PARADE: Passage Representation Aggregation for Document Reranking

CANJIA LI[*], University of Chinese Academy of Sciences, China and Max Planck Institute for Informatics
ANDREW YATES[†], University of Amsterdam, Netherlands and Max Planck Institute for Informatics
SEAN MACAVANEY, University of Glasgow, United Kingdom
BEN HE[†], University of Chinese Academy of Sciences, China
YINGFEI SUN[†], University of Chinese Academy of Sciences, China

Pre-trained transformer models, such as BERT and T5, have shown to be highly effective at ad-hoc passage and document ranking. Due to the inherent sequence length limits of these models, they need to process document passages one at a time rather than processing the entire document sequence at once. Although several approaches for aggregating passage-level signals into a document-level relevance score have been proposed, there has yet to be an extensive comparison of these techniques. In this work, we explore strategies for aggregating relevance signals from a document's passages into a final ranking score. We find that passage representation aggregation techniques can significantly improve over score aggregation techniques proposed in prior work, such as taking the maximum passage score. We call this new approach PARADE. In particular, PARADE can significantly improve results on collections with broad information needs where relevance signals can be spread throughout the document (such as TREC Robust04 and GOV2). Meanwhile, less complex aggregation techniques may work better on collections with an information need that can often be pinpointed to a single passage (such as TREC DL and TREC Genomics). We also conduct efficiency analyses and highlight several strategies for improving transformer-based aggregation.

CCS Concepts: • **Information systems → Document representation**; **Retrieval models and ranking**; **Language models**.

Additional Key Words and Phrases: document reranking, passage representation aggregation, pre-trained language models

ACM Reference Format:
Canjia Li, Andrew Yates, Sean MacAvaney, Ben He, and Yingfei Sun. 2023. PARADE: Passage Representation Aggregation for Document Reranking. 1, 1 (May 2023), 26 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

[*]This work was partially conducted while the author was an intern at the Max Planck Institute for Informatics.
[†]Corresponding author

Authors' addresses: Canjia Li, licanjia17@mails.ucas.ac.cn, University of Chinese Academy of Sciences, Beijing, China, Max Planck Institute for Informatics; Andrew Yates, a.c.yates@uva.nl, University of Amsterdam, Amsterdam, Netherlands, Max Planck Institute for Informatics; Sean MacAvaney, sean.macavaney@glasgow.ac.uk, University of Glasgow, Glasgow, United Kingdom; Ben He, benhe@ucas.ac.cn, University of Chinese Academy of Sciences, Beijing, China; Yingfei Sun, yfsun@ucas.ac.cn, University of Chinese Academy of Sciences, Beijing, China.

1 INTRODUCTION

Pre-trained language models (PLMs), such as BERT [22], ELECTRA [15] and T5 [80], have achieved state-of-the-art results on standard ad-hoc retrieval benchmarks. In this context, the success of PLMs mainly relies on learning contextualized representations of input sequences using the transformer encoder architecture [94]. The transformer uses a self-attention mechanism whose computational complexity is quadratic with respect to the input sequence's length. Therefore, PLMs generally limit the sequence's length (e.g., to 512 tokens) to reduce computational costs. Consequently, when applied to the ad-hoc ranking task, PLMs are commonly used to predict the relevance of passages or individual sentences [20, 106]. The max or k-max passage scores (e.g., top 3) are then aggregated to produce a document relevance score. Such score aggregation approaches have achieved state-of-the-art results on a variety of ad-hoc retrieval benchmarks.

Documents are often much longer than a single passage, however, and intuitively there are many types of relevance signals that can only be observed in a full document. For example, the *Verbosity Hypothesis* [82] states that relevant excerpts can appear at different positions in a document. It is not necessarily possible to account for all such excerpts by considering only the top passages. Similarly, the ordering of passages itself may affect a document's relevance; a document with relevant information at the beginning is intuitively more useful than a document with the information at the end [9, 41]. Empirical studies support the importance of full-document signals. Wu et al. study how passage-level relevance labels correspond to document-level labels, finding that more relevant documents also contain a higher number of relevant passages [98]. Additionally, experiments suggest that aggregating passage-level relevance scores to predict the document's relevance score outperforms the common practice of using the maximum passage score (e.g., [1, 5, 23]).

On the other hand, the amount of non-relevant information in a document can also be a signal, because relevant excerpts would make up a large fraction of an ideal document. IR axioms encode this idea in the first length normalization constraint (LNC1), which states that adding non-relevant information to a document should decrease its score [24]. Considering a full document as input has the potential to incorporate signals like these. Furthermore, from the perspective of training a supervised ranking model, the common practice of applying document-level relevance labels to individual passages is undesirable, because it introduces unnecessary noise into the training process.

In this work, we provide an extensive study on neural techniques for aggregating passage-level signals into document scores. We study how PLMs like BERT and ELECTRA can be applied to the ad-hoc document ranking task while preserving many document-level signals. We move beyond simple passage *score* aggregation strategies (such as Birch [106]) and study passage *representation* aggregation (PARADE). We find that aggregation over passage representations often outperforms passage score aggregation. We also confirm that hierarchical aggregation architectures like CNN and Transformer are more capable of capturing the diverse relevance signals from the passages within a document, which results in a more effective ranking model.

Since the utilization of the full-text increases memory requirements, we investigate using knowledge distillation to create smaller, more efficient passage representation aggregation models that remain effective. In summary, our contributions are:

- The formalization of passage *score* and *representation* aggregation strategies, showing how both can be trained end-to-end,
- A thorough comparison of passage aggregation strategies on a variety of benchmark datasets, demonstrating the value of passage representation aggregation,
- An ablation study of the contributions of different components that improves the representation aggregation effectiveness,

- An analysis of how to reduce the computational cost of PARADE by decreasing the model size,
- An analysis of how the effectiveness of PARADE is influenced by different hyper-parameters,
- An analysis into dataset characteristics that can influence which aggregation strategies are most effective on certain benchmarks
- An analysis of how reranking effectiveness is influenced by the first-stage retriever.

The rest of the paper is organized as follows: We summarize the related work in Section 2. We formalize our method in Section 3. The details of evaluation results are described in Section 4. A few in-depth analysis to the research questions are conducted in Section 5. Finally, we conclude the paper in Section 6.

2 RELATED WORK

We review four lines of related research related to our study.

Contextualized Language Models for IR. Many pre-BERT neural ranking models have been proposed, such as DSSM [39], DRMM [28], Duet [71], (Co-)PACRR [40, 41], and (Conv-)KNRM [21, 99]. However, compared to Transformer-based models, pre-BERT models often obtain relatively small improvements on TREC benchmarks [103], due to their relatively small model size (except the word embeddings). Benefiting from embedding contextualization, BERT-based IR models have been shown to be superior to these prior neural IR models. We briefly summarize related approaches here and refer the reader to a survey on transformers for text ranking by Lin et al. [57] for further details. These approaches use BERT as a relevance classifier in a cross-encoder configuration (i.e., BERT takes both a query and a document as input). Nogueira et al. first adopted BERT to passage reranking tasks [75] using BERT's `[CLS]` vector. Birch [106] and BERT-MaxP [20] explore using sentence-level and passage-level relevance scores from BERT for document reranking, respectively. CEDR proposed a joint approach that combines BERT's outputs with existing neural IR models and handled passage aggregation via a representation aggregation technique (averaging) [67]. In this work, we further explore techniques for passage aggregation and consider an improved CEDR variant as a baseline. We focus on the under-explored direction of representation aggregation by employing more sophisticated strategies, including using CNNs and Transformers.

On the pre-training side, researchers have designed pre-training objectives tailored for IR. PROP proposed a novel representative words prediction training task [63], while B-PROP further improves upon PROP by replacing PROP's classical unigram language model with a more powerful BERT-based contextual language model [64]. Other researchers trade off PLM effectiveness for efficiency by utilizing the PLM to improve document indexing [19, 77], pre-computing intermediate Transformer representations [27, 42, 47, 65], selecting query-aware key blocks within a document for input squeezing [48, 55], using the PLM to build sparse representations [25, 56, 66, 68, 73, 112, 114], weighting offline pseudo-query and document relevance [11], or reducing the number of Transformer layers [34, 36, 72].

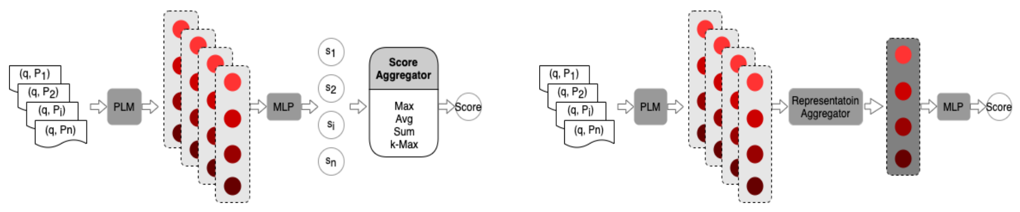
Several works have investigated approaches for improving the Transformer's efficiency by reducing the computational complexity of its attention module, e.g., Sparse Transformer [14], Big Bird [107] and Longformer [4]. QDS-Transformer tailors Longformer to the ranking task with query-directed sparse attention [43]. We note that representation-based passage aggregation is more effective than increasing the input text size using the aforementioned models, but representation aggregation could be used in conjunction with such models.

Passage-based Document Retrieval. Callan first experimented with paragraph-based and window-based methods of defining passages [8]. Several works drive passage-based document retrieval in the language modeling context [5, 59], indexing context [58], and learning to rank context [88]. In the realm of neural networks, HiNT demonstrated that aggregating representations of passage-level relevance can perform well in the context of pre-BERT models [23]. Wu et al. explicitly modeled the importance of passages based on position decay, passage length, length with position decay, exact match, etc [98]. They proposed a model that considers passage-level representations of relevance in order to predict the passage-level cumulative gain of each passage [97]. In this approach the final passage's cumulative gain can be used as the document-level cumulative gain. Our approaches share some similarities, but differ in that Wu et al. use passage-level labels to train their model and perform passage representation aggregation using a LSTM.

Besides passage-based methods, field-based methods use aggregation techniques across heterogenous document fields to improve ranking effectiveness. NRM-F [108] and DuetMF [70] aggregate representations from different document fields (e.g., title, body, anchor text). Passage aggregation methods can be considered as a special case of field aggregation approaches, where the fields are homogeneous rather than containing different types of information (e.g., title vs. body). Others have investigated sophisticated evidence aggregation approaches [111, 113]. Aggregating across different document fields [70, 108] or aggregating evidence from outer sources [111, 113] is interesting and has the potential to improve the reranking effectiveness. However, such explorations are orthogonal to the current work on aggregating *passage* representations.

