Understanding Performance of Long-Document Ranking Models through Comprehensive Evaluation and Leaderboarding

Anonymous ACL submission

Abstract

We evaluated 20+ Transformer models for ranking of long documents (including recent *LongP* models trained with FlashAttention) and compared them with a simple *FirstP* baseline, which applies the *same* model to the truncated input (at most 512 tokens). We used MS MARCO Documents v1 as a primary training set and evaluated both zero-shot transferred and fine-tuned models.

On MS MARCO, TREC DLs, and Robust04 no long-document model outperformed *FirstP* by more than 5% in NDCG and MRR (when averaged over all test sets). We conjectured this was not due to models' inability to process long context, but due to a positional bias of relevant passages, whose distribution was skewed towards the beginning of documents. We found direct evidence of this bias in some test sets, which motivated us to create *MS MARCO FarRelevant* (based on MS MARCO Passages) where the relevant passages were not present among the first 512 tokens.

Unlike standard collections where we saw *both* little benefit from incorporating longer contexts and *limited* variability in model performance (within a few %), experiments on MS MARCO FarRelevant uncovered *dramatic* differences among models. The *FirstP* models performed roughly at the random-baseline level in both zero-shot and fine-tuning scenarios. Simple aggregation models including MaxP and PARADE Attention had good zero-shot accuracy, but benefited little from fine-tuning. Most other models had poor zero-shot performance (sometimes at a random baseline level), but outstripped MaxP by as much as 13-28% after finetuning. Thus, the positional bias not only diminishes benefits of processing longer document contexts, but also leads to model overfitting to positional bias and performing poorly in a zero-shot setting when the distribution of relevant passages changes substantially. We make our software and data available.[1]

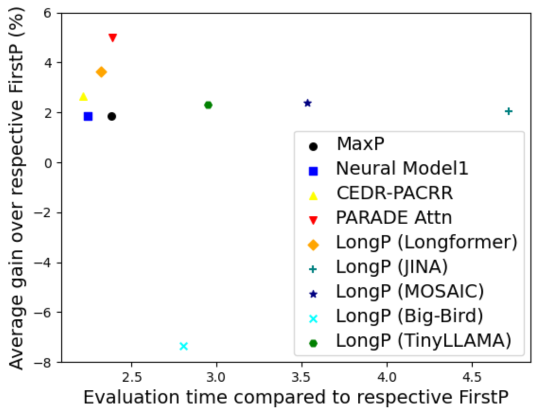

Figure 1: Average relative gain (in %) vs. relative increase in run-time compared to *respective FirstP* baselines on MS MARCO, TREC DL 2019-2021, and Robust04 (for a representative subset of models).

1 Introduction

Transformer models (Vaswani et al., 2017)—such as BERT (Devlin et al., 2019)—pretrained in a self-supervised manner considerably advanced state-of-the-art of core natural language processing (NLP) (Devlin et al., 2019; Radford et al., 2018) and information retrieval (Nogueira and Cho, 2019). However, due to quadratic cost of the self-attention with respect to an input sequence length, a number of "chunk-and-aggregate" approaches were proposed and evaluated (Dai and Callan, 2019; MacAvaney et al., 2019; Boytsov and Kolter, 2021; Li et al., 2024), but existing studies typically have *at least one* of the following shortcomings:

- Reliance *only* on *small-scale* query collections such as TREC DL (Craswell et al., 2020, 2022), Robust04 (Voorhees, 2004), and Gov2 Terabyte (Clark et al., 2005);
- Lacking *systematic* comparison with respective *FirstP* baselines, which consists in apply-

[1] https://anonymous.4open.science/r/long_doc_rank_model_analysis_v2-78E9/.

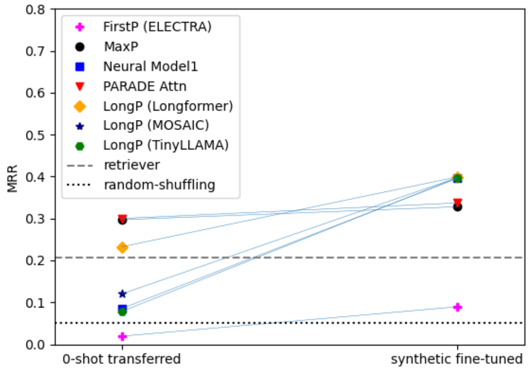

Figure 2: Zero-shot vs. fine-tuned performance on MS MARCO FarRelevant for a representative set of models.

ing the *same* model to input truncated to the first 512 tokens,

- Lacking comparison with *LongP* models—directly supporting long inputs—such as sparse-attention models Longformer and BigBird (Beltagy et al., 2020; Zaheer et al., 2020), or more recent full-attention models trained with FlashAttention (Dao et al., 2022);

- Using undisclosed seed-selection strategies, which can restrict reproducibility since there can be substantial (in the order of few %) differences due to using different seeds.

To fill this gap we evaluated over 20 recent models for ranking of long documents and carried out their systematic comparison using two popular document collections: MS MARCO Documents v1/v2 (Craswell et al., 2020) and Robust04 (Clarke et al., 2004), diverse query sets (both large and small) and multiple training seeds. We found that ranking models capable of processing long documents—including *LongP* models with sparse or full attention—showed little to no improvement compared to their *respective FirstP* baselines (which truncated documents to satisfy the input-sequence constraint of most off-the-shelf Transformer models, i.e., 512 tokens).

This finding is generally in line with previously reported results (see § B.4) and an ablation experiment showed that limited improvement over *FirstP* was not related to the choice of the backbone Transformer model (see Table 7). Furthermore, we used our best models to produce several high-ranking runs on a competitive leaderboard. This, in our view, strengthens the credibility of our evaluation.

From the efficiency-effectiveness plot in Fig. 1, we can see that all long-document models are at least 2× slower than respective *FirstP* baselines. The biggest average gain of merely 5% is achieved by the PARADE Attn model (with a BERT-base backbone) at the cost of being 2.5× slower than its *FirstP* baseline. All *LongP* models are even slower and show less improvement. Given such small benefits at the cost of a substantial slow-down, one could question practicality of such models and suggest using *FirstP* variants instead.

Our initial exploration prompted two *broad* research questions:

- **RQ1:** What is the reason for the lackluster performance of long-document models?

- **RQ2:** How much progress has the community made in improving long-document ranking models?

To answer these questions, we started with analyzing a distribution of relevant passages in the MS MARCO document collection and found evidence of a substantial positional bias, namely, relevant passages tended to appear in the beginning of documents. This finding—which partially answers **RQ1**—prompted an additional research question:

- **RQ3:** How robust are long-document models to the positional-bias of relevant passages?

To further support the relevance-bias hypothesis and answer **RQ3**, we constructed a new synthetic collection *MS MARCO FarRelevant* where relevant passages were not present among the first 512 tokens. Using MS MARCO FarRelevant, we evaluated zero-shot transferred as well as fine-tuned models and found the following (see Fig. 2):

- The *FirstP* models performed roughly at the random-baseline level in both zero-shot and fine-tuning modes (**RQ3**);

- Simple aggregation models including MaxP and PARADE Attention had good zero-shot accuracy, but benefited little from fine-tuning on MS MARCO FarRelevant (**RQ3**);

- In contrast, other long-document models had poor zero-shot performance (sometimes at a random baseline level), but outstripped *respective* MaxP baselines by as much as 13.3%-27.7% after finetuning (**RQ3**);

- Not only positional bias diminished benefits of processing longer document contexts, but it also lead to models' overfitting to the bias and performing poorly in a zero-shot setting when the distribution of relevant passages changed substantially (**RQ3**);

- Although PARADE Transformer models were more effective than other models on standard collections, their advantage was small (a few %). In contrast, on MS MARCO FarRelevant, PARADE Transformer (ELECTRA) outperformed the next competitor Longformer by 8% and PARADE Max (ELECTRA)—an early chunk-and-aggregate approach—by as much as 23.8% (**RQ2**).

