PARADE: Passage Representation Aggregation for Document Reranking

Canjia Li[1,3*], Andrew Yates[2], Sean MacAvaney[4], Ben He[1,3], Yingfei Sun[1]
[1] University of Chinese Academy of Sciences, Beijing, China
[2] Max Planck Institute for Informatics, Saarbrücken, Germany
[3] Institute of Software, Chinese Academy of Sciences, Beijing, China
[4] IR Lab, Georgetown University, Washington, DC, USA
licanjia17@mails.ucas.ac.cn, ayates@mpi-inf.mpg.de
sean@ir.cs.georgetown.edu, {benhe, yfsun}@ucas.ac.cn

Abstract

We present PARADE, an end-to-end Transformer-based model that considers document-level context for document reranking. PARADE leverages passage-level relevance representations to predict a document relevance score, overcoming the limitations of previous approaches that perform inference on passages independently. Experiments on two ad-hoc retrieval benchmarks demonstrate PARADE's effectiveness over such methods. We conduct extensive analyses on PARADE's efficiency, highlighting several strategies for improving it. When combined with knowledge distillation, a PARADE model with 72% fewer parameters achieves effectiveness competitive with previous approaches using BERT-Base. Our code is available at https://github.com/canjiali/PARADE.

1 Introduction

Pre-trained language models (PLMs), e.g., BERT (Devlin et al., 2019), ELECTRA (Clark et al., 2020) and T5 (Raffel et al., 2019), have achieved state-of-the-art results on standard ad-hoc retrieval benchmarks and in many NLP tasks. The success of PLMs mainly relies on learning contextualized representations of input sequences using the Transformer (Vaswani et al., 2017). The Transformer uses a self-attention mechanism whose computational complexity is quadratic with respect to the input sequence's length, so 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. (Dai and Callan, 2019b; Yilmaz et al., 2019). The max or k-max passage scores (e.g., top 3) are then aggregated to produce a document relevance score. Such 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* (Robertson and Walker, 1994) 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 (Hui et al., 2018). 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 (Fang et al., 2011). 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.

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 (Wu et al., 2019). Additionally, experiments in several works suggest that aggregating passage-level relevance scores to predict the document's relevance score outperforms the common practice of using the maximum passage's score (Bendersky and Kur-

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

land, 2008; Fan et al., 2018; Ai et al., 2018).

In this work, we study how PLMs like BERT can be applied to the ad-hoc document ranking task while preserving many document-level signals. To this end, we propose PARADE, an end-to-end document reranking model. PARADE predicts a document's relevance by learning passage-level relevance representations that are aggregated in a way that preserves document-level context. These aggregation approaches include 1) a passage weighting method, 2) a pooling technique, and 3) using the Transformer in a hierarchical way. PARADE is optimized end-to-end at the document level, which eliminates the noise introduced by using the document relevance label as a proxy for passage relevance labels. Since the utilization of full-text causes more memory usage, we investigate using knowledge distillation to create smaller, more efficient PARADE models that remain effective.

In the recent TREC-COVID challenge that studies the problem of identifying literature relevant to COVID-19 information needs, PARADE performed well and was among the top positions in the second round (as measured by nDCG@10). The details of our TREC-COVID submissions can be found in Appendix A.1.

Our contributions are threefold:

- The proposal of the end-to-end PARADE method for predicting a document's relevance by aggregating passage representations,
- An evaluation on standard TREC ad-hoc benchmark collections confirming the effectiveness of our approach, and
- Analyses of how PARADE's efficiency can be improved by decreasing the model size, and of how its effectiveness is influenced by the number of passages considered and by the initial ranking method.

2 Related Work

We review three lines of related research.

Contextualized Language Models for IR. Neural IR models like DSSM (Huang et al., 2013), DRMM (Guo et al., 2016), (Co-)PACRR (Hui et al., 2017, 2018), and (Conv-)KNRM (Xiong et al., 2017; Dai et al., 2018) have been proposed for the ad-hoc retrieval task. However, their contextual capacity is limited by using pre-trained unigram embeddings. Benefiting from BERT's pre-trained contextual embeddings, BERT-based IR models have been shown to be superior to neural IR models. Nogueira et al. first adopted BERT to passage reranking tasks (Nogueira and Cho, 2019) using BERT's `[CLS]` vector. Birch (Yilmaz et al., 2019) and BERT-MaxP (Dai and Callan, 2019b) explore the sentence-level and passage-level relevance signals using BERT for document reranking, respectively. CEDR proposed a joint approach that combines BERTs outputs with existing neural IR models (MacAvaney et al., 2019). Other researchers trade off PLM effectiveness for efficiency by utilizing the PLM to improve document indexing (Nogueira et al., 2019; Dai and Callan, 2019a), pre-computing intermediate Transformer representations (Khattab and Zaharia, 2020; MacAvaney et al., 2020a; Gao et al., 2020; Humeau et al., 2020), using the PLM to build sparse representations (MacAvaney et al., 2020b), or reducing the number of Transformer layers (Hofstätter et al., 2020b,a).

