer the BM25 one which is slightly better overall). The choice between Vanilla BERT and PARADE variants, the latter being slightly better than the former on standard IR collections, and slightly worse on the TREC 2019 DL collection, depends on the collection considered.

In the future, we plan on deploying the proposed block selection approach on more complex models, which could be a way to further improve the results obtained in this study. We also plan to investigate alternative negative sampling strategies as well as ways to accelerate the selection process based on learned models.

3

Late-interaction Based Model for Long Document Retrieval

3.1 Introduction

Information retrieval (IR) plays an important role in our daily life in the era of big data. Retrieving relevant documents given a query is the central part of many applications in our daily life, e.g., web search. Deep neural networks have shown great success on a variety of tasks including information retrieval [60, 49, 182, 27, 118] with pre-trained Transformer [166] architectures like BERT [30] leading to state-of-the-art performances [118, 102, 26, 83, 72, 89]. BERT-based neural IR approaches can be classified into three categories [72]: interaction-based methods, representation-based methods and late-interaction methods. This first method, like a vanilla BERT model [118], where the query tokens and document tokens are concatenated as BERT inputs and applied full self-attention, is viewed to be extremely effective [51] but suffers from high computational complexity. On the other hand, representation-based methods generate two representations [144] for a query and a document respectively. When the document representations can be pre-stored, this method enables efficient fast retrieval at the expense effectiveness. To take advantage of both approaches, late-interaction meth-

ods have been proposed, ColBERT [72] being certainly the most well-known representative of this category. In ColBERT, token level passage embeddings are pre-stored, which are then late interacted with query embeddings to produce a relevance score. This method is slightly less efficient than representation-based methods, but definitely more effective.

ColBERT was primarily used for passage ranking and, as most BERT methods, suffers from the major drawback that it cannot directly handle long documents. Although researchers [83, 89, 90, 54] has proposed some methods for long document information retrieval, they are designed for interaction-based methods that are computational expensive. So far, there have been no attempts to adapt late interaction methods to long documents.

We address this problem here through a BERT-based dense intra-ranking and contextualized late interaction (ICLI) with multi-task learning. Efficiency is guaranteed by the pre-calculation of self attention, and effectiveness by the fact that pre-stored token embeddings are interacted in a fine-grained way. To the best of our knowledge, this is the first attempt to adapt a late interaction method for long document retrieval. Experimental results show that the proposed approach obtains near SOTA level effectiveness while being efficient on such collections as TREC 2019.

3.2 Related Work

As already mentioned, neural IR can be classified into three types: interaction-based, representation-based [50] and late interaction-based methods. The first effective interaction-based neural IR approaches, as DRMM [49], KNRM [182] or Conv-KNRM [27], were proposed before the introduction of BERT. After transformer-based BERT models were proposed, the field of neural IR

has seen rapid improvements. Nogueira and Cho [118] proposed a simple usage of BERT for passage re-ranking where the query and passage tokens are concatenated and processed by the BERT and the [CLS] output of BERT is used to assess the relevance, and got results that outperformed other neural IR models largely. Hofstätter et al. [57] introduced the TK model which relies on transformers to learn contextualized embeddings and kernel matching for ranking. Early representation-based models as DSSM [60] were appealing because of their extremely low latency. These models have also benefited from BERT-based architectures, leading to so-called dense retrieval models [144, 183]. Despite their efficiency, representation-based models are however less effective than interaction-based models. A trade-off between the two approaches is realized with late interaction methods like ColBERT [72], which relies first on separate representations for queries and documents and which approximates the effectiveness of interaction-based methods in a late interaction step. It places the high latency passage processing by BERT to offline stage and the contextualized token embeddings are stored, which enable fine-grained late interaction. ColBERT is slightly less efficient than dense retrieval methods, but more effective. However, as other BERT-based models, this model cannot directly handle long documents, due to the complexity of the self-attention step.

To deal with long document retrieval, a variety of methods have been proposed, e.g., modified attention methods like Longformer and QDS-Transformer [3, 66], sliding window and local relevance aggregation like TKL [55], passage representation aggregation methods like PARADE [83], and the recent selecting key block/passage for evaluation methods like KeyBLD and IDCM [89, 54, 90]. Recently, the KeyBLD model family [89, 90], that first filters a long document by selecting key blocks on which to ground the document relevance, has shown SOTA level performance while also being memory efficient. In parallel, the IDCM model [54] was proposed, the core idea of

which is also to first select key passages on which to ground the document relevance. In this chapter, we introduce a late interaction method for long document retrieval based on the same idea but in a different way that enables both effectiveness and efficiency. The details are described in Section 3.3.

3.3 Method

As mentioned above, the selecting key block related methods [89, 54, 90] have been shown to achieve SOTA level effectiveness for long document information retrieval. Nevertheless, these models have been designed for interaction-based models which have high computational complexity due to their online self-attention computation and not good solutions for real world online search scenarios where low latency is crucial. We propose to extend them to late interaction retrieval methods for long documents by using a cascaded ranking approach based on dense intra-ranking and late interaction. However, designing this is non-trival for being both effective and efficient. In the following, we firstly introduce the overall architecture and then describe the details and the reason of some choices.

The overall architecture of the proposed method is depicted in Fig. 3.1 and described later.

3.3.1 Contextualized Document Embedding

The ColBERT model's efficiency comes from its offline pre-computed contextualized passage token embeddings. Here, we want to also rely on the pre-stored contextualized document embeddings to enable fast retrieval. Due to the quadratic complexity of self-attention mechanism, transformer based

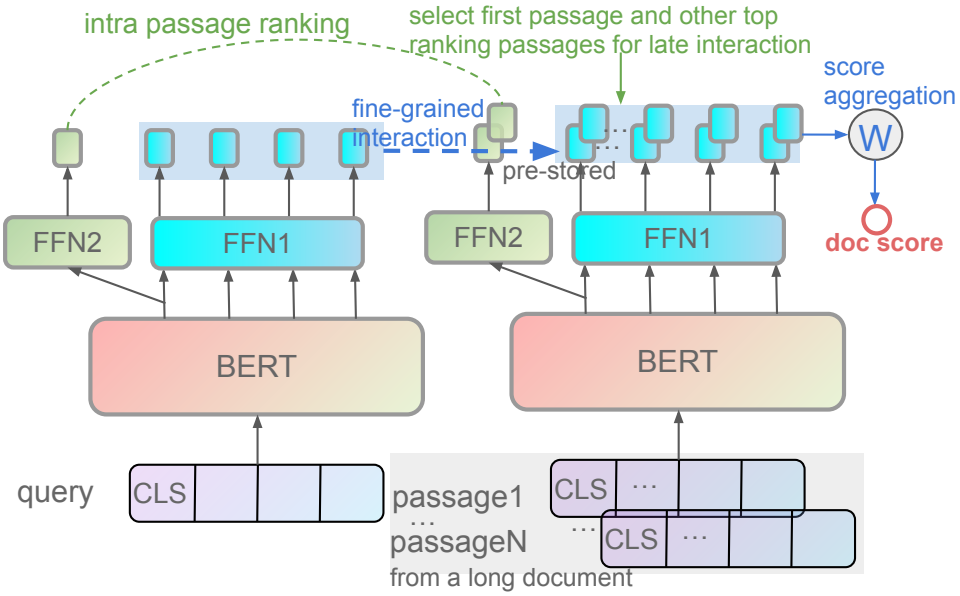


Fig. 3.1.: The architecture of proposed approach. Contextualized embeddings are calculated by the BERT model and a feedforward neural network for late interaction. The [CLS] embedding is also inputted to another feedforward neural network for passage ranking. The selected passages' late interaction scores are aggregated to obtain the document relevance score. The document tokens can be pre-stored.

models including BERT can only handle limited number of tokens, so they are not able to process long documents directly. To tackle this, following previous work [83, 89, 54], we firstly segment a long document into passages, which can be inputted to a BERT model separately to obtain contextualized embeddings. During training, the BERT model can be learned end-to-end and enables storing contextualized embeddings.

Given a document D, we segment it into passages $P_1 P_2 ... P_k$, which can be done in a sliding window way. For each passage, with a BERT tokenizer, we can obatin its tokens $p_1 p_2 ... p_m$, then BERT's [CLS] token is prepended to the tokens of each passage. The BERT model will compute the contextualized token embeddings $E_p^{cls} E_p^1 ... E_p^m$, where each E_p is of dimension 768 ($dim(E_p = 768)$) for a bert-base-uncased model (as shown in the right part of Fig. 3.1).

Each embedding E_p is then passed into a one-layer feedforward neural network FFN_1, referred to as *compressor1* (the blue module in Fig. 3.1), to obtain a low dimensional vector for late interaction: $V_p = FFN_1(E_p)$, where $dim(V_p) = 128$. It is worth mentioning that the [CLS] embedding is also passed to a one-layer feedforward neural network FFN_2, referred to as *compressor2* (the green module in Fig. 3.1), for dense intra-ranking of the passages in a document. This is to say, $V_p^{cls_1} = FFN_1(E_p^{cls})$, $V_p^{cls_2} = FFN_2(E_p^{cls})$ with, for each V_p^{cls}, $dim(V_p^{cls}) = 128$. The choice of using two compressors is based on the fact that a vector for intra-ranking should contain query/passage representation information while a vector participating into late interaction should be trained together with other tokens in a different way. Using two compressed vectors allows one to capture these differences and gives more flexibility.

Online computation is used during training. During deployment, the contextualized tokens of each document are pre-computed and stored for efficient late interaction.

3.3.2 Contextualized Query Embedding

Similar to a passage in a document, BERT's [CLS] token is prepended to the tokenized query tokens, which are passed to the BERT model to obtain contextualized query token embeddings $E_q^{cls}E_q^1...E_q^n$, with, for each E_q, $dim(E_q) = 768$. Then using FFN_1, one obtains a low dimensional vector V_q for each E_q. For the [CLS] token, $V_q^{cls_1}$ and $V_q^{cls_2}$ are also calculated.

Different from documents, the query is always computed online, which is also the same case for ColBERT and dense retrieval models. Nevertheless,

the computational cost is relatively small as the queries are shorter and only required to be computed once to retrieve different documents.

3.3.3 Intra-Ranking for Key Passage Filtering

The token embeddings of a long document can be pre-stored, however, they are normally too long which may result in high latency and may contain noisy information. Previous works [90, 54] first select key passages according to the query to make it more efficient and even more accurate. BM25 and learning based methods can be used where the later normally performs better as it enables semantic matching. In the case of late interaction, we also want to use this step and to use learning based method for informative passage selection. To rely on pre-stored embeddings, inspired by dense retrieval where a query and a passage are represented by a low dimensional vector respectively, normally using the [CLS] embedding, we want to rely on this which allows us to do semantic matching and is naturally part of the BERT model. To do so, the [CLS] embedding of a passage or a query from BERT is inputted to a compresser layer2 (the FFN_2 module in Fig. 3.1), to obtain the representation of a passage. Dot product is used during inference, to select the most informative chunks, with the pre-stored passage representations and the online query representation.

As the first passage of a document usually carries important information, as illustrated in [90], we always select this passage in our approach. This strategy is also consistent with truncation-based methods which truncate long documents and rely only on their beginning in order to apply BERT-based models. Given the representation $V_q^{cls_2}$ of a given query and pre-stored

representations $V_p^{cls_2}$ of passages in a given document, we use the standard dot product to score passages:

$$S_{q,p}^1 = V_q^{cls_2} \cdot (V_p^{cls_2})^T. \tag{3.1}$$

This selection process provides "top"-k passages for each query-document pair. Having the first passage in the "top"-k list furthermore allows one to train compressor2 through a standard ranking loss, as described below. During training, this step is eval model for PyTorch, which means no backpropagation.

The above approach finally amounts to learning a representation useful for selecting passages and is in line with previous work [90] that has shown that learning how to select key blocks in a document can outperform methods that select key blocks using standard ranking functions as BM25 [150].

3.3.4 Fine-Grained Late Interaction

After having selected the k passages, we adopt, for each passage, the late interaction approach of ColBERT to obtain the query-passage relevance score:

$$S_{q,p}^2 = \sum_{i \in [cls_1, 1...n]} \max_{j \in [cls_1, 1...m]} V_q^i \cdot (V_p^j)^T. \tag{3.2}$$

These scores are then simply aggregated through a weighted sum:

$$S_{q,d} = w_1 S_{q,p_1}^2 + ... + w_k S_{q,p_k}^2, \tag{3.3}$$

where the weights $\{w_1, ..., w_k\}$ are real numbers.

3.3.5 Multi-Task Learning

Since the [CLS] contextualized vectors are used for two tasks: intra passage ranking (Section 3.3.3) and late interaction (Section 3.3.4), we adopt a multi-task learning approach to ensure that both tasks are taken into account to fine-tune the BERT model. As mentioned before, (a) the first passage of a document usually contains relevant information and is always selected, (b) this first passage is furthermore always used by relatively strong baselines based on truncation, and (c) its use ensures the training of the first task. Indeed, we train the [CLS] vector to be used for intra-passage ranking using the first passage of documents as explained below. The second task directly relies on the scores of the selected passages (Eq. 3.2) and the document labels. The loss functions associated with these tow tasks are given in the following section.

3.3.6 Loss Functions

Following [54], we use the pairwise RankNet loss for each task, defined by:

$$\mathcal{L}(q, d^+, d^-; \Theta) = -\log(Sigmoid(S_{q,d^+} - S_{q,d^-})),$$

where q is a query, (d_q^+, d_q^-) is a positive and negative training document pair for q, Θ represents the parameters of the model and $S_{q,d}$ is the score provided by the model for document d. For task 1, the loss function is given by:

$$\mathcal{L}_1(q, d^+, d^-; \Theta) = RankNet(q, S^1_{q,p_{1+}}, S^1_{q,p_{1-}}),$$

where $S^1_{q,p_{1+}}$ (resp. $S^1_{q,p_{1-}}$) is the relevance score of the first passage of a positive (resp. negative) document for query q according to Eq. 3.1. For task 2, the loss is given by:

$$\mathcal{L}_2(q, d^+, d^-; \Theta) = RankNet(q, S_{q,d_+}, S_{q,d_-}),$$

where S_{q,d_+} (resp. S_{q,d_-}) is the relevance score of a positive (resp. negative) document for query q according to Eq. 3.3.

As the two losses may have different scales, we combine them to obtain the final loss $\mathcal{L}(q, d^+, d^-; \Theta)$ following [92], which adjusts the proposal of [71] to enforce positive regularization:

$$\mathcal{L} = \frac{1}{2\sigma_1^2}\mathcal{L}_1 + \frac{1}{2\sigma_2^2}\mathcal{L}_2 + \log(1+\sigma_1^2) + \log(1+\sigma_2^2),$$

where σ_1 and σ_2 are parameters of the model.

3.4 Experiments

In this section, we evaluate the proposed approach for both effectiveness and efficiency. Besides, an ablation study is performed to analyze the model.

3.4.1 Datasets

We make use here of the widely used test collection TREC 2019 Deep Learning Track Document Collection. We use two test sets: TREC 2019 and 2020 test queries, where the task is to rerank top 100 documents for each test query. The first test set is widely studied [66, 90, 54] on MS MARCO v1 corpus. We

here also evaluate it on MS MARCO v2 corpus[1], which is said to be larger, cleaner and more realistic. To be specific, for TREC 2019 and 2020 test set, we use the same official data with MS MARCO v1 corpus, where the training set is 'msmarco-doctrain-top100', validation set is ' msmarco-docdev-top100'. For TREC 2019 test set, we also additionally evaluate it on the MS MARCO v2 corpus, where the training set is 'docv2_train_top100', validation set is 'docv2_dev_top100'.

3.4.2 Baseline Models

The proposed approach is compared with 5 baselines:

- **BM25**: we use the BM25 [150] implementation of Anserini [185], with default hyperparameters;
- **BERT-CAT**: this is a BERT interaction-based [118] baseline;
- **TK**: this model [57] generates contextualized embeddings using Transformer and does kernel matching.
- **TKL**: this model [55] extends TK model for long documents;
- **ColBERT**: this is the SOTA late interaction model [72] for passage ranking.

In addition, we also propose to use BM25 to do intra-ranking to select passages. The other steps are unchanged but we do not need multi-task learning in this case as there is no learning of the selection module. Only late interaction loss, $\mathcal{L}_2$ is required. In this setting, we use the BM25 model

[1] `https://microsoft.github.io/msmarco/TREC-Deep-Learning-2021`

proposed in [89] which has provided very good results for block selection on several collections.

3.4.3 Experimental Settings

Our implementation is based on the matchmaker open-source framework[2], using automatic mixed precision [109]. For the MS MARCO v1 corpus, the goal is to rerank the official top 100 documents. For the MS MARCO v2 corpus, as TREC 2019 DL has no official top retrieved documents, we rely on Anserini [185] with default hyperparameters to obtain top 100 documents which are then used for reranking. All neural-IR models, except TKL, are trained for 40000 steps where each step contains 8 positive-negative pairs. For TKL, each step is set to 32 pairs and is trained for 10000 steps. All models are trainied using the pairwise RankNet loss. The negative documents in the positive-negative pairs are sampled randomly, from each query's official top 100 documents that have no label. We conduct 10 validations during training to store the best performing model.

For BERT-based models, the learning rate for BERT is set to 1.0e-05, the other learning rates are set to 1.0e-3. TK and TKL use 1.0e-4, the learning rate for kernels is 1.0e-3. Adam optimizer is used to train the models. For BERT-CAT, TK and ColBERT, we only consider the first 400 tokens; indeed, as these tokens are concatenated with the query, this ensures that one does not exceed the maximum allowed size of 512 tokens in BERT (note that previous studies, as [55], only considered 200 tokens for these kind models). The BERT model used for queries and documents is shared. The maximum document length is set to 3000 tokens, while the passage length is set to 200 tokens (without overlap as we did not see significant difference during preliminary

[2] `https://github.com/sebastian-hofstaetter/matchmaker`

experiments). This means that a document can split into a maximum of 30 passages (when the document length is shorter than 3000, the obtained passages can be less). For TKL, we set the passage size to 40, with an overlap of 10, as done in the original paper. Both TK and TKL use lowercased texts as GloVe [131] embeddings are lowercased version, while BERT based models use original texts (bert-base-uncased model is used and in this case every token will be made lowercase automatically). For the intra-ranking step, we select 4 passages (an extreme case is the original passage number is less than 4, if so we pad 0 as the scores).

To learn both the complete BERT model and the aggregation scores (Eq. 3.3), we first fix the aggregation scores to $[0.4, 0.3, 0.2, 0.1]$ and learn BERT, prior to fix BERT parameters and adjust the aggregation scores. The choice for the initial values for the aggregation scores is arbitrary and simply reflects the fact that passages with higher relevance scores are more important for deciding the relevance of the entire document.

Finally, results are reported according to NDCG@10 and MAP, two standard metrics on the collections considered.

3.4.4 Results

The results obtained on the TREC 2019 DL test collection with MS MARCO v1 and v2 are shown in Table 3.1. With MS MARCO v1, the proposed approach ICLI with dot product for passage filtering achieves SOTA results: for NDCG@10, it reaches 0.7048, which is 8.36% higer than ColBERT. With MS MARCO v2, it also obtains the best result on NDCG@10. Using BM25 to filter passages for late interaction, ICLI-BM25 is the second best method on MS MARCO v1, the best method on MAP and second best on NDCG@10 on

Tab. 3.1.: Results on *TREC 2019 DL* collection of MS MARCO v1 and v2 corpus. Best results are in **bold**.

Model	MS MARCO v1		MS MARCO v2	
	NDCG@10	MAP	NDCG@10	MAP
BM25	0.5176	0.2434	0.2368	0.0865
BERT-CAT	0.6519	0.2627	0.3754	0.1144
TK	0.5850	0.2491	0.3290	0.1086
TKL	0.6213	0.2656	0.3351	0.1108
ColBERT	0.6504	0.2688	0.3788	0.1133
ICLI-BM25	0.6806	0.2703	0.3926	**0.1160**
ICLI-dot	**0.7048**	**0.2768**	**0.4049**	0.1146

Tab. 3.2.: Results on *TREC 2020 DL* dataset, corpus MS MARCO v1. Best results are in **bold**.

Model	NDCG@10	MAP
BM25	0.5286	0.3793
BERT-CAT	**0.6211**	**0.4112**
TK	0.5732	0.3660
TKL	0.5677	0.3633
ColBERT	0.5951	0.3907
ICLI-BM25	0.5940	0.3783
ICLI-dot	0.6042	0.3938

MS MARCO v2. This proves that the proposed approach is effective compared to other late interaction methods.

The results obtained on the TREC 2020 DL test collection are displayed in Table 3.2. As once can see, BERT-CAT using only the first 400 tokens is the best performing method, while the proposed approach is only slightly better than ColBERT. A similar trend is observed on the TK and TKL models as TK, with only the first 400 tokens, obtains better results than TKL with 3000 tokens. These results are consistent with the findings reported in [52]: the authors also observed that for TKL, the 2K model outperforms the 4K model on TREC 2020.