Our key contributions are as follows:

- We carried a comprehensive evaluation of 20+ long-document ranking models, which included both the chunk-and-aggregate models as well as the models that directly supported long inputs (using both the standard collections MS MARCO Documents v1/v2 and Robust04 as well as the new synthetic collection MS MARCO FarRelevant);

- We contributed to the nascent field of analytical experimentation with a full control of outcomes by creating a new dataset MS MARCO FarRelevant, which we made available together with code.[2]

- Our study confirmed superiority of PARADE (Li et al., 2024) models, but also showed their limited benefits on standard collections, which we attributed to the existence of positional bias of relevant passages (in such collections);

- We used MS MARCO FarRelevant to support the positional-bias hypothesis as well as to demonstrate that best long-document ranking models substantially (by up to 27.7%) outperform simpler baselines (such as MaxP) when training/fine-tuning data is available. However, they can also suffer more from the distribution shift and perform much worse in the zero-shot scenario.

[2] https://anonymous.4open.science/r/long_doc_rank_model_analysis_v2-78E9/.

2 Methods

2.1 Related Work

Neural Ranking models have been a popular topic in recent years (Guo et al., 2019), but the success of early approaches was controversial (Lin, 2019). This changed with an introduction of a bi-directional encoder-only Transformer model BERT (Devlin et al., 2019), which was a successor of GPT (Radford et al., 2018) and ELMO (Peters et al., 2018). BERT was hugely successful and its resounding success can be attributed to a combination of the large model size and massive pre-training using self-supervision. A number of different Transformer models such as ELECTRA (Clark et al., 2020), and DEBERTA (He et al., 2021) improve upon BERT using different training strategies and/or datasets. However, due to their architectural similarities we—following Lin et al (Lin et al., 2021)—collectively call them as BERT models.

Nogueira and Cho were first to apply BERT to ranking of text documents (Nogueira and Cho, 2019). In the big-data regime—most notably in the TREC deep learning track (Craswell et al., 2020)—BERT models outperformed prior neural and non-neural approaches by a large margin. They were also quite successful for several small-scale query collections outperforming previous neural and traditional approaches (Li et al., 2024; MacAvaney et al., 2019; Dai and Callan, 2019).

Despite their impressive performance, neural models are susceptible to the distribution shift and learning superficial features. Several authors found that neural rankers applied to out-of-domain data do not always outperform BM25 (Thakur et al., 2021; Mokrii et al., 2021). They can also be confused by superficial text modifications such as adding distractor sentences (MacAvaney et al., 2022). Likewise, ranking performance can decrease if a query is reformulated (Penha et al., 2022). Weller et al. (Weller et al., 2023) showed that neural models are not effective to "spot" negation and often perform at random level in this respect. However, we are not aware of the prior work *systematically* studying robustness to positional biases of relevant passages.

The Transformer model (Vaswani et al., 2017) uses an attention mechanism (Bahdanau et al., 2015) where each sequence position can attend to all the positions in the previous layer. Because self-attention complexity is quadratic with respect to a sequence length, direct processing of long documents is not always practical. Thus, a vast majority of existing Transformer models limit the input length to be at most 512 (subword) tokens.

Until recently, there have been two general approaches to handling long documents: localization of attention and splitting documents into chunks each of which is processed separately. Attention-localization approaches combine a limited-span (i.e., a sliding window) attention with some form of a selective global attention. There are many such approaches proposed (see, e.g., a survey by Tay et al. 2020) and it would be infeasible to evaluate them all. Instead we consider two popular models: Longformer (Beltagy et al., 2020) and Big-Bird (Zaheer et al., 2020).

With a document-splitting approach, one has to split documents into several chunks, process each chunk separately, and aggregate results, e.g., by computing a maximum or a weighted prediction score (Yilmaz et al., 2019; Dai and Callan, 2019). With respect to training approaches, the MaxP and SumP models by Dai and Callan (Dai and Callan, 2019) assume that each chunk in a relevant document is relevant. However, this assumption is problematic as the degree of relevance varies from passage to passage. Yilmaz et al. (Yilmaz et al., 2019) work around this problem by training a MaxP BERT model on short documents and zero-transfer it to long documents. In this study we work around this problem by training all document-splitting approaches including MaxP (Dai and Callan, 2019) in the end-to-end fashion, i.e., by plugging aggregated document-level scores directly into a loss function (analogous to training of CEDR (MacAvaney et al., 2019) and PARADE (Li et al., 2024) models).

More recently, it has also become possible to train longer-context models using FlashAttention (Dao et al., 2022). FlashAttention computes attention exactly and it does not eliminate quadratic complexity. However, it dramatically speeds ups training while reducing memory requirements by using an IO-efficient computation approach.

Because our primary focus is accuracy and we aim to understand the limits of long-document models, we exclude from evaluation several recent models (e.g., (Hofstätter et al., 2021; Zou et al., 2021)) that achieve better efficiency-effectiveness trade-offs by pre-selecting certain document parts and feeding only selected parts into a BERT ranker.

Recently, several teams have focused on creating challenging benchmarks for long-document retrieval. A recent LoCo v1 (Saad-Falcon et al., 2024) benchmark has 12 datasets. Despite 11 out of 12 collections has average document lengths in the order of dozens of thousands tokens, the E5 model with a 512 token input limit achieves high NDCG@10 scores (in the range of 0.4-0.85) for seven out of 12 LoCo v1 datasets. This prompted Zhu et al., 2024 to propose a more challenging LongEmbed benchmark containing a mix of real and synthetic datasets (Zhu et al., 2024).

Table 1: Distribution of Start/End Positions of Relevant Passages Inside Documents

input chunk #	MS MARCO dev (estimated)		FIRA (Hofstätter et al., 2020b) (crowd-sourced)	
	start	end	start	end
1	85.9%	71.0%	83.8%	76.4%
2	9.1%	14.9%	9.9%	15.3%
3	2.6%	6.1%	2.3%	3.9%
4	1.2%	3.0%	2.2%	2.2%
5	0.6%	1.4%	0.7%	0.9%
6	0.6%	1.2%	0.4%	0.5%
6+	0.1%	2.5%	0.7%	0.7%

Chunk size is 477 BERT tokens.

Table 2: Document Statistics

data set	# of documents	average # of BERT tokens per document
MS MARCO v1	3.2M	1.4K
MS MARCO v2	12M	2K
Robust04	0.5M	0.6K
MS MARCO FarRelevant	0.53M	1.1K

2.2 Data

Our primary datasets include two MS MARCO *Documents* collections (v1 and v2) (Bajaj et al.,

Table 3: Query Statistics

	# of queries	avg. # of BERT tokens	avg. # of pos. judgements
MS MARCO v1			
MS MARCO train	352K	7	1
MS MARCO dev	5193	7	1
TREC DL 2019	43	7	153.4
TREC DL 2020	45	7.4	39.3
MS MARCO v2			
TREC DL 2021	57	9.8	143.9
Robust04			
title	250	3.6	69.6
description	250	18.7	69.6
MS MARCO FarRelevant			
train	50K	7.0	1
test	1K	7.0	1

2016; Craswell et al., 2020, 2022), Robust04 (Voorhees, 2004), and associated query sets. In addition, we created a collection *MS MARCO FarRelevant* by using passages and relevance judgments from the MS MARCO *Passages* collection.

Robust04 is a small collection of 0.5M documents that has a mixture of news articles and government documents some of which are quite long. Yet it has only a small number of queries (250), which makes it a challenging benchmark for training models in a low-data regime. Each query has a title and a description, which represent a brief information need and a more elaborate request (often a proper English prose), respectively. We use Robust04 in a cross-validation settings with folds established by Huston and Croft (Huston and Croft, 2014) provided via IR-datasets (MacAvaney et al., 2021). All datasets are in *English*. Document and query statistics are summarized in Tables 2 and 3.