Representation Aggregation Approaches for NLP. Representation learning has been shown to be powerful in many NLP tasks [6, 61]. For pre-trained language models, a text representation is learned by feeding the PLM with a formatted text like `[CLS] TextA [SEP]` or `[CLS] TextA [SEP] TextB [SEP]`. The vector representation of the prepended `[CLS]` token in the last layer is then regarded as either a text overall representation or a text relationship representation. Such representations can also be aggregated for tasks that requires reasoning from multiple scopes of evidence. Gear aggregates the claim-evidence representations by max aggregator, mean aggregator, or attention aggregator for fact checking [113]. Transformer-XH uses extra hop attention that bears not only in-sequence but also inter-sequence information sharing [111]. The learned representation is then adopted for either question answering or fact verification tasks. Several lines of work have explored hierarchical representations for document classification and summarization, including transformer-based approaches [60, 96, 104, 109]. In the context of ranking, SMITH [101], a long-to-long text matching model, learns a document representation with hierarchical sentence representation aggregation, which shares some similarities with our work. Rather than learning independent document (and query) representations, SMITH is a bi-encoder approach that learns separate representations for each. While such approaches have efficiency advantages, current bi-encoders do not match the effectiveness of cross-encoders, which are the focus of our work [57]. Furthermore, Luan et al. [62] demonstrate from both a theoretical and empirical perspective that the size of a document's representation must increase with its length in order to maintain an accurate ranking.

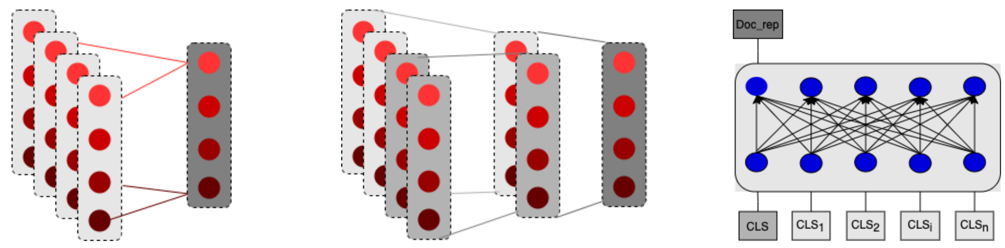
Knowledge Distillation. Knowledge distillation is the process of transferring knowledge from a large model to a smaller student model [2, 31]. Ideally, the student model performs well while consisting of fewer parameters. One line of research investigates the use of specific distilling objectives for intermediate layers in the BERT model [44, 90], which is shown to be effective in the IR context [10]. Turc et al. pre-train a family of compact BERT models and explore transferring task knowledge from large fine-tuned models [93]. Tang et al. distill knowledge from the BERT model into Bi-LSTM [92]. Tahami et al. propose a new cross-encoder architecture and transfer knowledge from this model to a bi-encoder model for fast retrieval [91]. Hofstätter et al. also proposes a cross-architecture knowledge distillation framework using a Margin Mean Squared Error loss in a pairwise training manner [32]. We demonstrate the approach in [91, 92] can be applied to



(a) Previous approaches: score aggregation

(b) PARADE: representation aggregation

Fig. 1. Comparison between score aggregation approaches and PARADE's representation aggregation mechanism.



(a) Max, Avg, Sum, and Attn Aggregators

(b) CNN Aggregator

(c) Transformer Aggregator

Fig. 2. Representation aggregators take passages' [CLS] representations as inputs and output a final document representation. Different from the other approaches, both PARADE–CNN and PARADE–Transformer employ a hierarchical architecture to aggregate the representations.

representation aggregation approaches to improve efficiency without substantial reductions in effectiveness.

3 METHOD

In this section, we formalize approaches for aggregating passage representations into document ranking scores. We make the distinction between the passage *score* aggregation techniques explored in prior work with passage *representation* aggregation (PARADE) techniques, which have received less attention in the context of document ranking. Given a query q and a document D, a ranking method aims to generate a relevance score $rel(q, D)$ that estimates to what degree document D satisfies the query q. As described in the following sections, we perform this relevance estimation by aggregating passage-level relevance representations into a document-level representation, which is then used to produce a relevance score.

3.1 Creating Passage Relevance Representations

As introduced in Section 1, a long document cannot be considered directly by the BERT model[1] due to its fixed sequence length limitation. As in prior work [8, 20], we split a document into passages that can be handled by BERT individually. To do so, a sliding window of 225 tokens is applied to the document with a stride of 200 tokens, formally expressed as $D = \{P_1, \ldots, P_n\}$ where n is the

[1]We refer to BERT since it is the most common PLM. In some of our later experiments, we consider the more recent and effective ELECTRA model [15]; the same limitations apply to it and to most PLMs.

number of passages. Afterward, these passages are taken as input to the BERT model for relevance estimation.

Following prior work [75], we concatenate a query q and passage P_i pair with a `[SEP]` token in between and another `[SEP]` token at the end. The special `[CLS]` token is also prepended, in which the corresponding output in the last layer is parameterized as a relevance representation $p_i^{cls} \in \mathcal{R}^d$, denoted as follows:

$$p_i^{cls} = \text{BERT}(q, P_i) \tag{1}$$

3.2 Score vs. Representation Aggregation

Previous approaches like BERT-MaxP [20] and Birch [106] use a feedforward network to predict a relevance score from each passage representation p_i^{cls}, which are then aggregated into a document relevance score with a score aggregation approach. Figure 1a illustrates common score aggregation approaches like max pooling ("MaxP"), sum pooling, average pooling, and k-max pooling. Different from score aggregation approaches, passage representation aggregation (PARADE) approaches generate an overall document relevance representation by aggregating passage representations directly (see Figure 1b). By leveraging the passages representations, PARADE has the advantage of modeling document-level relevance directly rather than splitting a document into passages independently. We describe the representation aggregators in the following sections.

3.3 Aggregating Passage Representations

Given the passage relevance representations $D^{cls} = \{p_1^{cls}, \ldots, p_n^{cls}\}$, PARADE summarizes D^{cls} into a single dense representation $d^{cls} \in \mathcal{R}^d$ in one of several different ways, as illustrated in Figure 2.

PARADE–Max utilizes a robust max pooling operation on the passage relevance features[2] in D^{cls}. As widely applied in Convolution Neural Networks (CNNs), max pooling has been shown to be effective in obtaining position-invariant features [87]. Herein, each element at index j in d^{cls} is obtained by a element-wise max pooling operation on the passage relevance representations over the same index.

$$d^{cls}[j] = \max(p_1^{cls}[j], \ldots, p_n^{cls}[j]) \tag{2}$$

PARADE–Attn assumes that each passage contributes differently to the relevance of a document to the query. A simple yet effective way to learn the importance of a passage is to apply a feed-forward network to predict passage weights:

$$w_1, \ldots, w_n = \text{softmax}(Wp_1^{cls}, \ldots, Wp_n^{cls}) \tag{3}$$

$$d^{cls} = \sum_{i=1}^{n} w_i p_i^{cls} \tag{4}$$

where softmax is the normalization function and $W \in \mathcal{R}^d$ is a learnable weight.

For completeness of study, we also introduce a **PARADE–Sum** that simply sums the passage relevance representations. This can be regarded as manually assigning equal weights to all passages (i.e., $w_i = 1$). In addition, we introduce another variant **PARADE–Avg** that is combined with document length normalization(i.e., $w_i = 1/n$).

PARADE–CNN, which operates in a hierarchical manner, stacks several Convolutional Neural Network (CNN) layers with a window size of $d \times 2$ and a stride of 2. In other words, the CNN filters

[2]Note that max pooling is performed on passage *representations*, not over passage relevance scores as in prior work.

Table 1. Collection statistics. (There are 43 test queries in DL'19 and 45 test queries in DL'20.)

Collection	# Queries	# Documents	# tokens / doc
Robust04	249	0.5M	0.7K
GOV2	149	25M	3.8K
Genomics	64	162K	6.5K
MSMARCO	43/45	3.2M	1.3K
ClueWeb12-B13	80	52M	1.9K

operate on every pair of passage representations without overlap. Specifically, we stack 4 layers of CNN, which halve the number of representations in each layer, as shown in Figure 2b. The output of the final CNN layer is then regarded as the document relevance representation.

PARADE–Transformer enables passage relevance representations to interact thoroughly by adopting the Transformer encoder [94] in a hierarchical way. Specifically, BERT's `[CLS]` token embedding and all p_i^{cls} are concatenated, resulting in an input $x^l = (emb^{cls}, p_1^{cls}, \ldots, p_n^{cls})$ that is consumed by several Transformer layers to exploit the ordering of and dependencies among passages. That is,

$$h = \text{LayerNorm}(x^l + \text{MultiHead}(x^l)) \tag{5}$$

$$x^{l+1} = \text{LayerNorm}(h + \text{FFN}(h)) \tag{6}$$

where LayerNorm is the layer-wise normalization as introduced in [3], MultiHead is the multi-head self-attention [94], and FFN is a two-layer feed-forward network with a ReLu activation in between. As shown in Figure 2c, the `[CLS]` vector of the last Transformer output layer, regarded as a pooled representation of the relevance between query and the whole document, is taken as d^{cls}.

Note that both PARADE–CNN and PARADE–Transformer are hierarchical models in the sense that a document level relevance representation is obtained by complex aggregation of passage level relevance representations, which is similar in spirit to pre-BERT models like HiNT [23], ARC [38], MatchPyramid [78], and Duet [71].

3.4 Generating Document Relevance Score

For all PARADE variants except PARADE–CNN, after obtaining the final d^{cls} embedding, a single-layer feed-forward network (FFN) is adopted to generate a relevance score, as follows:

$$rel(q, D) = W_d d^{cls} \tag{7}$$

where $W_d \in \mathcal{R}^d$ is a learnable weight. For PARADE–CNN, a FFN with one hidden layer is applied to every CNN representation, and the final score is determined by the sum of those FFN output scores[3].

3.5 Aggregation Complexity

We note that the computational complexity of representation aggregation techniques are dominated by the passage processing itself. In the case of PARADE–Max, Attn, and Sum, the methods are inexpensive. For PARADE–CNN and PARADE–Transformer, there are inherently fewer passages in a document than total tokens, and (in practice) the aggregation network is shallower than the transformer used for passage modeling.

4 EXPERIMENTS

4.1 Datasets

We experiment with several ad-hoc ranking collections. Robust04[4] is a newswire collection used by the TREC 2004 Robust track. GOV2[5] is a web collection crawled from US government websites used in the TREC Terabyte 2004–06 tracks. For Robust04 and GOV2, we consider both keyword (title) queries and description queries in our experiments. The Genomics dataset [29, 30] consists of scientific articles from the Highwire Press[6] with natural-language queries about specific genes, and was used in the TREC Genomics 2006–07 track. The MSMARCO document ranking dataset[7] is a large-scale collection and is used in TREC 2019–20 Deep Learning Tracks [16, 18]. To create document labels for the development and training sets, passage-level labels from the MSMARCO passage dataset are transferred to the corresponding source document that contained the passage. In other words, a document is considered relevant as long as it contains a relevant passage, and each query can be satisfied by a single passage. The ClueWeb12-B13 dataset[8] is a large-scale collection crawled from the web between February 10, 2012 and May 10, 2012. It is used for the NTCIR We Want Web 3 (WWW-3) Track [86]. The statistics of these datasets are shown in Table 1. The average number of tokens per document is calculated from the subset of documents retrieved by BM25 (i.e., the documents being reranked). Documents in GOV2 and Genomics are much longer than Robust04, making it more challenging to train an end-to-end ranker.