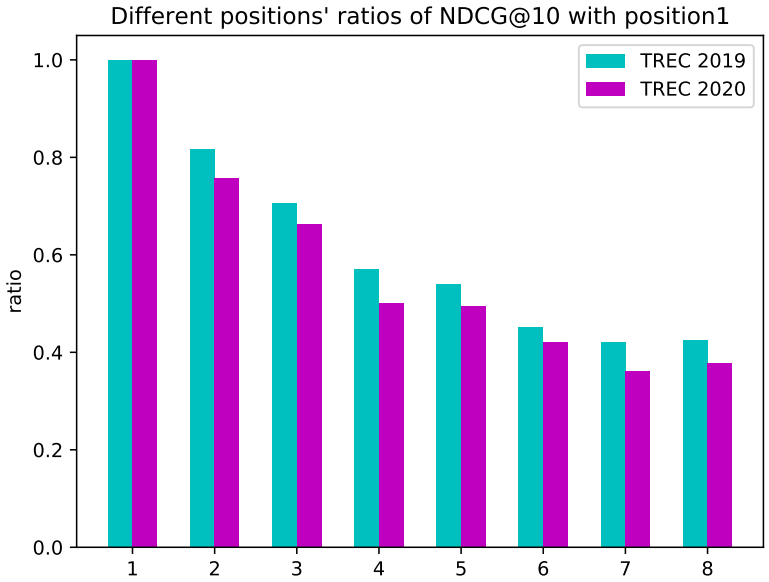


Fig. 3.2.: The NDCG@10 result of different positions compared with the first position.

To better understand what's happening on the TREC 2020 test collection, we display in Fig. 3.2, for both TREC 2019 and TREC 2020, the ratio of the NDCG@10 score of the i^{th} (from 1 to 8) passage of a document. To be specific, we use each passage's relevance score as the document relevance score to calculate the NDCG@10 with the label. Each passage's relevance score with the query is obtained using Sentence-BERT [144] pre-trained on MS MARCO passage collection[3]. We consider here the first 8 passages, each passage containing 400 tokens. As one can note, the relevance information decreases with the position, and, compared to passages in TREC 2019, passages in TREC 2020 tend to be less relevant when their position in the document increases. We believe that this, at least partly, explains the unexpected results observed here and in previous studies on TREC 2020. This said, the ICLI-dot model is still comparable with ColBERT and better than the TK family on this collection.

[3]The msmarco-distilbert-base-v4 version from `https://www.sbert.net/docs/pretrained-models/msmarco-v3.html`

Tab. 3.3.: Average reranking latencies (seconds) on *TREC 2019 DL* test set, corpus MS MARCO v1 for 100 documents with a query. Best result is in **bold**.

Model	Latency
BERT-CAT	1.0234
TKL	0.5002
ICLI-dot	**0.3420**

Tab. 3.4.: Ablation study on *TREC 2019 DL* dataset, corpus MS MARCO v1.

Model	NDCG@10	MAP
Complete	0.7048	0.2768
W/o compressor1	0.6798	0.2687
Compressor2 = compressor1	0.6975	0.2719
Score aggregation: equal weights	0.6634	0.2614
Score aggregation: learned weights	0.6646	0.2725

3.4.5 Reranking Latency

Following [72], latency is used to measure the average time (in seconds) for loading the pre-stored document vectors from disk to the GPU, for computing the query representation and for applying the fine-grained interaction module to rerank 100 documents for a query on the TREC 2019 MS MARCO v1 test collection (total 43 queries). Average latency results for reranking, on a RTX 8000 GPU, each query's top 100 documents for TREC 2019 test set are reported in Table 3.3. The average latency for BERT-CAT amounts to 1.0234, to 0.5002 for TKL, and to 0.3420 for ICLI-dot. This demonstrates that the proposed approach is more efficient than BERT-CAT, by a factor of 3, and TKL, by a factor of 1.46.

3.4.6 Ablation Study

Lastly, Table 3.4 reports the results of the ablation study we performed on the ICLI-dot model on MS MARCO v1. We provide in the first line the results obtained with the complete model. The second line corresponds to

removing the compressor for intra-ranking module, in which case the model uses the original 768 dimensional [CLS] embeddings (this embedding is also directly learned with task 1). The third line means without compressor2, in which case the model uses the same compressor for intra-ranking and late interaction. The fourth line corresponds to initializing the aggregation weights (Eq. 3.3) to $[0.25, 0.25, 0.25, 0.25]$. Lastly, the fifth line shows the results when directly learning the score aggregation weights. As one can note, comparing with the proposed design, the above modifications show lower results, suggesting the importance of using different compressors and validating our choice of learning weights for score aggregation by partly decoupling their learning from the one of the BERT model.

3.5 Conclusion

In this chapter, we have attempted to adapt late interaction methods for long document retrieval by first learning the [CLS] vectors for fast intra-passage ranking, and then by applying late interaction on contextualized token vectors to obtain the fine-grained relevance scores for each selected passage. Score aggregation and multi-task learning methods are furthermore used to combine the various ingredients of our approach. Experimental results demonstrate the efficiency and efficacy of the proposed approach on such collections as TREC 2019.

Part II

Domain Adaptation for Dense Retrieval and Conversational Search

4

Domain Adaptation for Dense Retrieval through Self-Supervision by Meticulous Pseudo-Relevance Labeling

4.1 Introduction

Information retrieval (IR) is playing a pivotal role in our daily life due to data explosion. Traditional IR approaches like BM25 [150] compute a similarity between a query and a document on the sole basis of the terms common to both. As such, they are unable to handle semantic matching between different surface forms. Neural information retrieval, with the advent of deep neural networks, has greatly improved IR systems through models which can capture the semantics of each term and compare them even if their surface form differs. A popular model in both Natural Language Processing (NLP) and IR is BERT [31], which is based on transformers [166] and is pre-trained

on large scale unlabeled collections through self-supervision; BERT can be used on a variety of downstream tasks through fine-tuning.

Neural IR models can be roughly classified into two categories [50]: interaction-based and representation-based (also called *dense retrieval*) approaches. Interaction-based models have been shown to perform better in average than dense retrieval models; on the other hand, dense retrieval (DR) models are faster than interaction-based models, since the document representations can be generated and stored in advance, and preferred if one needs to deploy a model at large scale. This said, recent studies like BEIR [157] showed that dense retrieval models trained on a source domain generalize less well than traditional models as BM25 and interaction-based models on out-of-distribution (OOD) data sets. Although training on target data sets with gold labels is a standard process, the annotation required may be both time consuming and expensive so that this approach can have limitation on many real world usages. It is thus important to address the issue at OOD scenarios for dense retrieval.

One of the goals of domain adaptation [173, 170] is to make a model that has been trained on one domain, called the source domain, to perform well on another domain, called the target domain, without using human labels on the latter. Recently, various domain adaption techniques for dense retrieval have been proposed. Domain generalization based on data generation is one type of approaches [170] which has been followed in Ma et al. [100] through a model called QGen which generates queries for the target domain using a query generator trained on the source domain. Along the same line, GPL [172] uses hard negatives and knowledge distillation and obtain state-of-the-art results on a number of BEIR data sets. However, the created queries are synthetic and may not resemble real target queries. Another popular and widely used approach is based on domain adversarial learning [170]. Very

recently, Xin et al. [181] proposed a model called MoDIR which adversially trains a dense retrieval encoder to learn domain-invariant representations for dense retrieval. However, such a learning objective may produce a poor embedding space and lead to unstable performance [172, 69].

In this chapter, we address **do**main generalization for **d**ense **re**trieval through **s**elf-**s**upervision by pseudo-relevance labeling (in short, DoDress). We first aim to build pseudo-relevance labels on the target domain using interaction-based models solely trained on the source domain as T53B [119] which act as re-ranker. The rationale for using interaction-based models in this setting lies in the fact that these models have been showed to behave relatively well on OOD data sets [157]. Note that the heavy T53B model is only used for producing pseudo-relevance labels before training the dense retrieval model so that the overall approach is still efficient during the online search stage. This method eliminates the requirement for human annotations and enables the model to use genuine queries and documents of the target domain. In addition, we investigate different negative sampling strategies: global random negative sampling, BM25 hard negatives, and SimANS [195] hard negatives based on current dense retrieval models being trained, to further improve the final dense retrieval model on the target domain.

Our contributions are twofold: first, we propose to use interaction-based model T5-3B trained on the source domain to produce pseudo-labels on the target domain; second, we further investigate different negative sampling strategies in order to improve the final dense retrieval model, to benefit from more informative training data. Experiments demonstrate the efficacy of our approach; they show in particular that it helps to improve the SOTA approach GPL when it is fine-tuned on the generated pseudo-labeled data.

4.2 Related Work

Wang et al. [170] present a survey paper about domain generalization on unseen domains. Domain generalization or adaptation can be categorized into three groups: data manipulation, representation learning and learning strategy. There are two kinds of techniques in the first group: data augmentation [134, 159, 153, 167] which is commonly used in image data (for example, altering the location, textual of objects and adding random noise), and data generation [142, 136, 193] which uses some models to generate new data to train a model. Representation learning group has domain-invariant representation learning (e.g., domain adversarial learning) [5, 40, 113] and feature disentanglement methods [86, 115, 94]. The third group has several categories, for example ensemble learning[105, 25], meta-learning [85, 34] and self-supervised learning based approaches (e.g., the task of solving jigsaw puzzles) [16, 65].

Similar strategies, such as domain generalization or transfer learning, are put forth by researchers for information retrieval. A strategy similar to the one adopted in this study is described in [111] which carries out a systematic evaluation of transfer ability of BERT-based neural ranking models. The authors additionally use BM25 to generate pseudo-relevance labels. They do not, however, focus on dense retrieval models which are known to require complex training methods and a large amount of data in a distinct situation [42]. Besides, only using BM25 to obtain pseudo-relevance labels might be a weak solution. For interaction-based models [81] or learning sentence embeddings [44, 171], some publications suggest self-supervised techniques. These methods are frequently used for pre-training, however they do not explicitly focus on domain generalization [172]. Ma et al. [100] proposes QGen, a generation approach to zero-shot learning for first-stage dense pas-

sage retrieval that makes use of synthetic question generation, allowing the construction of arbitrarily large yet noisy question-passage relevance pairs that are domain specific, in an effort to overcome the challenge that neural retrieval models need a large supervised training set to outperform conventional term-based approaches. The documents used to generate queries are viewed as positive and other in-batch instances are viewed as negative. Concurrently, Liang et al. [91] consider the two-tower dense passage retrieval architecture. Given that labeled data can be difficult to obtain and that neural retrieval models need a huge amount of data to be trained, they also suggest using synthetic queries produced by a large sequence-to-sequence (seq2seq) model for unsupervised domain adaptation. These two papers show the effectiveness of the query generation approach, which is also used in the GPL model [172]. GPL relies on a pre-trained T5 encoder-decoder [141] to generate queries from input passages. The input passages are seen as positive passages whereas similar passages retrieved using an existing dense retrieval model are constituted by the (hard) negative passages. The Margin-MSE loss [51] is used as knowledge distillation to teach the dense retrieval model to learn from an interaction-based model. Experimental results show a state-of-the-art effectiveness on several BEIR [157] collections.

Researchers have also explored alternative strategies for domain adaptation of dense retrieval models. Xin et al. [181] proposed a momentum adversarial domain invariant representation learning (MoDIR) approach, which introduces a momentum method to train a domain classifier that distinguishes source and target domains. The dense retrieval encoder is then trained in an adversarial manner to learn domain-invariant representations. A momentum queue that records embeddings from several prior batches is used in order to strike a balance between accuracy and efficiency [181]. This approach is used on a trained ANCE model [183]. The results vary from one data set to the other, with sometimes important improvements and sometimes marginal

gains or losses. Karouzos et al. [69] proposed UDALM for domain adaptation for sentiment classification through multi-task learning. It simultaneously learns the objective of the Masked Language Modeling (MLM) task on the target domain and the task from the source labeled data. However, this strategy was not designed for dense retrieval and, as mentioned in [172], it does not work well for dense retrieval.

In this chapter, we propose to do domain adaptation for dense retrieval through self-supervision by pseudo-relevance labeling. We apply the cutting-edge, domain-generalizable T53B interaction-based model [119] for pseudo-positive labeling . This model can produce more accurate pseudo-relevance labels, where top ranked documents are viewed as relevant to a given query. Additionally, different negative sampling strategies are investigated, especially with the SimANS [195] hard negatives sampled from the current DR models' retrieval list, to improve the model effectiveness, after training with the generated pseudo-labeled data.

4.3 Background

Dense retrieval [70, 183, 181] seeks to encode both queries and documents into a low-dimensional space with an encoder g, typically a BERT-like model. The retrieval status value (RSV) of a query and a document is then calculated with a simple similarity function in the low-dimensional space:

$$RSV(q,d)_{DR} = g(q) \cdot g(d) \quad (or\ RSV(q,d)_{DR} = cos(g(q), g(d)))\,,$$

where $g(q)$ (resp. $g(d)$) denotes the encoding of the query (resp. document). This enables a fast retrieval through a nearest neighbour search strategy [183].

BM25 BM25 is a widely used standard IR algorithm based on term matching. The RSV of a document with respect to a query is given by:

$$RSV(q,d)_{BM25} = \sum_{w \in q \cap d} IDF(w) \cdot \frac{tf_w}{k_1 \cdot (1 - b + b \cdot \frac{l_d}{l_{avg}}) + tf_w},$$

where $IDF(w)$ is the inverse document frequency, l_d is the length of document d, l_{avg} the average length of the documents in the data set, and k_1 and b two hyper-parameters

T53B By establishing a uniform framework that transforms all text-based language problems into a text-to-text format, T5 [141] explores the landscape of transfer learning for NLP and achieves state-of-the-art results on many benchmarks. Nogueira et al. [119] proposed to use T5 as an interaction-based model for information retrieval by relying on the following input representation:

Query: [q] Document: [d] Relevant: true or false

where $[q]$ and $[d]$ are replaced with the query and document texts. During training, the T5 model learns to generate the word "true" when the document is relevant to the query, and the word "false" when it is not. The relevance score for inference is then determined by the likelihood of producing "true" [119]:

$$RSV(q,d)_{T5} = softmax(Z_{true}) = \frac{e^{Z_{true}}}{e^{Z_{true}} + e^{Z_{false}}},$$

where Z_{true} and Z_{false} are the logits of output tokens.

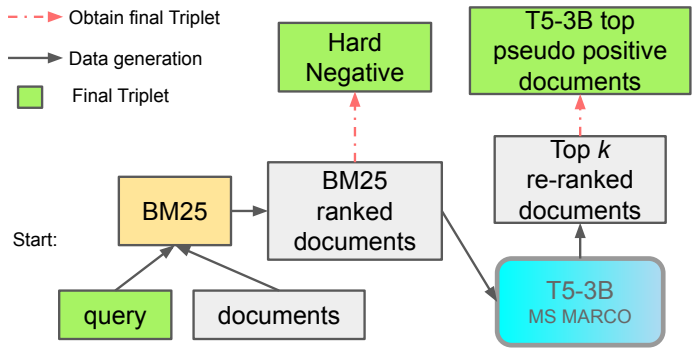


Fig. 4.1.: The overall pipeline of generating self-supervised data with BM25 hard negative sampling for pseudo-relevance labeling.

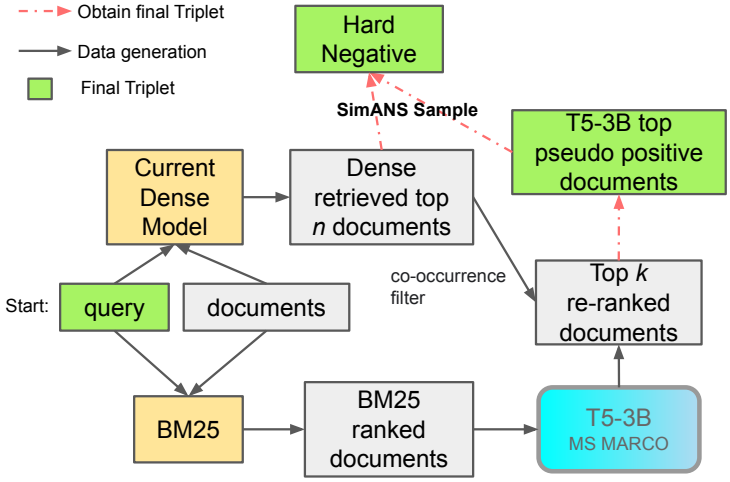


Fig. 4.2.: The overall pipeline of generating self-supervised data with meticulous pseudo-relevance labeling using SimANS hard negative sampling.

4.4 DoDress: Pseudo-Relevance Label Generation

One of our approach using BM25 hard negative sampling is described in Figure 4.1, and our best approach using SimANS hard negative sampling is described in Figure 4.2. We will describe them in detail below.

Generating Positive-Negative Training Pairs on the Target Domain

We simply propose here to consider the top k documents, obtained with the combination BM25&T53B in which T53B serves as a re-ranker, as relevant. k is an hyper-parameter which can be set according to different information, as, *e.g.*, the number of available queries and documents. Furthermore, for each pseudo-relevant query-document pair, we sample m documents and consider them as non-relevant (different negative mining strategies can be used, and they are described in Section 4.4.1 and Section 4.4.2). Thus, for each query, $k \times m$ query-document triplets (query, relevant document, non-relevant document) can be formed. The green blocks in Figure 4.1 and Figure 4.2 represent the generated triplet pseudo-training pairs, consisting the training data for domain adaptation.

4.4.1 Global and BM25 Hard Negative Sampling

Previously, we mentioned that the top ranking documents are viewed as positive instances for pseudo-relevance labeling, and now we discuss the negative instances. Apparently, one simple negative mining strategy is global random negative sampling: for all documents in the corpus, m documents randomly sampled except the positive instances are viewed as negative.

However, as pointed out by other researchers, in order for dense retrievers to reach their maximum capacity, fine-tuning pipelines often require heavily engineered strategies [43]. A key challenge in DR is to construct proper negative instances for learning its representations [70]. Previous global random negative instances might be too simple for the DR models, thus they might not be well trained on target domain. So, we further propose to use BM25 top ranking documents except the pseudo-relevant documents (top

T53B instances) as hard negative instances for training the DR models. The architecture is shown in Figure 4.1: the hard negative documents can be randomly sampled in the BM25 top ranking list, and positive documents are labeled from the top re-ranking list of T5-3B model after a first stage BM25 list.

4.4.2 Step Further: Meticulous Pseudo-Relevance Labeling with SimANS Hard Negative

Previous pseudo-labeling approach is based on top K reranked instances and global random negatives or BM25 hard negatives. Although results are overall good, we want to step further. Recently, researchers have shown hard negative sampling is vital for training a good dense retrieval model [183]. Xiong et al. [183] propose ANCE which selects hard training negatives using an asynchronously updated ANN index, showing the accuracy nearly matches BERT-based reranking model. The current DR model which is being fine-tuned retrieve the entire corpus and is formed as an ANN index.

Recently, Zhou et al. [195] show that existing negative sampling strategies suffer from the uninformative or false negative problem, and that the negatives ranked around the positives (for example BM25 scores or dense retrieval scores) are generally more informative and less likely to be false negatives compared to others. This led them to consider the sampling probability distribution [195] defined by:

$$p_i \propto \exp\left(-a(s(q,d_i) - s(q,\tilde{d}^+) - b)^2\right), \forall\, d_i \in \widetilde{\mathcal{D}}^-, \tag{4.1}$$

where the hyper-parameter a controls the density of the distribution, the hyper-parameter b controls the peak of the distribution, $\tilde{d}^+ \in \mathcal{D}^+$ is a randomly sampled positive, $\widetilde{\mathcal{D}}^-$ is the top-k ranked negatives.

In this chapter, we use this SimANS [195] approach with the proposed pseudo-positive labeling approach. This is to say, our positives are not ground-truth positives labeled by humans, instead by the T53B model. The architecture is shown in Figure 4.2. We use SimANS to select hard negatives in the top ranking of current DR models (i.e., D-BERT and GPL respectively). Note that the negatives ranked around the positives do not mean they are in the most top ranking list, they are in fact around the positive instance in current dense retrieval model list (e.g., a top-1 positive document from T53B list is viewed as pseudo-positive, and may be at the position 50 in the dense retrieval list, and we should sample the documents around position 50 as negatives). Thus more ambiguous and informative negatives can be sampled. During implementation, we found that several T53B top (e.g., 10) ranked documents are not in the top ranking list of DR models, we thus remove them if they are not in Top n (e.g., 500) list of dense retrieval scores. We assume this step ensures that if the ensemble models both give high relevant scores for the documents, they can less likely be false positive documents.

With above steps, the pseudo-positive documents and hard negatives can be obtained together with the queries, obtaining the triplet training data (the green blocks in Figure 4.2) used for the pairwise loss.

4.4.3 Improving GPL: Combining Pseudo-Relevance Labels and Pseudo-Queries

As mentioned before, the QGen and GPL approaches both rely on a query generator to generate pseudo-queries in order to train a dense retrieval model. We propose here to further train this dense retrieval model on the pseudo-relevance triplets described in Section 4.4. We believe one can gain

from this additional training on the target collection as pseudo-queries and pseudo-relevance labels rely on different sources of information and are complementary to each other. As we will see in the experimental section, this combination indeed improves the pseudo-query generation approach.

4.4.4 Pairwise Loss

In this chapter, we rely on the RankNet pairwise loss [9, 88] to train a dense retrieval model using the triplets generated above, defined by:

$$\mathcal{L}(q, d^+, d^-; \Theta) = -\log(\sigma(S_{q,d^+} - S_{q,d^-})),$$

where q is a query, (d_q^+, d_q^-) is a (positive,negative) training document pair for q, σ is the sigmoid function, Θ represents the parameters of the dense retrieval model, and $S_{q,d}$ is the score provided by the model for document d with respect to query q.