MS MARCO v1 was created from the MS MARCO reading comprehension dataset (Bajaj et al., 2016) and it has two *related* collections: passages and documents. MS MARCO v1 comes with *large* query sets, which is particularly useful for training and testing models in the big-data regime. These query sets consist of question-like queries sampled from the Bing search engine log with subsequent filtering (Craswell et al., 2020). Note that queries are not necessarily proper English questions, e.g., "lyme disease symptoms mood", but they are answerable by a short passage retrieved from a set of about 3.6M Web documents (Bajaj et al., 2016).

MS MARCO v1 test sets were created in two stages, where initially relevance judgements were created for the passage variant of the dataset. Then, document-level relevance labels were created by transferring passage-level relevance to original documents from which passages were extracted. To assess positional bias, we mapped relevant passages (from the MS MARCO Passage collection) to their positions in documents. Because document and passage texts were collected at different times this lead to some content divergence (Craswell et al., 2020) and made exact mapping impossible: In particular, Hofstätter et al. 2020b were able to match only 32% of the passages:

We deemed such mapping insufficient: To obtain a more comprehensive mapping we resorted to approximate matching and were able to match about 85% of the passages. We manually inspected a sample of matched passages to ensure that the matching procedure was reliable. Moreover, the distribution of positions of relevant passages matched that of a related FIRA dataset (Hofstätter et al., 2020b), where such information was collected by crowdsourcing. Positional bias information is summarized in Table 1.

Relevance labels in the training and development sets are "sparse": There is about one positive example per query without explicit negatives. In addition to sparse relevance judgements—separated into training and developments subsets—there is a small number (98) of queries that have "dense" judgements provided by NIST assessors for TREC 2019 and 2020 deep learning (DL) tracks (Craswell et al., 2020).

MS MARCO v2 collections was created for TREC 2021 DL track. It is an expanded version of MS MARCO v1 and uses a subset of sparse relevance judgements from MS MARCO v1. In the training set, newly added documents do not have any (positive or negative) judgments, which created a bias and made MS MARCO v2 training set less useful than that of MS MARCO v1.

The MS MARCO FarRelevant collection was created from the MS MARCO passage collection in such a way that each document contained exactly one relevant passage and this passage did not start before token 512 (see algorithm in the Appendix § B.1). Moreover, we created just a single relevant document for each training or testing query. MS MARCO FarRelevant is a variant of a the needle-in-the-haystack test (Saad-Falcon et al., 2024; Zhu et al., 2024). It is designed to be textually similar to MS MARCO Documents but with different positional biases for relevant passages. Due MS MARCO having a non-commercial license, MS MARCO FarRelevant has the same licensing restriction.

Although we generated about 7K test queries and about 500K training queries, we used only 50K and 1K queries for fine-tuning and testing, respectively. On one hand, this was sufficient for accurate training and testing and, on the other hand, it reduced experimentation time and cost.

2.3 Overview of Selected Methods

Due to space constraints, a detailed description is given in the Appendix § A. In summary, all methods can be divided into split-and-aggregate (*SplitP*) methods and *LongP* methods that "natively" support longer documents inputs. *SplitP* use either simple aggregating operations (averaging, summing, taking the maximum) or an aggregator neural network. CEDR (MacAvaney et al., 2019), PARADE Attention (Li et al., 2024), and Neural Model 1 (Boytsov and Kolter, 2021) aggregate using simple neural networks, whereas PARADE Transformer models aggregator is a smaller Transformer (Li et al., 2024).

We focused on cross-encoding rankers, which process queries concatenated with documents (Nogueira and Cho, 2019). As a reference point we also tested a bi-encoding E5-4K model, which had strong performance on LongEmbed benchmark with context sizes under 4K tokens (Zhu et al., 2024). E5-4K was tested as a ranking model and only in the zero-shot mode (without fine-tuning).

Nearly all rankers use only BERT models (i.e., bi-directional encoder-only Transformers) and have in total 100M-200M parameters (see Table 6). In addition, inspired by a recent success of LLM-rankers (Pradeep et al., 2023; Ma et al., 2023), we tested a much larger cross-encoding decoder-only ("causal") Transformer model. Specifically we chose a 1B-parameter TinyLLAMA model due to its impressive performance for its relatively small size (Zhang et al., 2024).

3 Experiments

3.1 Setup

We trained each cross-encoding ranking model using *three* seeds, except the bi-encoder model E5 (Zhu et al., 2024), which was evaluated only in the zero-shot mode. To compute statistical significance, we averaged query-specific metric values over these seeds. Due to space constraints, additional experimental details are provided in the Appendix § B.2. Moreover, in the main part of the paper we only show results for the mean reciprocal rank (MRR) and the non-discounted cumulative gain at rank k (NDCG@K). Additional precision-related metrics are computed in the Appendix (see § B.5).

3.2 Results

Our main experimental results for MS MARCO, TREC DL 2019-2021, and Robust04 are presented in Table 4. Table 5 and Fig. 2 show results for MS MARCO FarRelevant. In the Appendix (see B.4) we show that we can match or outperform key prior results, which, we believe, boosts the trustworthiness of our experiments.

We abbreviate names of several PARADE models: Note that *PARADE Attn* denotes a PARADE Attention model. The *PARADE Transf* or *P. Transf* prefix denotes PARADE Transformer models where an aggregator Transformer can be either trained from scratch (*Transf-RAND-L2*) or initialized with a pretrained model (*Transf-PRETR-L6*). L2 and L6 denote the number of aggregating layers (two and six, respectively).[3]

Unless explicitly specified, the backbone Transformer model for *SplitP* methods is BERT-base (Devlin et al., 2019). Although using other backbones such as ELECTRA (Clark et al., 2020) and DEBERTA (He et al., 2021) can improve an overall accuracy, we observe a bigger gain compared to a *FirstP* baseline when we use BERT-base (see § B.4 in the Appendix).

To ease understanding and simplify presentation, we display key results for a representative sample of models in Fig. 1 and Fig. 2 (in § 1). Moreover, in Table 4 we present only a single aggregate number for all TREC DL query sets, which is obtained by combining all the queries and respective relevance judgements (i.e., we post an overall average rather than an average over the mean values for 2019, 2020, and 2020).

From Fig. 1 and Table 4 we learn that the maximum average gain over respective *FirstP* baselines is only 5% (when measured using MRR or NDCG@K). Gains are much smaller for a number of models, which even underperform their *FirstP* baselines on one or more dataset and some of these differences are statistically significant. In particular, this is true for CEDR-DRMM, CEDR-KNRM (MacAvaney et al., 2019), JINA (**?**) and MOSAIC (Portes et al., 2023) on the MS MARCO development set.

We can also see that the *LongP* variant of the Longformer model appears to have a relatively strong performance, but so does the *FirstP* version of Longformer. Thus, we think that a good performance of Longformer on MS MARCO and Robust04 collections can be largely explained by better pretraining compared to the original BERT-base model rather than to its ability to ability to process long contexts. Moreover, FirstP (ELECTRA) and FirstP (DEBERTA) are even more accurate than FirstP (Longformer) and perform comparably well (or better) with chunk-and-aggregate

[3]Note, however, that *Transf-PRETR-L2* has only four attention heads.

Table 4: Ranking Performance on MS MARCO, TREC DL, and Robust04.