4.2 Baselines

We compare PARADE against the following traditional and neural baselines, including those that employ other passage aggregation techniques.

BM25 is an unsupervised ranking model based on IDF-weighted counting [83]. The documents retrieved by BM25 also serve as the candidate documents used with reranking methods.

BM25+RM3 is a query expansion model based on RM3 [50]. We used Anserini's [102] implementations of BM25 and BM25+RM3. Documents are indexed and retrieved with the default settings for keywords queries. For description queries, we set $b = 0.6$ and changed the number of expansion terms to 20.

Birch aggregates sentence-level evidence provided by BERT to rank documents [106]. Rather than using the original Birch model provided by the authors, we train an improved "Birch-Passage" variant. Unlike the original model, Birch-Passage uses passages rather than sentences as input, it is trained end-to-end, it is fine-tuned on the target corpus rather than being applied zero-shot, and it does not interpolate retrieval scores with the first-stage retrieval method. These changes bring our Birch variant into line with the other models and baselines (e.g., using passages inputs and no interpolating), and they additionally improved effectiveness over the original Birch model in our pilot experiments.

ELECTRA-MaxP adopts the maximum score of passages within a document as an overall relevance score [20]. However, rather than fine-tuning BERT-base on a Bing search log, we improve performance by fine-tuning on the MSMARCO passage ranking dataset. We also use the more recent and effective pre-trained ELECTRA model rather than BERT [15, 110].

[3]In pilot experiments, we found this approach to be more robust than considering only the final CNN representation.
[4]https://trec.nist.gov/data/qa/T8_QAdata/disks4_5.html
[5]http://ir.dcs.gla.ac.uk/test_collections/gov2-summary.htm
[6]https://www.highwirepress.com/
[7]https://microsoft.github.io/TREC-2019-Deep-Learning
[8]http://lemurproject.org/clueweb12/

ELECTRA-KNRM is a kernel-pooling neural ranking model based on query-document similarity matrix [99]. We set the kernel size as 11. Different from the original work, we use the embeddings from the pre-trained ELECTRA model for model initialization.

CEDR-KNRM (Max) combines the advantages from both KNRM and pre-trained model [67]. It digests the kernel features learned from KNRM and the `[CLS]` representation as ranking features. We again replace the BERT model with the more effective ELECTRA. We also use a more effective variant that performs max-pooling on the passages' `[CLS]` representations, rather than averaging the representations.

T5-3B performs ranking in a sequence-to-sequence generation context using the pre-trained T5 model [76]. For the document reranking task, it utilizes the same score max-pooling technique as in BERT-MaxP [20]. Due to its large size and expensive training, we present the values reported by [76] in their zero-shot setting, rather than training it ourselves.

BioBERT-base [51] is a BERT variant pre-trained on a biomedical corpus. We use BioBERT in place of ELECTRA-base on the TREC Genomics task, because it is better suited for specialized medical domain data.

We do not consider pre-BERT methods that utilize aggregation approaches (e.g., [23, 108]), since we focus on studying passage representation aggregation vs. score aggregation in the context of pre-trained language models.

4.3 Training

To prepare the ELECTRA model for the ranking task, we first fine-tune ELECTRA on the MSMARCO passage ranking dataset [74]. The fine-tuned ELECTRA model is then used to initialize PARADE's PLM component. For PARADE–Transformer we use two randomly initialized transformer encoder layers with the same hyperparameters (e.g., number of attention heads, hidden size, etc.) used by ELECTRA-base. Training of PARADE and the baselines was performed on a single Google TPU v3-8 using a pairwise hinge loss. We use the Tensorflow implementation of PARADE available in the Capreolus toolkit [105]; a standalone implementation is also available[9]. We train on the top 1,000 documents returned by a first-stage retrieval method; documents that are labeled relevant in the ground-truth are taken as positive samples and all other documents serve as negative samples. We use BM25+RM3 for first-stage retrieval on Robust04 and BM25 on the other datasets with parameters tuned on the dev sets via grid search. We train for 36 "epochs" consisting of 4,096 pairs of training examples with a learning rate of 3e-6, warm-up over the first ten epochs, and a linear decay rate of 0.1 after the warm-up. Due to its larger memory requirements, we use a batch size of 16 with CEDR and a batch size of 24 with all other methods. Other than this difference, the PARADE variants and other methods are trained in the same way. Each instance comprises a query and all split passages in a document.

Documents are split into a maximum of 16 passages. As we split the documents using a sliding window of 225 tokens with a stride of 200 tokens, a maximum number of 3,250 tokens in each document are retained. The maximum passage sequence length is set as 256. Documents with fewer than the maximum number of passages are zero padded. The padded passage representations are masked when doing passage representation aggregation. For documents longer than required, the first and last passages are always kept while the remaining are uniformly sampled as in [20].

4.4 Evaluation

Following prior work [20, 67], we use 5-fold cross-validation. We set the reranking threshold to 1000 on the test fold as trade-off between latency and effectiveness. The reported results aggregate

[9]`https://github.com/canjiali/PARADE/`

queries across all test folds. Performance is measured in terms of the MAP, Precision, ERR and nDCG ranking metrics using `trec_eval`[10] with different cutoffs. On Robust04 and Gov2, the cutoff is 20 following prior work on these collections [57]. On TREC-DL and NTCIR, the cutoff is 10 following the official guidelines. [16, 18, 86]. For NTCIR WWW-3, the results are reported using `NTCIREVAL`[11]. The details of the evaluation metrics are described as follows.

The **Precision** metric is defined as the ratio of relevant documents in the retrieved documents at a cutoff k. **MAP** evaluates a retrieval system from the trade-off between precision and recall and is usually measured at a deeper cutoff than precision. It's defined as:

$$MAP = \frac{1}{|Q|} \sum_{j=1}^{|Q|} \frac{1}{m_j} \sum_{k=1}^{m_j} Precision(R_{jk}) \tag{8}$$

where $|Q|$ is the number of queries and $|m_j|$ is the number of relevant documents to the query Q_j. R_{jk} is the ranking position of a relevant document d_k. In other words, MAP takes into account the precision of all recalled documents.

Both Precision and MAP consider documents as either relevant (label 1) or irrelevant (label 0) while the **nDCG** evaluates a ranking system in a finer grain:

$$DCG@k = \sum_{i=1}^{k} \frac{2^{rel_i} - 1}{\log(1+i)} \tag{9}$$

where rel_i is the label of a document at position i, measured by its relevance to the query (3 as highly relevant, 2 as relevant, 1 as partially relevant, and 0 as irrelevant). The DCG is then normalized by IDCG as

$$nDCG@k = \frac{DCG@k}{IDCG@k} \tag{10}$$

where the IDCG is the DCG value of the ideal ranked list to the query so as to ensure that nDCG is in the range from 0 to 1.

The **ERR** metric is inspired by the cascade user model that assumes a user only will continue examining the next document when the last perceived document does not satisfy the user's information need, formally:

$$ERR@k = \sum_{r=1}^{k} \frac{1}{r} \prod_{i=1}^{r-1} (1 - R_i) R_r \tag{11}$$

where R_i represents the relevance degrade of a document $R_i = \frac{2^{rel_i - 1}}{2^{rel_{max}}}$. As ERR is not normalized, there is a normalized version of ERR, called nERR, that is normalized by the aforementioned ideal ranked list [85].

4.5 Main Results

The reranking effectiveness of PARADE on the two commonly-used Robust04 and GOV2 collections are shown in Table 2 and Table 3, respectively. Considering the three approaches that do not introduce any new weights, PARADE–Max is usually more effective than PARADE–Avg and PARADE–Sum, though the results are mixed on GOV2. PARADE–Max is consistently better than PARADE–Attn on Robust04, but PARADE–Attn sometimes outperforms PARADE–Max on GOV2. The two variants that consume passage representations in a hierarchical manner, PARADE–CNN and PARADE–Transformer, consistently outperforms the four other variants. This confirms the effectiveness of the passage representation aggregation approaches.

[10] `https://trec.nist.gov/trec_eval`
[11] `http://research.nii.ac.jp/ntcir/tools/ntcireval-en.html`

Table 2. Ranking effectiveness of PARADE on the *Robust04* collection. Best performance is in bold. Significant difference between PARADE–Transformer and the corresponding method is marked with † ($p < 0.05$, two-tailed paired t-test). We also report the current best-performing model (T5-3B from [76]).

	Robust04 Title			**Robust04 Description**		
	MAP	P@20	nDCG@20	MAP	P@20	nDCG@20
BM25	0.2531†	0.3631†	0.4240†	0.2249†	0.3345†	0.4058†
BM25+RM3	0.3033†	0.3974†	0.4514†	0.2875†	0.3659†	0.4307†
Birch	0.3763	0.4749†	0.5454†	0.4009†	0.5120†	0.5931†
ELECTRA-MaxP	0.3183†	0.4337†	0.4959†	0.3464†	0.4731†	0.5540†
T5-3B (from [76])	-	-	-	0.4062	-	0.6122
ELECTRA-KNRM	0.3673†	0.4755†	0.5470†	0.4066	**0.5255**	0.6113
CEDR-KNRM (Max)	0.3701†	0.4769†	0.5475†	0.3975†	0.5219	0.6044†
PARADE-Avg	0.3352†	0.4464†	0.5124†	0.3640†	0.4896†	0.5642†
PARADE-Sum	0.3526†	0.4711†	0.5385†	0.3789†	0.5100†	0.5878†
PARADE-Max	0.3711†	0.4723†	0.5442†	0.3992†	0.5217	0.6022
PARADE-Attn	0.3462†	0.4576†	0.5266†	0.3797†	0.5068†	0.5871†
PARADE-CNN	**0.3807**	0.4821†	0.5625	0.4005†	0.5249	0.6102
PARADE-Transformer	0.3803	**0.4920**	**0.5659**	**0.4084**	**0.5255**	**0.6127**

Table 3. Ranking effectiveness of PARADE on the *GOV2* collection. Best performance is in bold. Significant difference between PARADE–Transformer and the corresponding method is marked with † ($p < 0.05$, two-tailed paired t-test).

	GOV2 Title			**GOV2 Description**		
	MAP	P@20	nDCG@20	MAP	P@20	nDCG@20
BM25	0.3056†	0.5362†	0.4774†	0.2407†	0.4705†	0.4264†
BM25+RM3	0.3350†	0.5634†	0.4851†	0.2702†	0.4993†	0.4219†
Birch	0.3406†	0.6154†	0.5520†	0.3270	0.6312†	0.5763†
ELECTRA-MaxP	0.3193†	0.5802†	0.5265†	0.2857†	0.5872†	0.5319†
ELECTRA-KNRM	0.3469†	0.6342†	0.5750†	0.3269	0.6466	0.5864†
CEDR-KNRM (Max)	0.3481†	0.6332†	0.5773†	**0.3354**†	0.6648	0.6086
PARADE-Avg	0.3174†	0.6225†	0.5741†	0.2924†	0.6228†	0.5710†
PARADE-Sum	0.3268†	0.6218†	0.5747†	0.3075†	0.6436†	0.5879†
PARADE-Max	0.3352†	0.6228†	0.5636†	0.3160†	0.6275†	0.5732†
PARADE-Attn	0.3306†	0.6359†	0.5864†	0.3116†	0.6584	0.5990
PARADE-CNN	0.3555†	0.6530	0.6045	0.3308	**0.6688**	**0.6169**
PARADE-Transformer	**0.3628**	**0.6651**	**0.6093**	0.3269	0.6621	0.6069

Considering the baseline methods, PARADE–Transformer significantly outperforms the Birch and ELECTRA-MaxP score aggregation approaches for most metrics on both collections. PARADE–Transformer's ranking effectiveness is comparable with T5-3B on the Robust04 collection while using only 4% of the parameters, though it is worth noting that T5-3B is being used in a zero-shot

Table 4. Ranking effectiveness of PARADE on the *Genomics* collection. Significant difference between PARADE–Transformer and the corresponding method is marked with † ($p < 0.05$, two-tailed paired t-test). The top neural results are underlined, and the top overall scores are in bold.