4.5 Experiments

4.5.1 Data Sets

The MS MARCO passage ranking data set [116] is used as the source domain data. We want to experiment in an extreme scenario where no test queries can be seen even without human labels. This is to say, we need to generate the pseudo-training data with the training queries which is not in the test set. To do so, we experiment on 3 target domain data sets from the BEIR benchmark [157]. They are FiQA, finance question answering [104] which

contains 6000 training queries, BioASQ biomedical question answering [160] (following [172], irrelevant documents are randomly eliminated, leaving 1M documents) which contains 3243 training queries from original collection[1], and Robust04, news documents [168] which contains 250 queries. Different topics and tasks are covered by these chosen data sets. For Robust04, we select the first 100 queries as training and development set, and the last 150 queries are used as test set.

4.5.2 Experimental Setting

Our implementation is based on the matchmaker open-source framework[2] which is modified with mean pooling and dense evaluation[3] using automatic mixed precision [109]. On the target collection, the training triplets are generated according to the approaches described in Section 4.4, using BM25+T5 to produce pseudo-relevant documents by considering the top k documents as relevant, and with different negative sampling strategies. Following previous work and for fair comparison, the transformer architecture of the dense retrieval model, referred to as D-BERT, is DistilBERT [152] with 6 layers. D-BERT is firstly trained on the source domain. We conduct two groups of experiments. For domain adaptation, For the first group, the dense retrieval model is based on D-BERT, and for the second group, the dense retrieval model is based on GPL. D-BERT and GPL are trained using the RankNet pairwise loss on both sets of triplets (obtained by BM25+T5 and different negative sampling strategies). Note that GPL is first trained on the target pseudo-queries it generates and associated documents prior to be trained on the target triplets. The T5 model used is the 3B version that is trained on MS

[1] http://participants-area.bioasq.org/Tasks/8b/trainingDataset/
[2] https://github.com/sebastian-hofstaetter/matchmaker
[3] https://github.com/UKPLab/gpl/blob/main/gpl/toolkit/evaluation.py

MARCO passage ranking data set[4]. The MiniLM cross-encoder used is the *ms-marco-MiniLM-L-6-v2* version[5] from Sentence Transformers [144].

For constructing the training set, we select the number k of top documents to be considered as relevant according to the number of queries (to generate enough pairs) and documents (as a large number of documents enables sampling more negative documents). For each relevant document, we select m documents from the target collection not present in the top k list of the query, with different negative sampling strategies. These documents are considered as not relevant. Table 4.1 displays the number of queries (not in the test set), the value selected for k (in parenthesis) and the number m of non relevant documents per relevant document. For BioASQ for example, the number of triplets in the training set is $3193 \times 2 \times 15 = 95790$. At the end, each data set has a number of triplets in the training set in the range of 50000 to 100000. We also construct a development set to select the hyper-parameters of the models on each collection. For each query in the development set, the top 10 documents are considered relevant and 90 randomly selected documents (not in the top 10) as non relevant. This choice is dictated by the fact that we need a sufficient number of relevant documents for evaluation purposes and that we have a limited number of queries for the development set. However, to counterbalance the risk of considering as relevant documents which in fact are not, the top two documents are labeled as '2' and the following eight ones as '1'. The non-relevant documents are labeled as '0', thus leading to 3-level relevance judgements for each data set. The best model is saved according to the NDCG@10 score on the development set evaluated every 1K steps.

Following [172], a maximum sequence length of 350 with mean pooling and dot-product similarity is used. For all data sets, we use a batch size of

[4] `https://huggingface.co/castorini/monot5-3b-msmarco`
[5] `https://huggingface.co/cross-encoder/ms-marco-MiniLM-L-6-v2`

8, which means 8 positive-negative pairs, and a learning rate of 2e-6 with Adam optimizer for 10K training steps. Cosine LR schedule [98] is also used for learning rate decay.

For SimANS, the hyper-parameter a and b are set to 0.5 and 0 respectively for all experiments.

Tab. 4.1.: The top k selected as positive and m as negative for each data set. The number in parentheses is used for generating training data, remaining for Dev set. Top k as relevant, m as non-relevant.

data set	#queries (exclude test)	#docs	k	m
FiQA	6000 (5960)	57K	1	10
BioASQ	3243 (3193)	1M	2	15
Robust04	100 (90)	528K	15	67

4.5.3 Baselines

Following [172], we compare the proposed approaches with zero-shot models, with pre-training approaches and with recent SOTA approaches to domain adaptation.

Zero-shot models

Baseline zero-shot models comprise BM25 based on Anserini [185] with default parameters which obtains top 100 documents for each query and does not require to be trained is compared (these BM25 ranking lists are further used to generate pseudo-relevance training data in this chapter) and the dense retrieval model D-BERT solely trained on the source collection with hard negatives and the MarginMSE loss with the *ms-marco-MiniLM-L-6-v2* model taken as the teacher model (it is further used as a start point for domain adaptation).

Pre-training based models

We compare with SimCSE [44], ICT [81] and TSDAE [171]. These models are all firstly pre-trained in a self-supervised way on the target data set and then fine-tuned on MS MARCO.

Domain adaptation approaches

We compare here with four recent SOTA approaches: MoDIR [181] which is based on ANCE [183] and relies on an adversarial training, UDALM [69] which is based on multi-task learning, and QGen [100] and GPL [172] which are approaches based on query generation.

In addition, we make use of the interaction-based models BM25+CE and BM25+T53B which can be seen as strong baselines due to the good behaviour of interaction-based models in OOD settings [157] but which are nevertheless inefficient at inference. These models re-rank the top 100 BM25 ranked list, using a *ms-marco-MiniLM-L-6-v2* and T53B cross encoders respectively.

4.5.4 Results and Analysis

Table 4.3, Table 4.4 and 4.5 display the results obtained with the different models and approaches. The results reported for BM25+CE, UDALM, MoDIR, SimCSE, ICT, TDSAE, QGen and TSDAE+GPL are from [172]. Since we test the Robust04 on the last 150 queries, for BM25+CE, GPL and TSDAE+GPL, we load the trained checkpoints of D-BERT and Wang et al. [172][6], and evaluate them on the last 150 queries. The notation "DoDress-BM25 (D-BERT)" (respectively "DoDress-T53B (D-BERT)") corresponds to the D-BERT

[6] `https://huggingface.co/GPL`

dense retrieval model pre-trained on MS MARCO and fine-tuned on the target data using the pseudo-relevance labels generated using BM25 (respectively using BM25+T5). The notation (GPL) means the same for GPL, which is again first trained on the target pseudo-queries it generates and associated documents prior to be trained on the target triplets.

We analyze the results by answering three research questions.

RQ1 Do BM25+T53B top positives help domain generalization for dense retrieval models?

From Table 4.3, we see DoDress-T53B (D-BERT) improves over D-BERT on FiQA dataset with all the three negative sampling strategies, DoDress-T53B (GPL) shows a similar trend compared to GPL. On Robust04 dataset, from Table 4.4, we see a similar trends for DoDress-T53B (D-BERT) and DoDress-T53B (GPL). Specifically, DoDress-T53B (GPL) with all three sampling strategies can outperform GPL and TADAE + GPL. With the SimANS negative mining strategy, DoDress-T53B (D-BERT) shows a 11.5% $((43.6-39.1)\div 39.1)$ improvement over D-BERT, and DoDress-T53B (GPL) shows a 8.6% improvement over GPL. From Table 4.5, however, the approach with global random negative sampling strategy fails, while the proposed approach with other two negative sampling strategies improve the dense retrieval model D-BERT and GPL respectively. These approach shows that proposed pseudo-relevance labeling approach can help dense retrieval models generalize to new domains, but different negative sampling strategy is vital for final effect.

RQ2 What is the impact of different negative sampling strategies and which is the best?

From Table 4.3, Table 4.4 and Table 4.5, we see a overall ascending trend of the three different negative sampling strategies. The global random negative

method, as the name shows, does not sample hard negatives. It randomly select negatives from the entire corpus, e.g., on BioASQ, the 2M documents, and consider them as negative. However, although it improves the dense retrieval models on FiQA and Robust04, it fails on BioASQ dataset. This might be due to the uninformative negatives: they may be too easy for the DR models on target domain.

Regarding the BM25 hard negative and SimANS negative sampling strategies, they shows better performances than the global random negative strategy, and improves D-BERT and GPL on all three datasets. These results show hard negative sampling is important for the proposed pseudo-relevance labeling data generation approach.

The proposed approach with SimANS hard negative sampling performs consistently the best on all the datasets, better than the global random negative and BM25 hard negative strategies, showing that better negative sampling strategy can also further improve the proposed approach.

RQ3 What is the effect of the proposed pseudo-relevance labeling approach with SimANS hard negative sampling compared with the baseline models?

We now analyze the proposed approach with SimANS hard negative sampling strategy comparing with the baselines. We have shown with SimANS, DoDress-T53B (D-BERT) and DoDress-T53B (GPL) consistently improves over D-BERT and GPL respectively and performs the best in the three negative sampling strategies. We now compare it with other baselines.

BM25 is a standard first stage retrieval algorithm and does not require to be trained, viewed as a zero-shot model. Although it is extremely simple compared with recent neural-IR approaches, it is a strong baseline and perform

well on new domains. Especially on BioASQ, BM25 approach even outperforms the BM25 + CE reranking approach on new domain. Our proposed approach outperform it on FiQA and Robust04. Specifically, DoDress-T53B (GPL) with SimANS is the only approach that outperform BM25 on Robust04, where GPL and TSDAE + GPL fail. On BioASQ, all DR models is lower than BM25, while DoDress-T53B (GPL) is the best in them.

For UDALM, MoDIR (ANCE) and the three pre-training based approach SimCSE, ICT and TSDAE, we report the results obtained on FiQA and BioASQ from [172]. As one can note, on FiQA, DoDress-T53B (D-BERT) and DoDress-T53B (GPL) with SimANS hard negatives reach 31.0 and 34.9 respectively on NDCG@10, while MoDIR (ANCE), the best method among the previous ones, only reaches 29.6. On BioASQ, the best baseline is TSDAE with 55.5, while our DoDress-T53B (D-BERT) and DoDress-T53B (GPL) are 60.6 and 65.3 respectively, where the latter one has a 17.7% improvement.

The generation based models are previous state-of-the-art, and are mainly based on the generated pseudo-queries. For example, a T-5 sequence-to-sequence model can be trained on MS MARCO to mimic to generate pseudo-queries, given the large query-document pairs. Then this T5 query generator can be used on the documents on target domain, obtaining pseudo-queries, and the corresponding documents used are viewed as positives. Our approach is slightly different, by also taking into the original queries and documents simultaneously into account. Then an effective reranking model T5-3B is used for pseudo-positive annotation, and different negative sampling strategies are explored for negative annotation. In Table 4.3, we see TSDAE + GPL performs the best in the baselines, with a score of 34.4, better than GPL's 32.8. The proposed approach DoDress-T53B (GPL) obtains 34.9, higher than them. On Robust04, in Table 4.4, however, TSDAE + GPL is worse than GPL: 40.7 and 41.9 respectively. Our proposed approach with SimANS hard negatives,

both DoDress-T53B (D-BERT) and DoDress-T53B (GPL) outperform them, showing the effectiveness of the approach. On BioASQ dataset, DoDress-T53B (GPL) obtains better results than the best model GPL in the baselines.

RQ4 Does sampling hard negatives in the retrieval list of current DR model perform better? Can SimANS further improve this?

We want to see if sampling the hard negatives in the retrieval list of current dense retrieval model is better than sampling the hard negative in the BM25 list. Previous BM25 hard negatives which are used in above experiments, are from random hard negative sampling without using SimANS sampling. So we conduct a further experiment using random negative sampling from current DR model's top list, to compare with sampling from BM25 top list. The results are shown in Table 4.2. We can see that sampling hard negatives from D-BERT's top retrieval list performs better than from BM25 top list. This may be explained by the fact that hard negatives from the top list of the DR model are more informative to train the model for domain adaptation.

Tab. 4.2.: Results of DoDress-BM25 (D-BERT) on Robust04 with different random hard negative sampling source.

Random sampling from	nDCG@10 (%)
BM25 top list	41.6
GPL top retrieval list	42.6

Besides, we want to see whether SimANS hard negative sampling can further improve the retrieval results. From Table 4.4, we see that DoDress-BM25 (D-BERT) using SimANS sampling approach obtains 43.6, while in Table 4.2, it is 42.6, showing that SimANS can further improve the result by sampling more ambiguous negatives than random sampling from the top ranking list of current DR model.

In conclusion, the above results demonstrate the effectiveness of the proposed approach which provides state-of-the-art generalization results of dense re-

trieval models on several collections. Besides, by exploring different negative sampling strategies, we show hard negative sampling is vital. By using a SimANS negative sampling strategy from current DR model's ranking list, the results is the best.

Tab. 4.3.: Domain Adaptation Result of FiQA (during training only use training set queries).

model	nDCG@10 (%)
Zero-Shot Models	
D-BERT	26.7
BM25 (Anserini)	23.6
Re-Ranking with Cross-Encoders (Upper Bound)	
BM25 + CE	33.1
BM25 + T53B	39.2
Previous Domain Adaptation Methods	
UDALM	23.3
MoDIR (ANCE)	29.6
Pre-Training based: Target → D-BERT	
SimCSE	26.7
ICT	27.0
TSDAE	29.3
Generation-based (Previous SOTA)	
QGen	28.7
GPL	32.8
TSDAE + GPL	34.4
Proposed: T53B, Global Random Neg	
DoDress-T53B (D-BERT)	27.3
DoDress-T53B (GPL)	33.0
Proposed: T53B, BM25 Hard Neg	
DoDress-BM25 (D-BERT)	30.4
DoDress-BM25 (GPL)	34.2
Proposed: T53B, SimANS Hard Neg	
DoDress-T53B (D-BERT)	**31.0**
DoDress-T53B (GPL)	**34.9**

4.6 Conclusion

This chapter studies whether one can benefit from existing re-ranking based IR models, pre-trained on MS MARCO, to generate pseudo-relevance labels on an unannotated target collection. These labels with sampled negatives are then used to fine-tune dense retrieval models on the target collection.

Tab. 4.4.: Domain Adaptation Result of Robust04 (training and development set use the first 100 queries, test set is the last 150 queries).

model	nDCG@10 (%)
Zero-Shot Models	
D-BERT	39.1
BM25 (Anserini)	44.4
Re-Ranking with Cross-Encoders (Upper Bound)	
BM25 + CE	45.8
BM25 + T53B	51.8
Generation-based (Previous SOTA)	
GPL	41.9
TSDAE + GPL	40.7
Proposed: T53B, Global Random Neg	
DoDress-T53B (D-BERT)	40.5
DoDress-T53B (GPL)	43.2
Proposed: T53B, BM25 Hard Neg	
DoDress-BM25 (D-BERT)	41.6
DoDress-BM25 (GPL)	43.3
Proposed: T53B, SimANS Hard Neg	
DoDress-T53B (D-BERT)	**43.6**
DoDress-T53B (GPL)	**45.5**

Tab. 4.5.: Domain Adaptation Result of BioASQ (during training only use training set queries).

model	nDCG@10 (%)
Zero-Shot Models	
D-BERT	53.6
BM25 (Anserini)	73.0
Re-Ranking with Cross-Encoders (Upper Bound)	
BM25 + CE	72.8
BM25 + T53B	76.1
Previous Domain Adaptation Methods	
UDALM	33.1
MoDIR (ANCE)	47.9
Pre-Training based: Target → D-BERT	
SimCSE	53.2
ICT	55.3
TSDAE	55.5
Generation-based (Previous SOTA)	
QGen	56.5
GPL	62.8
TSDAE + GPL	61.6
Proposed: T53B, Global Random Neg	
DoDress-T53B (D-BERT)	52.9
DoDress-T53B (GPL)	62.0
Proposed: T53B, BM25 Hard Neg	
DoDress-BM25 (D-BERT)	58.6
DoDress-BM25 (GPL)	64.7
Proposed: T53B, SimANS Hard Neg	
DoDress-T53B (D-BERT)	**60.6**
DoDress-T53B (GPL)	**65.3**

Our study reveals that this approach works well where the pseudo-labels are generated using a T53B model to re-rank a first stage BM25 list, and that it helps improve the generalization results of the GPL model which also makes use of generated queries and associated relevant documents on the target collection.

We further study the importance of using hard negative sampling strategies. A BM25 hard negative sampling approach shows consistently improvement over the base dense retrieval models after training using the generated data, which global random negative sampling is not always good. By incorporating a recent hard negative sampling strategy SimANS, which sampled hard negatives from the current dense retrieval model's top retrieval lists, shows the best results, obtaining new state-of-the-art results on the data sets for DR model's domain generalization.

5 Domain Adaptation for Conversational Search

5.1 Introduction

Conversational document search, which is to find relevant documents from collections of documents in response to user queries in a conversational context, is often referred to as "conversational search" as documents are the typical output generated by the system [41]. In the preceding section, we presented a novel approach that enhances the generative capability of dense retrieval models, which are commonly used retrieval architectures. Currently, conversational search, involving the interaction with a system through natural conversations for information retrieval purposes, has gained significant attention as an increasingly popular research area and is acknowledged as a crucial frontier within the realm of information retrieval [192, 23]. In recent times, ChatGPT[1] has gained widespread popularity worldwide. Despite being a generation-based model at its core, its ability to comprehend conversations showcases the advantages of conversational search, wherein the system also needs to grasp the user's information needs across multiple turns. The objective of conversational search is to build such a model that return the most relevant results to users during conversations.

[1] https://openai.com/blog/chatgpt

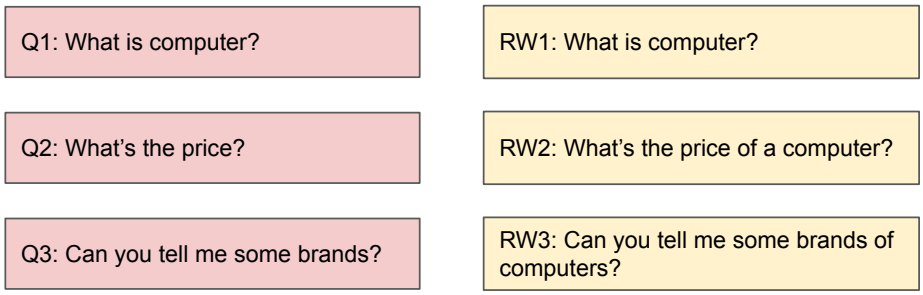


Fig. 5.1.: An example of conversational search user queries, and the rewritten queries or user intentions.

The challenge in conversational search lies in the understanding of queries. Human conversations exhibit contextualization, conciseness, and reliance on prior knowledge. Figure 5.1 shows an example of conversational search's queries (the left part) with the rewritten queries or the user intentions (the right part). Given a query, for example the third one, the system needs to understand the omission or user intention, by taking account into previous queries. Conversational search often involves referencing and omitting information from previous turns, resulting in ambiguities, which pose challenges for search systems in accurately understanding the underlying information needs, making it more complex compared to ad hoc retrieval [190]. Recently, two main approaches have been proposed to address this challenge. The first approach involves reformulating conversational queries into fully specified, context-independent queries that can be effectively processed by existing information retrieval systems [189, 108, 145, 162]. The second approach, termed as conversational dense retrieval (CDR), directly utilizes a dense retrieval architecture. The query encoder takes into account both the current query and the history of queries to generate a representation that captures the user's information needs [190, 106, 93]. The relevance scores of documents can then be obtained by matching the representations.

Despite the simplicity and efficiency of utilizing dense retrieval models, a well-known issue is the requirement of a large amount of training data. Additionally, annotating relevance labels for a target dataset with human experts can be expensive. To address this data scarcity issue, recent research has proposed various approaches. Mao et al. [106] transform web search sessions into conversational search sessions to train the CDR model. Yu et al. [190] learn a student query encoder with concatenated queries to mimic teacher embeddings on oracle reformulated queries using CANARD [35], and then train the model on the target dataset with human judgments. Lin et al. [93] use ColBERT [72] to rerank the BM25 list for human rewritten queries in CANARD with documents in target domain, and thus obtain pseudo-labels.

Although these approaches partially address the data scarcity problem of CDR, they still suffer from domain gaps in the training data. Therefore, obtaining relevance information for queries and documents in the target domain remains a significant challenge.

In this chapter, we propose a pseudo-relevance labeling approach for conversational queries and documents in the target dataset, also assuming a difficult scenario where there are no human rewritten queries for conversational queries. Our solution involves training a T5-Large [141] model to generate rewritten queries given conversational queries, which can then be used on the target dataset to generate rewritten queries. Once the rewritten queries are obtained, pseudo-relevance labeling procedures can be employed, as in the previous chapter. In a common few-shot setting with limited query numbers, this pseudo-relevance data can serve as a complement for the related CDR models, enabling domain adaptation on the target dataset.

5.2 Related Work

Conversational search differs from standard document search in that its queries are related to previous turns and often contain ambiguities. Two commonly proposed approaches for addressing the challenges of conversational search are:

- Query rewriting module: This approach relies on a module that rewrites the conversational queries into a more standard format that can be effectively handled by existing information retrieval systems.

- Dense retrieval technique: This approach leverages the dense retrieval architecture, termed as conversational dense retrieval (CDR), where the query encoder of the dense retrieval model is trained to directly understand conversational queries and generate representations that capture the user's information needs.

As to the first kind, researchers mainly rely on rule-based approach or learning based approach. For example, Mele et al. [108] propose an utterance rewriting approach that enriches the current utterance with context keywords. Their rewriting approach aims to maintain the conversation context since natural-language utterances might be not self-explanatory. They utilize a linguistically-driven approach that involves sophisticated components such as Part-of-speech tagging, named entity resolution, dependency parsing, and co-reference resolution to analyze the text and identify crucial texts for utterance rewriting. These components form the core modules of their approach. Experimental results show that the use of well-formed and self-expressive utterances consistently improves the precision of results. Among their various rewriting techniques, the one that employs topic detection and its possible variations (referred to as "Topic Shift") yields results closest to manually

rewritten utterances compared to other methods. However, there still exists a substantial disparity between the results obtained from their approach and manually rewritten utterances.