Retriever / Ranker	MS MARCO dev	TREC DL (2019-2021)	Robust04 title	Robust04 description	Avg. gain over FirstP
	MRR	**NDCG@10**	**NDCG@20**		
retriever	0.312	0.629	0.428	0.402	–
FirstP (BERT)	0.394	0.632	0.475	0.527	–
FirstP (Longformer)	0.404	0.643	0.483	0.540	–
FirstP (ELECTRA)	0.417	0.662	0.492	0.552	–
FirstP (DEBERTA)	0.415	0.672	0.534	0.596	–
FirstP (Big-Bird)	0.408	0.656	0.507	0.560	–
FirstP (JINA)	0.422	0.654	0.488	0.532	–
FirstP (MOSAIC)	0.423	0.643	0.453	0.538	–
FirstP (TinyLLAMA)	0.395	0.615	0.431	0.473	–
FirstP (E5-4K) **zero-shot**	0.380	0.641	0.438	0.429	–
AvgP	0.389 (−1.3%)	0.642 (+1.5%)	0.478 (+0.5%)	0.531 (+0.9%)	+0.4%
MaxP	0.392 (−0.4%)	0.644^a (+1.9%)	0.488^a (+2.6%)	0.544^a (+3.3%)	+1.9%
MaxP (ELECTRA)	0.414 (−0.6%)	0.659 (−0.5%)	0.502 (+2.0%)	0.563 (+2.1%)	+0.8%
MaxP (DEBERTA)	0.402^a (−3.2%)	0.671 (−0.1%)	0.535 (+0.2%)	0.609^a (+2.2%)	-0.2%
SumP	0.390 (−1.0%)	0.639 (+1.0%)	0.486 (+2.2%)	0.538 (+2.1%)	+1.1%
CEDR-DRMM	0.385^a (−2.3%)	0.629 (−0.5%)	0.466 (−2.0%)	0.533 (+1.3%)	-0.9%
CEDR-KNRM	0.379^a (−3.8%)	0.630 (−0.3%)	0.483 (+1.7%)	0.535 (+1.7%)	-0.2%
CEDR-PACRR	0.395 (+0.3%)	0.643^a (+1.6%)	0.496^a (+4.3%)	0.549^a (+4.2%)	+2.6%
Neural Model1	0.398 (+0.9%)	0.650^a (+2.8%)	0.484 (+1.8%)	0.537 (+1.9%)	+1.8%
PARADE Attn	0.416^a (+5.5%)	0.652^a (+3.1%)	0.503^a (+5.7%)	0.556^a (+5.6%)	**+5.0%**
PARADE Attn (ELECTRA)	0.431^a (+3.3%)	0.680^a (+2.7%)	0.523^a (+6.4%)	0.581^a (+5.3%)	+4.4%
PARADE Attn (DEBERTA)	0.422^a (+1.6%)	**0.688**^{a} (+2.4%)	**0.549**^{a} (+2.9%)	**0.615**^{a} (+3.2%)	+2.5%
PARADE Avg	0.392 (−0.6%)	0.646^a (+2.1%)	0.483 (+1.5%)	0.534 (+1.5%)	+1.1%
PARADE Max	0.405^a (+2.7%)	0.655^a (+3.5%)	0.489^a (+2.8%)	0.548^a (+4.0%)	+3.3%
PARADE Transf-RAND-L2	0.419^a (+6.3%)	0.655^a (+3.6%)	0.488^a (+2.8%)	0.548^a (+4.1%)	+4.2%
PARADE Transf-RAND-L2 (ELECTRA)	**0.433**^{a} (+3.9%)	0.670 (+1.2%)	0.523^a (+6.3%)	0.574^a (+3.9%)	+3.8%
PARADE Transf-PRETR-L6	0.402^a (+1.9%)	0.643 (+1.6%)	0.494^a (+4.0%)	0.554^a (+5.1%)	+3.2%
PARADE Transf-PRETR-LATEIR-L6	0.398 (+1.1%)	0.626 (−0.9%)	0.450^a (−5.2%)	0.501^a (−4.9%)	-2.5%
LongP (Longformer)	0.412^a (+1.9%)	0.668^a (+3.9%)	0.500^a (+3.6%)	0.568^a (+5.1%)	+3.6%
LongP (Big-Bird)	0.397^a (−2.9%)	0.651 (−0.7%)	0.452^a (−10.9%)	0.477^a (−14.9%)	-7.3%
LongP (JINA)	0.416^a (−1.5%)	0.665^a (+1.7%)	0.503^a (+2.9%)	0.558^a (+4.9%)	+2.0%
LongP (MOSAIC)	0.421 (−0.4%)	0.664^a (+3.3%)	0.456 (+0.6%)	0.570^a (+6.0%)	+2.4%
LongP (TinyLLAMA)	0.402^a (+1.7%)	0.608 (−1.1%)	0.452^a (+4.8%)	0.505^a (+6.7%)	+3.0%
LongP (E5-4K) **zero-shot**	0.353^a (−7.1%)	0.649 (+1.3%)	0.439 (+0.1%)	0.434 (+1.1%)	-1.1%

In each column we show a relative gain with respect model's respective *FirstP* baseline: The last column shows the average relative gain of *FirstP*. Best numbers are in **bold**: Results are averaged over three seeds. Unless specified explicitly, the backbone is **BERT-base**.
Statistical significant differences with respect to this baseline are denoted using the superscript superscript **a**.
p-value threshold is 0.01 for an MS MARCO development collection and 0.05 otherwise.

document models that uses BERT-base as the backbone model. This is a fair comparison aiming to demonstrate that on a typical test collection the benefits of long-context models are so small that comparable benefits can be obtained by finding or training a more effective *FirstP* model. *FirstP* models are more efficient during inference and they can be pretrained using a larger number of tokens for the same cost (so they could potentially perform better).

Our analysis of position of relevance passages in MS MARCO as well as results by Hofstätter et al. 2020b provide strong evidence that limited benefits of long-context models are not due inability to process long context, but rather due to a positional bias of relevant passages, which tended to be among the first 512 document tokens (see Table 1).

To further support this hypothesis, we carried out two sets of experiments using the MS MARCO FarRelevant collection, where a relevant passage was never in the first chunk. We carried out both the zero-shot experiment (evaluation of the model trained on MS MARCO) as well fine-tuning experiment using 50K MS MARCO FarRelevant queries. Because *FirstP* models perform poorly in this setting we use different baselines, namely, Longformer and *MaxP* models. For models with ELECTRA and DEBERTA backbones we compare with MaxP (ELECTRA) and MaxP (DEBERTA), respectively. Otherwise, the baseline is MaxP (BERT). From Fig. 2 and Table 5, we make the following key observations:

Table 5: Model Ranking Performance on MS MARCO FarRelevant.

Retriever / Ranker	zero-shot transferred		fine-tuned	
Random shuffling of top-100	0.052		0.052	
Retriever	0.207		0.207	
FirstP (BERT)	0.016^{b}		0.090^{b}	
FirstP (Longformer)	0.017^{b}		0.091^{b}	
FirstP (ELECTRA)	0.019^{b}		0.089^{b}	
FirstP (Big-Bird)	0.021^{b}		0.089^{b}	
FirstP (JINA)	0.018^{b}		0.088^{b}	
FirstP (MOSAIC)	0.018^{b}		0.089^{b}	
FirstP (TinyLLAMA)	0.020^{b}		0.079^{b}	
FirstP (E5-4K)	0.015^{ab}		–	
AvgP	0.154^{ab}	(-48.1%)	0.365^{ab}	$(+11.4\%)$
MaxP	0.297^{b}		0.328^{b}	
MaxP (ELECTRA)	0.328^{b}		0.349^{b}	
MaxP (DEBERTA)	0.298^{b}		0.332^{b}	
SumP	0.211^{ab}	(-28.8%)	0.327^{b}	(-0.4%)
CEDR-DRMM	0.157^{ab}	(-47.3%)	0.372^{ab}	$(+13.3\%)$
CEDR-KNRM	0.055^{ab}	(-81.5%)	0.382^{a}	$(+16.4\%)$
CEDR-PACRR	0.209^{ab}	(-29.6%)	0.393^{a}	$(+19.9\%)$
Neural Model1	0.085^{ab}	(-71.3%)	0.396^{a}	$(+20.6\%)$
PARADE Attn	0.300^{b}	$(+1.0\%)$	0.337^{b}	$(+2.8\%)$
PARADE Attn (ELECTRA)	$\mathbf{0.338}^{b}$	$(+3.3\%)$	0.354^{b}	$(+1.6\%)$
PARADE Attn (DEBERTA)	0.307^{b}	$(+3.2\%)$	0.343^{b}	$(+3.4\%)$
PARADE Avg	0.274^{ab}	(-7.6%)	0.322^{b}	(-1.7%)
PARADE Max	0.291^{b}	(-2.1%)	0.330^{b}	$(+0.6\%)$
PARADE Transf-RAND-L2	0.243^{a}	(-18.2%)	0.419^{ab}	$(+27.7\%)$
P. Transf-RAND-L2 (ELECTRA)	0.229^{a}	(-30.2%)	$\mathbf{0.432}^{ab}$	$(+23.8\%)$
PARADE Transf-PRETR-L6	0.267^{ab}	(-10.0%)	0.413^{a}	$(+26.0\%)$
P. Transf-PRETR-LATEIR-L6	0.244^{a}	(-18.0%)	0.358^{ab}	$(+9.2\%)$
LongP (Longformer)	0.233^{a}	(-21.7%)	0.399^{a}	$(+21.7\%)$
LongP (Big-Bird)	0.126^{ab}	(-57.4%)	0.401^{a}	$(+22.1\%)$
LongP (JINA)	0.069^{ab}	(-76.9%)	0.372^{ab}	$(+13.4\%)$
LongP (MOSAIC)	0.120^{ab}	(-59.6%)	0.397^{a}	$(+21.2\%)$
LongP (TinyLLAMA)	0.078^{ab}	(-73.6%)	0.397^{a}	$(+21.1\%)$
LongP (E5-4K)	0.057^{ab}	(-80.7%)	N/A (zero-shot only)	