	MAP	**P@20**	**nDCG@20**
BM25	0.3108	0.3867	0.4740
TREC Best	**0.3770**	**0.4461**	**0.5810**
Birch	0.2832	0.3711	0.4601
BioBERT-MaxP	0.2577	0.3469	0.4195†
BioBERT-KNRM	0.2724	0.3859	0.4605
CEDR-KNRM (Max)	0.2486	0.3516†	0.4290
PARADE-Avg	0.2514†	0.3602	0.4381
PARADE-Sum	0.2579†	0.3680	0.4483
PARADE-Max	<u>0.2972</u>	<u>0.4062</u>†	<u>0.4902</u>
PARADE-Attn	0.2536†	0.3703	0.4468
PARADE-CNN	0.2803	0.3820	0.4625
PARADE-Transformer	0.2855	0.3734	0.4652

setting. CEDR-KNRM and ELECTRA-KNRM, which both use some form of representation aggregation, are significantly worse than PARADE–Transformer on title queries and have comparable effectiveness on description queries. Overall, PARADE–CNN and PARADE–Transformer are consistently among the most effective approaches, which suggests the importance of performing complex representation aggregation on these datasets.

Results on the Genomics dataset are shown in Table 4. We first observe that this is a surprisingly challenging task for neural models. Unlike Robust04 and GOV2, where transformer-based models are clearly state-of-the-art, we observe that all of the methods we consider almost always underperform a simple BM25 baseline, and they perform well below the best-performing TREC submission. It is unclear whether this is due to the specialized domain, the smaller amount of training data, or some other factor. Nevertheless, we observe some interesting trends. First, we see that PARADE approaches can outperform score aggregation baselines. However, we note that statistical significance can be difficult to achieve on this dataset, given the small number of queries. Next, we notice that PARADE–Max performs the best among neural methods. This is in contrast with what we observed on Robust04 and GOV2, and suggests that hierarchically aggregating evidence from different passages is not required on the Genomics dataset.

4.6 Results on the TREC DL Track and NTCIR WWW-3 Track

We study the effectiveness of PARADE on the TREC DL Track and NTCIR WWW-3 Track. We report results in this section and refer the readers to the TREC and NTCIR task papers for details on the specific hyperparameters used [53, 54].

Results from the TREC Deep Learning Track are shown in Table 5. For simplicity of study, we only report the results of the two most representative PARADE variants: PARADE–Max (representation aggregation without a hierarchical architecture) and PARADE–Transformer (representation aggregation with a hierarchical architecture). In TREC DL'19, we include comparisons with top-performaning runs from TREC: `ucas_runid1` [13] used BERT-MaxP [20] as the reranking method, `TUW19-d3-re` [35] is a Transformer-based non-BERT method, and `idst_bert_r1` [100] utilizes

Table 5. Ranking effectiveness of PARADE on TREC DL Track document ranking task. PARADE's best results are underlined. The top overall results of of each track are in bold.

Year	Group	Runid	MAP	nDCG@10
2019	TREC	BM25	0.237	0.517
		ucas_runid1 [13]	0.264	0.644
		TUW19-d3-re [35]	0.271	0.644
		idst_bert_r1 [100]	**0.291**	**0.719**
	Ours	PARADE–Max	0.287	0.679
		PARADE–Transformer	0.274	0.650
2020	TREC	BM25	0.379	0.527
		fr_doc_roberta [49]	0.442	0.640
		ICIP_run1 [12]	0.433	0.662
		d_d2q_duo [79]	**0.542**	**0.693**
	Ours	PARADE–Max	0.420	0.613
		PARADE–Transformer	0.403	0.601

Table 6. Ranking effectiveness of PARADE on NTCIR WWW-3 task. PARADE's best results are underlined. The best results of the Track are in bold.

Model	nDCG@10	Q@10	nERR@10
BM25	0.5748.	0.5850	0.6757
Technion-E-CO-NEW-1 [81]	0.6581	0.6815	0.7791
KASYS-E-CO-NEW-1 [89]	**0.6935**	**0.7123**	0.7959
PARADE–Max	0.6337	0.6556	0.7395
PARADE–Transformer	0.6897	0.7016	**0.8090**

structBERT [95], which strengthens on the task of sentence order prediction. All PARADE variants outperform `ucas_runid1` and `TUW19-d3-re` in terms of nDCG@10, but cannot outperform `idst_bert_r1`. Since this run's pre-trained structBERT model is not publicly available, we are not able to embed it into PARADE and make a fair comparison. In TREC DL'20, the best TREC run `d_d2q_duo` is a T5-3B model. Moreover, PARADE–Max again outperforms PARADE–Transformer, which is in line to the Genomics results in Table 4 and in contrast to results on Robust04 and GOV2 in Table 2 and Table 3. We explore this further in Section 5.5.

Results from the NTCIR WWW-3 Track are shown in Table 6. `KASYS-E-CO-NEW-1` is a Birch-based method [106] that uses BERT-Large and `Technion-E-CO-NEW-1` is a cluster-based method. As shown in Table 6, PARADE–Transformer's effectiveness is comparable with `KASYS-E-CO-NEW-1` across metrics. On this benchmark, PARADE–Transformer outperforms PARADE–Max by a large margin.

5 ANALYSIS

In this section, we consider the following research questions:

- **RQ1:** What factors contribute most to PARADE's effectiveness?
- **RQ2:** How does PARADE perform when compared with transformers that support long text?
- **RQ3:** How can BERT's efficiency be improved while maintaining its effectiveness?

Table 7. Source of effectiveness in PARADE. **PRA:** Passage Representation Aggregation, **HM:**Hierarchical model, **DT:**Document-level Training

	Source			**Robust04**			**GOV2**		
Model	HM	PRA	DT	MAP	P@20	nDCG@20	MAP	P@20	nDCG@20
ELECTRA-MaxP				0.3183	0.4337	0.4959	0.3193	0.5802	0.5265
ELECTRA-MaxP (e2e)			✓	0.3766	0.4757	0.5444	0.3389	0.6134	0.5467
PARADE-Max		✓	✓	0.3711	0.4723	0.5442	0.3352	0.6228	0.5636
PARADE-Transformer	✓	✓	✓	**0.3803**	**0.4920**	**0.5659**	**0.3628**	**0.6651**	**0.6093**

- **RQ4:** How sensitive is PARADE to the hyper-parameters for selecting passages in a document?
- **RQ5:** When is the complex representation aggregation approach e.g., CNN and Transformer preferable to the simple aggregation approach in the light of dataset characteristics?
- **RQ6:** Is it beneficial to rerank documents from a more effective first-stage retriever? In particular, is reranking BM25+RM3 better than reranking BM25?

5.1 Source of Effectiveness (RQ1)

The major components of PARADE are 1) passage representation aggregation, 2) hierarchical model, and 3) end-to-end document-level training. To study these factors, we first trained an ELECTRA-MaxP model end-to-end, i.e., the max-pooling is conducted during training rather than training on individual passages separately. MaxP score aggregation [20] exhibits two limitations. First, a passage model is trained by taking all passages in a relevant document as relevant and all passages in an irrelevant document as irrelevant. This introduces label noise into the training as not all passages in a relevant document are necessarily relevant. Second, a document score is obtained by max pooling the passage scores during inference while no document level signal is supervised during training. This introduces a mismatch between training and testing.

As illustrated in Table 7, by overcoming such limitations, the end-to-end trained ELECTRA-MaxP (e2e) outperforms ELECTRA-MaxP by a large margin. This finding confirms the advantage of using document labels to train a document retrieval model rather than splitting a document into independent passages, which relies on the unrealistic assumption that all passages have the same relevance label. The second difference between PARADE and ELECTRA-MaxP [20] is that passage representations are aggregated rather than passage relevance scores. From Table 7, the effectiveness difference between ELECTRA-MaxP (e2e) and PARADE–Max is minor in terms of MAP on the Robust04 collection, but improves by a larger margin on GOV2 and with other metrics on Robust04. Aggregating passage representations is arguably better than aggregating passage scores, which is also confirmed by Zamani et al. [108] where they study aggregating field representations vs. scores. The third key factor in PARADE is the hierarchical passage model, which is adopted in PARADE-Transformer and PARADE-CNN. Notably, PARADE-Transformer improves the performance over the simple max-pooling aggregation approach on both collections. In summary, a hierarchical passage representation aggregation model trained at the document-level makes a more effective document retrieval model. In the next section, we study the difference between PARADE and the more recent long-text Transformers.

5.2 Comparison with Long-Text Transformers (RQ2)

Recently, a line of research focuses on reducing the redundant computation cost in the transformer block, allowing models to support longer sequences. Most approaches design novel sparse attention

Table 8. Comparison with transformers that support longer text sequences on the Robust04 collection.

Model	**nDCG@20**	**ERR@20**
Sparse-Transformer [43]	0.449	0.119
Longformer-QA [43]	0.448	0.113
Transformer-XH [43]	0.450	0.123
QDS-Transformer [43]	0.457	0.126
Longformer [7]	0.500	-
Big-Bird [7]	0.452	-
PARADE–Transformer	**0.565**	**0.149**

mechanism for efficiency, which makes it possible to input longer documents as a whole for ad-hoc ranking. These methods restrict the attention for a word using a sliding attention mask, whereas PARADE restricts the attention from the input by considering one predefined passage unit at a time. Hence, PARADE learns the relevance of stationary passages as opposed to the sliding attention window for a word in the long-text Transformers. We consider the results reported by Jiang et al. [43] and Boytsov et al. [7] to compare some of these approaches with passage representation aggregation. The results are shown in Table 8. In this comparison, long-text transformer approaches achieve similar effectiveness and underperform PARADE–Transformer by a large margin. However, it is worth noting that approaches in Jiang et al. [43] use the `[CLS]` representation as features for a downstream model rather than using it to predict a relevance score directly as by Boytsov et al. [7] and PARADE, which contribute to the difference in effectiveness. Using the `[CLS]` representation to predict a relevance score is more effective as confirmed by the comparison between Longformer-QA [43] and Longformer [7], which are the same model trained differently. Apart from the higher ranking effectiveness, Boytsov et al. [7] argue that PARADE is also 1.5x faster to train and evaluate than Longformer. We consider the question of how to further improve PARADE's efficiency in Section 5.3.