While several works have recently investigated approaches for improving the Transformer's efficiency by reducing the computational complexity of its attention module, none of these approaches have been shown to be effective for the document ranking task. The Sparse Transformer (Child et al., 2019) and Reformer (Kitaev et al., 2020) focus on text generation. We were unable to effectively use Transformer-XL (Dai et al., 2019) in pilot experiments, while Longformer (Beltagy et al., 2020) is an interesting contemporaneous work. We note that PARADE could be used in conjunction with such models.

Passage-based Document Retrieval. Given the increasing lengths of documents in full-text collections, Callan first experimented with paragraph-based and window-based methods of defining passages (Callan, 1994). Several works drive passage-based document retrieval in the language modeling context (Liu and Croft, 2002; Bendersky and Kurland, 2008), indexing context (Lin, 2009), and learning to rank context (Sheetrit et al., 2020). 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 (Fan et al., 2018). Others have investigated sophisticated evidence aggregation approaches (Zhao et al., 2020; Zhou et al., 2019). Wu et al. explicitly modeled the importance of passages based on position decay, passage length, length with position decay, exact match, etc (Wu et al., 2019). In a contemporaneous study, they proposed a model that considers passage-level representations of relevance in order to predict the passage-level cumulative gain of each passage (Wu et al., 2020). In this approach the final passage's cumulative gain can be used as the document-level cumulative gain. Our approaches share some similarities, but theirs differs in that they use passage-level labels to train their model and perform passage representation aggregation using a LSTM.

Knowledge Distillation. Knowledge distillation is the process of transferring knowledge from a large model to a smaller student model (Ba and Caruana, 2014; Hinton et al., 2015). 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 (Jiao et al., 2019; Sun et al., 2019). Turc et al. pre-train a family of compact BERT models and explore transferring task knowledge from large fine-tuned models (Turc et al., 2019). Tang et al. distill knowledge from the BERT model into Bi-LSTM (Tang et al., 2019). Tahami et al. propose a new cross-encoder architecture and transfer knowledge from this model to a bi-encoder model for fast retrieval (Tahami et al., 2020). We demonstrate this approach can be applied to PARADE to improve efficiency without substantial reductions in effectiveness.

3 Method

In this section, we present the proposed PARADE method for end-to-end document reranking. 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.

Representing a Document as Passages. As introduced in Section 1, a long document cannot be considered directly by the BERT model due to its fixed sequence length limitation. Following (Dai and Callan, 2019b; Callan, 1994), we split a document into passages that can be handled by BERT individually. To do so, a sliding window of 150 words is applied to the document with a stride of 100 words, formally expressed as $D = \{P_1, \ldots, P_n\}$ where n is the number of passages. Afterward, these passages are taken as input to the BERT model for relevance estimation.

Creating Passage Relevance Representations. Following prior work (Nogueira and Cho, 2019), we concatenate a pair of query q and passage P_i 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:

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

Aggregating Passage Relevance 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 three different ways, coined as PARADE$_{\text{Max}}$, PARADE$_{\text{Attn}}$, and PARADE.

PARADE$_{\text{Max}}$ utilizes a robust max-pooling operation on the passage relevance features[1] in D^{cls}. As widely applied in Convolution Neural Network, max-pooling has been shown to be effective in obtaining position-invariant features (Scherer et al., 2010). 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$_{\text{Attn}}$ 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:

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

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

where $\mathrm{softmax}$ is the normalization function and $W \in \mathcal{R}^d$ is a learnable weight. For completeness of study, we also introduce a **PARADE$_{\text{Avg}}$** that simply averages the passage relevance representations. This can be regarded as manually assigning equal weights to all passages (i.e., $w_i = 1/n$).

PARADE$_{\text{Transformer}}$, which as shorthand we simply call **PARADE**, enables passage relevance

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

representations to interact by adopting the Transformer (Vaswani et al., 2017) in a hierarchical way. Specifically, BERT's `[CLS]`[2] 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 Transformer layers to exploit the ordering of and dependencies among passages. That is,

$$h = \mathrm{LayerNorm}(x^l + \mathrm{MultiHead}(x^l)) \quad (5)$$

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

where LayerNorm is the layer-wise normalization as introduced in (Ba et al., 2016), MultiHead is the multi-head self-attention (Vaswani et al., 2017), and FFN is a two-layer feed-forward network with a ReLu activation in between.

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}. The sequence length of the Transformer layers in PARADE is equal to the number of passages used in a document, usually dozens, hence this approach adds only a small amount of computation compared with PARADE$_{\text{Attn}}$ and PARADE$_{\text{Max}}$.

Generating the Relevance Score. For all three PARADE variants, after obtaining the final d^{cls} embedding, a single-layer feed-forward network is adopted to generate a relevance score, as follows:

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

where $W_d \in \mathcal{R}^d$ is a learnable weight.

4 Experiments

4.1 Dataset

We experiment with two ad-hoc collections: Robust04[3] and GOV2[4]. Both are common TREC benchmarks. Robust04 is a newswire collection used by the TREC 2004 Robust track. GOV2 is a Web collection crawled from government Websites. We consider both keyword (title) queries and description queries in our experiments. The statistics of these two datasets are shown in Table 1. Note that the average document length is obtained only from the documents returned by BM25. Documents in GOV2 are much longer than Robust04, making it more challenging to train an end-to-end ranker.