Another research approach in query rewriting involves learning-based models, particularly sequence-to-sequence models, which are designed to generate target sequences from source sequences. Ren et al. [145] formulate the task of conversational query understanding as context-aware query reformulation, with the goal of transforming the conversational query into a format that is compatible with search engines in order to effectively address user's information needs. A general recurrent neural network (RNN) is employed in the encoding process of the sequence-to-sequence approach, while another RNN is utilized for sequential generation of the target sequence. This generation process takes into account the RNN state and incorporates the embedding of the previously generated word, aided by an attention mechanism. Several variations of sequence-to-sequence models are proposed, including concatenated sequence-to-sequence, which concatenates history queries with the current query, and pair sequences-to-sequence, which employs distinct RNNs to encode history queries and the current query. Experimental results show that the best-performing model accurately reformulates more than half of the conversational queries, demonstrating the potential of sequence-to-sequence modeling for this specific task. Later works focus on using the Transformer architectures, especially the decoder part. Vakulenko et al. [162] propose a question rewriting model that leverages a unidirectional Transformer decoder for encoding the input sequence and decoding the output sequence. The input to the model encompasses the question as well as previous conversation turns. During the training process, the objective is to predict the output tokens based on the ground truth question rewrites that are generated by human annotators.

After the conversational queries have been rewritten into conversation turns, standard retrieval models can be employed, treating the rewritten queries as regular search queries. This can include term matching based approaches such as BM25, neural information retrieval (IR) approaches like BERT, dense retrieval models, and other relevant techniques.

The second research approach for conversational search involves the utilization of dense retrieval architecture. In this approach, history queries and the current query are combined and input together to the query encoder. During training, the objective is to align the representations of the concatenated queries and the positive document, while separating them from negative documents. This approach is referred to as conversational dense retrieval (CDR). One of the advantages of CDR is that, by leveraging deep learning and training with a large amount of data, the query encoder can learn to generate effective representations, and it does not need the query rewriter module. Additionally, dense retrieval is known for its fast retrieval speed, as document processing can be done offline. This makes CDR efficient and straightforward in terms of both speed and simplicity.

However, one common issue is the need of large amount of data to train the query encoder with ground-truth rewritten queries, which is difficult and expensive to obtain for target dataset. Training dense retrieval is challenging since dense retrieval models need more relevance-oriented supervision signals to be effective [190]. Besides, different from standard query encoder of DR models, the query encoder of CDR needs to capture the user search intention in the conversation, making it to be more difficult, and not able to reuse pre-trained query encoder (e.g., trained on MS MARCO).

Related papers of CDR have been proposed that try to alleviate this data scarcity issue. Mao et al. [106] present a data augmentation method called ConvTrans that can automatically transform easily-accessible web search

sessions into conversational search sessions to alleviate the data scarcity issue for conversational dense retrieval. Although user's multiturn interaction with the web search engine is similar to the multi-turn interaction in the conversational search session to some extent, directly using raw web search sessions may not be effective for training the conversational session encoder due to the gaps. To address these gaps, the authors develop a graph-based reorganization method and a specific conversational query rewriter composed of two T5 models, and then generate pseudo-conversational search sessions. Experiments show it can achieve comparable retrieval performance as it trained on high-quality but expensive artificial conversational search data.

Yu et al. [190] propose a teacher-student framework called ConvDR that improves the few-shot ability of CDR by learning from an well trained ad hoc dense retriever. They learn a student query encoder with the concatenated queries to mimic the teacher embeddings on oracle reformulated queries on CANARD [35] dataset, with mean squared error (MSE) loss, and otained good results. However, human judgements of target datasets are still required. A similar approach to this chapter is called CQE [93]. They use the conversational queries and human rewritten queries in the conversational query reformulation dataset CANARD [35] for the target datasets. ColBERT [72] is used to rerank the BM25 top list with the oracle rewritten queries in CANARD, and top 3 are viewed as relevant, other reranked top ones are viewed as negative. Although good results are obtained, this approach has the potential problem: they assume the target datasets can be well searched by CANARD queries. Apparently this is not always a truth, for example in standard dense retrieval dataset, robust04 queries are not related to TREC-COVID, which is a scenario for conversational search if the target is not happened to be related to CANARD.

Although approaches like ConvDR [190] and CQE [93] can utilize the large conversational query rewritten dataset to train CDR model, they still suffer from domain gaps in the training data. In this chapter, we propose a pseudo-relevance labeling approach on the target conversational search dataset based on the pseudo-labeling of previous chapter. To deal with the difficulty of obtaining human rewritten queries on target datasets, we propose to use T5-Large which is pre-trained on CANARD, to rewrite conversational queries. Then we use the rewritten queries, and T5-3B from MS MARCO, to generate pseudo-relevance labeling. The hard negatives are obtained with current dense model and SimANS method [195]. For target datasets like TREC CAsT-19 [29], which only contain a few training queries, training the query encoder from scrach is not a best solution as this is a few-shot setting. If in this scenario, our proposed method can be viewed as a complement step of ConvDR [190] and CQE [93]: the DR model can be firstly trained using these approaches, and then train it on target dataset using our proposed pseudo-labels.

5.3 Domain Adaptation for Conversational Dense Retrieval: Leveraging Pseudo-Relevance Labels Generated with T5-Large Rewritten Queries

As discussed above, our objective is to generate pseudo-labels without relying on human rewritten queries in the target dataset, considering the challenges and costs associated with annotating a new conversational dataset in real-

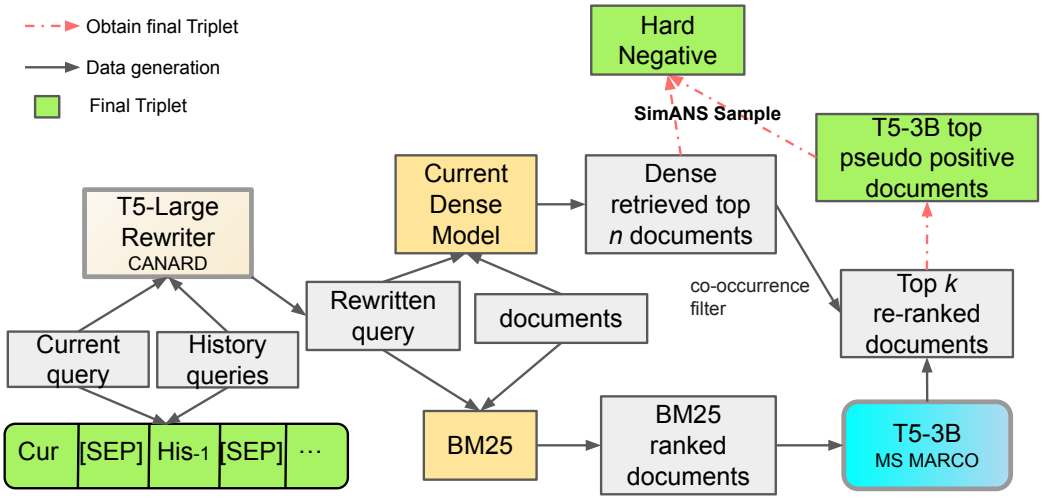


Fig. 5.2.: Overall pipeline of generating pseudo-data for conversational dense retrieval. The rewritten query is a human language style sentence that represent whole user intention, and the current dense model (pre-trained on MS MARCO) can retrieve top documents.

world scenarios. Compared with Chapter 4, this pseudo-relevance data generation process needs an additional module that acts as query rewriter.

The overall pipeline is depicted in Figure 5.2, wherein the architecture primarily incorporates an additional T5-Large model [141] in the left section to generate rewritten queries. Further details are given below.

5.3.1 Quantifying the Requirement of Pseudo-Relevance Data for Training Conversational Dense Retrieval Model

We begin by analyzing the data required to train a conversational dense retrieval model. As discussed in Section 5.2, the query encoder in conversational dense retrieval is trained to accept concatenated queries, which include the current query and history queries, and encode them into embeddings

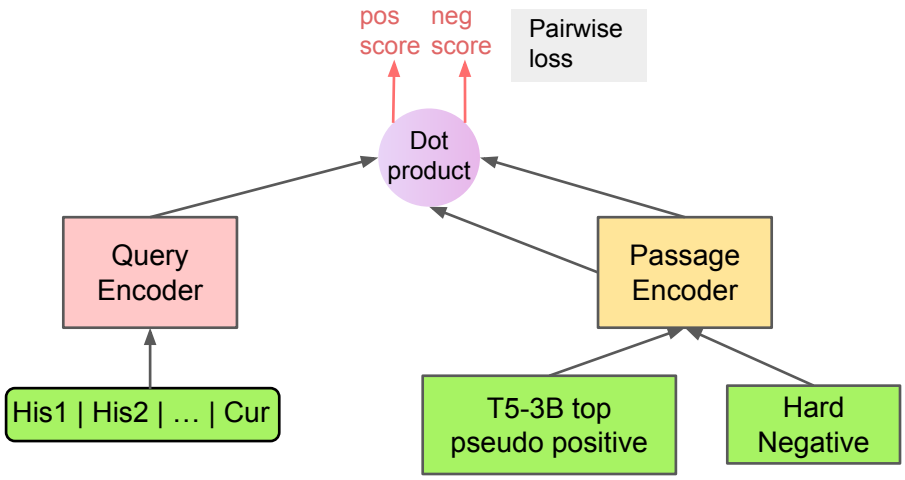


Fig. 5.3.: After generating pseudo-labeling data, now do domain adaptation for the dense retrieval model for target conversational search corpus. Now the encoders of the dense model (pre-trained on MS MARCO) are no longer shared, and the query encoder is trained to generate whole representation of the conversation (concatenated, not rewritten by T5-Large, since for fast online search).

that encapsulate the overall user intentions. To train a conversational dense model, relevance labels for some documents at each turn in the conversation are required. However, obtaining ground-truth query representations that encompass the entire user intention, including history turns in a conversation, is challenging. To address this, we propose the use of a "machine oracle" that can generate rewritten queries based on the current query and history queries or conversations. In this study, we achieve this by training a T5-Large sequence-to-sequence model on the CANARD dataset [35], which is a dataset for learning to rewrite conversational queries. This trained T5-Large model serves as the oracle for the target conversational search dataset, enabling the rewriting of each query in a conversation. Subsequently, with the rewritten queries, we can employ a similar pseudo-data generation process as described in Chapter 4. The detailed procedures are outlined below.

5.3.2 T5-Large Query Rewriter Module

This module is shown in the left part of Figure 5.2. We utilize the T5 model's special token "</s>" to concatenate the current query and history queries. The current query is placed at the beginning, followed by the recent history queries (with farther history queries located towards the end), following a similar approach as described in [106]:

$$< /s > Current < /s > History_{-1} < /s > History_{-2} ... \quad (5.1)$$

The T5-Large model is trained using the CANARD dataset [35] to generate rewritten queries that comprehensively capture the user intentions, with ground-truth human rewritten queries serving as labels. Once trained, this model can effectively rewrite conversational queries from target datasets into the desired format.

5.3.3 Generating Pseudo-Relevance Data on Target Dataset

Pseudo-Positive We initially leverage the trained T5-Large model to generate de-contextualized queries denoted as *ReQ* for each turn in the conversations. Subsequently, we employ BM25 to retrieve a list of top-ranked documents based on the generated *ReQ*. Following that, we utilize the T5-3B model, which has been trained on MS MARCO, to rerank the BM25 list. The resulting top-k documents from this process can be considered as pseudo-relevant instances.

SimANS Hard Negative Following Section 4.4.2, we use the SimANS [195] to sample hard negatives. Also with the T5-Large rewritten queries: *ReQ*, we use the dense retrieval model to retrieve the whole corpus and obtain DR score list. In previous step for obtaining pseudo-positive instances, we also have the top ranking list by T5-3B. With the DR score list and T5-3B list and SimANS [195] hard negative sampling, as shown in section 4.4.2, then we can obtain hard negative instances. These procedures are shown in the right part of Figure 5.2.

Training the Conversational Dense Retrieval Model The architecture for training the conversational dense model using training triplets is illustrated in Figure 5.3. To train the dense retrieval model, we concatenate the history and current queries, and aim to teach the query encoder to generate a de-contextualized query representation. Consequently, each line of the training triplet file can be presented in the following format:

[ConcatenateQ; Texts of a Pseudo-Positive Document; Texts of a Pseudo-Negative Document],

where $ConcatenateQ$ follows the format in [93]:

$$History_1 | \ History_2 | ... | \textit{Current}$$

The triplet training file can serve as training data for training a CDR model from a source domain. If the generated data is large enough, it even could be used to train a query encoder from scratch.

5.4 Experiment on Conversational Search

Data Set Used

We utilize the TREC CAsT 2019 (CAsT-19) dataset [29] as the experimental conversational search dataset. CAsT-19 comprises of 30 training topics and 20 test topics, with each topic representing a conversational search session consisting of queries from multiple turns. The dataset contains a total of 269 training queries, and for each query, the neural network needs to comprehend the overall user intention along with the history of previous queries. Notably, in this chapter, we investigate an extreme scenario in which we do not have access to human rewritten queries and relevance labels for CAsT-19, resulting in an almost zero-shot scenario.

Following what has been done for BioASQ in [172]: the authors randomly remove irrelevant passages from the whole 15M corpus to make the final corpus size to 1M, we randomly remove irrelevant passages of the TREC CAsT-19's 38M corpus to 2M, for efficient experiments.

5.4.1 T5 Rewriter Training

The conversational rewriter used is the T5-Large version[2]. We train the model on CANARD dataset which contains 31526 training instances repeatedly for 80K instances, and evaluate the T5-Large model on development set for every 20000 instances and save the best model. The learning rate is 5e-5 and batch size is 4 using AdamW optimizer [97]. The BLEU [127] results of the final T5-Large model are presented in Table 5.1.

[2] `https://huggingface.co/t5-large`

Tab. 5.1.: BLEU scores of different approaches for rewriting conversational queries on CANARD dataset. First four results are from [35]. Human accuracy (*) is computed from a small subset of the validation set.

Method	Dev	Test
Copy	33.84	36.25
Pronoun Sub	47.72	47.44
Seq2Seq	51.37	49.67
Human Rewrites*	59.92	
T5-Large	59.5	57.9

Tab. 5.2.: BLEU scores of T5-Large for rewriting conversational queries on TREC CAsT-19 test set, compared to human rewritten queries.

Method	CAsT-19
T5-Large	64.35

We can see the T5-Large model can obtain near human accuracy for rewriting conversational queries on CANARD dataset.

The T5-Large model, trained on the CANARD dataset, can be employed to rewrite conversational queries for TREC CAsT-19. The BLEU score, as shown in Table 5.2, is compared with human rewritten queries and indicates a favorable outcome.

5.4.2 Baselines and Training

Zero-shot baselines The BERT-dot-v5[3] models trained on MS MARCO are used as zero-shot baselines. To address the challenge of conversational queries, we experiment with four methods: (1) using only the current query to retrieve documents, (2) concatenating the history and current queries as described in section 5.3.3 (it should be noted that the query encoder is not specifically trained to handle this format), (3) using T5-Large rewritten queries for retrieval, and (4) the upper bound method with human rewritten

[3] https://huggingface.co/sentence-transformers/msmarco-bert-base-dot-v5

queries from the dataset. However, it should be noted that the latter approach may not be efficient in real-world scenarios, as relying on a T5-Large model for online query conversion could be slow. We will present the results of these models as zero-shot baselines. In addition, we also include BM25 using Anserini [185] with official rewritten queries and T5-Large rewritten queries as a baseline.

ConvDR This approach [190] teaches the query encoder to learn to mimic the human rewritten queries' representation produced by the teacher encoder. We train this for 40k steps with the batch size of 8 and a learning rate of 2e-6. We observed that the performance remained similar across different training intervals, namely 10k, 40k, and 80k steps. We adopt this approach as a baseline trained on CANARD with an abundance of queries.

CQE This approach [93] uses CANARD's queries and human rewritten queries, to search the target CAsT-19 corpus. Following [93], we train it for 120k steps with a batch size of 8, which is comparable to their original paper's training process of 20k steps with a batch size of 96. The learning rate used in our training process is set to 2e-6.

Training As discussed above, the CAsT-19 dataset comprises only 269 queries across all conversations, making it a few-shot learning scenario. So we use the generated data to fine-tune a base model which is firstly trained on a domain. This is to say, we further conduct experiments based on the checkpoints of baseline models ConvDR and CQE. Following [190, 93], the passage encoder is fixed and the query encoder is trained. The parameters for data generation are shown in Table 5.3, while other parameters are the same as in Chapter 4. Our fine-tuning strategy is similar to the previous

Tab. 5.3.: The top k selected as positive and m as negative for CAsT-19. The number in parentheses is used for generating training data, remaining for Dev set. Top k as relevant, m as non-relevant.

data set	#queries	#docs	k	m
CAsT-19	269 (219)	2M	5	100

Chapter, using the RankNet pairwise loss [9, 88] with a batch size of 8 and learning rate of 2e-6. We train on our (few shot) pseudo-target data for 2000 steps and evaluate on the development set every 500 steps, saving the best models. Finally, we report the results of the trained models, and the evaluation metric is based on NDCG@3, consistent with previous research [190, 93].

5.4.3 Experiment Result

Experiment results are presented in Table 5.4.

Tab. 5.4.: Domain Adaptation Result of Cast19.

model	nDCG@3 (%)
Zero-Shot Models	
BERT-dot-v5(current)	33.4
BERT-dot-v5(concatenation)	27.2
BERT-dot-v5(T5Rewrite)	53.2
BERT-dot-v5(Human) (Upper Bound)	58.9
BM25(Human)	37.0
BM25(T5Rewrite)	31.2
Re-Ranking with Cross-Encoders	
T5-3B rerank T5Rewrite	56.7
Related Work	
ConvDR (teachQ) (BERT-dot-v5)	55.4
CQE (BERT-dot-v5)	53.7
Proposed Approach	
T53B(filter), SimANS Neg, based ConvDR	
DoDress-T53B (BERT-dot-v5)	**58.0**
T53B(filter), SimANS Neg, based CQE	
DoDress-T53B (BERT-dot-v5)	**57.6**

The analysis of the results involves addressing several research questions.

RQ1 In the context of conversational search, how do standard dense retrieval (DR) models exclusively trained on MS MARCO perform as zero-shot models?

We begin by presenting the results of using the standard DR model for conversational search. The best baseline among zero-shot models is BERT-dot-v5(Human), which constitutes an upper bound as it uses human rewritten queries on the target dataset. This model largely outperforms BM25(Human). In comparison, using the current query or the concatenation of queries with the BERT-dot-v5 model yields lower results than the human rewritten queries, respectively 33.4 and 27.2 compared to 58.9. This means that, for this dataset, although DR models can successfully retrieve documents when given real human rewritten queries and outperform BM25 approach, they do not perform well when they are not specifically fine-tuned to understand the conversational queries or when they are not given good rewritten queries. When using T5 rewritten queries, the performance becomes closer to the one obtained with human rewritten queries. In addition, using T5-3B with the T5 rewritten queries as a reranking model yields very good results, can outperform BM25 by a large margin and is close to the upper bound. The proposed T5-Large rewriter is a relative good solution for a DR model to automatically understand the queries. However, it may not be ideal for online search, as it involves the use of another large sequence-to-sequence model.

RQ2 What about the conversation dense retrieval models in the related paper compared with other baselines?

The experimental results presented in Table 5.4 show that ConvDR and CQE achieve respective scores of 55.4 and 53.7, outperforming the standard zero-shot dense retrieval (DR) model baselines except the upper bound using

human rewritten queries. These two models benefit from the large scale conversational query rewritten dataset CANARD, which enables the query encoder to learn effective representations for conversational queries. These reproduced results provide evidence for the effectiveness of CDR models.

RQ3 Does the proposed in-domain pseudo-relevance data generation approach effectively enhance the performance of CDR models?

Previous results of ConvDR and CQE have shown the effectiveness of CDR models using CANARD dataset. However, the queries in CANARD differ from those in target dataset, it is essential to train the models using in-domain queries. In this study, we aim to investigate whether the proposed in-domain pseudo-relevance data generation approach can enhance the performance of CDR models. Results of proposed models based on ConvDR and CQE show 58.0 and 57.6 respectively. Comaring with ConvDR and CQE, they have an improvement of 4.7% and 7.3% respectively. These results demonstrate that using target in-domain data and pseudo-relevance labels allows one to adapt a CDR model pre-trained on CANARD to a new target domain. This is due to the fact that the proposed pseudo-relevance labeling approach enables the CDR models to see real queries and respective documents on the target domain, resulting in better adaptation. Overall, our proposed approach achieves comparable performance to the one obtained by the BERT-dot-v5 model with real human rewritten queries, without requiring human annotations on the target dataset.

Additional experiment: Comparing with further training the query encoder with in-domain MSE loss. The ConvDR approach, as proposed by Yu et al. [190], involves training a dense retrieval model using CANARD human rewritten queries in a teacher-student manner (with MSE loss). We also want to know, what if we further train the model like this manner using the

in-domain rewritten queries? So we conduct another experiment with the T5-Large rewritten queries generated by us. The detail training procedure is the same as the ConvDR experiment and the training is based on the checkpoint of ConvDR model shown in Table 5.4. The query encoder is finetuned with a batch size of 8 for 500 training steps, and evaluate every 100 steps to save the best checkpoint.