5.3 Reranking Effectiveness vs. Efficiency (RQ3)

While BERT-based models are effective at producing high-quality ranked lists, they are computationally expensive. However, the reranking task is sensitive to efficiency concerns, because documents must be reranked in real time after the user issues a query. In this section we consider two strategies for improving PARADE's efficiency.

Using a Smaller BERT Variant. As smaller models require fewer computations, we study the reranking effectiveness of PARADE when using pre-trained BERT models of various sizes, providing guidance for deploying a retrieval system. To do so, we use the pre-trained BERT provided by Turc et al. [93]. In this analysis we change several hyperparameters to reduce computational requirements: we rerank the top 100 documents from BM25, train with a cross-entropy loss using a single positive or negative document, reduce the passage length 150 tokens, and reduce the stride to 100 tokens. We additionally use BERT models in place of ELECTRA so that we can consider models warmed up from LM distillation (i.e., distillation using self-supervised PLM objectives in the pretraining stage) rather than using ranker distillation alone (i.e., using student models that are randomly initialized). Gao et al. [26] found that LM distillation is more effective than ranker distillation alone. From Table 9, it can be seen that as the size of models is reduced, their effectiveness usually declines. The hidden layer size (#6 vs #7, #8 vs #9) plays a more critical role for performance than the number of layers (#3 vs #4, #5 vs #6). An example is the comparison between models #7 and #8. Model

Table 9. PARADE–Transformer's effectiveness using BERT models of varying sizes on Robust04 title queries. L and H stand for number of hidden layers and hidden size, respectively. The distilled models are all distilled from BERT-Base. Significant improvements of distilled over non-distilled models are marked with †. ($p < 0.01$, two-tailed paired t-test.)

			Robust04		Robust04 (Distilled)		Parameter	Inference Time
ID	Model	L / H	P@20	nDCG@20	P@20	nDCG@20	Count	(ms / doc)
1	BERT-Large	24 / 1024	0.4508	0.5243	-	-	360M	15.93
2	BERT-Base	12 / 768	0.4486	0.5252	-	-	123M	4.93
3	\	10 / 768	0.4420	0.5168	0.4494†	0.5296†	109M	4.19
4	\	8 / 768	0.4428	0.5168	0.4490†	0.5231	95M	3.45
5	BERT-Medium	8 / 512	0.4303	0.5049	0.4388†	0.5110	48M	1.94
6	BERT-Small	4 / 512	0.4257	0.4983	0.4365†	0.5098†	35M	1.14
7	BERT-Mini	4 / 256	0.3922	0.4500	0.4046†	0.4666†	13M	0.53
8	\	2 / 512	0.4000	0.4673	0.4038	0.4729	28M	0.74
9	BERT-Tiny	2 / 128	0.3614	0.4216	0.3831†	0.4410†	5M	0.18

#8 performs better; it has fewer layers but contains more parameters. The number of parameters and inference time are also given in Table 9 to facilitate the study of trade-offs between model complexity and effectiveness.

Distilling Knowledge from a Large Model. To further explore the limits of smaller PARADE models, we apply knowledge distillation to leverage knowledge from a large teacher model. We use PARADE–Transformer trained with BERT-Base on the target collection as the teacher model. Smaller student models then learn from the teacher at the output level. We use mean squared error as the distilling objective, which has been shown to work effectively [91, 92]. The learning objective penalizes the student model based on both the ground-truth and the teacher model:

$$L = \alpha \cdot L_{CE} + (1 - \alpha) \cdot ||z^t - z^s||^2 \tag{12}$$

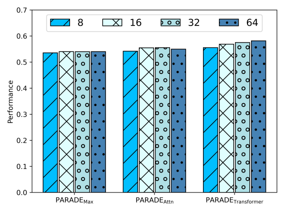
where L_{CE} is the cross-entropy loss with regard to the logit of the student model and the ground truth, α weights the importance of the learning objectives, and z^t and z^s are logits from the teacher model and student model, respectively.

As shown in Table 9, the nDCG@20 of distilled models always increases. The PARADE model using 8 layers (#4) can achieve comparable results with the teacher model. Moreover, the PARADE model using 10 layers (#3) can outperform the teacher model with 11% fewer parameters. The PARADE model trained with BERT-Small achieves a nDCG@20 above 0.5, which outperforms BERT-MaxP using BERT-Base, while requiring only 1.14 ms to perform inference on one document. Thus, when reranking 100 documents, the inference time for each query is approximately 0.114 seconds.

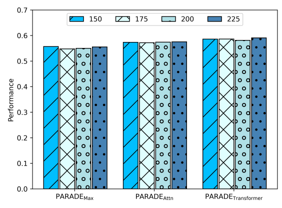
5.4 Hyper-parameter Sensitivity (RQ4)

In this section, we study the hyper-parameter sensitivity of PARADE with respect to the number of passages, window size, and window stride. We conduct such ablation study on the GOV2 collection given that these documents are longer on average than in Robust04. When studying one specific hyper-parameter, we fix other hyper-parameters to the values used in Section 4.3.

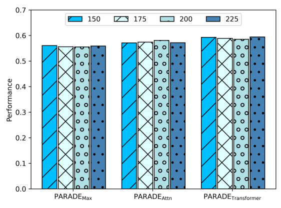
Number of Passages. One hyper-parameter in PARADE is the maximum number of passages being used, i.e., preserved data size, which is studied to answer RQ3 in this section. Figure 3 depicts nDCG@20 of PARADE–Transformer with the number of passages varying from 8 to 64. Generally, larger preserved data size results in better performance for PARADE–Transformer, which suggests that a document can be better understood from document-level context with more



(a) Number of Passages.

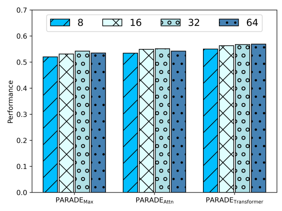


(b) Window Size.

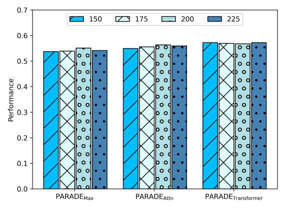


(c) Window Stride.

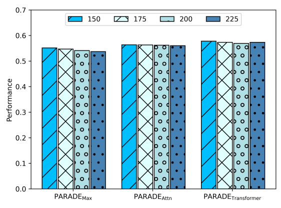
Fig. 3. Hyper-parameter sensitivity of PARADE on *Gov2* collection using title queries. nDCG@20 is reported.



(a) Number of Passages.



(b) Window Size.



(c) Window Stride.

Fig. 4. Hyper-parameter sensitivity of PARADE on *Gov2* collection using description queries. nDCG@20 is reported.

preservation of its content. For PARADE–Max and PARADE–Attn, however, the performance degrades a little when using 64 passages. Both max pooling (Max) and simple attention mechanism (Attn) have limited capacity and are challenged when dealing with such longer documents. The PARADE–Transformer model is able to improve nDCG@20 as the number of passages increases, demonstrating its superiority in detecting relevance when documents become much longer.

However, considering more passages also increases the number of computations performed. One advantage of the PARADE models is that the number of parameters remains constant as the number of passages in a document varies. Thus, we consider the impact of varying the number of passages considered between training and inference. As shown in Table 10, rows indicate the number of passages considered at training time while columns indicate the number used to perform inference. The diagonal indicates that preserving more of the passages in a document consistently improves nDCG. Similarly, increasing the number of passages considered at inference time (columns) or at training time (rows) usually improves nDCG. In conclusion, the number of passages considered plays a crucial role in PARADE's effectiveness. When trading off efficiency for effectiveness, PARADE models' effectiveness can be improved by training on more passages than will be used at inference time. This generally yields a small nDCG increase.

Window Size and Stride. The other two parameters are the window size and the window stride used for selecting passages. The window size controls the length of a passage while the stride controls how much consecutive passages overlap. As it can be seen from Figure 3 and Figure 4, PARADE is very stable in a wide range of hyper-parameters. Such phenomenon is consistent with early work on passage-based document retrieval [8].

Table 10. Reranking effectiveness of PARADE–Transformer using various preserved data size on *GOV2* title dataset. nDCG@20 is reported. The indexes of columns and rows are number of passages being used.

Train \ Eval	8	16	32	64
8	*0.5554*	0.5648	0.5648	0.5680
16	0.5621	*0.5685*	0.5736	0.5733
32	0.5610	0.5735	*0.5750*	0.5802
64	0.5577	0.5665	0.5760	*0.5815*

Table 11. Percentage of documents with a given number of relevant passages.

# Relevant passages	**GOV2**	**DL19**	**DL19**	**DL20**	**MS MARCO**	**Genomics**
		(FiRA)	(Ours)	(Ours)	train / dev	2006
1	38%	66%	66%	67%	99% / 98%	62%
1–2	60%	87%	86%	81%	100% / 100%	80%
3+	40%	13%	14%	19%	0% / 0%	20%

5.5 Dataset Characteristics: Complex Aggregation vs. Simple Aggregation (RQ5)

Focused Nature of Queries. While PARADE variants are effective across a range of datasets and the PARADE–Transformer variant is generally the most effective, this is not always the case. In particular, PARADE–Max outperforms PARADE–Transformer on both years of TREC DL and on TREC Genomics. We hypothesize that this difference in effectiveness is a result of the focused nature of queries in both collections. Such queries may result in a lower number of highly relevant passages per document, which would reduce the advantage of using more complex aggregation methods like PARADE–Transformer and PARADE–CNN. In other words, when a focused query can be answered well by a short highly-relevant passage, we expect considering the entire document to yield less benefit.

While documents in MS MARCO may contain other passages related to the query, these passages are not *necessary* to satisfy the query's information need, which can be answered by a single passage by nature of the collection's construction. The fact that TREC DL queries come from MS MARCO suggests that the queries in both TREC DL collections *can* also be sufficiently answered by a single highly relevant passage.

Passage-document Mapping. We consider this hypothesis by using passage-level relevance judgments to compare the number of highly relevant passages per document in various collections. To do so, we use mappings between relevant passages and documents for those collections with passage-level judgments available: TREC DL, TREC Genomics, and GOV2.

- **TREC DL (Ours).** We create a mapping between the MS MARCO document and passage collections by using the MS MARCO Question Answering (QnA) collection to map passages to document URLs. This mapping can then be used to map between passage and document judgments in DL'19 and DL'20. However, the DL'21 overview paper notes that the resulting counts of relevant passages may be unreliable due to construction artifacts [17].
- **TREC DL (FiRA).** With DL'19, we additionally use the FIRA passage relevance judgments [37] to map between documents and passages. The FIRA judgments were created by asking annotators to identify relevant passages in every DL'19 document with a relevance label of 2 or 3 (i.e., the two highest labels). Thus, FIRA provides reliable counts of relevant passages, but this mapping

contains fewer queries and documents than the mapping we constructed covering DL'19 and DL'20.
- **TREC Genomics.** In the case of TREC Genomics, we use the mapping provided by TREC.
- **GOV2.** We use the sentence-level relevance judgments available in WebAP [45, 46], which cover a subset of 82 queries.