Collection	# Queries	# Documents	# tokens / doc
Robust04	249	0.5M	0.7k
GOV2	149	25M	3.8k

Table 1: Collection statistics.

4.2 Baselines

We compare PARADE against the following traditional and neural baselines:

BM25 is an unsupervised ranking model based on IDF-weighted counting (Robertson et al., 1995). 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 (Lavrenko and Croft, 2001). We used Anserini's (Yang et al., 2018) 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 (MS) and **Birch (MS→ MB)** aggregate sentence-level evidence provided by BERT to rank documents (Yilmaz et al., 2019). Birch (MS) is the fine-tuned BERT-Large model on the MSMARCO passage dataset while Birch (MS→MB) is further fine-tuned on TREC MicroBlog datasets. We use BM25 rather than BM25+RM3 as an initial ranking method for a fair comparison.

BERT-MaxP (MS) adopts the maximum score of passages within a document as an overall relevance score (Dai and Callan, 2019b). However, rather than fine-tuning BERT-base on a Bing search log, we improve performance by fine-tuning on the MSMARCO passage ranking dataset.

4.3 Training PARADE

To prepare the BERT model for the ranking task, we first fine-tune BERT on the MSMARCO passage ranking dataset (Nguyen et al., 2016). The fine-tuned BERT model is then used to initialize PARADE's BERT component. Training of PARADE was performed on a single Google TPU v3-8 using a cross entropy loss where $rel(q, D)$ in Equation 7 is the logits. We train on the top 1,000 documents returned by BM25; documents that are labeled relevant in the ground-truth are taken as positive samples and all other documents as negative samples. We train PARADE for 3 epochs with batches of 32 training instances. Each instance

[2] In our pilot study, there is no difference among using `[CLS]`, `[SEP]`, and `[UNK]` as the prepended token in x^l.

[3] `https://trec.nist.gov/data/qa/T8_QAdata/disks4_5.html`

[4] `http://ir.dcs.gla.ac.uk/test_collections/gov2-summary.htm`

	Robust04				GOV2			
	Title		Description		Title		Description	
Model	P@20	nDCG@20	P@20	nDCG@20	P@20	nDCG@20	P@20	nDCG@20
BM25	0.3631	0.4240	0.3345	0.4058	0.5362	0.4774	0.4705	0.4264
BM25+RM3	0.3821	0.4407	0.3661	0.4255	0.5634	0.4851	0.4966	0.4212
Birch (MS)	0.3616	0.4227	0.3341	0.4053	0.5352	0.4722	0.4701	0.4260
Birch (MS→MB)	0.4404	0.5137	0.4211	0.5069	0.6409	0.5608	0.5973	0.5307
BERT-MaxP (MS)	0.4277	0.4931	0.4522	0.5453	0.6356	0.5600	0.6087	0.5506
PARADE$_{\text{Avg}}$	0.4251†	0.4917†	0.4482†‡	0.5324†‡	0.6107†	0.5362†	0.5872†	0.5288†
PARADE$_{\text{Max}}$	0.4432†§	0.5115†§	0.4657†‡§	0.5487†‡	0.6319†	0.5399†	0.6148†	0.5419†
PARADE$_{\text{Attn}}$	0.4410†§	0.5134†§	0.4614†‡§	0.5517†‡	0.6319†	0.5554†	0.6198†‡	0.5513†
PARADE	**0.4486**†§	**0.5252**†§	**0.4661**†‡§	**0.5605**†‡§	**0.6530**†§	**0.5750**†	**0.6299**†‡§	**0.5674**†‡
PARADE (ELECTRA)	**0.4604**†‡§	**0.5399**†‡§	**0.4717**†‡§	**0.5713**†‡§	**0.6678**†‡§	**0.5851**†‡	**0.6470**†‡§	**0.5762**†‡

Table 2: Reranking effectiveness of different models on *Robust04* and *GOV2* dataset. Best results are in **bold**. Significant improvements over Birch (MS), Birch (MS→MB) and BERT-MaxP (MS) are marked with †, ‡ and §, respectively. ($p < 0.01$, two-tailed paired t-test.)

comprises a query and all split passages in a document. We use a learning rate of 3e-6 with warm-up over the first 10 proportions of training steps. Training takes approximately 2.5 hours for each fold on the Robust04 collection. Further details on fine-tuning and hyper-parameters are available in Appendix A.3 and A.4.

4.4 Evaluation

Following prior work (Dai and Callan, 2019b; MacAvaney et al., 2019), we use 5-fold cross-validation. We set the reranking threshold to 100 on the test fold as trade-off between latency and effectiveness. The reported results are based on the average of all test folds. Performance is measured in terms of the P@20 and nDCG@20 ranking metrics using `trec_eval`[5].

4.5 Results

The reranking effectiveness of PARADE is shown in Table 2. It can be seen that the performance of PARADE$_{\text{Max}}$ and PARADE$_{\text{Attn}}$ is comparable, while nDCG@20 of PARADE$_{\text{Attn}}$ can always surpass PARADE$_{\text{Max}}$. PARADE$_{\text{Avg}}$ underperforms other models by a large margin, which confirms that passages differ in their contributions to the overall relevance of a document. PARADE consistently outperforms the other models across both datasets, suggesting that the multi-head self-attention mechanism in the Transformer is a superior method for passage-level relevance aggregation.