The experimental result for NDCG@3 is 56.0, which outperforms the ConvDR model trained on CANARD dataset (55.4), but is lower than the result obtained using the proposed pseudo-relevance data generation approach (58.0). These findings suggest that while teaching the query encoder with target domain queries to mimic the representations of the teacher encoder with T5-Large rewritten queries can lead to slight improvements, the proposed approach of using pseudo-relevance data generation with standard pairwise loss is more effective in adapting the model.

5.5 Conclusion

In this chapter, we leverage the proposed pseudo-relevance labeling approach in the context of conversational search task. We incorporate a query rewritten module that utilizes T5-Large sequence-to-sequence model trained on CANARD dataset to automatically rewrite conversational queries in the target dataset without relying on human experts. This transforms the task into a standard retrieval problem, and pseudo-relevance labels are generated using T5-3B model and SimANS hard negative mining strategy. The CDR models can be firstly trained using other approaches like ConvDR with CANARD dataset. Subsequently, by further fine-tuning on the generated data, with the domain adaptation step, CDR models can further be enhanced, show-

ing the importance of domain adaptation and the effectiveness of proposed pseudo-data generation pipeline.

Part III

Differentiable Listwise Loss Functions Based on Approximate Rank Indicators

6

Listwise Learning to Rank Based on Approximate Rank Indicators

6.1 Introduction

Learning to rank [95] is a sub-field of Machine Learning and Information Retrieval (IR) that aims at learning, from some training data, functions able to rank a set of objects – typically a set of documents for a given query. Learning to rank is currently one of the privileged approaches to build IR systems. This said, one important problem faced with learning to rank is that the metrics considered to evaluate the quality of a system, and the losses they underlie, are usually not differentiable. This is typically the case in IR: popular IR metrics such as precision at K, mean average precision or normalized discounted cumulative gain, are neither continuous nor differentiable. As such, state-of-the-art optimization techniques, such as stochastic gradient descent, cannot be used to learn systems that optimize their values.

To address this problem, researchers have followed two main paths. The first one consists in replacing the loss associated with a given metric by a

surrogate loss which is easier to optimize. A surrogate loss typically upper bounds the true loss and, if consistent, asymptotically (usually when the number of samples tends to infinity) behaves like it. One of the main advantages of surrogate losses lies in the fact that it is sometimes possible to rely on an optimization problem that is convex and thus relatively simple to solve. However, Calauzènes et al. [14, 13] have shown that convex and consistent surrogate ranking losses do not always exist, as for example for the mean average precision or the expected reciprocal rank. The second solution is to identify *differentiable approximations* of the metrics considered. Typically, such approximations converge towards the true metrics when an hyperparameter that controls the quality of the approximation tends to a given value. One of the main advantages in using a differentiable approximation of a metric is the fact that one directly approximates the true loss, the quality of the approximation being controlled by an hyperparameter and not the number of samples considered. One of the main disadvantages of differentiable approximations is that the optimization problem obtained is in general non-convex. That said, the recent success of deep learning shows that solving non-convex optimization problems can nonetheless lead to state-of-the-art systems.

We follow here this latter path and study differentiable approximations of standard IR metrics. We focus on one ingredient at the core of ranking metrics: the rank indicator function. We show how one can define high-quality, differentiable approximations of the rank indicator and how these lead to good approximations of the losses associated with standard IR metrics. Our contributions are two three-fold:

- We introduce SmoothI ("smoothie"), a novel differentiable approximation of the rank indicator function that can be used in various ranking metrics and losses.

- We further empirically illustrate the behavior of our proposal on both learning to rank features and standard, text-based features, and show that it is, in both cases, very competitive compared to previous approaches.

6.2 Related Work

Listwise approaches are widely used in IR as they directly address the ranking problem [15, 180]. A first category of methods developed for listwise learning to rank aimed at building surrogates for non-differentiable loss functions based on a ranking of the objects. In this line, RankCosine [140] used a loss function based on the cosine of two rank vectors while ListNet [15] adopted a cross-entropy loss. ListMLE and its extensions [80, 180] introduced a likelihood loss and a theoretical framework for statistical consistency (extended in [79, 78, 179]), while [68] and [7, 143, 163] studied surrogate loss functions for P@K and NDCG, respectively. Lastly, LambdaRank [10] used a logistic loss weighted by the cost, according to the targeted evaluation metric, of swapping two documents. This approach has then been extended to tree-based ensemble methods in LambdaMART [11], and finally generalized in LambdaLoss [174], the best performing method according to [174] in this family.

If surrogate losses are interesting as they can lead to simpler optimization problems, they are sometimes only loosely related to the target loss, as pointed out in Bruch et al. [8]. A typical example is the Top-K loss proposed in [4] (see also [20, 184] for a study of the relations between evaluation metrics and surrogate losses). Furthermore, using a notion of consistency based on the concept of calibration developed in [155], Calauzènes et al. [14,

13] have shown that convex and consistent surrogate ranking losses do not always exist, as for example for the mean average precision or the expected reciprocal rank. Researchers have thus directly studied differentiable approximations of loss functions and evaluation metrics, from SoftRank [156], which proposed a smooth approximation of NDCG, to the recent differentiable approximation of MAP, called ListAP, in the context of image retrieval [146]. Some of the proposed approaches are based on a soft approximation of the position function [178] or of the rank indicator [18], from which one can derive differentiable approximations of most standard IR metrics. However, [178] is specific to DCG whereas [18] assumes that the inverse of the rank function is known. Qin et al. [139] proposed differentiable approximations of P@K, MAP, P@K and NDCG@K, recently used in [8], based of the composition of two approximation functions, namely the position and the truncation functions. In contrast, our approach makes use of a single approximation, that of the rank indicator, for all losses and metrics considered, and thus reduces the risk of composing errors of different approximations.

More recently, different studies, mostly in the machine learning community, have been dedicated to differentiable approximations of the sorting and ranking functions. A fundamental relation between optimal transport and generalized sorting is for example provided in [24], with an approximation based on Sinkhorn quantiles (note that [188] also exploits optimal transport for listwise document ranking, without however proving that the approximation used is correct). [6] have focused on devising fast approximations of the sorting and ranking functions by casting differentiable sorting and ranking as projections onto the convex hull of all permutations. In the context of K-NN classification, Plötz and Roth [132] proposed a recursive formulation of an approximation of the ranking function. However, no theoretical guarantees are provided, neither for this approximation nor for the K-NN loss it is used in. A more general framework, based on unimodal row-stochastic matrices,

is introduced in [48] with an approximation of the sorting operator which is used in [133] to derive a differentiable approximation to NDCG. It can be shown that the approximate rank indicator matrix our approach leads to is a unimodal row-stochastic matrix, so that our proposal can be used in their framework as well. [135] further improved the above proposal by simplifying it, an approach referred to as SoftSort. Lastly, we want to mention the approach developed by Kong et al. [75] who propose an adaptive projection method, called Rankmax, that projects, in a differentiable manner, a score vector onto the (n, k)-simplex. This method is particularly well adapted to multi-class classification. Its application to IR metrics remains however to be studied.

6.3 Differentiable IR Metrics

For a given query, an IR system returns a list of ranked documents. The ranking is based on scores provided by the IR system, scores that we assume here to be *strictly positive and distinct*[1] and that will be denoted by $\mathcal{S} = \{S_1, \ldots, S_N\}$ for a list of N documents. To assess the validity of an IR system, one uses gold standard collections in which the true relevance scores of documents are known, and IR metrics that assess to which extent the IR system is able to place documents with higher relevance scores at the top of the ranked list it returns. The most popular metrics are certainly the precision at K (denoted by P@K) which measures the precision in the list of top-K documents, its extension Mean Average Precision (MAP), as well the Normalized Discounted Cumulative Gain at K (NDCG@K) which can take into account graded relevance judgements.

[1] This is not a restriction *per se* as one can add an arbitrary large value to the scores without changing their ranking, and ties can be broken randomly.

P@K is the average over queries of P@K_q, defined for a given query q by:

$$\text{P@}K_q = \frac{1}{K}\sum_{r=1}^{K} rel_q(j_r), \tag{6.1}$$

where j_r is the r^{th} highest document in the list of scores $\mathcal{S}$ (*i.e.*, the document with the r^{th} largest score in $\mathcal{S}$), $rel_q(j)$ is a binary relevance score that is 1 if document j is relevant to q and 0 otherwise. MAP is the average over queries of AP_q defined by:

$$\text{AP}_q = \frac{1}{\sum_{j=1}^{N} rel_q(j)}\sum_{K=1}^{N} rel_q(j_K)P@K_q, \tag{6.2}$$

The normalized discounted cumulative gain at rank K, NDCG@K, is the average over queries of NDCG@K_q, defined for a given query q by:

$$\text{NDCG@}K_q = \frac{1}{N_K^q}\sum_{r=1}^{K}\frac{2^{rel_q(j_r)}-1}{\log_2(k+1)}, \tag{6.3}$$

where $rel_q(j)$ is now a (not necessarily binary) positive, bounded relevance score for document j with respect to query q (higher values correspond to higher relevance) and N_K^q a query-dependent normalizing constant. The standard NDCG metric corresponds to NDCG@N [64].

As the reader may have noticed, the common building block and main ingredient of the above IR metrics (Eqs. 6.1, 6.2, 6.3) is the relevance score of the document at any rank r, namely $rel_q(j_r)$. If one can define a "good" differentiable approximation of $rel_q(j_r)$, then one obtains a "good" differentiable approximation of IR metrics. The goal of this chapter is to introduce such a differentiable approximation, while giving "good" a precise meaning.

6.3.1 SmoothI: Smooth Rank Indicators

The relevance score of the document at any rank r in a list of N documents can be rewritten as:

$$rel_q(j_r) = \sum_{j=1}^{N} rel_q(j) I_j^r,$$

where I_j^r is the rank indicator function at rank r, which is equal to 1 if j is the r^{th} highest document in the list and 0 otherwise. Thus, the rank indicator function at rank r can be defined by:

$$I_j^r = \begin{cases} 1 & \text{if } j = \underset{\substack{j' \in \{1,\dots,N\} \\ \forall l < r, I_{j'}^l = 0}}{\operatorname{argmax}} S_{j'}, \\ 0 & \text{otherwise.} \end{cases}$$

Given the strict positivity assumption on the scores, the argmax above can be equivalently expressed as:

$$\underset{\substack{j' \in \{1,\dots,N\} \\ \forall l < r, I_{j'}^l = 0}}{\operatorname{argmax}} S_{j'} = \underset{j' \in \{1,\dots,N\}}{\operatorname{argmax}} S_{j'} \prod_{l=1}^{r-1} (1 - I_{j'}^l),$$

as the product $\prod_{l=1}^{r-1}(1 - I_{j'}^l)$ is 0 for the $(r-1)$ highest documents.

A widespread smooth approximation of the argmax is the parameterized softmax. It has been employed in, *e.g.*, [112] in the context of deep k-means clustering, in [132] in the context of neural nearest neighbor networks, as well as in [63, 103] within a Gumbel-softmax distribution employed to approximate categorical samples. The parameterized softmax defines, for any rank $r \in \{1, \dots, N\}$ and document $j \in \{1, \dots, N\}$, a smooth rank indicator $I_j^{r,\alpha}$ of the form:

$$I_j^{r,\alpha} = \frac{e^{\alpha S_j} \prod_{l=1}^{r-1} (1 - I_j^{l,\alpha})}{\sum_{j'} e^{\alpha S_{j'}} \prod_{l=1}^{r-1} (1 - I_{j'}^{l,\alpha})},$$

where α is an hyperparameter that plays the role of an inverse temperature guaranteeing that $I_j^{r,\alpha}$ converges to the true rank indicator function I_j^r (for any document j and rank r) when $\alpha \to +\infty$.

Numerical approximations in the above formulation may however lead in practice to choosing a document at a given rank r that was already selected at a lower rank. Indeed, it may be that for a document j of rank $l' < r$, $I_j^{l',\alpha}$ gets a value slightly below 1, with the risk that $S_{j'} \prod_{l=1}^{r-1}(1 - I_{j'}^{l,\alpha})$ takes the highest value for $j' = j$ and, in turn, that j is selected for both ranks l' and r. We thus slightly modify the above formulation by introducing an additional hyperparameter, leading, for any rank $r \in \{1, \ldots, N\}$ and document $j \in \{1, \ldots, N\}$, to:

$$I_j^{r,\alpha} = \frac{e^{\alpha S_j \prod_{l=1}^{r-1}(1-I_j^{l,\alpha}-\delta)}}{\sum_{j'} e^{\alpha S_{j'} \prod_{l=1}^{r-1}(1-I_{j'}^{l,\alpha}-\delta)}}. \tag{6.4}$$

The hyperparameter $\delta \in (0, 0.5)$ controls the mass of the distribution that is allocated to the $(r-1)$ highest documents: a larger δ leads to further reducing the contribution of the $(r-1)$ highest documents in the distribution at rank r. We refer to the above approximation of the rank indicator function as SmoothI.

Figure 6.1 further shows how SmoothI can be integrated in a neural retrieval system. It simply consists of the last element that is used to compute the overall loss corresponding to the desired IR metrics.

6.3.2 Gradient Stabilization in Neural Architectures

In pilot experiments, we found that the recursive computation in $I_j^{r,\alpha}$ (Eq. 6.4) could sometimes lead to numerical instability when computing its gradient

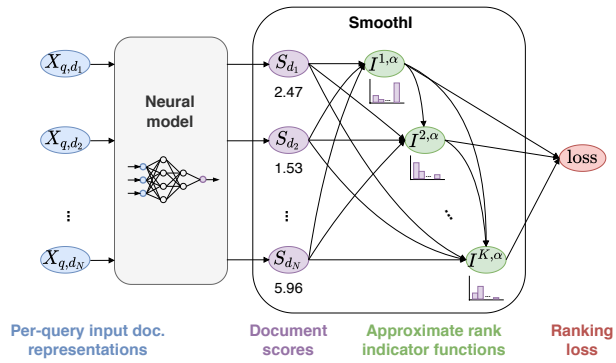


Fig. 6.1.: Illustration of SmoothI and its positioning in a neural retrieval system. Given a query q, the document representations $\{X_{q,d_i}\}_{i=1}^{N}$ are first passed through a neural model which outputs a set of scores $\{S_{d_i}\}_{i=1}^{N}$. The scores are then processed by the SmoothI module, yielding smooth rank indicators $\{I^{r,\alpha}\}_{r=1}^{K}$ up to rank K, which are ultimately used to calculate the ranking loss.

with respect to the scores $\mathcal{S}$. To alleviate this issue, we adopted a simple solution which consists in applying the stop-gradient operator to $\prod_{l=1}^{r-1}(1 - I_{j'}^{l,\alpha} - \delta)$ in the definition of $I_j^{r,\alpha}$ to "prune" the computation graph in the backward pass. This operator, which was used in previous works such as [122], acts as the identity function in the forward pass and sets the partial derivatives of its argument to zero in the backward pass, leading to the following slightly modified definition of $I_j^{r,\alpha}$ which we use in practice:

$$I_j^{r,\alpha} = \frac{e^{\alpha S_j \mathrm{sg}\left[\prod_{l=1}^{r-1}(1-I_j^{l,\alpha}-\delta)\right]}}{\sum_{j'} e^{\alpha S_{j'} \mathrm{sg}\left[\prod_{l=1}^{r-1}(1-I_{j'}^{l,\alpha}-\delta)\right]}},$$

where $\mathrm{sg}[\cdot]$ is the stop-gradient operator. In other words, we consider that the lower-rank smooth indicators $I_{j'}^{l,\alpha}$ $(l < r)$ in $I_j^{r,\alpha}$ are constant with respect to $\mathcal{S}$.

6.3.3 Application to IR Metrics

Based on the proposed smooth rank indicators, one can obtain simple approximations of IR metrics by replacing $rel_q(j_r)$ with $\sum_{j=1}^{N} rel_q(j) I_j^{r,\alpha}$, leading to the following approximation for P@K_q:

$$\text{P@}K_q^\alpha = \frac{1}{K} \sum_{r=1}^{K} \sum_{j=1}^{N} rel_q(j) I_j^{r,\alpha},$$

from which one obtains the following approximation of AP_q:

$$\text{AP}_q^\alpha = \frac{1}{\sum_{j=1}^{N} rel_q(j)} \sum_{K=1}^{N} \left(\sum_{j=1}^{N} rel_q(j) I_j^{K,\alpha} \right) \text{P@}K_q^\alpha.$$

Similarly, the approximation for NDCG@K_q is given by:

$$\text{NDCG@}K_q^\alpha = \frac{1}{N_K^q} \sum_{r=1}^{K} \frac{2^{\sum_{j=1}^{N} rel_q(j) I_j^{r,\alpha}} - 1}{\log_2(k+1)}.$$

6.4 Experiments

We conducted both feature-based learning to rank and text-based IR experiments to validate SmoothI's ability to define high-quality differentiable approximations of IR metrics, and hence meaningful listwise losses. In particular, our evaluation seeks to address the following two questions: On learning to rank collections, how does SmoothI compare to state-of-the-art listwise approaches? Do neural models for text-based IR (*e.g.*, models based on BERT) benefit from SmoothI's listwise loss?

In the remainder of this section, we first describe the experimental setup of the learning to rank experiments, then we discuss the learning to rank results. Finally, we detail our experiments with BERT on text-based IR.

6.4.1 Learning to Rank Experimental Setup

Datasets. To evaluate our approach, we conducted learning to rank experiments on standard, publicly available datasets, namely LETOR 4.0 MQ2007, MQ2008 and MSLR-Web30K [137], respectively containing 1,692/69,623, 784/15,211 and 31,531/3,771,125 queries/documents, and the Yahoo learning to rank Set-1 dataset [17], containing 29,921/709,877 queries/documents. In these datasets, each query-document pair is associated with a feature vector. We rely on the standard 5-fold train/validation/test split for the LETOR collections and the standard train/validation/test split for YLTR. In the remainder, MSLR-Web30K and Yahoo learning to rank Set-1 will respectively be referred to as Web30K and YLTR. The statistics of the different datasets for their respective train/validation/test folds are detailed in Table 6.1.

Tab. 6.1.: Statistics of the learning to rank datasets, averaged over 5 folds.

	#queries			#docs		
	train	val	test	train	val	test
MQ2007	1,015	338	338	41,774	13,925	13,925
MQ2008	470	157	157	9,127	3,042	3,042
Web30K	18,919	6,306	6,306	2,262,675	754,225	754,225
YLTR	19,944	2,994	6,983	473,134	71,083	165,660

Baseline methods. Differentiable approximations of IR metrics (and their associated losses) can be classified under two categories: surrogate losses and direct approximations. Among approaches based on surrogate losses, we have retained the three state-of-the-art approaches **ListNET** [15], **ListMLE** [180], and **LambdaLoss** [174]. The latter is considered the best performing method in the Lambda* family (LambdaRank, LambdaMART, LambdaLoss) [174], and we thus omit the comparison against LambdaMART and LambdaRank. Among approaches based on direct approximations, we have retained the recently proposed **ListAP** [146] and the state-of-the-art method **Approx**

[139], which was also recently used in [8]. In addition, we also considered losses derived from recent approaches for differentiable sorting and ranking [6, 24, 135]. We used here as baselines the most recent representatives of these approaches, namely **OT** [24], which frames differentiable sorting as an optimal transport problem, **FastSort** [6], which devises an efficient differentiable approximation based on projections onto the convex hull of permutations, and **SoftSort** [135], which proposes a continuous relaxation of the sorting operator based on unimodal row-stochastic matrices and is comparable, both in terms of method and results, to the NeuralSort method introduced in [48].

As LambdaLoss, Approx and SmoothI can be used to approximate different IR metrics, we defined several variants for each approach, respectively optimizing P@$\{1, 5, 10\}$, NDCG@$\{1, 5, 10, N\}$ and MAP, and selected the best variant based on the validation performance. The losses defined by ListNET, ListAP[2], Approx and SmoothI were implemented in PyTorch [129], using our own implementation for ListNET, Approx and SmoothI[3]. For ListMLE and LambdaLoss, we relied on the TF-Ranking library [128]. OT[4], FastSort[5] and SoftSort[6] all propose differentiable approximations of the position function. This is the same position function as the one used by Approx for computing an approximation of NDCG@N. We thus directly used this latter approximation from the outputs of OT, FastSort and SoftSort (note that Approx uses, in addition to the approximation of the position function, an approximation of the truncation function for computing approximations of truncated IR-metrics P@K and NDCG@K [139]). For evaluating the performance of the different

[2] https://github.com/almazan/deep-image-retrieval
[3] https://github.com/ygcinar/SmoothI
[4] https://github.com/google-research/google-research/tree/master/soft_sort
[5] https://github.com/google-research/fast-soft-sort
[6] https://github.com/sprillo/softsort

approaches, we used the fast Python implementation of the TREC evaluation tool [164], which calls the trec_eval evaluation metrics[7] from Python.