In each column we show a relative gain over models' respective *MaxP* baseline. For *LongP* models, the gain is over *MaxP* (BERT). Best numbers are in **bold**: Our results are averaged over three seeds. Unless specified explicitly, the backbone is BERT-base. Statistically significant differences from a respective *MaxP* baseline are denoted with the superscript **a**. Statistical significant differences with respect to *Longformer* are denoted with the superscript **b**. p-value threshold is 0.01.

- The *FirstP* models performed roughly at the random-baseline level in both zero-shot and fine-tuning modes (**RQ3**). Surprisingly, E5-4K performance is also at a random-baseline level despite its competitive performance on LongEmbed benchmark (Zhu et al., 2024), MS MARCO, and Robust04 (see Table 4);

- Simple aggregation models including MaxP and PARADE Attention had good zero-shot accuracy, but benefited little from fine-tuning on MS MARCO FarRelevant (**RQ3**);

- In contrast, other long-document models had poor zero-shot performance (sometimes at a a random baseline level), but outstripped *respective* MaxP baselines by as much as 13.3%-27.7% after finetuning (**RQ3**);

- Not only positional bias diminished benefits of supporting longer document contexts, but it also lead to model overfitting to the bias and performing poorly in a zero-shot setting when the distribution of relevant passages changed substantially;

- Although PARADE Transformer models were more effective than other models on standard collections, their advantage was small (a few %). In contrast, on MS MARCO FarRelevant, PARADE Transformer (ELECTRA) outperformed the next competitor Longformer by 8% and PARADE Max (ELECTRA)—an early chunk-and-aggregate approach—by as much as 23.8% (**RQ2**).

Note that no *LongP* model outperformed the best chunk-and-aggregate approaches (while being also slower). Compared to simple aggregation models such as MaxP (ELECTRA) and PARADE Attention (ELECTRA), *LongP* models have at least 1.4× lower MRR in the zero-shot setting. In fact, in this setting three out of four *LongP* models—except Longformer—have a very low MRR with JINA being at the random-baseline level. *LongP* models also do not outperform PARADE Transformer model in the zero-shot setting and are at least 8% worse after fine-tuning. In this setting, three out of four *LongP* models have MRR scores ≈ 0.4 that are not statistical different from that of Longformer.

4 Conclusion

We carried a comprehensive evaluation of 20+ long-document ranking models, which included both chunk-and-aggregate approaches and *LongP* models that directly support long inputs, using standard IR collections as well as a synthetic new dataset MS MARCO FarRelevant. These experiments helped us expose the bias in the distribution of relevant information (a trend to appear in the beginning of documents) and to demonstrate that MS MARCO FarRelevant is a hard benchmark even for models that supported long inputs. We made our code and MS MARCO FarRelevant available.[4]

[4] https://anonymous.4open.science/r/long_doc_rank_model_analysis_v2-78E9/.

5 Limitations

Our paper has several limitations related primarily to the choice of datasets, models, and the strength of evidence for the positional bias of relevant passages.

First of all, our evaluation uses only cross-encoding ranking models. With an exception of E5-4K model, which is used in the zero-shot ranking mode, we do not train or evaluate bi-encoding models (typically used to create query and document embeddings for the first-stage retrieval). We nonetheless believe that—given a large number of proposals for long-document ranking—a reproduction and evaluation of cross-encoding long-document rankers is a sufficiently important topic that alone warrants a publication.

Moreover, as we explain below, we also use cross-encoding rankers as a tool to detect and expose bias in the position of relevant information. In that, cross-encoders are easier to train using standard (rather than high-memory) GPUs with mini-batch size one and gradient accumulation. They also typically require only one epoch to converge (only a few models need two or three epochs). In contrast, bi-encoders are trained using large batches with in-batch negatives for multiple epochs (e.g., Karpukhin et al. 2020 reports using at least 40 epochs).

Second, we focus on popular *English* document collections: MS MARCO Documents v1/v2 (Craswell et al., 2020) and Robust04 (Clarke et al., 2004). However, we have to restrict the choice of datasets to make multi-seed evaluations of 20+ models feasible. Despite this limitation, identifying bias in commonly used collections is an important task on its own. Moreover, strong performance of *FirstP* baselines was also noticed in other collections: Gao and Callan 2022 showed this for ClueWeb09 (and Robust04). Zhu et al. 2024 noticed a strong E5 *FirstP* performance on many LoCo datasets (Saad-Falcon et al., 2024).

While good performance of *FirstP* models strongly suggests a positional bias in relevant passages, we believe this alone is not sufficient evidence. Additionally—using the structure of the MS MARCO datasets—we attempt to directly identify positions of relevant passages. In that we could not map about 15% of the passages to documents, because these documents were changed after the passages were extracted. Although the failure to map 15% of passages can potentially bias the estimates for the distribution of relevant passages, we think it is unlikely because document updates were likely affected by the same positional biases as their prior versions. Moreover, our results are also supported by the FIRA experiment (Hofstätter et al., 2020b), where relevant positions were identified manually for a sample of documents used in TREC Deep Learning track (Craswell et al., 2020, 2022).

One can also argue that limited gains over *FirstP* baselines can be attributed to models' inability to process long contexts. To counter this argument, we trained and evaluated a large number of diverse cross-encoding ranking models, which included both split-and-aggregate models as well as models directly supporting long inputs. However, we can still test only a limited number of models: One might always argue that there are untested architectures that would outperform *FirstP* baselines by a much larger margin.

To demonstrate that selected models can, in principle, benefit from long contexts and decisively outperform simple baselines such as *FirstP* and even *MaxP* models we trained and/or evaluated them on a synthetic needle-in-the-haystack collection MS MARCO FarRelevant. This is still a limiting experiment, because synthetic collections—with documents composed from randomly selected passages—are imperfect proxies for real-life datasets.

In summary, we provided three types of evidence for positional bias of relevant passages: strong performance of *FirstP* models on standard collections, direct estimation of the distribution of relevant passages, and experimentation with the synthetic collection MS MARCO FarRelevant where relevant passages distribution was not skewed towards the beginning of a document. Each experiment provided imperfect/limited evidence on its own, but together they strongly support the existence of relevance position bias.

Finally, in contrast to some recent studies extending input contexts with dozens of thousands of tokens (Zhu et al., 2024; Saad-Falcon et al., 2024), we truncated documents to have at most 1431 BERT tokens. This limitation, however, did not prevent us from answering our key research questions. In particular, as we showed and explained in the Appendix § B.3, using larger inputs only marginally improved outcomes for popular IR collections such as MS MARCO, Robust04 or ClueWeb09. At the same time, when we trained models on MS MARCO and applied them to MS MARCO FarRelevant in a zero-shot mode, we observed a large (at least 17%) decrease in MRR with many models struggling to outperform a random-shuffling baseline. This indicates that even short-document collections can be quite challenging.