Compared with other baseline models, Birch has two innate advantages: it uses the BERT-Large model with 3x more parameters than BERT-Base, and it is an ensemble model that additionally considers the original ranking scores. Nevertheless, PARADE still outperform it, especially on description queries.[6] For BERT-MaxP, the reported results are better than those reported in (Dai and Callan, 2019b) with approximately a 0.02 nDCG@20 increase on Robust04 title queries. On the Robust04 collection with deeper judgments, PARADE outperforms BERT-MaxP significantly.

When applying PARADE to the more recent and efficiently trained LM model ELECTRA-Base (Clark et al., 2020), PARADE's performance increases substantially. This model significantly improves over all baselines on nDCG@20 for the Robust04 collection and P@20 for both collections. These results illustrate that as Transformer pre-training techniques advance, PARADE is able to take advantage of improved pre-trained models.

5 Analysis

In this section, we consider the following research questions:

- **RQ1:** How can BERT's efficiency be improved while maintaining its effectiveness?
- **RQ2:** How does the number of document passages preserved influence effectiveness?
- **RQ3:** Is it beneficial to rerank documents from a more effective initial ranking method? In particular, is reranking BM25+RM3 better than reranking BM25?

Additionally, we study the effectiveness of PARADE on the TREC-COVID Challenge in Ap-

[5] `https://trec.nist.gov/trec_eval`

[6] Note the Birch results presented here are lower than those in the original work, because we rerank 100 documents. PARADE continues to outperform Birch when reranking 1,000 documents in a comparable setting, as shown later in Table 5.

			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

Table 3: PARADE's effectiveness using BERT models of varying sizes on Robust04 title queries. Significant improvements of distilled over non-distilled models are marked with †. ($p < 0.01$, two-tailed paired t-test.)

pendix A.1, on the TREC 2019 DL document ranking task in Appendix A.2, and the impact of fine-tuning on different domains in Appendix A.3.

5.1 Reranking Effectiveness vs. Efficiency

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, which also answers RQ1.

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 an important guidance for deploying a retrieval system. Pre-trained BERT models of various sizes were provided by (Turc et al., 2019). From Table 3, it can be seen that as the size of models is reduced, their effectiveness decline monotonously. 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 #8 performs better; it has fewer layers but contains more parameters.

The number of parameters and inference time are also given in Table 3 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 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 (Tahami et al., 2020; Tang et al., 2019). The learning objective penalizes the student model based on both the ground-truth and the teacher model:

$$L = \alpha \cdot L_{CE} + (1 - \alpha) \cdot ||z^t - z^s||^2 \qquad (8)$$

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 3, 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. The inference time for each query is only 0.114 second by reranking top 100 documents.

5.2 Number of Passages Considered

One hyper-parameter in PARADE is the maximum number of passages being used, i.e., preserved data size, which is studied to answer RQ2 in this section. We consider title queries on the GOV2 dataset given that these documents are longer on average than in Robust04. Figure 1 depicts nDCG@20 of PARADE with the number of passages varying from 8 to 64. Generally, larger preserved data size results in better performance for PARADE, which suggests that a document can be better un-

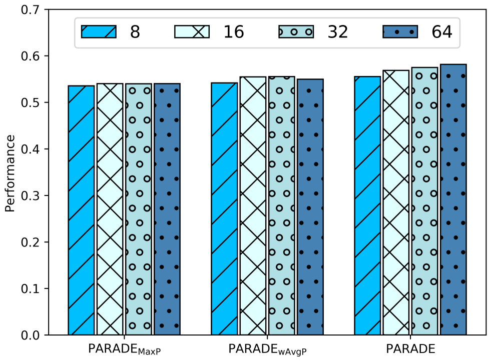


Figure 1: Reranking effectiveness of PARADE when different number of passages are being used on *Gov2* title dataset. nDCG@20 is reported.

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 4: Reranking effectiveness of PARADE 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.

derstood from document-level context with more preservation of its content. For PARADE$_{\text{Max}}$ and PARADE$_{\text{Attn}}$, 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. PARADE is able to improve nDCG@20 as the number of passages increases, demonstrating its superiority in identifying relevant and non-relevant documents 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 4, 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.