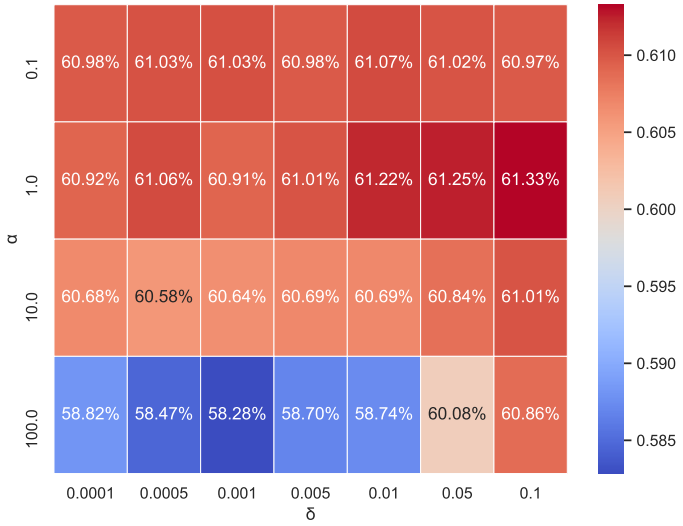


Fig. 6.2.: NDCG performance (averaged over 5 folds) of SmoothI on MQ2007's validation set with different α and δ.

Lastly, in order to have a fair comparison of the losses defined by the different methods in the context of modern neural end-to-end approaches, we used the same fully-connected feedforward neural network for all methods. It is composed of an input layer followed by batch normalization, a 1024-dimensional hidden layer with ReLU activation again followed by batch normalization, and a fully-connected output layer that provides a score for each document in the list. The mini-batch size is chosen as 128 (for learning to rank experiments), and the network parameters are optimized by the Adam optimizer [73], with an initial learning rate in the range of $\{10^{-2}, 10^{-3}\}$. Each model is trained with 50 epochs and the parameters (weights) leading to the lowest validation error are selected. The hyperparameters α and β for Approx are searched over $\{0.1, 1, 10, 10^2\}$. Similarly, for SmoothI, a line search was performed on the hyperparameter α in the range $\{0.1, 1, 10, 10^2\}$. The hyperparameter δ was simply set to 0.1 as we found in

[7] `https://github.com/usnistgov/trec_eval`

Tab. 6.2.: Learning to rank retrieval results. Mean test performance is calculated over 5 folds for MQ2007, MQ2008 and Web30K, and 5 random initializations for YLTR as no predefined folds are available. Best results are in bold and " † " indicates a model significantly worse than the best one according to a paired t-test with Bonferroni correction at 5%.

	P@1	P@5	NDCG@1	NDCG@5	NDCG	P@1	P@5	NDCG@1	NDCG@5	NDCG
			MQ2007					MQ2008		
ListNet	0.463	0.412†	0.420	0.422†	0.603†	0.392†	0.318†	0.339†	0.422†	0.514†
ListMLE	0.442†	0.397†	0.395†	0.405†	0.594†	0.415†	0.337†	0.365†	0.445†	0.526†
LambdaLoss	0.452†	0.403†	0.407†	0.415†	0.601†	0.441	0.337†	0.385	0.457†	0.540
ListAP	0.457†	0.405†	0.405†	0.414†	0.600†	0.420	0.330†	0.371	0.442†	0.532†
Approx	0.479	0.419	0.430	0.430	0.611	0.457	0.349	0.401	0.471	0.549
OT	0.451†	0.405†	0.406†	0.414†	0.602†	0.431	0.342†	0.382	0.461†	0.542
FastSort	0.461	0.405†	0.413†	0.417†	0.599†	0.430	0.332†	0.371	0.450†	0.537†
SoftSort	0.469	0.413†	0.425	0.426†	0.608	0.411†	0.335†	0.360†	0.449†	0.534†
SmoothI (ours)	**0.488**	**0.424**	**0.441**	**0.439**	**0.612**	**0.459**	**0.353**	**0.402**	**0.477**	**0.550**
			Web30K					YLTR		
ListNet	0.694†	0.649†	0.496†	0.483†	0.741†	0.858†	0.814†	0.726†	0.741†	0.857†
ListMLE	0.620†	0.544†	0.404†	0.383†	0.646†	**0.874**	**0.829**	0.724†	0.746	**0.859**
LambdaLoss	0.697†	0.617†	0.497†	0.466†	0.691†	0.868†	0.822†	0.731†	0.743†	0.854†
ListAP	0.715†	0.658†	0.503†	0.483†	0.733†	0.820†	0.768†	0.685†	0.686†	0.820†
Approx	0.767†	0.716	0.544†	0.523†	0.754	0.870	0.828	0.731†	0.745†	0.858
OT	0.682†	0.637†	0.459†	0.456†	0.729†	0.846†	0.793†	0.710†	0.719†	0.842†
FastSort	0.722†	0.660†	0.525†	0.494†	0.738†	0.857†	0.812†	0.724†	0.729†	0.851†
SoftSort	0.724†	0.669†	0.521†	0.500†	0.747†	0.861†	0.814†	0.729†	0.739†	0.854†
SmoothI (ours)	**0.776**	**0.717**	**0.552**	**0.530**	**0.754**	0.869†	0.826†	**0.735**	**0.748**	0.858

pilot experiments that this value consistently gave the best results on the validation set. This is illustrated in Fig. 6.2 which shows the NDCG@N scores on the validation set of MQ2007. We can observe that no matter what the choice of α is, setting $\delta = 0.1$ leads to the best results, or close to the best result for $\alpha = 0.1$. For SoftSort, the temperature is searched over $\{0.1, 1, 10, 10^2\}$. The regularization strength of FastSort and OT is searched over $\{10^{-2}, 0.1, 1, 10\}$. Our experiments were run on an Intel Xeon server with a Nvidia GTX 1080 Ti GPU.

6.4.2 Learning to Rank Results

In this section, we study the retrieval performance of the different learning to rank losses derived from SmoothI and baseline approaches. Table 6.2 presents the learning to rank results, averaged over 5 folds for MQ2007, MQ2008 and Web30K, each fold using a different random initialization,

and averaged over five random initializations for YLTR. We reported the significance using a paired Student t-test with Bonferroni correction at 5% significance level. For space reasons, we only show in Table 6.2 the mean results according to P@$\{1, 5\}$, NDCG@$\{1, 5, N\}$, and let the reader refer to Section 5 of the Supplementary Material for the P@10, MAP and NDCG@10 metrics as well as the standard errors around the mean. For LambdaLoss, Approx and SmoothI, Table 6.2 contains the best results obtained across the variants – optimizing different metrics – of each approach.

As one can notice, SmoothI is the best performing method on MQ2007, MQ2008 and Web30K. On MQ2007 and MQ2008, SmoothI outperforms all other methods for P@$\{1, 5\}$ and NDCG@$\{1, 5, N\}$. Approx is, on these collections, the second best method. On Web30K, SmoothI significantly outperforms all methods on P@1, NDCG@1, NDCG@5, and all methods but Approx on P@5 and NDCG. The results are more contrasted on YLTR. On the one hand, SmoothI and ListMLE are on par according to the NDCG-based metrics as they respectively obtained the best performance in terms of NDCG@$\{1, 5\}$ and NDCG@N, with significant differences only at cutoff 1. On the other hand, in terms of precision-based metrics, ListMLE outperformed all other approaches except Approx.

Turning to the listwise losses obtained from the differentiable sorting approaches (OT, FastSort, and SoftSort), we observe that these methods demonstrate competitive performance on the learning to rank task. SmoothI nonetheless outperformed all of these approaches, in particular on Web30K on which the differences are significant for all metrics. In summary, over all the collections, we conclude that SmoothI proves to be very competitive on learning to rank with respect to traditional listwise losses and differentiable sorting approaches.

Tab. 6.3.: Text-based retrieval results on Robust04: mean test performance ± standard error calculated over 5 folds. The best results are in bold and "†" indicates a model significantly worse than the best one according to a paired t-test at 5%.

	P@1	P@5	P@10	P@20	MAP
vanilla-BERT [102]	0.631±0.026	0.544±0.028	0.474±0.028†	0.396±0.019	0.236±0.006†
vanilla-BERT (Approx-NDCG@N loss)	**0.651±0.023**	0.529±0.020†	0.465±0.025†	0.392±0.021†	0.237±0.008†
vanilla-BERT (SmoothI-NDCG@N loss)	0.635±0.014	**0.562±0.025**	**0.494±0.024**	**0.407±0.019**	**0.245±0.007**
	NDCG@1	NDCG@5	NDCG@10	NDCG@20	NDCG
vanilla-BERT [102]	0.592±0.022	0.528±0.024	0.493±0.023†	0.464±0.020†	0.434±0.010
vanilla-BERT (Approx-NDCG@N loss)	**0.602±0.017**	0.521±0.017†	0.490±0.020†	0.462±0.018†	0.436±0.010
vanilla-BERT (SmoothI-NDCG@N loss)	0.601±0.010	**0.548±0.017**	**0.515±0.019**	**0.480±0.017**	**0.441±0.007**

As a complement to this experiment, we investigate in Section 6 of the Supplementary Material how the choice of the optimized metric influences SmoothI's performance. This study highlights that optimizing NDCG@N leads to the best performance according to any IR metric and thus constitutes an overall safe choice. We also discuss the efficiency of the different approaches in Section 7 of the Supplementary Material. Overall, all approaches but ListMLE, LambdaLoss and OT – which are significantly slower than the other approaches on different datasets – are comparable and scale reasonably well.

6.4.3 Experiments on Text-based IR

To further validate the efficacy of SmoothI, we conducted experiments on text-based information retrieval, *i.e.*, with raw texts as input. In particular, the task consists here in optimizing a given neural model to appropriately rank the documents for each query, where the documents and queries are raw texts. This differs from the previous sections which focus on feature-based learning to rank, *i.e.*, where each query-document pair is represented by a feature vector.

Experimental setup. The standard TREC Robust04 collection, which consists of 250 queries and 0.5M documents, is used here as the text-based IR collection. For queries, we used the keyword version which corresponds to the title fields of the TREC topics [26, 107]. We experimented with **vanilla BERT** as the neural ranking model, using the pretrained uncased BERT-base version. This model is at the core of recent state-of-the-art IR models [32, 26, 102, 83]. We make use here of the version proposed by [102], which is slightly better than the other ones.

Most text-based IR neural models based on BERT are trained with a pointwise or pairwise loss, and not a listwise loss [83, 102]. This is not really surprising as the calculation of the loss requires that the representations of all the documents to be ranked for a query hold together in memory, which can lead to a prohibitive memory cost for BERT if the list of documents associated to a query is large. To overcome this potential problem when using a listwise loss such as SmoothI, we computed the loss only on the documents of the training batch, where each batch contains two pairs of (relevant, non-relevant) documents associated to one query. For each query, one thus has a list of four documents, which are all fed to the vanilla BERT model as a list of query-document pairs. The input of the BERT models for each query-document pair is obtained by concatenating the [CLS] token, query tokens, the [SEP] token and document tokens. From BERT's output [CLS] vector, a dense layer generates the relevance score for the corresponding query-document pair. Following MacAvaney et al. [102], documents are truncated at 800 tokens in order to handle documents longer than the capacity of BERT. In this case, a document is split into two inputs and the [CLS] vectors from each split are averaged to get BERT's output [CLS] vector.

We use as baselines the standard vanilla BERT model [102] as well as its version with Approx-NDCG@N, which is the second best performing method

in our previous comparison. We compare both approaches to the vanilla BERT with SmoothI-NDCG@N, the best method overall in our previous comparison. All models are trained for 100 epochs using Adam optimizer with a learning rate of $2 \cdot 10^{-5}$ for BERT, as suggested in MacAvaney et al. [102], and 10^{-3} for the top dense layer, which is a common default value. As mentioned before, the batch size is set to four and gradient accumulation is used every eight steps [102]. We furthermore followed a five-fold cross validation protocol: the models are trained on the training set (corresponding to three folds), tuned on the validation set (one fold) with early stopping, and evaluated on the test set (the remaining fold). We use the standard re-ranking setting and re-rank the top-150 documents returned by BM25 [148]. The hyperparameters α and δ for SmoothI are set to 1.0 and 0.1 respectively. The hyperparameter α for Approx-NDCG@N is set to 1.0 (note that only one hyperparameter is needed for approximating NDCG@N with Approx as the second hyperparameter relates to the truncation function used for the truncated IR-metrics P@K and NDCG@K [139]). The random seed integer was set to 66 and we ran our experiments on an Intel Xeon server with a Nvidia GTX 1080 Ti GPU.

Results. Table 6.3 reports the text-based retrieval performance, averaged over 5 folds, of the standard vanilla BERT model and the two vanilla BERT models with the listwise losses Approx-NDCG@N and SmoothI-NDCG@N. The best results are in bold and "†" indicates a model significantly worse than the best one according to a paired t-test at 5%. Note that we observed no significant differences between Approx-NDCG@N and the pairwise hinge loss. In contrast, one can observe that vanilla BERT performs better when it is trained with SmoothI-NDCG@N and achieves the highest scores on all metrics but P@1 and NDCG@1 for which it is on par with the other approaches. The improvement over the original vanilla BERT model with the

pairwise hinge loss is in particular significant on P@10, MAP, NDCG@10 and NDCG@20. The improvement over Approx-NDCG@N is significant on P@5, P@10, P@20, MAP, NDCG@5, NDCG@10 and NDCG@20. Furthermore, the vanilla BERT model with SmoothI-NDCG@N achieves 0.480 on NDCG@20 on the TREC Robust04 collection, which is the best result this model has achieved on this collection to our knowledge [32, 102, 101].

6.5 Conclusion

We presented in this study a unified approach to build differentiable approximations of IR metrics (P@K, MAP and NDCG@K) on the basis of an approximation of the rank indicator function. We further showed that the errors associated with these approximations decrease exponentially with an inverse temperature-like hyperparameter that controls the quality of the approximations. We also illustrated the efficacy and efficiency of our approach on four standard collections based on learning to rank features, as well as on the popular TREC Robust04 text-based collection. All in all, our proposal, referred to as *SmoothI*, constitutes a tool for differentiable ranking that proved very competitive compared with previous approaches on several collections, either based on learning to rank or textual features.

We also want to stress that the approach we proposed is more general and can directly be applied to other losses, such as the K-NN loss studied in Grover et al. [48], and functions that are directly based on the rank indicator. Among such functions, we are particularly interested in the ranking function, which aims at ordering the documents in decreasing order of their scores, the sorting function, which aims at ordering the scores, and the position function, which aims at providing, for each document, its rank in the ordered

list of scores. We plan to study, on the basis of the development given in this chapter, differentiable approximations of these functions in a near future.

7 Conclusion

7.1 Conclusion

In this thesis, we presented several contributions to improve information retrieval (IR) models, including adapting IR models to long documents, to new domains, and proposing differentiable listwise loss functions based on approximate rank indicators. Chapter 1 briefly introduces the definition of information retrieval, the limitations and the contributions proposed. These contributions are present in three parts.

The first part is about adapting Transformer based IR models to long document retrieval. In chapter 2, a two-stage framework is proposed. For a long document, it is firstly segmented into short passages or blocks. Then these short blocks are judged the relevance information with the query using standard IR approaches like BM25 or learning based approaches. With this step, few top ranking blocks can be obtained, which are then combined to form a short document, or chosen to be input to an IR model like PARADE . This is reminiscent of the way humans assess the relevance of a document for a given query and this step enables lower memory usage and less noise information for better identifying the relevance information. The experiments on standard IR collections like Robust 04 and MQ 2007 show the our framework based on vanilla BERT IR model and PARADE model both overall significantly improves the base models perform better than other baseline models. We further conduct experiments on TREC 2019 DL Track collection.

The results also demonstrate a state-of-the-art level result. Among these proposed variants, the learning based block selecting mechanism performs in average better than the one based on standard IR methods, however, the later ones have lower latencies. In practice, one may choose to use the TF-IDF and BM25 versions of Vanilla BERT and PARADE if latency constraints are important. The choice between Vanilla BERT and PARADE variants, the latter being slightly better than the former on standard IR collections, and slightly worse on the TREC 2019 DL collection, depends on the collection considered.

In chapter 3, the idea of selecting key blocks is extended to late interaction models. We propose a similar late interaction based retrieval model like ColBERT for long document retrieval. The model not only should have the ability to generate token level representations for late interaction usage, but also should have the ability to generate passage level representations which are used as passage relevance information judgement. Given a query and with the representations obtained, top related passages are filtered with dot product operations on the passage level representations and their fine-grained relevance information are judged with token level representations. Then with a weighting scheme, a long document's relevance score is obtained. The model's common abilities to generate token level and passage level representations are trained by multi-task learning . Experimental results demonstrate the efficiency and efficacy of the proposed approach on such collections as TREC 2019.

The second part is related to domain adaptation for dense retrieval and conversational dense retrieval models. This is achieved by proposing a pseudo-relevance labeling approach to generate self-supervised training data on target domain datasets. As recent study BEIR shows , dense retrieval (DR) models that are trained on the source domain like MS MARCO, perform not

well when they are used on the target domain data. In chapter 4, we propose this approach to deal with the issue of dense retrieval. The pseudo-positive instances are identified by a T5-3B IR model trained on the source domain data MS MARCO, which re-ranks the BM25 lists of the target data according to the queries. The top re-ranked documents are viewed as positive. Different negative sampling strategies are investigated for obtaining better negative instances. Among them, the hard negative sampling strategy according to SimANS performs the best. Experiments demonstrate the proposed approach leads to improved performance of the DR model and the state-of-the-art query generation approach GPL when fine-tuned on the pseudo-relevance labeled data. In chapter 5, this approach is adapted to conversational search scenario. The conversational search task is more difficult as the queries in the conversation may be ambiguous and contain omissions and references. Besides, obtaining rewritten queries for these queries and obtaining relevance labels are difficult and expensive. In this chapter, we use an T5-Large model that is trained on a large conversational query rewritten dataset CANARD, to rewrite queries for the target datasets. Once the rewritten queries are obtained, pseudo-relevance labels can be obtained in a similar way as done for dense retrieval models. Experiments show when further training on the generate training data, conversational dense retrieval models can achieve better performances on the target domain data.

The third part is related to listwise loss functions. Chapter 6 introduces an approximation of the rank indicator function, which can be used in various ranking metrics and losses. This is achieved through a softmax-based approximation. With these differentiable loss functions, one is able to train neural IR models in a better listwise way. The efficacy and efficiency of our approach are illustrated on four standard collections based on learning to rank features, as well as on the popular TREC Robust04 text-based collection.

In conclusion, this thesis proposes several approaches to address the important limitations of information retrieval models especially the current deep learning era. We hope these contributions can give insights for research and building better real world information retrieval applications.

7.2 Future Direction

There are several potential directions for future research to enhance or build upon the ideas presented in this thesis.

In terms of adapting information retrieval models to long documents, there is potential for exploring more complex models or other modalities that incorporate the proposed selecting key block framework. For instance, this framework may assist in developing a model for long audio or video retrieval. Additionally, alternative negative sampling strategies and ways to accelerate the selection process through learned models can be further explored. As exemplified in Chapter 2 and Chapter 3, the training process relies on pairwise loss, with negatives either obtained through human labels or random sampling. Exploring novel approaches to extract more informative negatives could lead to further performance improvements. Moreover, in order to accelerate the selection process, it may be beneficial to incorporate dense representations into models such as KeyB(PARADEk)$_{BinB}$ and KeyB(PARADEk)$_{BinB2}$, which can facilitate fast retrieval of relevant passages from long documents.

In the context of domain adaptation for information retrieval, it is important to investigate domain shift issues not only for dense retrieval models, but also for interaction based and late interaction based models. We plan to study these issues and develop approaches to address them. Additionally, other

methods of incorporating in-domain knowledge into models are planed be explored to improve their performance, such as utilizing knowledge graphs. In specific domains, expert knowledge can be crucial in determining the relevance of information. This can benefit not only interaction based and late interaction based models, but also the pseudo-relevance labeling process for dense retrieval models, making it more reliable.

Furthermore, the thesis introduces a retrieval-based conversational system for information retrieval. While ChatGPT, a generation-based system, has gained considerable attention, the reliability of its generated contents remains a concern. One potential line of research involves investigating the combination of retrieval-based and generation-based systems to produce more reliable conversational responses. For instance, we plan to propose a two-stage conversational system: the first stage is a standard conversational search system, and the second stage is a generation-based system that rearranges the retrieved relevant passages into natural language content. This approach may help alleviate the issue of generating "fake" content that can arise with generation-based systems.

7.3 Papers Accepted during this Thesis

- Minghan Li, and Eric Gaussier. "KeyBLD: selecting key blocks with local pre-ranking for long document information retrieval." Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval. 2021.

- Thibaut Thonet, Yagmur Gizem Cinar, Eric Gaussier, Minghan Li, Jean-Michel Renders. "Listwise Learning to Rank Based on Approximate Rank Indi-

cators." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 36. No. 8. 2022.

- Minghan Li, and Eric Gaussier. "BERT-based Dense Intra-ranking and Contextualized Late Interaction via Multi-task Learning for Long Document Retrieval." Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval. 2022.

- Minghan Li, and Eric Gaussier. "Intra-document Block Pre-ranking for BERT-based Long Document Information Retrieval-Abstract." CIRCLE (Joint Conference of the Information Retrieval Communities in Europe). 2022.

- Minghan Li, Diana Nicoleta Popa, Johan Chagnon, Yagmur Gizem Cinar, Eric Gaussier. "The power of selecting key blocks with local pre-ranking for long document information retrieval." ACM Transactions on Information Systems 41.3 (2023): 1-35.

- Minghan Li, and Eric Gaussier. "Adaptation de domaine pour la recherche dense par annotation automatique." CORIA. 2023.

Bibliography

[1] Joshua Ainslie, Santiago Ontanon, Chris Alberti, et al. “ETC: Encoding Long and Structured Inputs in Transformers”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2020, pp. 268–284 (cit. on p. 18).

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. “Layer normalization”. In: *arXiv preprint arXiv:1607.06450* (2016) (cit. on p. 37).