6 Ethics Statement

We believe our study does not pose any ethical concerns. We do not collect any new data with the help of human annotators and we do not use human or animal subjects in our study. Although we do discover a positional bias in existing retrieval collections, we are not aware of any potential risks or harms that can be caused by the exposure of this bias.

In terms of the environmental impact, our computational requirements are rather modest, because we only fine-tuned our models rather than trained them from scratch. These models were also rather small by modern standards. Except 1B-parameter TinyLLAMA (Zhang et al., 2024), each model has about 100M parameters (see Table 6 for details). Despite training and testing 20+ models with three seeds, we estimate to have spent only about 6400 GPU hours for our main experiments. 96% of the time we used NVIDIA A10 (or similarly-powerful) RTX 3090 GPUs and 4% of the time we used NVIDIA A6000.

We believe this is roughly equivalent to training a single 1B-parameter TinyLLAMA model, which required about 3400 GPU hours using a more powerful NVIDIA A100. This, in turn, this is only a tiny fraction of compute required to train LLAMA2 models (2% compared to a 7B LLAMA2 smodel).[5]

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015*.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. 2016. MS MARCO: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

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

Adam Berger and John Lafferty. 1999. Information retrieval as statistical translation. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 222–229.

Leonid Boytsov and Zico Kolter. 2021. Exploring classic and neural lexical translation models for information retrieval: Interpretability, effectiveness, and efficiency benefits. In *ECIR (1)*, volume 12656 of *Lecture Notes in Computer Science*, pages 63–78. Springer.

Peter F. Brown, Stephen Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

Charles L. A. Clark, Falk Scholer, and Ian Soboroff. 2005. The trec 2005 terabyte track. In *TREC*.

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.

Charles L. A. Clarke, Nick Craswell, and Ian Soboroff. 2004. Overview of the TREC 2004 terabyte track. In *TREC*, volume 500-261 of *NIST Special Publication*. National Institute of Standards and Technology (NIST).

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. 2020. Overview of the TREC 2019 deep learning track. *CoRR*, abs/2003.07820.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, Ellen M. Voorhees, and Jimmy Lin. 2022. Overview of the TREC 2021 deep learning track.

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

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *NeurIPS*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. pages 4171–4186.

Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE v2: Sparse lexical and expansion model for information retrieval. *CoRR*, abs/2109.10086.

[5] https://github.com/microsoft/Llama-2-Onnx/blob/main/MODEL-CARD-META-LLAMA-2.md

Chengzhen Fu, Enrui Hu, Letian Feng, Zhicheng Dou, Yantao Jia, Lei Chen, Fan Yu, and Zhao Cao. 2022. Leveraging multi-view inter-passage interactions for neural document ranking. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, WSDM '22, page 298–306, New York, NY, USA. Association for Computing Machinery.

Luyu Gao and Jamie Callan. 2022. Long document re-ranking with modular re-ranker. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, page 2371–2376, New York, NY, USA. Association for Computing Machinery.

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.

Jiafeng Guo, Yixing Fan, Liang Pang, Liu Yang, Qingyao Ai, Hamed Zamani, Chen Wu, W Bruce Croft, and Xueqi Cheng. 2019. A deep look into neural ranking models for information retrieval. *Information Processing & Management*, page 102067.

Mandy Guo, Joshua Ainslie, David Uthus, Santiago Ontanon, Jianmo Ni, Yun-Hsuan Sung, and Yinfei Yang. 2022. LongT5: Efficient text-to-text transformer for long sequences. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 724–736, Seattle, United States. Association for Computational Linguistics.

Michael Günther, Jackmin Ong, Isabelle Mohr, Alaeddine Abdessalem, Tanguy Abel, Mohammad Kalim Akram, Susana Guzman, Georgios Mastrapas, Saba Sturua, Bo Wang, Maximilian Werk, Nan Wang, and Han Xiao. 2023. Jina embeddings 2: 8192-token general-purpose text embeddings for long documents. *Preprint*, arXiv:2310.19923.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021. Debertav3: Improving deberta using electra-style pretraining with gradient-disentangled embedding sharing. *Preprint*, arXiv:2111.09543.

Sebastian Hofstätter, Bhaskar Mitra, Hamed Zamani, Nick Craswell, and Allan Hanbury. 2021. Intra-document cascading: Learning to select passages for neural document ranking. In *SIGIR*, pages 1349–1358. ACM.

Sebastian Hofstätter, Markus Zlabinger, and Allan Hanbury. 2020a. Interpretable & time-budget-constrained contextualization for re-ranking. In *ECAI*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, pages 513–520. IOS Press.

Sebastian Hofstätter, Markus Zlabinger, Mete Sertkan, Michael Schröder, and Allan Hanbury. 2020b. Fine-grained relevance annotations for multi-task document ranking and question answering. In *CIKM*, pages 3031–3038. ACM.

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 Huston and W Bruce Croft. 2014. A comparison of retrieval models using term dependencies. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 111–120.

Nasreen Abdul Jaleel, James Allan, W. Bruce Croft, Fernando Diaz, Leah S. Larkey, Xiaoyan Li, Mark D. Smucker, and Courtney Wade. 2004. Umass at TREC 2004: Novelty and HARD. In *TREC*, volume 500-261 of *NIST Special Publication*. National Institute of Standards and Technology (NIST).

Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *EMNLP (1)*, pages 6769–6781. Association for Computational Linguistics.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over BERT. In *SIGIR*, pages 39–48. ACM.

Canjia Li, Andrew Yates, Sean MacAvaney, Ben He, and Yingfei Sun. 2024. PARADE: passage representation aggregation for document reranking. *ACM Trans. Inf. Syst.*, 42(2):36:1–36:26.

Jimmy Lin. 2019. The neural hype and comparisons against weak baselines. In *ACM SIGIR Forum*, volume 52, pages 40–51. ACM New York, NY, USA.

Jimmy Lin, Rodrigo Nogueira, and Andrew Yates. 2021. *Pretrained Transformers for Text Ranking: BERT and Beyond*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. 2023. Fine-tuning llama for multi-stage text retrieval. *CoRR*, abs/2310.08319.

Sean MacAvaney, Sergey Feldman, Nazli Goharian, Doug Downey, and Arman Cohan. 2022. ABNIRML: analyzing the behavior of neural IR models. *Trans. Assoc. Comput. Linguistics*, 10:224–239.

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

Sean MacAvaney, Andrew Yates, Sergey Feldman, Doug Downey, Arman Cohan, and Nazli Goharian. 2021. Simplified data wrangling with ir-datasets. In *SIGIR*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

Iurii Mokrii, Leonid Boytsov, and Pavel Braslavski. 2021. *A Systematic Evaluation of Transfer Learning and Pseudo-Labeling with BERT-Based Ranking Models*, page 2081–2085. Association for Computing Machinery, New York, NY, USA.

Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. 2020. On the stability of fine-tuning BERT: misconceptions, explanations, and strong baselines. *CoRR*, abs/2006.04884.

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

Rodrigo Nogueira and Jimmy Lin. 2019. From doc2query to docTTTTTquery. *MS MARCO passage retrieval task publication*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8026–8037.

Gustavo Penha, Arthur Câmara, and Claudia Hauff. 2022. Evaluating the robustness of retrieval pipelines with query variation generators. In *ECIR (1)*, volume 13185 of *Lecture Notes in Computer Science*, pages 397–412. Springer.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, pages 2227–2237.

Jacob Portes, Alexander R Trott, Sam Havens, DANIEL KING, Abhinav Venigalla, Moin Nadeem, Nikhil Sardana, Daya Khudia, and Jonathan Frankle. 2023. MosaicBERT: A bidirectional encoder optimized for fast pretraining. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023. Rankvicuna: Zero-shot listwise document reranking with open-source large language models. *CoRR*, abs/2309.15088.

Ofir Press, Noah Smith, and Mike Lewis. 2022. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations*.

Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2021. Rocketqa: An optimized training approach to dense passage retrieval for open-domain question answering. In *NAACL-HLT*, pages 5835–5847. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding with unsupervised learning. *Technical report, OpenAI*.