Due to the difficulty of mapping between passages and documents, these MS MARCO mappings come with several caveats. Our mapping covers nearly the entire MS MARCO collection, but it is limited by the fact that DL's passage-level relevance judgments may not be complete, and by the fact that all relevant passages in a MS MARCO document do not necessarily appear in the MS MARCO passage collection. The FIRA mapping covers only highly-relevant DL'19 documents, but the passage annotations are complete and it was created by human annotators with quality control. Thus, it includes all relevant passages in the annotated documents, but not all documents were annotated.

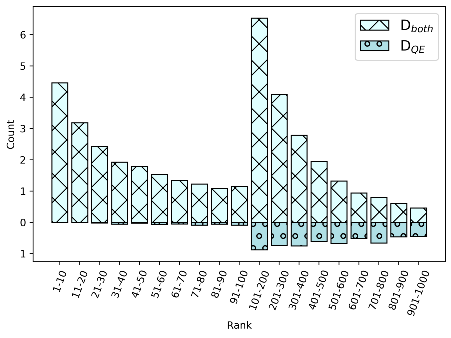
Relevance Label Alignment. We compare passage judgments across collections by using each collection's annotation guidelines to align their relevance labels with MS MARCO's definition of a relevant passage as one that is *sufficient* to answer the question query. With GOV2 we consider passages with a relevance label of 3 or 4 to be relevant. With DL documents we consider a label of 2 or 3 to be relevant and passages with a label of 3 to be relevant. With FIRA we consider label 3 to be relevant. With Genomics we consider labels 1 or 2 to be relevant.

We align the maximum passage lengths in GOV2 to FIRA's maximum length so that they can be directly compared. To do so, we convert GOV2's sentence judgments to passage judgments by collapsing sentences following a relevant sentence into a single passage with a maximum passage length of 130 tokens, as used by FIRA[12]. We note that this process can only *decrease* the number of relevant passages per document observed in GOV2, which we expect to have the highest number. With the DL collections using the MS MARCO mapping, the passages are much smaller than these lengths, so collapsing passages could only *decrease* the number of relevant passages per document. We note that Genomics contains "natural" passages that can be longer; this should be considered when drawing conclusions. In all cases, the relevant passages comprise a small fraction of the document.

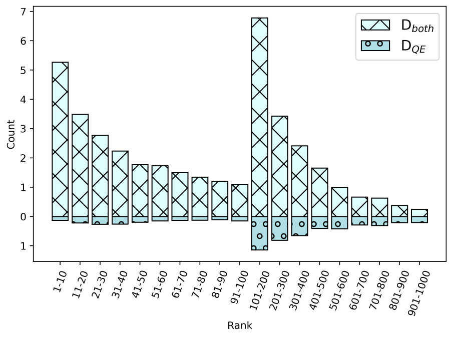
Results. In each collection, we calculate the number of relevant passages per document using the collection's associated document and passage judgments. The results are shown in Table 11. First, considering the GOV2 and MS MARCO collections that we expect to lie at opposite ends of the spectrum, we see that 38% of GOV2 documents contain a single relevant passage, whereas 98–99% of MS MARCO documents contain a single relevant passage. This confirms that MS MARCO documents contain only 1–2 highly relevant passages per document by nature of the collection's construction, though this collection may still contain unjudged passages that are relevant but not necessary to answer the query. The percentages are the lowest on GOV2 as expected. While we would prefer to put these percentages in the context of another collection like Robust04, the lack of passage-level judgments on such collections prevents us from doing so. Second, considering the Deep Learning collections, we see that DL'19 and DL'20 exhibit similar trends regardless of whether our potentially-noisy mapping or the smaller, more reliable FIRA mapping is used. In these collections, the majority of documents contain a single relevant passage and the vast majority of documents contain one or two relevant passages. We call this a *maximum passage bias*[13]. The

[12]Applying the same procedure to both FIRA and WebAP with longer maximum lengths did not substantially change the trend.

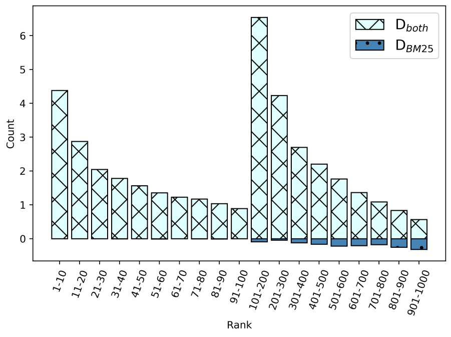
[13]Others have observed another dataset bias in the QA datasets used for passage retrieval [33]: the answers favors earlier positions in the paragraphs, especially in the MS MARCO collection.



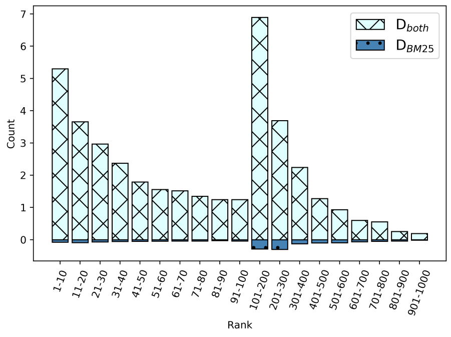
(a) Ranking with BM25+RM3



(b) Reranking with PARADE (BM25+RM3)



(c) Ranking with BM25



(d) Reranking with PARADE (BM25)

Fig. 5. (Re)ranking distributions by different models. The X-axis represents the ranking position bins while Y-axis represents the average number of relevant documents dropped in each bin.

fact that the queries are shared with MS MARCO likely contributes to this observation, since we know the vast majority of MS MARCO question queries *can* be answered by a single passage. Third, considering Genomics 2006, we see that this collection is similar to the DL collections. The majority of documents contain only one relevant passage, and the vast majority contain one or two relevant passages. Thus, this analysis supports our hypothesis that the difference in PARADE–Transformer's effectiveness across collections is related to the number of relevant passages per document in these collections. PARADE–Max performs better when the number is low, which may reflect the reduced importance of aggregating relevance signals across passages on these collections.

5.6 Is Reranking Effectiveness Influenced by the First-stage Retriever? (RQ6)

Can PARADE be further boosted by reranking a stronger first-stage retriever? Exhaustively testing all first-stage retrievers is prohibitively expensive since PARADE must be trained separately with each for a fair comparison (due to the fact that its input distributions may change). In this section, we narrow the first-stage retriever to lexical matching methods, and study whether reranking from a stronger first-stage retriever brings better reranking effectiveness (RQ 6). We compare the reranking effectiveness of PARADE that reranks BM25 [84] and BM25+RM3 [50], respectively. BM25+RM3 is a query expansion methods based on pseudo-relevance feedback that select representative terms in the first round retrieval for expansion. Generally, BM25+RM3 yields better retrieval performance than BM25 [52, 69].

Table 12. (Re)ranking effectiveness of different first-stage retrieval and reranking models.

Model	Recall@100	Recall@1k	MAP@100	MAP@1k	P@20	nDCG@20
BM25	0.4137	0.6989	0.2154	0.2531	0.3631	0.4240
BM25+RM3	0.4517	0.7549	0.2451	0.2903	0.3821	0.4407
PARADE-Transformer (BM25)	0.4996	0.6989	0.2889	0.3280	0.4562	0.5291
PARADE-Transformer (BM25+RM3)	0.5058	0.7549	0.2943	0.3407	0.4548	0.5303
PARADE-Transformer (BM25+RM3, Ensemble)	0.5347	0.7549	0.3167	0.3635	0.4733	0.5411

To simplify the analysis, we focus on the ranking distribution of relevant documents. On the Robust04 collection with title queries, we examine the top 1,000 documents retrieved by BM25 and BM25+RM3. We then divide all relevant documents retrieved into three partitions, D_{both}, D_{BM25} and D_{QE}, defined as follows:

- D_{both}: the relevant documents retrieved by both BM25 and BM25+RM3
- D_{BM25}: the relevant documents retrieved by BM25 but not retrieved by BM25+RM3
- D_{QE}: the relevant documents retrieved by BM25+RM3 but not retrieved by BM25

For all methods, D_{both} is the same; differences come from D_{BM25}, D_{QE}, and non-relevant documents. In total, Count(D_{both}) = 9863, Count(D_{QE}) = 1538, and Count(D_{BM25}) = 409, which means that BM25 and BM25+RM3 share a large number of relevant documents.

The most effective PARADE-Transformer is adopted as a reranker. The (re-)ranking effectiveness of these models is shown in Table 12. Replacing BM25 with BM25+RM3 increases Recall@1k by about 8% and MAP@1k by about 4%, which may be a result of the nearly 1,000 relevant documents introduced by RM3. The differences for the other metrics are minor, with RM3 slightly reducing P@20. These findings are in line with recent work demonstrating that there is little difference in effectiveness between reranking BM25 and reranking BM25+RM3 [76].

To investigate why there is little difference between reranking BM25 and BM25+RM3 for metrics considering top positions, we provide four sub-figures in Figure 5 that depict the number of relevant documents placed in different position bins (averaged by the number of queries). Figures 5a, 5b, 5c, 5d depict the ranking distribution of BM25+RM3, PARADE (reranking BM25+RM3), BM25, PARADE (reranking BM25), respectively. Due to the change in bin size from 10 to 100, there is a steep increase in the bin 101-200 across all figures. The distribution is mono-decreasing if the bin size is unchanged. It can be seen that:

- From figures 5a and 5c, the documents from D_{QE} and D_{BM25} are more likely to be ranked at the low positions (behind 100) by the initial ranking models, which suggests that both models are less confident in these documents. For BM25+RM3, it might be that the documents from D_{QE} are mostly retrieved by the expanded terms; for BM25, it may be these documents are retrieved by terms with lower weights.
- Comparing Figure 5a with 5b, as well as Figure 5c with 5d, the documents from D_{QE} and D_{BM25} can be boosted to higher positions by PARADE. Mostly, documents in D_{QE} are retrieved using the expanded terms. PARADE can boost these documents without even knowing these terms, which confirms contextualization benefits by BERT.
- Comparing Figure 5c with 5d, it can be seen that a large amount of documents from D_{BM25}, especially the documents behind position 100, are boosted to higher positions, which closes the large gap in MAP between BM25 and BM25+RM3 as shown in Table 12.

The advantage of using BM25+RM3 may be that its relevance scores are good source for model ensemble. As shown in Table 12, an ensemble method that linearly interpolates the scores achieves the best results. In conclusion, while BM25+RM3 does retrieve more relevant documents than

BM25, these documents are not effectively utilized by the reranking methods. Simply adopting BM25 as a first-stage retriever can result in a simple retrieval pipeline that performs well.

6 CONCLUSION

We formalized the passage score and representation aggregation strategies for document reranking (PARADE), and demonstrated PARADE's effectiveness on ad-hoc benchmark collections. The effectiveness comes from a combination of 1) document-level training, 2) hierarchical architecture, and 3) passage representation aggregation. The passage representation method also outperforms long-text Transformers taking the whole document as a single input. Particularly, PARADE can cooperate with diverse relevance signals from the full text, rather than focusing on a single passage. It is robust to several hyper-parameters for selecting passages, i.e., number of passages, window size, and stride. A dataset characteristic, i.e., maximum passage bias, is later explored. It reveals that representation aggregation strategies can be less effective when such bias exists on the dataset. For efficiency, we investigated how model size affects performance, finding that knowledge distillation on PARADE boosts the performance of smaller PARADE models while substantially reducing their parameters. Finally, the reranking behaviour of PARADE w.r.t different first-stage retrievers is studied, finding that simply adopting BM25 as a first-stage retriever can result in a simple retrieval pipeline that performs well.