5.3 Understanding Reranking Behavior

Query expansion methods based on pseudo-relevance feedback, like RM3 (Lavrenko and Croft, 2001) and NPRF (Li et al., 2018), have been shown to increase the effectiveness of a search system. The use of PRF methods in prior work on BERT ranking models varies, however. Thus, in this section we consider the question (i.e., RQ3) of whether reranking a stronger initial ranking method (e.g., RM3) improves retrieval results. To do so, we compare the reranking effectiveness of PARADE on top of BM25 and BM25+RM3. To simplify the analysis, we focus on the ranking distribution of relevant documents. On the Robust04 dataset 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:

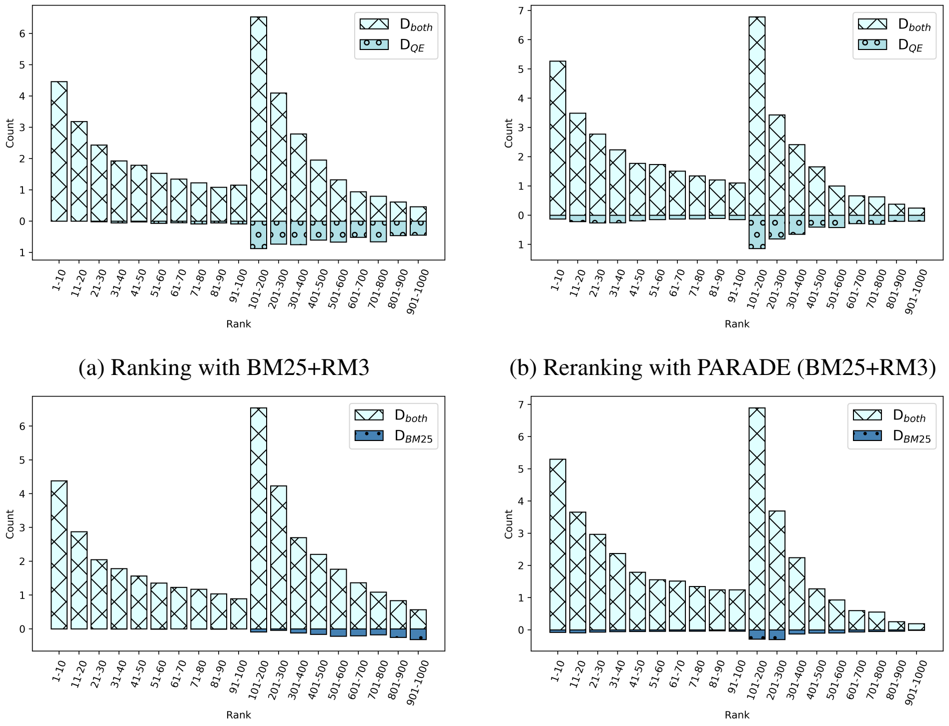
- D_{both}: the relevant documents retrieved by both BM25 and BM25+RM3
- D_{BM25}: the relevant documents retrieved by BM25 but not retrieved by BM25+RM3
- D_{QE}: the relevant documents retrieved by BM25+RM3 but not retrieved by BM25

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.

Different from the previous setting, we set the reranking threshold to 1,000 to increase recall. The most effective PARADE is adopted as a reranker. The (re-)ranking effectiveness of these models is shown in Table 5. 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 effec-

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 (BM25)	0.4996	0.6989	0.2889	0.3280	0.4562	0.5291
PARADE (BM25+RM3)	0.5058	0.7549	0.2943	0.3407	0.4548	0.5303
PARADE (BM25+RM3, Ensemble)	0.5347	0.7549	0.3167	0.3635	0.4733	0.5411

Table 5: (Re)ranking effectiveness of different models.



(a) Ranking with BM25+RM3

(b) Reranking with PARADE (BM25+RM3)

(c) Ranking with BM25

(d) Reranking with PARADE (BM25)

Figure 2: (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.

tiveness between reranking BM25 and reranking BM25+RM3 (Nogueira et al., 2020).

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 2 that depict the number of relevant documents placed in different position bins (averaged by the number of queries). Figures 2a, 2b, 2c, 2d 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 2a and 2c, 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 2a with 2b, as well as Figure 2c with 2d, 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 2c with 2d, 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 5.

The advantage of using BM25+RM3 may be that its relevance scores are good source for model ensemble. As shown in Table 5, 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. BM25+RM3 is thus more of a reranking method than an initial ranking method.

6 Conclusion

We proposed the PARADE end-to-end document reranking model and demonstrated its effectiveness on two TREC ad-hoc benchmark collections. Our results indicate the importance of incorporating diverse relevance signals from the full text into ad-hoc ranking, rather than basing it on a single passage. We additionally investigated how model size and the initial ranking method affect performance. Knowledge distillation on PARADE boosts the performance of smaller PARADE models while substantially reducing their parameters.

Acknowledgments

This work was supported in part by Google Cloud and the TensorFlow Research Cloud.

References

Qingyao Ai, Brendan O'Connor, and W. Bruce Croft. 2018. A neural passage model for ad-hoc document retrieval. In *ECIR*, volume 10772 of *Lecture Notes in Computer Science*, pages 537–543. Springer.

Jimmy Ba and Rich Caruana. 2014. Do deep nets really need to be deep? In *NIPS*, pages 2654–2662.

Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR*, abs/1607.06450.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *CoRR*, abs/2004.05150.

Michael Bendersky and Oren Kurland. 2008. Utilizing passage-based language models for document retrieval. In *ECIR*, volume 4956 of *Lecture Notes in Computer Science*, pages 162–174. Springer.

James P. Callan. 1994. Passage-level evidence in document retrieval. In *SIGIR*, pages 302–310. ACM/Springer.

Xuanang Chen, Canjia Li, Ben He, and Yingfei Sun. 2019. UCAS at TREC-2019 deep learning track. In *TREC*.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *CoRR*, abs/1904.10509.