Stephen Robertson. 2004. Understanding inverse document frequency: on theoretical arguments for IDF. *Journal of Documentation*, 60(5):503–520.

Alexander M Rush. 2018. The annotated transformer. In *Proceedings of workshop for NLP open source software (NLP-OSS)*, pages 52–60.

Jon Saad-Falcon, Daniel Y. Fu, Simran Arora, Neel Guha, and Christopher Ré. 2024. Benchmarking and building long-context retrieval models with loco and M2-BERT. *CoRR*, abs/2402.07440.

Jianlin Su, Murtadha H. M. Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. 2023. Roformer: Enhanced transformer with rotary position embedding. *Preprint*, arXiv:2104.09864.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2020. Efficient transformers: A survey. *CoRR*, abs/2009.06732.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *NeurIPS Datasets and Benchmarks*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*, pages 5998–6008.

Ellen Voorhees. 2004. Overview of the trec 2004 robust retrieval track. In *TREC*.

Orion Weller, Dawn J. Lawrie, and Benjamin Van Durme. 2023. Nevir: Negation in neural information retrieval. *CoRR*, abs/2305.07614.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

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

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. 2021. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *ICLR*. OpenReview.net.

Zeynep Akkalyoncu Yilmaz, Shengjin Wang, Wei Yang, Haotian Zhang, and Jimmy Lin. 2019. Applying BERT to document retrieval with birch. In *EMNLP/IJCNLP (3)*, pages 19–24. Association for Computational Linguistics.

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*.

George Zerveas, Navid Rekabsaz, Daniel Cohen, and Carsten Eickhoff. 2021. Coder: An efficient framework for improving retrieval through contextualized document embedding reranking. *ArXiv*, abs/2112.08766.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. 2024. Tinyllama: An open-source small language model. *CoRR*, abs/2401.02385.

Xinyu Zhang, Andrew Yates, and Jimmy Lin. 2021. Comparing score aggregation approaches for document retrieval with pretrained transformers. In *ECIR (2)*, volume 12657 of *Lecture Notes in Computer Science*, pages 150–163. Springer.

Dawei Zhu, Liang Wang, Nan Yang, Yifan Song, Wenhao Wu, Furu Wei, and Sujian Li. 2024. Longembed: Extending embedding models for long context retrieval. *CoRR*, abs/2404.12096.

Lixin Zou, Shengqiang Zhang, Hengyi Cai, Dehong Ma, Suqi Cheng, Shuaiqiang Wang, Daiting Shi, Zhicong Cheng, and Dawei Yin. 2021. Pre-trained language model based ranking in baidu search. In *KDD*, pages 4014–4022. ACM.

Table 6: Number of Model Parameters

Model family	# of params.
PARADE Transformer	132-148M
Longformer	149M
BigBird	127M
JINA	137M
MOSAIC	137M
DEBERTA-based models	184M
TinyLLAMA-based models	1034M
Other BERT- and ELECTRA-based models	≈110 M

A Ranking with Cross-Encoding Long-Document Models

In this section, we describe long-document cross-encoding models in more details. With an exception of TinyLLAMA (Zhang et al., 2024) all models use only smallish bi-directional encoder-only Transformers (Vaswani et al., 2017) with 100-200M parameters in total (see Table 6). TinyLLAMA is a so-called LLM-ranker: a "causal" decoder-only Transformer that has about 1B parameters.

We assume that an input text is split into small chunks of texts called *tokens*. Although tokens can be complete English words, Transformer models usually split text into sub-word units (Wu et al., 2016).

The length of a document d—denoted as $|d|$—is measured in the number of tokens. Because neural networks cannot operate directly on text, a sequence of tokens $t_1 t_2 \ldots t_n$ is first converted to a sequences of d-dimensional embedding vectors $w_1 w_2 \ldots w_n$ by an *embedding* network. These embeddings are context-independent, i.e., each token is always mapped to the same vector (Collobert et al., 2011; Mikolov et al., 2013).

For a detailed description of Transformer models, please see the annotated Transformer guide (Rush, 2018) as well as the recent survey by Lin et al. (Lin, 2019), which focuses on the use of BERT-style cross-encoding models for ranking and retrieval. For this paper, it is necessary to know only the following basic facts:

- BERT is an encoder-only model, which converts a sequence of tokens $t_1 t_2 \ldots t_n$ to a sequence of d-dimensional vectors $w_1 w_2 \ldots w_n$.

These vectors—which are token representations from the *last* model layer—are commonly referred to as contextualized token embeddings (Peters et al., 2018);

- BERT operates on word pieces (Wu et al., 2016) rather than on complete words;

- The vocabulary includes two special tokens: [CLS] (an aggregator) and [SEP] (a separator);

- Using a *pooled* representation of token vectors $w_1 w_2 \ldots w_n$, a linear layer is used to produce a ranking score. A nearly universal pooling approach in cross-encoding rankers is to use the vector of the [CLS] token, i.e., w_1. However, we learned that some models (e.g., JINA (Günther et al., 2023)) converge well *only* with mean pooling, i.e., they use $\frac{1}{n}\sum_{i=1}^{n} w_i$.

A "vanilla" BERT ranker (dubbed as monoBERT by Lin et al. (Lin, 2019)) uses a single fully-connect layer F as a prediction head, which converts the last-layer representation of the [CLS] token (i.e., a contextualized embedding of [CLS]) into a scalar (Nogueira and Cho, 2019). It makes a prediction based on the following sequence of tokens: [CLS] q [SEP] d [SEP], where q is a query and d is a document.

An alternative approach is to aggregate contextualized embeddings of regular tokens using a shallow neural network (MacAvaney et al., 2019; Boytsov and Kolter, 2021; Khattab and Zaharia, 2020) (possibly together with the contextualized embedding of [CLS]) . This was first proposed by MacAvaney et al. (MacAvaney et al., 2019) who also found that incorporating [CLS] improves performance. However, Boytsov and Kolter proposed a shallow aggregating network that does not use the output of the [CLS] token and achieved the same accuracy on MS MARCO datasets (Boytsov and Kolter, 2021).

Replacing the standard BERT model in the vanilla BERT ranker with a BERT variant that "natively" supports longer documents (e.g., Big-Bird (Zaheer et al., 2020)) is, perhaps, the simplest way to deal with long documents. We collectively call these models as LongP models. For a typical BERT model, however, long documents and queries need to be split or truncated so that the overall number of tokens does not exceed 512. In the *FirstP* mode, we process only the first chunk and ignore the truncated text. In the *SplitP* mode, each chunk is processed separately and the results are aggregated. In the remaining of this section, we discuss these approaches in detail.

A.1 LongP models

In our work, we benchmark both sparse-attention and full-attention models. Sparse attention LongP models include two popular options: Longformer (Beltagy et al., 2020) and Big-Bird (Zaheer et al., 2020). In that, we use the same approach to score documents as with the vanilla BERT ranker, namely, concatenating queries with documents and making a prediction based on the contextualized embedding of the [CLS] token (Nogueira and Cho, 2019). Both Big-Bird and Longformer use a combination of the local, "scattered" (our terminology), and global attention. The local attention utilizes a sliding window of a constant length where each token attends to each other token within this window. In the case of the global attention, certain tokens can attend to *all* other tokens and vice-versa, In Big-Bird, only special tokens such as [CLS] can attend globally. In Longformer, the user have to select such tokens explicitly. Following Beltagy et al. (Beltagy et al., 2020), who applied this technique to question-answering, we "place" global attention only on query tokens. Unlike the global attention, the scattered attention is limited to restricted sub-sets of tokens, but these subsets do not necessarily have locality. In Big-Bird the scattered attention relies on random tokens, whereas Longformer uses a dilated sliding-window attention with layer- and head-specific dilation.