[3] Iz Beltagy, Matthew E. Peters, and Arman Cohan. *Longformer: The Long-Document Transformer*. 2020. arXiv: 2004.05150 (cit. on pp. 18, 19, 64, 65, 71).

[4] Leonard Berrada, Andrew Zisserman, and M. Pawan Kumar. “Smooth Loss Functions for Deep Top-k Classification”. In: *Proceedings of the 6th International Conference on Learning Representations*. 2018 (cit. on p. 137).

[5] Gilles Blanchard, Aniket Anand Deshmukh, Ürun Dogan, Gyemin Lee, and Clayton Scott. “Domain generalization by marginal transfer learning”. In: *The Journal of Machine Learning Research* 22.1 (2021), pp. 46–100 (cit. on p. 92).

[6] Mathieu Blondel, Olivier Teboul, Quentin Berthet, and Josip Djolonga. “Fast Differentiable Sorting and Ranking”. In: *Proceedings of the 37th International Conference on Machine Learning*. 2020, pp. 950–959 (cit. on pp. 138, 146).

[7] Sebastian Bruch. “An Alternative Cross Entropy Loss for Learning-to-Rank”. In: *arXiv:1911.09798* (2019) (cit. on p. 137).

[8] Sebastian Bruch, Masrour Zoghi, Michael Bendersky, and Marc Najork. “Revisiting Approximate Metric Optimization in the Age of Deep Neural Networks”. In: *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2019 (cit. on pp. 137, 138, 146).

[9] Christopher J. C. Burges. *From RankNet to LambdaRank to LambdaMART: An Overview*. Tech. rep. Microsoft Research, 2010 (cit. on pp. 100, 128).

[10] Christopher J. C. Burges, Robert Ragno, and Quoc Viet Le. "Learning to Rank with Nonsmooth Cost Functions". In: *Proceedings of the 21st Annual Conference on Neural Information Processing Systems*. 2007 (cit. on p. 137).

[11] Christopher J. C. Burges, Krysta M. Svore, Paul N. Bennett, Andrzej Pastusiak, and Qiang Wu. "Learning to Rank Using an Ensemble of Lambda-Gradient Models." In: *Journal of Machine Learning Research* (2011) (cit. on p. 137).

[12] Stefan Büttcher and Charles LA Clarke. "A document-centric approach to static index pruning in text retrieval systems". In: *Proceedings of the 15th ACM international conference on Information and knowledge management*. 2006, pp. 182–189 (cit. on p. 20).

[13] Clément Calauzènes and Nicolas Usunier. "On ranking via sorting by estimated expected utility". In: *Proceedings of the 34th Annual Conference on Neural Information Processing Systems*. 2020 (cit. on pp. 136, 138).

[14] Clément Calauzènes, Nicolas Usunier, and Patrick Gallinari. "On the (Non-)existence of Convex, Calibrated Surrogate Losses for Ranking". In: *Proceedings of the 26th Annual Conference on Neural Information Processing Systems*. 2012, pp. 197–205 (cit. on pp. 136, 137).

[15] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. "Learning to Rank: From Pairwise Approach to Listwise Approach". In: *Proceedings of the 24th International Conference on Machine Learning*. 2007, pp. 129–136 (cit. on pp. 137, 145).

[16] Fabio M Carlucci, Antonio D'Innocente, Silvia Bucci, Barbara Caputo, and Tatiana Tommasi. "Domain generalization by solving jigsaw puzzles". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 2229–2238 (cit. on p. 92).

[17] Olivier Chapelle and Yi Chang. "Yahoo! Learning to Rank Challenge Overview". In: *Proceedings of the 2010 International Conference on Yahoo! Learning to Rank Challenge*. 2010, pp. 1–24 (cit. on p. 145).

[18] Olivier Chapelle and Mingrui Wu. "Gradient Descent Optimization of Smoothed Information Retrieval Metrics". In: *Information Retrieval* 13.3 (2010), pp. 216–235 (cit. on p. 138).

[19] Tianqi Chen, Thierry Moreau, Ziheng Jiang, et al. "TVM: an automated end-to-end optimizing compiler for deep learning". In: *Proceedings of the 13th USENIX conference on Operating Systems Design and Implementation*. 2018, pp. 579–594 (cit. on p. 19).

[20] Wei Chen, Tie-Yan Liu, Yanyan Lan, Zhiming Ma, and Hang Li. "Ranking Measures and Loss Functions in Learning to Rank". In: *Proceedings of the 23rd Annual Conference on Neural Information Processing Systems*. 2009, pp. 315–323 (cit. on p. 137).

[21] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. "Generating Long Sequences with Sparse Transformers". In: *CoRR* abs/1904.10509 (2019) (cit. on pp. 18, 64, 65).

[22] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M Voorhees. "Overview of the trec 2019 deep learning track". In: *arXiv preprint arXiv:2003.07820* (2020) (cit. on pp. 64, 67).

[23] J Shane Culpepper, Fernando Diaz, and Mark D Smucker. "Research frontiers in information retrieval: Report from the third strategic workshop on information retrieval in lorne (swirl 2018)". In: *ACM SIGIR Forum*. Vol. 52. 1. ACM New York, NY, USA. 2018, pp. 34–90 (cit. on p. 113).

[24] Marco Cuturi, Olivier Teboul, and Jean-Philippe Vert. "Differentiable Ranking and Sorting using Optimal Transport". In: *Proceedings of the 33rd Annual Conference on Neural Information Processing Systems*. 2019 (cit. on pp. 138, 146).

[25] Antonio D'Innocente and Barbara Caputo. "Domain generalization with domain-specific aggregation modules". In: *German Conference on Pattern Recognition*. Springer. 2018, pp. 187–198 (cit. on p. 92).

[26] Zhuyun Dai and Jamie Callan. "Deeper text understanding for IR with contextual neural language modeling". In: *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2019, pp. 985–988 (cit. on pp. 2, 12, 17, 46, 65, 69, 151).

[27] Zhuyun Dai, Chenyan Xiong, Jamie Callan, and Zhiyuan Liu. "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*. 2018, pp. 126–134 (cit. on pp. 69, 70).

[28] Zihang Dai, Zhilin Yang, Yiming Yang, et al. "Transformer-XL: Attentive Language Models beyond a Fixed-Length Context". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019 (cit. on pp. 17, 18).

[29] Jeffrey Dalton, Chenyan Xiong, and Jamie Callan. "TREC CAsT 2019: The conversational assistance track overview". In: *arXiv preprint arXiv:2003.13624* (2020) (cit. on pp. 120, 125).

[30] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *CoRR* abs/1810.04805 (2018) (cit. on pp. 2, 12, 16, 69).

[31] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "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)*. 2019, pp. 4171–4186 (cit. on p. 89).

[32] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "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)*. 2019, pp. 4171–4186 (cit. on pp. 151, 153).

[33] Ming Ding, Chang Zhou, Hongxia Yang, and Jie Tang. "CogLTX: Applying BERT to Long Texts". In: *Advances in Neural Information Processing Systems*. Vol. 33. 2020, pp. 12792–12804 (cit. on pp. 13, 19, 20, 23, 35).

[34] Yingjun Du, Jun Xu, Huan Xiong, et al. "Learning to learn with variational information bottleneck for domain generalization". In: *European Conference on Computer Vision*. Springer. 2020, pp. 200–216 (cit. on p. 92).

[35] Ahmed Elgohary, Denis Peskov, and Jordan Boyd-Graber. "Can You Unpack That? Learning to Rewrite Questions-in-Context". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 2019, pp. 5918–5924 (cit. on pp. 115, 119, 122, 123, 126).

[36] Yixing Fan, Jiafeng Guo, Yanyan Lan, et al. "Modeling diverse relevance patterns in ad-hoc retrieval". In: *The 41st international ACM SIGIR conference on research & development in information retrieval*. 2018, pp. 375–384 (cit. on pp. 2, 11).

[37] Hui Fang. "A re-examination of query expansion using lexical resources". In: *proceedings of ACL-08: HLT*. 2008, pp. 139–147 (cit. on p. 29).

[38] Hui Fang and ChengXiang Zhai. "Semantic term matching in axiomatic approaches to information retrieval". In: *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. 2006, pp. 115–122 (cit. on p. 29).

[39] Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. Bradford Books, 1998 (cit. on p. 29).

[40] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, et al. “Domain-adversarial training of neural networks”. In: *The journal of machine learning research* 17.1 (2016), pp. 2096–2030 (cit. on p. 92).

[41] Jianfeng Gao, Chenyan Xiong, Paul Bennett, and Nick Craswell. *Neural approaches to conversational information retrieval*. Vol. 44. Springer Nature, 2023 (cit. on p. 113).

[42] Luyu Gao and Jamie Callan. “Condenser: a Pre-training Architecture for Dense Retrieval”. In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 2021, pp. 981–993 (cit. on p. 92).

[43] Luyu Gao and Jamie Callan. “Unsupervised Corpus Aware Language Model Pre-training for Dense Passage Retrieval”. In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2022, pp. 2843–2853 (cit. on p. 97).

[44] Tianyu Gao, Xingcheng Yao, and Danqi Chen. “SimCSE: Simple Contrastive Learning of Sentence Embeddings”. In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 2021, pp. 6894–6910 (cit. on pp. 92, 104).

[45] Yanjie Gao, Yu Liu, Hongyu Zhang, et al. “Estimating gpu memory consumption of deep learning models”. In: *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2020, pp. 1342–1352 (cit. on p. 37).

[46] Quentin Grail, Julien Perez, and Eric Gaussier. “Globalizing BERT-based Transformer Architectures for Long Document Summarization”. In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*. Ed. by Paola Merlo, Jörg Tiedemann, and Reut Tsarfaty. Association for Computational Linguistics, 2021, pp. 1792–1810 (cit. on p. 17).

[47] Scott Gray, Alec Radford, and Diederik P Kingma. “Gpu kernels for block-sparse weights”. In: *arXiv preprint arXiv:1711.09224* 3 (2017) (cit. on p. 19).

[48] Aditya Grover, Eric Wang, Aaron Zweig, and Stefano Ermon. “Stochastic Optimization of Sorting Networks via Continuous Relaxations”. In: *Proceedings of the 7th International Conference on Learning Representations*. 2019 (cit. on pp. 139, 146, 153).

[49] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. "A deep relevance matching model for ad-hoc retrieval". In: *Proceedings of the 25th ACM international on conference on information and knowledge management*. 2016, pp. 55–64 (cit. on pp. 2, 11, 15, 17, 38, 64, 69, 70).

[50] Jiafeng Guo, Yixing Fan, Liang Pang, et al. "A deep look into neural ranking models for information retrieval". In: *Information Processing & Management* 57.6 (2020), p. 102067 (cit. on pp. 1, 2, 11, 14, 16, 70, 90).

[51] Sebastian Hofstätter, Sophia Althammer, Michael Schröder, Mete Sertkan, and Allan Hanbury. "Improving efficient neural ranking models with cross-architecture knowledge distillation". In: *arXiv preprint arXiv:2010.02666* (2020) (cit. on pp. 3, 69, 93).

[52] Sebastian Hofstätter and Allan Hanbury. "Evaluating Transformer-Kernel Models at TREC Deep Learning 2020". In: *TREC*. 2020 (cit. on p. 82).

[53] Sebastian Hofstätter, Bhaskar Mitra, Hamed Zamani, Nick Craswell, and Allan Hanbury. "Intra-Document Cascading: Learning to Select Passages for Neural Document Ranking". In: *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*. 2021, pp. 1349–1358 (cit. on pp. 20, 65, 66).

[54] Sebastian Hofstätter, Bhaskar Mitra, Hamed Zamani, Nick Craswell, and Allan Hanbury. "Intra-document cascading: Learning to select passages for neural document ranking". In: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2021, pp. 1349–1358 (cit. on pp. 70–73, 75, 77, 78).

[55] Sebastian Hofstätter, Hamed Zamani, Bhaskar Mitra, Nick Craswell, and Allan Hanbury. "Local Self-Attention over Long Text for Efficient Document Retrieval". In: *Proc. of SIGIR*. 2020 (cit. on pp. 71, 79, 80).

[56] Sebastian Hofstätter, Hamed Zamani, Bhaskar Mitra, Nick Craswell, and Allan Hanbury. "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*. 2020, pp. 2021–2024 (cit. on pp. 18, 65).

[57] Sebastian Hofstätter, Markus Zlabinger, and Allan Hanbury. "Interpretable & Time-Budget-Constrained Contextualization for Re-Ranking". In: *ECAI 2020*. IOS Press, 2020, pp. 513–520 (cit. on pp. 71, 79).

[58] Sebastian Hofstätter, Markus Zlabinger, and Allan Hanbury. "Interpretable & Time-Budget-Constrained Contextualization for Re-Ranking". In: *ECAI 2020 - 24th European Conference on Artificial Intelligence*. Vol. 325. IOS Press, 2020, pp. 513–520 (cit. on pp. 16–18, 65).

[59] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. "Convolutional neural network architectures for matching natural language sentences". In: *Advances in neural information processing systems*. 2014, pp. 2042–2050 (cit. on p. 15).

[60] Po-Sen Huang, Xiaodong He, Jianfeng Gao, et al. "Learning deep structured semantic models for web search using clickthrough data". In: *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. 2013, pp. 2333–2338 (cit. on pp. 2, 11, 15, 69, 71).

[61] Kai Hui, Andrew Yates, Klaus Berberich, and Gerard De Melo. "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*. 2018, pp. 279–287 (cit. on p. 16).

[62] Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. "PACRR: A Position-Aware Neural IR Model for Relevance Matching". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 2017, pp. 1049–1058 (cit. on pp. 2, 11, 16, 17, 65).

[63] Eric Jang, Shixiang Gu, and Ben Poole. "Categorical Reparameterization with Gumbel-Softmax". In: *Proceedings of the 5th International Conference on Learning Representations*. 2017 (cit. on p. 141).

[64] Kalervo Järvelin and Jaana Kekäläinen. "Cumulated Gain-based Evaluation of IR Techniques". In: *ACM Transactions on Information Systems* 20.4 (2002) (cit. on p. 140).

[65] Seogkyu Jeon, Kibeom Hong, Pilhyeon Lee, Jewook Lee, and Hyeran Byun. "Feature stylization and domain-aware contrastive learning for domain generalization". In: *Proceedings of the 29th ACM International Conference on Multimedia*. 2021, pp. 22–31 (cit. on p. 92).

[66] Jyun-Yu Jiang, Chenyan Xiong, Chia-Jung Lee, and Wei Wang. "Long Document Ranking with Query-Directed Sparse Transformer". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*. 2020, pp. 4594–4605 (cit. on pp. 4, 12, 19, 63–65, 71, 78).

[67] Chris Kamphuis, Arjen P de Vries, Leonid Boytsov, and Jimmy Lin. "Which BM25 do you mean? A large-scale reproducibility study of scoring variants". In: *European Conference on Information Retrieval*. Springer. 2020, pp. 28–34 (cit. on p. 33).

[68] Purushottam Kar, Harikrishna Narasimhan, and Prateek Jain. "Surrogate Functions for Maximizing Precision at the Top". In: *Proceedings of the 32nd International Conference on Machine Learning*. 2015, pp. 189–198 (cit. on p. 137).

[69] Constantinos Karouzos, Georgios Paraskevopoulos, and Alexandros Potamianos. "UDALM: Unsupervised Domain Adaptation through Language Modeling". In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2021, pp. 2579–2590 (cit. on pp. 91, 94, 104).

[70] Vladimir Karpukhin, Barlas Oguz, Sewon Min, et al. "Dense Passage Retrieval for Open-Domain Question Answering". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2020, pp. 6769–6781 (cit. on pp. 3, 94, 97).

[71] Alex Kendall, Yarin Gal, and Roberto Cipolla. "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 7482–7491 (cit. on pp. 5, 78).

[72] Omar Khattab and Matei Zaharia. "Colbert: Efficient and effective passage search via contextualized late interaction over bert". In: *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 2020, pp. 39–48 (cit. on pp. 3, 5, 69–71, 79, 84, 115, 119).

[73] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *Proceedings of the 3rd International Conference on Learning Representations*. 2015 (cit. on p. 147).

[74] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. *Reformer: The Efficient Transformer*. 2020. arXiv: 2001.04451 (cit. on p. 18).

[75] Weiwei Kong, Walid Krichene, Nicolas Mayoraz, Steffen Rendle, and Li Zhang. "Rankmax: An Adaptive Projection Alternative to the Softmax Function". In: *Proceedings of the 34th Annual Conference on Neural Information Processing Systems*. 2020 (cit. on p. 139).

[76] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105 (cit. on p. 16).

[77] Robert Krovetz. "Viewing morphology as an inference process". In: *Artificial intelligence* 118.1-2 (2000), pp. 277–294 (cit. on p. 47).

[78] Yanyan Lan, Jiafeng Guo, Xueqi Cheng, and Tie-Yan Liu. "Statistical Consistency of Ranking Methods in A Rank-Differentiable Probability Space". In: *Proceedings of the 26th Annual Conference on Neural Information Processing Systems*. 2012, pp. 1241–1249 (cit. on p. 137).

[79] Yanyan Lan, Tie-Yan Liu, Zhiming Ma, and Hang Li. "Generalization analysis of listwise learning-to-rank algorithms". In: *Proceedings of the 26th Annual International Conference on Machine Learning*. 2009, pp. 577–584 (cit. on p. 137).

[80] Yanyan Lan, Yadong Zhu, Jiafeng Guo, Shuzi Niu, and Xueqi Cheng. "Position-Aware ListMLE: A Sequential Learning Process for Ranking". In: *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence*. 2014, pp. 449–458 (cit. on p. 137).

[81] Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. "Latent Retrieval for Weakly Supervised Open Domain Question Answering". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019, pp. 6086–6096 (cit. on pp. 92, 104).

[82] Canjia Li, Yingfei Sun, Ben He, et al. "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*. 2018, pp. 4482–4491 (cit. on pp. 2, 11).

[83] Canjia Li, Andrew Yates, Sean MacAvaney, Ben He, and Yingfei Sun. "PARADE: Passage representation aggregation for document reranking". In: *arXiv preprint arXiv:2008.09093* (2020) (cit. on pp. 2–5, 12, 13, 17, 19, 36, 46, 47, 65, 69–71, 73, 151).

[84] Cheng Li, Mingyang Zhang, Michael Bendersky, et al. "Multi-view embedding-based synonyms for email search". In: *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2019, pp. 575–584 (cit. on p. 29).

[85] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy Hospedales. "Learning to generalize: Meta-learning for domain generalization". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018 (cit. on p. 92).

[86]Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. “Deeper, broader and artier domain generalization”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 5542–5550 (cit. on p. 92).

[87]Hang Li. “Learning to rank for information retrieval and natural language processing”. In: *Synthesis lectures on human language technologies* 4.1 (2011), pp. 1–113 (cit. on pp. 1, 11).

[88]Minghan Li and Eric Gaussier. “BERT-based Dense Intra-ranking and Contextualized Late Interaction via Multi-task Learning for Long Document Retrieval”. In: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2022, pp. 2347–2352 (cit. on pp. 21, 100, 128).

[89]Minghan Li and Eric Gaussier. “KeyBLD: Selecting Key Blocks with Local Pre-ranking for Long Document Information Retrieval”. In: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2021, pp. 2207–2211 (cit. on pp. 38, 48, 69–73, 80).

[90]Minghan Li, Diana Nicoleta Popa, Johan Chagnon, Yagmur Gizem Cinar, and Eric Gaussier. “The Power of Selecting Key Blocks with Local Pre-ranking for Long Document Information Retrieval”. In: *arXiv preprint arXiv:2111.09852* (2021) (cit. on pp. 70–72, 75, 76, 78).

[91]Davis Liang, Peng Xu, Siamak Shakeri, et al. “Embedding-based zero-shot retrieval through query generation”. In: *arXiv preprint arXiv:2009.10270* (2020) (cit. on p. 93).

[92]Lukas Liebel and Marco Körner. “Auxiliary tasks in multi-task learning”. In: *arXiv preprint arXiv:1805.06334* (2018) (cit. on pp. 5, 78).

[93]Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. “Contextualized Query Embeddings for Conversational Search”. In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 2021, pp. 1004–1015 (cit. on pp. 4, 6, 114, 115, 119, 120, 124, 127, 128).

[94]Chang Liu, Xinwei Sun, Jindong Wang, et al. “Learning causal semantic representation for out-of-distribution prediction”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 6155–6170 (cit. on p. 92).

[95]Tie-Yan Liu. *Learning to rank for information retrieval*. Springer, 2011, pp. I–XVII, 1–285 (cit. on pp. 1, 11, 135).

[96] Yinhan Liu, Myle Ott, Naman Goyal, et al. “Roberta: A robustly optimized bert pretraining approach”. In: *arXiv preprint arXiv:1907.11692* (2019) (cit. on pp. 18, 65).

[97] Ilya Loshchilov and Frank Hutter. “Decoupled weight decay regularization”. In: *arXiv preprint arXiv:1711.05101* (2017) (cit. on p. 125).

[98] Ilya Loshchilov and Frank Hutter. “SGDR: Stochastic Gradient Descent with Warm Restarts”. In: *International Conference on Learning Representations*. 2017 (cit. on p. 103).

[99] Jing Lu, Gustavo Hernández Ábrego, Ji Ma, Jianmo Ni, and Yinfei Yang. “Neural passage retrieval with improved negative contrast”. In: *arXiv preprint arXiv:2010.12523* (2020) (cit. on p. 48).

[100] Ji Ma, Ivan Korotkov, Yinfei Yang, Keith Hall, and Ryan McDonald. “Zero-shot Neural Passage Retrieval via Domain-targeted Synthetic Question Generation”. In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. 2021, pp. 1075–1088 (cit. on pp. 4, 90, 92, 104).