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.

Gordon V. Cormack, Charles L A Clarke, and Stefan Buettcher. 2009. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *SIGIR*.

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

Zhuyun Dai and Jamie Callan. 2019a. Context-aware sentence/passage term importance estimation for first stage retrieval. *CoRR*, abs/1910.10687.

Zhuyun Dai and Jamie Callan. 2019b. Deeper text understanding for IR with contextual neural language modeling. In *SIGIR*, pages 985–988. ACM.

Zhuyun Dai, Chenyan Xiong, Jamie Callan, and Zhiyuan Liu. 2018. Convolutional neural networks for soft-matching n-grams in ad-hoc search. In *WSDM*, pages 126–134. ACM.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc Viet Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. In *ACL (1)*, pages 2978–2988. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Yixing Fan, Jiafeng Guo, Yanyan Lan, Jun Xu, Chengxiang Zhai, and Xueqi Cheng. 2018. Modeling diverse relevance patterns in ad-hoc retrieval. In *SIGIR*, pages 375–384. ACM.

Hui Fang, Tao Tao, and Chengxiang Zhai. 2011. Diagnostic evaluation of information retrieval models. *ACM Trans. Inf. Syst.*, 29(2).

Luyu Gao, Zhuyun Dai, and James P. Callan. 2020. EARL: Speedup transformer-based rankers with pre-computed representation. *ArXiv*, abs/2004.13313.

Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. 2016. A deep relevance matching model for ad-hoc retrieval. In *CIKM*, pages 55–64. ACM.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531.

Sebastian Hofstätter, Hamed Zamani, Bhaskar Mitra, Nick Craswell, and Allan Hanbury. 2020a. Local self-attention over long text for efficient document retrieval. In *SIGIR*, pages 2021–2024. ACM.

Sebastian Hofstätter, Markus Zlabinger, and Allan Hanbury. 2019. TU wien @ TREC deep learning '19 - simple contextualization for re-ranking. In *TREC*.

Sebastian Hofstätter, Markus Zlabinger, and Allan Hanbury. 2020b. Interpretable & time-budget-constrained contextualization for re-ranking. *CoRR*, abs/2002.01854.

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*, pages 2333–2338. ACM.

Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. 2017. PACRR: A position-aware neural IR model for relevance matching. In *EMNLP*, pages 1049–1058. Association for Computational Linguistics.

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*, pages 279–287. ACM.

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.

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.

Omar Khattab and Matei Zaharia. 2020. ColBERT: Efficient and effective passage search via contextualized late interaction over bert. In *SIGIR*.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. In *ICLR*. OpenReview.net.

Victor Lavrenko and W. Bruce Croft. 2001. Relevance-based language models. In *SIGIR*, pages 120–127. ACM.

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*, pages 4482–4491. Association for Computational Linguistics.

Jimmy J. Lin. 2009. Is searching full text more effective than searching abstracts? *BMC Bioinform.*, 10.

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

Sean MacAvaney, Franco Maria Nardini, Raffaele Perego, Nicola Tonellotto, Nazli Goharian, and Ophir Frieder. 2020a. Efficient document re-ranking for transformers by precomputing term representations. In *SIGIR*.

Sean MacAvaney, Franco Maria Nardini, Raffaele Perego, Nicola Tonellotto, Nazli Goharian, and Ophir Frieder. 2020b. Expansion via prediction of importance with contextualization. In *SIGIR*.

Sean MacAvaney, Andrew Yates, Arman Cohan, and Nazli Goharian. 2019. CEDR: contextualized embeddings for document ranking. In *SIGIR*, pages 1101–1104. ACM.

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. *CoRR*, abs/1611.09268.

Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with BERT. *CoRR*, abs/1901.04085.

Rodrigo Nogueira, Zhiying Jiang, and Jimmy Lin. 2020. Document ranking with a pre-trained sequence-to-sequence model. *CoRR*, abs/2003.06713.

Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. 2019. Document expansion by query prediction. *CoRR*, abs/1904.08375.

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.

Stephen E. Robertson and Steve Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR*, pages 232–241. ACM/Springer.

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

Dominik Scherer, Andreas C. Müller, and Sven Behnke. 2010. Evaluation of pooling operations in convolutional architectures for object recognition. In *ICANN (3)*, volume 6354 of *Lecture Notes in Computer Science*, pages 92–101. Springer.

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

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. Patient knowledge distillation for BERT model compression. In *EMNLP*.

Amir Vakili Tahami, Kamyar Ghajar, and Azadeh Shakery. 2020. Distilling knowledge for fast retrieval-based chat-bots. *CoRR*, abs/2004.11045.

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.

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.

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*, pages 5998–6008.

Ellen M. Voorhees, Tasmeer Alam, Steven Bedrick, Dina Demner-Fushman, William R. Hersh, Kyle Lo, Kirk Roberts, Ian Soboroff, and Lucy Lu Wang. 2020. TREC-COVID: constructing a pandemic information retrieval test collection. *CoRR*, abs/2005.04474.