ACKNOWLEDGMENTS

This work was supported in part by the National Natural Science Foundation of China (Grant No. 62272439), Google Cloud, and the Google TPU Research Cloud (TRC).

REFERENCES

[1] Qingyao Ai, Brendan O'Connor, and W. Bruce Croft. 2018. A Neural Passage Model for Ad-hoc Document Retrieval. In *ECIR (Lecture Notes in Computer Science, Vol. 10772)*. Springer, 537–543.
[2] Jimmy Ba and Rich Caruana. 2014. Do Deep Nets Really Need to be Deep?. In *NIPS*. 2654–2662.
[3] Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer Normalization. *CoRR* abs/1607.06450 (2016).
[4] Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The Long-Document Transformer. *CoRR* abs/2004.05150 (2020).
[5] Michael Bendersky and Oren Kurland. 2008. Utilizing Passage-Based Language Models for Document Retrieval. In *ECIR (Lecture Notes in Computer Science, Vol. 4956)*. Springer, 162–174.
[6] Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. 2013. Representation Learning: A Review and New Perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.* 35, 8 (2013), 1798–1828.
[7] Leonid Boytsov, Tianyi Lin, Fangwei Gao, Yutian Zhao, Jeffrey Huang, and Eric Nyberg. 2022. Understanding Performance of Long-Document Ranking Models through Comprehensive Evaluation and Leaderboarding. *CoRR* abs/2207.01262 (2022).
[8] James P. Callan. 1994. Passage-Level Evidence in Document Retrieval. In *SIGIR*. ACM/Springer, 302–310.
[9] M. Catena, O. Frieder, Cristina Ioana Muntean, F. Nardini, R. Perego, and N. Tonellotto. 2019. Enhanced News Retrieval: Passages Lead the Way! *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (2019).
[10] Xuanang Chen, Ben He, Kai Hui, Le Sun, and Yingfei Sun. 2020. Simplified TinyBERT: Knowledge Distillation for Document Retrieval. *CoRR* abs/2009.07531 (2020).
[11] Xuanang Chen, Ben He, Kai Hui, Yiran Wang, Le Sun, and Yingfei Sun. 2021. Contextualized Offline Relevance Weighting for Efficient and Effective Neural Retrieval. In *SIGIR*. ACM, 1617–1621.
[12] Xuanang Chen, Ben He, Le Sun, and Yingfei Sun. 2020. ICIP at TREC-2020 Deep Learning Track. In *TREC (NIST Special Publication, Vol. 1266)*. National Institute of Standards and Technology (NIST).
[13] Xuanang Chen, Canjia Li, Ben He, and Yingfei Sun. 2019. UCAS at TREC-2019 Deep Learning Track. In *TREC*.
[14] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating Long Sequences with Sparse Transformers. *CoRR* abs/1904.10509 (2019).
[15] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. In *ICLR*. OpenReview.net.

[16] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2020. Overview of the TREC 2020 deep learning track. In *TREC*.

[17] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Jimmy Lin. 2022. Overview of the TREC 2021 deep learning track. In *Text REtrieval Conference (TREC)*. TREC. https://www.microsoft.com/en-us/research/publication/overview-of-the-trec-2021-deep-learning-track/

[18] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. 2019. Overview of the TREC 2019 deep learning track. In *TREC*.

[19] Zhuyun Dai and Jamie Callan. 2019. Context-Aware Sentence/Passage Term Importance Estimation For First Stage Retrieval. *CoRR* abs/1910.10687 (2019).

[20] Zhuyun Dai and Jamie Callan. 2019. Deeper Text Understanding for IR with Contextual Neural Language Modeling. In *SIGIR*. ACM, 985–988.

[21] Zhuyun Dai, Chenyan Xiong, Jamie Callan, and Zhiyuan Liu. 2018. Convolutional Neural Networks for Soft-Matching N-Grams in Ad-hoc Search. In *WSDM*. ACM, 126–134.

[22] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*.

[23] Yixing Fan, Jiafeng Guo, Yanyan Lan, Jun Xu, Chengxiang Zhai, and Xueqi Cheng. 2018. Modeling Diverse Relevance Patterns in Ad-hoc Retrieval. In *SIGIR*. ACM, 375–384.

[24] Hui Fang, Tao Tao, and Chengxiang Zhai. 2011. Diagnostic Evaluation of Information Retrieval Models. *ACM Trans. Inf. Syst.* 29, 2, Article 7 (2011), 42 pages.

[25] Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE v2: Sparse lexical and expansion model for information retrieval. *arXiv preprint arXiv:2109.10086* (2021).

[26] Luyu Gao, Zhuyun Dai, and Jamie Callan. 2020. Understanding BERT Rankers Under Distillation. In *Proceedings of the ACM International Conference on the Theory of Information Retrieval (ICTIR 2020)*.

[27] Luyu Gao, Zhuyun Dai, and James P. Callan. 2020. EARL: Speedup Transformer-based Rankers with Pre-computed Representation. *ArXiv* abs/2004.13313 (2020).

[28] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. 2016. A Deep Relevance Matching Model for Ad-hoc Retrieval. In *CIKM*. ACM, 55–64.

[29] William Hersh, Aaron Cohen, Lynn Ruslen, and Phoebe Roberts. 2007. TREC 2007 Genomics Track Overview. In *TREC*.

[30] William Hersh, Aaron M. Cohen, Phoebe Roberts, and Hari Krishna Rekapalli. 2006. TREC 2006 Genomics Track Overview. In *TREC*.

[31] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the Knowledge in a Neural Network. *CoRR* abs/1503.02531 (2015).

[32] Sebastian Hofstätter, Sophia Althammer, Michael Schröder, Mete Sertkan, and Allan Hanbury. 2020. Improving Efficient Neural Ranking Models with Cross-Architecture Knowledge Distillation. *CoRR* abs/2010.02666 (2020).

[33] Sebastian Hofstätter, Aldo Lipani, Sophia Althammer, Markus Zlabinger, and Allan Hanbury. 2021. Mitigating the Position Bias of Transformer Models in Passage Re-ranking. In *ECIR (1) (Lecture Notes in Computer Science, Vol. 12656)*. Springer, 238–253.

[34] Sebastian Hofstätter, Hamed Zamani, Bhaskar Mitra, Nick Craswell, and Allan Hanbury. 2020. Local Self-Attention over Long Text for Efficient Document Retrieval. In *SIGIR*. ACM, 2021–2024.

[35] Sebastian Hofstätter, Markus Zlabinger, and Allan Hanbury. 2019. TU Wien @ TREC Deep Learning '19 - Simple Contextualization for Re-ranking. In *TREC*.

[36] Sebastian Hofstätter, Markus Zlabinger, and Allan Hanbury. 2020. Interpretable & Time-Budget-Constrained Contextualization for Re-Ranking. *CoRR* abs/2002.01854 (2020).

[37] Sebastian Hofstätter, Markus Zlabinger, Mete Sertkan, Michael Schröder, and Allan Hanbury. 2020. Fine-Grained Relevance Annotations for Multi-Task Document Ranking and Question Answering. In *CIKM*. ACM, 3031–3038.

[38] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional Neural Network Architectures for Matching Natural Language Sentences. In *NIPS*. 2042–2050.

[39] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry P. Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *CIKM*. ACM, 2333–2338.

[40] Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. 2017. PACRR: A Position-Aware Neural IR Model for Relevance Matching. In *EMNLP*. Association for Computational Linguistics, 1049–1058.

[41] Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. 2018. Co-PACRR: A Context-Aware Neural IR Model for Ad-hoc Retrieval. In *WSDM*. ACM, 279–287.

[42] Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux, and Jason Weston. 2020. Poly-encoders: Architectures and Pre-training Strategies for Fast and Accurate Multi-sentence Scoring. In *ICLR*. OpenReview.net.

[43] Jyun-Yu Jiang, Chenyan Xiong, Chia-Jung Lee, and Wei Wang. 2020. Long Document Ranking with Query-Directed Sparse Transformer. In *EMNLP (Findings)*. Association for Computational Linguistics, 4594–4605.

[44] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2019. TinyBERT: Distilling BERT for Natural Language Understanding. *CoRR* abs/1909.10351 (2019).

[45] Mostafa Keikha, Jae Hyun Park, and W Bruce Croft. 2014. Evaluating answer passages using summarization measures. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*. 963–966.

[46] Mostafa Keikha, Jae Hyun Park, W Bruce Croft, and Mark Sanderson. 2014. Retrieving passages and finding answers. In *Proceedings of the 2014 Australasian Document Computing Symposium*. 81–84.

[47] Omar Khattab and Matei Zaharia. 2020. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. In *SIGIR*.

[48] Youngwoo Kim, Razieh Rahimi, Hamed R. Bonab, and James Allan. 2021. Query-driven Segment Selection for Ranking Long Documents. In *CIKM*. ACM, 3147–3151.

[49] Julien Knafou, Matthew Jeffryes, Sohrab Ferdowsi, and Patrick Ruch. 2020. SIB Text Mining at TREC 2020 Deep Learning Track. In *TREC (NIST Special Publication, Vol. 1266)*. National Institute of Standards and Technology (NIST).

[50] Victor Lavrenko and W. Bruce Croft. 2001. Relevance-Based Language Models. In *SIGIR*. ACM, 120–127.

[51] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics* 36, 4 (2020), 1234–1240.

[52] Canjia Li, Yingfei Sun, Ben He, Le Wang, Kai Hui, Andrew Yates, Le Sun, and Jungang Xu. 2018. NPRF: A Neural Pseudo Relevance Feedback Framework for Ad-hoc Information Retrieval. In *EMNLP*. Association for Computational Linguistics, 4482–4491.

[53] Canjia Li and Andrew Yates. [n.d.]. MPII at the TREC 2020 Deep Learning Track. ([n. d.]).

[54] Canjia Li and Andrew Yates. 2020. MPII at the NTCIR-15 WWW-3 Task. In *Proceedings of NTCIR-15*.

[55] Minghan Li and Éric Gaussier. 2021. KeyBLD: Selecting Key Blocks with Local Pre-ranking for Long Document Information Retrieval. In *SIGIR*. ACM, 2207–2211.

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

[57] Jimmy Lin, Rodrigo Nogueira, and Andrew Yates. 2020. Pretrained transformers for text ranking: Bert and beyond. *arXiv preprint arXiv:2010.06467* (2020).

[58] Jimmy J. Lin. 2009. Is searching full text more effective than searching abstracts? *BMC Bioinform.* 10 (2009).

[59] Xiaoyong Liu and W. Bruce Croft. 2002. Passage retrieval based on language models. In *CIKM*. ACM, 375–382.

[60] Yang Liu and Mirella Lapata. 2019. Hierarchical Transformers for Multi-Document Summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 5070–5081.