[101] Xinyu Ma, Jiafeng Guo, Ruqing Zhang, et al. “PROP: Pre-training with Representative Words Prediction for Ad-hoc Retrieval”. In: *arXiv:2010.10137* (2020) (cit. on p. 153).

[102] Sean MacAvaney, Andrew Yates, Arman Cohan, and Nazli Goharian. “CEDR: Contextualized embeddings for document ranking”. In: *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2019, pp. 1101–1104 (cit. on pp. 2, 12, 16, 38, 46, 48, 69, 150–153).

[103] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. “The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables”. In: *Proceedings of the 5th International Conference on Learning Representations*. 2017 (cit. on p. 141).

[104] Macedo Maia, Siegfried Handschuh, André Freitas, et al. “WWW’18 Open Challenge: Financial Opinion Mining and Question Answering”. In: *Companion Proceedings of the The Web Conference 2018*. WWW ’18. Lyon, France: International World Wide Web Conferences Steering Committee, 2018, pp. 1941–1942 (cit. on p. 100).

[105] Massimiliano Mancini, Samuel Rota Bulo, Barbara Caputo, and Elisa Ricci. “Best sources forward: domain generalization through source-specific nets”. In: *2018 25th IEEE international conference on image processing (ICIP)*. IEEE. 2018, pp. 1353–1357 (cit. on p. 92).

[106] Kelong Mao, Zhicheng Dou, Hongjin Qian, et al. "ConvTrans: Transforming Web Search Sessions for Conversational Dense Retrieval". In: *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. 2022, pp. 2935–2946 (cit. on pp. 114, 115, 118, 123).

[107] Ryan McDonald, George Brokos, and Ion Androutsopoulos. "Deep Relevance Ranking Using Enhanced Document-Query Interactions". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 2018, pp. 1849–1860 (cit. on p. 151).

[108] Ida Mele, Cristina Ioana Muntean, Franco Maria Nardini, et al. "Topic propagation in conversational search". In: *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 2020, pp. 2057–2060 (cit. on pp. 114, 116).

[109] Paulius Micikevicius, Sharan Narang, Jonah Alben, et al. "Mixed Precision Training". In: *International Conference on Learning Representations*. 2018 (cit. on pp. 47, 48, 80, 101).

[110] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. "Distributed Representations of Words and Phrases and their Compositionality". In: *Advances in Neural Information Processing Systems*. Ed. by C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger. Vol. 26. Curran Associates, Inc., 2013 (cit. on p. 15).

[111] Iurii Mokrii, Leonid Boytsov, and Pavel Braslavski. "A systematic evaluation of transfer learning and pseudo-labeling with bert-based ranking models". In: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2021, pp. 2081–2085 (cit. on p. 92).

[112] Maziar Moradi Fard, Thibaut Thonet, and Eric Gaussier. "Deep k-Means: Jointly Clustering with k-Means and Learning Representations". In: *arXiv:1806.10069* (2018) (cit. on p. 141).

[113] Saeid Motiian, Marco Piccirilli, Donald A Adjeroh, and Gianfranco Doretto. "Unified deep supervised domain adaptation and generalization". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 5715–5725 (cit. on p. 92).

[114] Cristina Ioana Muntean, Franco Maria Nardini, Raffaele Perego, Nicola Tonellotto, and Ophir Frieder. "Weighting Passages Enhances Accuracy". In: *ACM Transactions on Information Systems (TOIS)* 39.2 (2020), pp. 1–11 (cit. on p. 20).

[115] Hyeonseob Nam, HyunJae Lee, Jongchan Park, Wonjun Yoon, and Donggeun Yoo. “Reducing domain gap by reducing style bias”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 8690–8699 (cit. on p. 92).

[116] Tri Nguyen, Mir Rosenberg, Xia Song, et al. “MS MARCO: A Human Generated MAchine Reading COmprehension Dataset”. In: *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, December 9, 2016*. Ed. by Tarek Richard Besold, Antoine Bordes, Artur S. d’Avila Garcez, and Greg Wayne. Vol. 1773. CEUR Workshop Proceedings. CEUR-WS.org, 2016 (cit. on p. 100).

[117] Tri Nguyen, Mir Rosenberg, Xia Song, et al. “MS MARCO: A human generated machine reading comprehension dataset”. In: *CoCo@ NIPS*. 2016 (cit. on p. 47).

[118] Rodrigo Nogueira and Kyunghyun Cho. “Passage Re-ranking with BERT”. In: *arXiv preprint arXiv:1901.04085* (2019) (cit. on pp. 2, 3, 12, 16, 46, 69, 71, 79).

[119] Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. “Document Ranking with a Pretrained Sequence-to-Sequence Model”. In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. 2020, pp. 708–718 (cit. on pp. 91, 94, 95).

[120] Henrik Nottelmann and Norbert Fuhr. “From retrieval status values to probabilities of relevance for advanced IR applications”. In: *Information retrieval* 6.3 (2003), pp. 363–388 (cit. on p. 33).

[121] Kezban Dilek Onal, Ye Zhang, Ismail Sengor Altingovde, et al. “Neural information retrieval: At the end of the early years”. In: *Information Retrieval Journal* 21.2 (2018), pp. 111–182 (cit. on p. 15).

[122] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. “Neural Discrete Representation Learning”. In: *Proceedings of the 31st Annual Conference on Neural Information Processing Systems*. 2017, pp. 6309–6318 (cit. on p. 143).

[123] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. “Representation learning with contrastive predictive coding”. In: *arXiv preprint arXiv:1807.03748* (2018) (cit. on p. 18).

[124]Hamid Palangi, Li Deng, Yelong Shen, et al. “Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval”. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 24.4 (2016), pp. 694–707 (cit. on pp. 2, 11, 15).

[125]Liang Pang, Yanyan Lan, Jiafeng Guo, et al. “Deeprank: A new deep architecture for relevance ranking in information retrieval”. In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 2017, pp. 257–266 (cit. on pp. 2, 11, 15, 19, 20, 38, 44, 46, 48).

[126]Liang Pang, Yanyan Lan, Jiafeng Guo, et al. “Text matching as image recognition”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 30. 1. 2016 (cit. on pp. 2, 11).

[127]Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. “Bleu: a method for automatic evaluation of machine translation”. In: *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*. 2002, pp. 311–318 (cit. on p. 125).

[128]Rama Kumar Pasumarthi, Sebastian Bruch, Xuanhui Wang, et al. “TF-Ranking: Scalable TensorFlow Library for Learning-to-Rank”. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2019, pp. 2970–2978 (cit. on p. 146).

[129]Adam Paszke, Sam Gross, Francisco Massa, et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Proceedings of the 33rd Annual Conference on Neural Information Processing Systems*. 2019, pp. 8024–8035 (cit. on p. 146).

[130]Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, et al. “Scikit-learn: Machine learning in Python”. In: *the Journal of machine Learning research* 12 (2011), pp. 2825–2830 (cit. on p. 32).

[131]Jeffrey Pennington, Richard Socher, and Christopher D Manning. “Glove: Global vectors for word representation”. In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543 (cit. on pp. 46, 81).

[132]Tobias Plötz and Stefan Roth. “Neural Nearest Neighbors Networks”. In: *Proceedings of the 32nd Conference on Neural Information Processing Systems*. 2018 (cit. on pp. 138, 141).

[133]Przemyslaw Pobrotyn and Radoslaw Bialobrzeski. “NeuralNDCG: Direct Optimisation of a Ranking Metric via Differentiable Relaxation of Sorting”. In: *CoRR* abs/2102.07831 (2021) (cit. on p. 139).

[134] Aayush Prakash, Shaad Boochoon, Mark Brophy, et al. “Structured domain randomization: Bridging the reality gap by context-aware synthetic data”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 7249–7255 (cit. on p. 92).

[135] Sebastian Prillo and Julian Eisenschlos. “SoftSort: A Continuous Relaxation for the argsort Operator”. In: *Proceedings of the 37th International Conference on Machine Learning*. 2020, pp. 7793–7802 (cit. on pp. 139, 146).

[136] Fengchun Qiao, Long Zhao, and Xi Peng. “Learning to learn single domain generalization”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 12556–12565 (cit. on p. 92).

[137] Tao Qin and Tie-Yan Liu. “Introducing LETOR 4.0 Datasets”. In: *arXiv:1306.2597* (2013) (cit. on p. 145).

[138] Tao Qin and Tie-Yan Liu. “Introducing LETOR 4.0 datasets”. In: *arXiv preprint arXiv:1306.2597* (2013) (cit. on pp. 22, 58).

[139] Tao Qin, Tie-Yan Liu, and Hang Li. “A General Approximation Framework for Direct Optimization of Information Retrieval Measures”. In: *Information Retrieval, 13(4)* (2010) (cit. on pp. 138, 146, 152).

[140] Tao Qin, Xu-Dong Zhang, Ming-Feng Tsai, et al. “Query-level loss functions for information retrieval”. In: *Information Processing & Management* 44.2 (2008), pp. 838–855 (cit. on p. 137).

[141] Colin Raffel, Noam Shazeer, Adam Roberts, et al. “Exploring the limits of transfer learning with a unified text-to-text transformer.” In: *J. Mach. Learn. Res.* 21.140 (2020), pp. 1–67 (cit. on pp. 5, 6, 93, 95, 115, 121).

[142] Mohammad Mahfujur Rahman, Clinton Fookes, Mahsa Baktashmotlagh, and Sridha Sridharan. “Multi-component image translation for deep domain generalization”. In: *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2019, pp. 579–588 (cit. on p. 92).

[143] Pradeep Ravikumar, Ambuj Tewari, and Eunho Yang. “On NDCG Consistency of Listwise Ranking Methods”. In: *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*. 2011, pp. 618–626 (cit. on p. 137).

[144] Nils Reimers and Iryna Gurevych. “Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks”. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 2019, pp. 3982–3992 (cit. on pp. 3, 27, 69, 71, 83, 102).

[145] Gary Ren, Xiaochuan Ni, Manish Malik, and Qifa Ke. “Conversational query understanding using sequence to sequence modeling”. In: *Proceedings of the 2018 World Wide Web Conference*. 2018, pp. 1715–1724 (cit. on pp. 114, 117).

[146] Jérôme Revaud, Jon Almazán, Rafael S. Rezende, and César Roberto de Souza. “Learning With Average Precision: Training Image Retrieval With a Listwise Loss”. In: *Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision*. 2019, pp. 5106–5115 (cit. on pp. 138, 145).

[147] Stephen Robertson and Hugo Zaragoza. *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc, 2009 (cit. on pp. 1, 32).

[148] Stephen E Robertson and Steve Walker. “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*. 1994, pp. 232–241 (cit. on p. 152).

[149] Stephen E. Robertson and Steve Walker. “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, 3-6 July 1994 (Special Issue of the SIGIR Forum)*. Ed. by W. Bruce Croft and C. J. van Rijsbergen. 1994 (cit. on p. 28).

[150] Stephen E. Robertson and Hugo Zaragoza. “The Probabilistic Relevance Framework: BM25 and Beyond”. In: *Found. Trends Inf. Retr.* 3.4 (2009), pp. 333–389 (cit. on pp. 32, 76, 79, 89).

[151] Gerard Salton, Anita Wong, and Chung-Shu Yang. “A vector space model for automatic indexing”. In: *Communications of the ACM* 18.11 (1975), pp. 613–620 (cit. on p. 1).

[152] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. “DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter”. In: *arXiv preprint arXiv:1910.01108* (2019) (cit. on p. 101).

[153] Shiv Shankar, Vihari Piratla, Soumen Chakrabarti, et al. “Generalizing Across Domains via Cross-Gradient Training”. In: *International Conference on Learning Representations*. 2018 (cit. on p. 92).

[154] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. "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*. 2014, pp. 101–110 (cit. on pp. 2, 11, 15).

[155] Ian Steinwart. "How to compare Different Loss Functions and Their Risks". In: *Constructive Approximation* 26.2 (2007), pp. 225–287 (cit. on p. 137).

[156] Michael Taylor, John Guiver, Stephen Robertson, and Tom Minka. "SoftRank: Optimizing Non-Smooth Rank Metrics". In: *Proceedings of the 1st International Conference on Web Search and Data Mining*. 2008, pp. 77–86 (cit. on p. 138).

[157] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. "BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models". In: *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*. 2021 (cit. on pp. 4, 90, 91, 93, 100, 104).

[158] James Thorne, Andreas Vlachos, Oana Cocarascu, Christos Christodoulopoulos, and Arpit Mittal. "The Fact Extraction and VERification (FEVER) Shared Task". In: *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*. 2018, pp. 1–9 (cit. on p. 18).

[159] Josh Tobin, Rachel Fong, Alex Ray, et al. "Domain randomization for transferring deep neural networks from simulation to the real world". In: *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE. 2017, pp. 23–30 (cit. on p. 92).

[160] George Tsatsaronis, Georgios Balikas, Prodromos Malakasiotis, et al. "An overview of the BIOASQ large-scale biomedical semantic indexing and question answering competition". In: *BMC bioinformatics* 16.1 (2015), p. 138 (cit. on p. 101).

[161] Peter D Turney. "Mining the web for synonyms: PMI-IR versus LSA on TOEFL". In: *European conference on machine learning*. Springer. 2001, pp. 491–502 (cit. on p. 29).

[162] Svitlana Vakulenko, Shayne Longpre, Zhucheng Tu, and Raviteja Anantha. "Question rewriting for conversational question answering". In: *Proceedings of the 14th ACM international conference on web search and data mining*. 2021, pp. 355–363 (cit. on pp. 114, 117).

[163] Hamed Valizadegan, Rong Jin, Ruofei Zhang, and Jianchang Mao. "Learning to Rank by Optimizing NDCG Measure". In: *Proceedings of the 23rd Annual Conference on Neural Information Processing Systems*. 2009 (cit. on p. 137).

[164] Christophe Van Gysel and Maarten de Rijke. "Pytrec_eval: An Extremely Fast Python Interface to trec_eval". In: *Proceedings of the 41st International ACM SIGIR conference on Research and Development in Information Retrieval*. 2018 (cit. on p. 147).

[165] Christophe Van Gysel and Maarten de Rijke. "Pytrec_eval: An extremely fast python interface to trec_eval". In: *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 2018, pp. 873–876 (cit. on p. 49).

[166] Ashish Vaswani, Noam Shazeer, Niki Parmar, et al. "Attention is all you need". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 2017, pp. 6000–6010 (cit. on pp. 2, 12, 16, 37, 69, 89).

[167] Riccardo Volpi, Hongseok Namkoong, Ozan Sener, et al. "Generalizing to unseen domains via adversarial data augmentation". In: *Advances in neural information processing systems* 31 (2018) (cit. on p. 92).

[168] Ellen Voorhees. "Overview of the TREC 2004 Robust Retrieval Track". en. In: Special Publication (NIST SP), National Institute of Standards and Technology, Gaithersburg, MD, 2005-08-01 2005 (cit. on p. 101).

[169] Shengxian Wan, Yanyan Lan, Jun Xu, et al. "Match-SRNN: Modeling the Recursive Matching Structure with Spatial RNN". In: *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI*. 2016, pp. 2922–2928 (cit. on p. 16).

[170] Jindong Wang, Cuiling Lan, Chang Liu, et al. "Generalizing to unseen domains: A survey on domain generalization". In: *IEEE Transactions on Knowledge and Data Engineering* (2022) (cit. on pp. 90, 92).

[171] Kexin Wang, Nils Reimers, and Iryna Gurevych. "TSDAE: Using Transformer-based Sequential Denoising Auto-Encoderfor Unsupervised Sentence Embedding Learning". In: *Findings of the Association for Computational Linguistics: EMNLP 2021*. 2021, pp. 671–688 (cit. on pp. 92, 104).

[172] Kexin Wang, Nandan Thakur, Nils Reimers, and Iryna Gurevych. "GPL: Generative Pseudo Labeling for Unsupervised Domain Adaptation of Dense Retrieval". In: *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Seattle, United States: Association for Computational Linguistics, July 2022, pp. 2345–2360 (cit. on pp. 6, 90–94, 101–104, 107, 125).

[173] M. Wang and W. Deng. "Deep visual domain adaptation: a survey". In: *Neurocomputing* (2018) (cit. on p. 90).

[174] Xuanhui Wang, Cheng Li, Nadav Golbandi, Michael Bendersky, and Marc Najork. "The LambdaLoss Framework for Ranking Metric Optimization". In: *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 2018 (cit. on pp. 137, 145).

[175] Yining Wang, Liwei Wang, Yuanzhi Li, Di He, and Tie-Yan Liu. "A theoretical analysis of NDCG type ranking measures". In: *Conference on learning theory*. PMLR. 2013, pp. 25–54 (cit. on p. 4).

[176] Thomas Wolf, Lysandre Debut, Victor Sanh, et al. "HuggingFace's Transformers: State-of-the-art natural language processing". In: *arXiv preprint arXiv:1910.03771* (2019) (cit. on p. 47).

[177] Ho Chung Wu, Robert WP Luk, Kam-Fai Wong, and KL Kwok. "A retrospective study of a hybrid document-context based retrieval model". In: *Information processing & management* 43.5 (2007), pp. 1308–1331 (cit. on pp. 13, 20).

[178] Mingrui Wu, Yi Chang, Zhaohui Zheng, and Hongyuan Zha. "Smoothing DCG for Learning to Rank: A Novel Approach Using Smoothed Hinge Functions". In: *Proceedings of the 18th ACM Conference on Information and Knowledge Management*. 2009, pp. 1923–1926 (cit. on p. 138).

[179] Fen Xia, Tie-Yan Liu, and Hang Li. "Statistical Consistency of Top-k Ranking". In: *Proceedings of the 23rd Annual Conference on Neural Information Processing Systems*. 2009, pp. 2098–2106 (cit. on p. 137).

[180] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. "Listwise approach to learning to rank: theory and algorithm". In: *Proceedings of the 25th International Conference on Machine Learning*. 2008 (cit. on pp. 137, 145).

[181] Ji Xin, Chenyan Xiong, Ashwin Srinivasan, et al. "Zero-Shot Dense Retrieval with Momentum Adversarial Domain Invariant Representations". In: *Findings of the Association for Computational Linguistics: ACL 2022*. 2022, pp. 4008–4020 (cit. on pp. 3, 4, 91, 93, 94, 104).

[182] Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. "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*. 2017, pp. 55–64 (cit. on pp. 2, 11, 15, 17, 69, 70).

[183] Lee Xiong, Chenyan Xiong, Ye Li, et al. "Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval". In: *International Conference on Learning Representations*. 2020 (cit. on pp. 3, 71, 93, 94, 98, 104).

[184] Jun Xu, Tie-Yan Liu, Min Lu, Hang Li, and Wei-Ying Ma. "Directly optimizing evaluation measures in learning to rank". In: *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2008, pp. 107–114 (cit. on p. 137).

[185] Peilin Yang, Hui Fang, and Jimmy Lin. "Anserini: Reproducible ranking baselines using Lucene". In: *Journal of Data and Information Quality (JDIQ)* 10.4 (2018), pp. 1–20 (cit. on pp. 22, 45, 79, 80, 103, 127).

[186] Zhilin Yang, Peng Qi, Saizheng Zhang, et al. "HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 2018, pp. 2369–2380 (cit. on p. 18).

[187] Zhuliang Yao, Shijie Cao, Wencong Xiao, Chen Zhang, and Lanshun Nie. "Balanced sparsity for efficient dnn inference on gpu". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 5676–5683 (cit. on p. 19).

[188] Haitao Yu, Adam Jatowt, Hideo Joho, et al. "WassRank: Listwise Document Ranking Using Optimal Transport Theory". In: *Proceedings of the 12th ACM International Conference on Web Search and Data Mining*. 2019, pp. 24–32 (cit. on p. 138).

[189] Shi Yu, Jiahua Liu, Jingqin Yang, et al. "Few-shot generative conversational query rewriting". In: *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 2020, pp. 1933–1936 (cit. on p. 114).

[190] Shi Yu, Zhenghao Liu, Chenyan Xiong, Tao Feng, and Zhiyuan Liu. "Few-shot conversational dense retrieval". In: *Proceedings of the 44th International ACM SIGIR Conference on research and development in information retrieval*. 2021, pp. 829–838 (cit. on pp. 4, 6, 114, 115, 118–120, 127, 128, 130).

[191] Manzil Zaheer, Guru Guruganesh, Avinava Dubey, et al. *Big Bird: Transformers for Longer Sequences*. 2021. arXiv: 2007.14062 (cit. on pp. 18, 19).

[192] Hamed Zamani, Johanne R. Trippas, Jeff Dalton, and Filip Radlinski. *Conversational Information Seeking*. 2023. arXiv: 2201.08808 [cs.IR] (cit. on p. 113).

[193] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. "mixup: Beyond Empirical Risk Minimization". In: *International Conference on Learning Representations*. 2018 (cit. on p. 92).

[194] Chen Zhao, Chenyan Xiong, Corby Rosset, et al. "Transformer-XH: Multi-Evidence Reasoning with eXtra Hop Attention". In: *International Conference on Learning Representations*. 2020 (cit. on pp. 18, 64, 65).

[195] Kun Zhou, Yeyun Gong, Xiao Liu, et al. "SimANS: Simple Ambiguous Negatives Sampling for Dense Text Retrieval". In: *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: Industry Track*. Abu Dhabi, UAE: Association for Computational Linguistics, Dec. 2022, pp. 548–559 (cit. on pp. 91, 94, 98, 99, 120, 124).

List of Figures

List of Tables