Lucy Lu Wang, Kyle Lo, Yoganand Chandrasekhar, Russell Reas, Jiangjiang Yang, Darrin Eide, Kathryn Funk, Rodney Kinney, Ziyang Liu, William Merrill, Paul Mooney, Dewey Murdick, Devvret Rishi, Jerry Sheehan, Zhihong Shen, Brandon Stilson, Alex D. Wade, Kuansan Wang, Chris Wilhelm, Boya Xie, Douglas Raymond, Daniel S. Weld, Oren Etzioni, and Sebastian Kohlmeier. 2020a. CORD-19: the covid-19 open research dataset. *CoRR*, abs/2004.10706.

Wei Wang, Bin Bi, Ming Yan, Chen Wu, Jiangnan Xia, Zuyi Bao, Liwei Peng, and Luo Si. 2020b. Structbert: Incorporating language structures into pre-training for deep language understanding. In *ICLR*. OpenReview.net.

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.

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*, pages 605–614. ACM.

Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017. End-to-end neural adhoc ranking with kernel pooling. In *SIGIR*, pages 55–64. ACM.

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 pretrained language modeling. In *TREC*.

Peilin Yang, Hui Fang, and Jimmy Lin. 2018. Anserini: Reproducible ranking baselines using lucene. *J. Data and Information Quality*, 10(4):16:1–16:20.

Zeynep Akkalyoncu Yilmaz, Shengjin Wang, Wei Yang, Haotian Zhang, and Jimmy Lin. 2019. Applying BERT to document retrieval with birch. In *EMNLP*.

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.

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)*, pages 892–901. Association for Computational Linguistics.

A Appendices

A.1 Results on the TREC-COVID Challenge

	runid	nDCG@10	P@5	bpref	MAP
1	**mpiid5_run3**	0.6893	0.8514	0.5679	0.3380
2	**mpiid5_run2**	0.6864	0.8057	0.4943	0.3185
3	SparseDenseSciBert	0.6772	0.7600	0.5096	0.3115
4	**mpiid5_run1**	0.6677	0.7771	0.4609	0.2946
5	UIowaS_Run3	0.6382	0.7657	0.4867	0.2845

Table 6: Ranking effectivenes of different retrieval systems in the TREC-COVID Round 2.

	runid	nDCG@10	P@5	bpref	MAP
1	covidex.r3.t5_lr	0.7740	0.8600	0.5543	0.3333
2	BioInfo-run1	0.7715	0.8650	0.5560	0.3188
3	UIowaS_Rd3Borda	0.7658	0.8900	0.5778	0.3207
4	udel_fang_lambdarank	0.7567	0.8900	0.5764	0.3238
11	sparse-dense-SBrr-2	0.7272	0.8000	0.5419	0.3134
13	**mpiid5_run2**	0.7235	0.8300	0.5947	0.3193
16	**mpiid5_run1** (Fusion)	0.7060	0.7800	0.6084	0.3010
43	**mpiid5_run3** (Attn)	0.3583	0.4250	0.5935	0.2317

Table 7: Ranking effectivenes of different retrieval systems in the TREC-COVID Round 3.

	runid	nDCG@20	P@20	bpref	MAP
1	UPrrf38rrf3-r4	0.7843	0.8211	0.6801	0.4681
2	covidex.r4.duot5.lr	0.7745	0.7967	0.5825	0.3846
3	UPrrf38rrf3v2-r4	0.7706	0.7856	0.6514	0.4310
4	udel_fang_lambdarank	0.7534	0.7844	0.6161	0.3907
5	run2_Crf_A_SciB_MAP	0.7470	0.7700	0.6292	0.4079
6	run1_C_A_SciB	0.7420	0.7633	0.6256	0.3992
7	**mpiid5_run1**	0.7391	0.7589	0.6132	0.3993

Table 8: Ranking effectiveness of different retrieval systems in the TREC-COVID Round 4.

In response to the urgent demand for reliable and accurate retrieval of COVID-19 academic literature, TREC has been developing the TREC-COVID challenge to build a test collection during the pandemic (Voorhees et al., 2020). The challenge uses the CORD-19 data set (Wang et al., 2020a), which is a dynamic collection enlarged over time. There are supposed to be 5 rounds for the researchers to iterate their systems. TREC develops a set of COVID-19 related topics, including queries (keyword based), questions, and narratives. A retrieval system is supposed to generate a ranking list corresponding to these queries.

We began submitting PARADE runs to TREC-COVID from Round 2. The Round 5 results are not yet available at the time of writing. By using PARADE, we are able to utilize the full-text of the COVID-19 academic papers. We used the question topics since it works much better than other types of topics. In all rounds, we employ the full PARADE model. In Round 3, we additionally tested $\text{PARADE}_{\text{Attn}}$ and a combination of PARADE and $\text{PARADE}_{\text{Attn}}$ using reciprocal rank fusion (Cormack et al., 2009).