[61] Zhiyuan Liu, Yankai Lin, and Maosong Sun. 2020. *Representation Learning for Natural Language Processing*. Springer.

[62] Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. 2021. Sparse, Dense, and Attentional Representations for Text Retrieval. *Transactions of the Association for Computational Linguistics* 9 (04 2021), 329–345. https://doi.org/10.1162/tacl_a_00369

[63] Xinyu Ma, Jiafeng Guo, Ruqing Zhang, Yixing Fan, Xiang Ji, and Xueqi Cheng. 2021. PROP: Pre-training with Representative Words Prediction for Ad-hoc Retrieval. In *WSDM*. ACM, 283–291.

[64] Xinyu Ma, Jiafeng Guo, Ruqing Zhang, Yixing Fan, Yingyan Li, and Xueqi Cheng. 2021. B-PROP: Bootstrapped Pre-training with Representative Words Prediction for Ad-hoc Retrieval. In *SIGIR*. ACM, 1318–1327.

[65] Sean MacAvaney, Franco Maria Nardini, Raffaele Perego, Nicola Tonellotto, Nazli Goharian, and Ophir Frieder. 2020. Efficient Document Re-Ranking for Transformers by Precomputing Term Representations. In *SIGIR*.

[66] Sean MacAvaney, Franco Maria Nardini, Raffaele Perego, Nicola Tonellotto, Nazli Goharian, and Ophir Frieder. 2020. Expansion via Prediction of Importance with Contextualization. In *SIGIR*.

[67] Sean MacAvaney, Andrew Yates, Arman Cohan, and Nazli Goharian. 2019. CEDR: Contextualized Embeddings for Document Ranking. In *SIGIR*. ACM, 1101–1104.

[68] Antonio Mallia, Omar Khattab, Torsten Suel, and Nicola Tonellotto. 2021. Learning passage impacts for inverted indexes. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1723–1727.

[69] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to information retrieval*. Cambridge university press.

[70] Bhaskar Mitra and Nick Craswell. 2019. DUET AT TREC 2019 DEEP LEARNING TRACK. In *TREC (NIST Special Publication, Vol. 1250)*. National Institute of Standards and Technology (NIST).

[71] Bhaskar Mitra, Fernando Diaz, and Nick Craswell. 2017. Learning to Match using Local and Distributed Representations of Text for Web Search. In *WWW*. ACM, 1291–1299.

[72] Bhaskar Mitra, Sebastian Hofstätter, Hamed Zamani, and Nick Craswell. 2020. Conformer-Kernel with Query Term Independence for Document Retrieval. *CoRR* abs/2007.10434 (2020).

[73] Thong Nguyen, Sean MacAvaney, and Andrew Yates. 2023. A Unified Framework for Learned Sparse Retrieval. In *Advances in Information Retrieval: 45th European Conference on Information Retrieval, ECIR 2023, Dublin, Ireland, April 2–6, 2023, Proceedings, Part III.* Springer, 101–116.

[74] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. In *CoCo@NIPS (CEUR Workshop Proceedings, Vol. 1773).* CEUR-WS.org.

[75] Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage Re-ranking with BERT. *CoRR* abs/1901.04085 (2019).

[76] Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document Ranking with a Pretrained Sequence-to-Sequence Model. In *Findings of EMNLP.*

[77] Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. 2019. Document Expansion by Query Prediction. *CoRR* abs/1904.08375 (2019).

[78] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2016. Text Matching as Image Recognition. In *AAAI.* AAAI Press, 2793–2799.

[79] Ronak Pradeep, Xueguang Ma, Xinyu Zhang, Hang Cui, Ruizhou Xu, Rodrigo Frassetto Nogueira, and Jimmy Lin. 2020. H2oloo at TREC 2020: When all you got is a hammer... Deep Learning, Health Misinformation, and Precision Medicine. In *TREC (NIST Special Publication, Vol. 1266).* National Institute of Standards and Technology (NIST).

[80] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *CoRR* abs/1910.10683 (2019).

[81] Fiana Raiber and Oren Kurland. 2020. The Technion at the WWW-3 Task: Cluster-Based Document Retrieval. *Proceedings of NTCIR-15* (2020), 247–248.

[82] Stephen E. Robertson and Steve Walker. 1994. Some Simple Effective Approximations to the 2-Poisson Model for Probabilistic Weighted Retrieval. In *SIGIR.* ACM/Springer, 232–241.

[83] Stephen E. Robertson, Steve Walker, Micheline Hancock-Beaulieu, Mike Gatford, and A. Payne. 1995. Okapi at TREC-4. In *TREC.*

[84] Joseph Rocchio. 1971. Relevance feedback in information retrieval. *The Smart retrieval system-experiments in automatic document processing* (1971), 313–323.

[85] Tetsuya Sakai, Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Topic Set Size Design with the Evaluation Measures for Short Text Conversation. In *AIRS (Lecture Notes in Computer Science, Vol. 9460).* Springer, 319–331.

[86] Tetsuya Sakai, Sijie Tao, Zhaohao Zeng, Yukun Zheng, Jiaxin Mao, Zhumin Chu, Yiqun Liu, Maria Maistro, Zhicheng Dou, Nicola Ferro, and Ian Soboroff. 2020. Overview of the NTCIR-15 We Want Web with CENTRE (WWW-3) Task. In *Proceedings of NTCIR-15.* to appear.

[87] Dominik Scherer, Andreas C. Müller, and Sven Behnke. 2010. Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition. In *ICANN (3) (Lecture Notes in Computer Science, Vol. 6354).* Springer, 92–101.

[88] Eilon Sheetrit, Anna Shtok, and Oren Kurland. 2020. A passage-based approach to learning to rank documents. *Inf. Retr. J.* 23, 2 (2020), 159–186.

[89] Kohei Shinden, Atsuki Maruta, and Makoto P Kato. 2020. KASYS at the NTCIR-15 WWW-3 Task. *Proceedings of NTCIR-15* (2020), 235–238.

[90] Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. Patient Knowledge Distillation for BERT Model Compression. In *EMNLP.*

[91] Amir Vakili Tahami, Kamyar Ghajar, and Azadeh Shakery. 2020. Distilling Knowledge for Fast Retrieval-based Chat-bots. *CoRR* abs/2004.11045 (2020).

[92] Raphael Tang, Yao Lu, Linqing Liu, Lili Mou, Olga Vechtomova, and Jimmy Lin. 2019. Distilling Task-Specific Knowledge from BERT into Simple Neural Networks. *CoRR* abs/1903.12136 (2019).

[93] Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-Read Students Learn Better: The Impact of Student Initialization on Knowledge Distillation. *CoRR* abs/1908.08962 (2019).

[94] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NIPS.* 5998–6008.

[95] Wei Wang, Bin Bi, Ming Yan, Chen Wu, Jiangnan Xia, Zuyi Bao, Liwei Peng, and Luo Si. 2020. StructBERT: Incorporating Language Structures into Pre-training for Deep Language Understanding. In *ICLR.* OpenReview.net.

[96] Chuhan Wu, Fangzhao Wu, Tao Qi, and Yongfeng Huang. 2021. Hi-Transformer: Hierarchical Interactive Transformer for Efficient and Effective Long Document Modeling. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers).* Association for Computational Linguistics, Online, 848–853. `https://doi.org/10.18653/v1/2021.acl-short.107`

[97] Zhijing Wu, Jiaxin Mao, Yiqun Liu, Jingtao Zhan, Yukun Zheng, Min Zhang, and Shaoping Ma. 2020. Leveraging Passage-level Cumulative Gain for Document Ranking. In *WWW*. ACM / IW3C2, 2421–2431.

[98] Zhijing Wu, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. 2019. Investigating Passage-level Relevance and Its Role in Document-level Relevance Judgment. In *SIGIR*. ACM, 605–614.

[99] Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017. End-to-End Neural Ad-hoc Ranking with Kernel Pooling. In *SIGIR*. ACM, 55–64.

[100] Ming Yan, Chenliang Li, Chen Wu, Bin Bi, Wei Wang, Jiangnan Xia, and Luo Si. 2019. IDST at TREC 2019 Deep Learning Track: Deep Cascade Ranking with Generation-based Document Expansion and Pre-trained Language Modeling. In *TREC*.

[101] Liu Yang, Mingyang Zhang, Cheng Li, Michael Bendersky, and Marc Najork. 2020. Beyond 512 Tokens: Siamese Multi-depth Transformer-based Hierarchical Encoder for Long-Form Document Matching. In *CIKM*. ACM, 1725–1734.

[102] Peilin Yang, Hui Fang, and Jimmy Lin. 2018. Anserini: Reproducible Ranking Baselines Using Lucene. *J. Data and Information Quality* 10, 4 (2018), 16:1–16:20.

[103] Wei Yang, Kuang Lu, Peilin Yang, and Jimmy Lin. 2019. Critically Examining the "Neural Hype": Weak Baselines and the Additivity of Effectiveness Gains from Neural Ranking Models. In *SIGIR*. ACM, 1129–1132.

[104] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical Attention Networks for Document Classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 1480–1489.

[105] Andrew Yates, Kevin Martin Jose, Xinyu Zhang, and Jimmy Lin. 2020. Flexible IR pipelines with Capreolus. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 3181–3188.

[106] Zeynep Akkalyoncu Yilmaz, Shengjin Wang, Wei Yang, Haotian Zhang, and Jimmy Lin. 2019. Applying BERT to Document Retrieval with Birch. In *EMNLP*.

[107] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big Bird: Transformers for Longer Sequences. In *NeurIPS*.

[108] Hamed Zamani, Bhaskar Mitra, Xia Song, Nick Craswell, and Saurabh Tiwary. 2018. Neural Ranking Models with Multiple Document Fields. In *WSDM*. ACM, 700–708.

[109] Xingxing Zhang, Furu Wei, and Ming Zhou. 2019. HIBERT: Document Level Pre-training of Hierarchical Bidirectional Transformers for Document Summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 5059–5069.

[110] Xinyu Zhang, Andrew Yates, and Jimmy Lin. 2021. Comparing Score Aggregation Approaches for Document Retrieval with Pretrained Transformers. In *Advances in Information Retrieval: 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28 – April 1, 2021, Proceedings, Part II*. Springer-Verlag, Berlin, Heidelberg, 150–163. https://doi.org/10.1007/978-3-030-72240-1_11

[111] Chen Zhao, Chenyan Xiong, Corby Rosset, Xia Song, Paul N. Bennett, and Saurabh Tiwary. 2020. Transformer-XH: Multi-Evidence Reasoning with eXtra Hop Attention. In *ICLR*. OpenReview.net.

[112] Tiancheng Zhao, Xiaopeng Lu, and Kyusong Lee. 2020. Sparta: Efficient open-domain question answering via sparse transformer matching retrieval. *arXiv preprint arXiv:2009.13013* (2020).

[113] Jie Zhou, Xu Han, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2019. GEAR: Graph-based Evidence Aggregating and Reasoning for Fact Verification. In *ACL (1)*. Association for Computational Linguistics, 892–901.

[114] Shengyao Zhuang and Guido Zuccon. 2021. TILDE: Term independent likelihood moDEl for passage re-ranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1483–1492.