Results from TREC-COVID Rounds 2-4 are shown in Table 6, Table 7, and Table 8, respectively.[7] In Round 2, PARADE achieves the highest nDCG, further supporting its effectiveness.[8] In Round 3, our runs are not as competitive as the previous round. One possible reason is that the collection doubles from Round 2 to Round 3, which can introduce more inconsistencies between training and testing data as we trained PARADE on Round 2 data and tested on Round 3 data. In particular, our run `mpiid5_run3` performed poorly. We found that it tends to retrieve more documents that are not likely to be included in the judgment pool. When considering the bpref metric that takes only the judged documents into account, its performance is comparable to that of the other variants. As measured by nDCG, PARADE's performance improved in Round 4 (Table 8), but is again outperformed by other approaches. It is worth noting that the PARADE runs were created by single models (excluding the fusion run from Round 3), whereas e.g. the `UPrrf38rrf3-r4` run in Round 4 is an ensemble of more than 20 runs.

A.2 Results on the TREC 2019 DL Document Ranking Task

The MSMARCO document ranking dataset[9] is a large-scale collection and is used in TREC 2019 Deep Learning track (Craswell et al., 2019). There are 367k, 5193, and 43 queries for training, development, and test set respectively. 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 results are shown in Table 9. We include comparisons with competitive runs from TREC: `ucas_runid1` (Chen et al., 2019) used BERT-

[7] Further details and system descriptions can be found at `https://ir.nist.gov/covidSubmit/archive.html`

[8] To clarify, our run type is feedback, not manual.

[9] `https://microsoft.github.io/TREC-2019-Deep-Learning`

group	runid	MAP	nDCG@10
TREC	BM25	0.237	0.517
	ucas_runid1 (Chen et al., 2019)	0.264	0.644
	TUW19-d3-re (Hofstätter et al., 2019)	0.271	0.644
	idst_bert_r1 (Yan et al., 2019)	0.291	0.719
Ours	PARADE$_{\text{Avg}}$	0.269	0.662
	PARADE$_{\text{Max}}$	0.287	0.679
	PARADE$_{\text{Attn}}$	0.285	0.677
	PARADE	0.274	0.650

Table 9: Ranking effectiveness on TREC 2019 DL Track document task test set.

Fine-tuned model	P@20	nDCG@20
BERT-Base	0.4333	0.4970
BERT-Base (Bing)	0.4223	0.4930
BERT-Base (MSMARCO)	0.4486	0.5252
BERT-Large	0.4408	0.5046
BERT-Large (MSMARCO)	0.4508	0.5243

Table 10: Rereanking effectiveness of PARADE using different fine-tuned BERT models on *Robust04* dataset with Title queries.

MaxP (Dai and Callan, 2019b) as the reranking method, `TUW19-d3-re` (Hofstätter et al., 2019) is a Transformer-based non-BERT method, and `idst_bert_r1` (Yan et al., 2019) utilizes structBERT (Wang et al., 2020b), which is intended to strengthen the modeling of sentence relationships. 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 contrast with the previous results, the other variants outperform PARADE in this setting.

A.3 Effectiveness of Domain Adaptation

As previously described, the BERT model used in PARADE is fine-tuned on the MSMARCO passage ranking dataset before being embedded into PARADE. This training set consists of approximately 400M tuples of query, relevant passage, and nonrelevant passage. The dev set and test set consist of approximately 6,900 and 6,800 queries, respectively. For each passage, we use BERT's `[CLS]` vector as in Equation 1 to a single-layer feed-forward network to obtain the probability of being relevant. We follow the training setup in (Nogueira and Cho, 2019) and fine-tune the model with a batch size of 32 for 400k iterations. After that, the fine-tuned model is used as weight initialization in the BERT layers of PARADE.

As mentioned in (Yilmaz et al., 2019), fine-tuning BERT on different domains can result in different model effectiveness. We verify the performance of PARADE using the BERT models fine-tuned on the above mentioned MSMARCO domain as well as Bing search log[10]. Results on Robust04 when using the original BERT model, a BERT model fine-tuned on Bing search logs, and using a BERT model fine-tuned on MSMARCO are shown in Table 10. It can be seen that fine-tuning on MSMARCO outperforms the other approaches by a large margin. Considering the model size, BERT-Base shows comparable ability to BERT-Large while requiring fewer parameters.

A.4 Hyper-parameters

In our pilot study, we tune learning rates from {1e-6, 3e-6, 5e-6, 1e-5, 5e-5}, α for knowledge distillation from {0.25, 0.5, 0.75}, numbers of Transformer layers from 1 to 4, and the numbers of training epochs from 1 to 10. Then we fix the learning rate as 3e-6, the number of Transformer layers as 2, the number of training epochs as 3, and α as 0.75 for all experiments. For PARADE, the configuration of Transformer layers (e.g., number of attention heads, hidden size, etc.) is the same as the Transformer block being used in BERT.

Documents are split into passages. We set the maximum number of passages in each document as 16 and 32 for Robust04 and GOV2 respectively. As we split the documents using a sliding window of 150 words with a stride of 100 words, a maximum number of 1650 words in each document are retained on the Robust04 collection while 3250 on GOV2. The maximum sequence length in BERT is set as 256. When running PARADE, documents with less number of required passages are padded and later masked out by passage level masks. For documents longer than required, the first and last passages are always kept while the remaining are selected using a uniform sampling strategy as in (Dai and Callan, 2019b).

[10] `http://boston.lti.cs.cmu.edu/appendices/SIGIR2019-Zhuyun-Dai`