Full-attention models include JINABert (Günther et al., 2023), TinyLLAMA (Zhang et al., 2024), and MosaicBERT (Portes et al., 2023), henceforth, simply JINA, TinyLLAMA and MOSAIC. All these models use a recently proposed FlashAttention (Dao et al., 2022) to efficiently process long-contexts as well as special positional embeddings that can generalize to document lengths not seen during training. In that, JINA and MOSAIC use AliBi (Press et al., 2022), while TinyLLAM uses ROPE embeddings (Su et al., 2023). JINA and MOSAIC are bi-directional encoder-only Transformer model whereas TinyLLAMA is a unidirectional (sometimes called causal) decoder-only Transformer model (Vaswani et al., 2017).

In addition architectural difference, models differ in pretraining strategies. MOSAIC relies primarily on the masked language (MLM) objective without next sentence prediction (NSP). JINA uses this approach as a first step, following a RoBERTa pretraining strategy (Liu et al., 2019) and fine-tuning on retrieval and classification tasks with mean-pooled representations. TinyLLAMA was trained using an autoregressive language modeling objective (Zhang et al., 2024). We found that JINA lost an ability to effectively pool on the `[CLS]` token and we used mean-pooling instead. We also use mean pooling for TinyLLAMA. For MOSAIC we used pooling on `[CLS]`.

A.2 SplitP models

SplitP models differ in partitioning and aggregation approaches. Documents can be split into either disjoint or overlapping chunks. In the first case, documents are split in a greedy fashion so that each document chunk except possibly the last one is exactly 512 tokens long after being concatenated with a (padded) query and three special tokens. In the second case, we use a sliding window approach with a window size and stride that are not tied to the maximum length of BERT input.

Greedy partitioning into disjoint chunks CEDR models (MacAvaney et al., 2019) and the Neural Model 1 (Boytsov and Kolter, 2021) use the first method, which involves:

- tokenizing the document d;
- greedily splitting a tokenized document d into m disjoint chunks: $d = d_1 d_2 \ldots d_m$;
- generating m token sequences `[CLS]` q `[SEP]` d_i `[SEP]` by concatenating the query with document chunks;
- processing each sequence with a BERT model to generate contextualized embeddings for regular tokens as well as for `[CLS]`.

The outcome of this procedure is m `[CLS]`-vectors cls_i and n contextualized vectors $w_1 w_2 \ldots w_n$ (one for *each* document token t_i) that are aggregated in a model-specific ways.

MacAvaney et al. (MacAvaney et al., 2019) use contextualized embeddings as a direct replacement of context-free embeddings in the following neural architectures: KNRM (Xiong et al., 2017), PACRR (Hui et al., 2018), and DRMM (Guo et al., 2016). To boost performance, they incorporate `[CLS]`-vectors in a model-specific way. We call the respective models as *CEDR-KNRM*, *CEDR-PACRR*, and *CEDR-DRMM*.

They also proposed an extension of the vanilla BERT ranker that makes a prediction using the average `[CLS]` token: $\frac{1}{m}\sum_{i=1}^{m} cls_i$ by passing it through a linear projection layer. We call this method *AvgP*.

The Neural Model 1 (Boytsov and Kolter, 2021) uses the same greedy partitioning approach as CEDR, but a different aggregator network, which does not use the embeddings of the `[CLS]` token. This network is a neural parametrization of the classic Model 1 (Berger and Lafferty, 1999; Brown et al., 1993).

Sliding window approach The BERT MaxP/SumP (Dai and Callan, 2019) and PARADE (Li et al., 2024) models use a sliding window approach. Assume w is the size of the window and s is the stride. Then the processing can be summarized as follows:

- tokenizing, the document d into sub-words $t_1 t_2 \ldots t_n$;
- splitting a tokenized document d into m possibly overlapping chunks $d_i = t_{i \cdot s} t_{i \cdot s+1} \ldots t_{i \cdot s+w-1}$: Trailing chunks may have fewer than w tokens.
- generating m token sequences `[CLS]` q `[SEP]` d_i `[SEP]` by concatenating the query with document chunks;
- processing each sequence with a BERT model to generate a last-layer output for each sequence `[CLS]` token.

The outcome of this procedure is m `[CLS]`-vectors cls_i, which are subsequently aggregated in a model-specific ways. Note that PARADE and MaxP/SumP models do not use contextualized embeddings of regular tokens.

BERT MaxP/SumP These models (Dai and Callan, 2019) use a linear layer F to produce m relevance scores $F(cls_i)$. Then complete document scores are computed as $\max_{i=1}^{m} F(cls_i)$ and $\sum_{i=1}^{m} F(cls_i)$ for the MaxP and SumP models, respectively.

PARADE These models (Li et al., 2024) can be divided into two groups. The first group includes PARADE average, PARADE max, and PARADE attention, which all use simple approaches to produce an aggregated representation of m `[CLS]`-vectors cls_i. To compute a relevance score these aggregated representations are passed through a linear layer F.

In particular, PARADE average and PARADE max combine cls_i using averaging and the element-wise maximum operation, respectively to generate aggregated representation of m [CLS] tokens cls_i.[6] The PARADE attention model uses a learnable attention (Bahdanau et al., 2015) vector C to compute a scalar weight w_i of each i as follows: $w_1 w_2 \ldots w_m = softmax(C \cdot cls_1, C \cdot cls_2, \ldots, C \cdot cls_m)$. These weights are used to compute the aggregated representation as $\sum_{i=1}^{m} w_i cls_i$

PARADE Transformer models combine [CLS]-vectors cls_i with an additional *aggregator* transformer model $AggregTransf()$. The input of the aggregator Transformer is sequence of cls_i vectors prepended with a learnable vector C, which plays a role of a [CLS] embedding for $AggregTransf()$. The last-layer representation of the first vector is passed through a linear layer F to produce a relevance score:

$$F(AggregTransf(C, cls_1, cls_2, \ldots, cls_m)[0]) \quad (1)$$

An aggregator Transformer can be either pretrained or randomly initialized. In the case of a pretrained transformer, we completely discard the embedding layer. Furthermore, if the dimensionality of cls_i vectors is different from the dimensionality of input embeddings in $AggregTransf$, we project cls_i using a linear transformation.

Miscellaneous models We attempted to implement additional state-of-the-art models (Gao and Callan, 2022; Fu et al., 2022). Gao and Callan (Gao and Callan, 2022) introduced a late-interaction model MORES+, which is a modular long document reranker that uses a sequence-to-sequence transformer in a non-auto-regressive mode. In MORES+ document chunks are first encoded using the encoder-only Transformer model. Then they use a modified decoder Transformer for joint query-to-all-document-chunk cross-attention: This modification changes a causal Transformer into an encoder-only bi-directional Transformer model. As of the moment of writing, the MORES+ model holds the first position on a competitive MS MARCO document leaderboard.[7]. However, the authors provide only incomplete implementation which does not fully match the description in the paper (i.e., crucial details are missing). We reimplemented this model to the best of our understanding, but our implementation failed to outperform even BM25.

Inspired by this approach, we managed to implement a late-interaction variant of the PARADE model, which we denoted as PARADE-LATEIR. Similar to the original PARADE model, it splits documents into overlapping chunks. However, it then encodes chunks and queries independently. Next, it uses an interaction Transformer to (1) mix these representations, and (2) combine output using an aggregator Transformer. In total, the model uses three backbone encoder-only Transformers: All of these Transformers are initialized using pretrained models.

Fu et al. (Fu et al., 2022) proposed a multi-view interactions-based ranking model (MIR). They implement inter-passage interactions via a multi-view attention mechanism, which enables information propagation at token, sentence, and passage levels. Due to the computational complexity, they restrict these interactions to a set of salient/pivot tokens. However, the paper does not provide enough details regarding the choices of these tokens. There is no software available and authors did not respond to our clarification requests. Thus, this model is also excluded from our evaluation.

We additionally implemented both the encoder-only and the encoder-decoder variant of LongT5 (Guo et al., 2022) as well as RoFormer (with ROPE embeddings) (Su et al., 2024). We eventually had to abandon them due to poor convergence (LongT5) and/or CUDA crashes (RoFormer).

B Experiments: Additional Information, Ablations, and Detailed Results

B.1 MS MARCO FarRelevant Creation Algorithm

The MS MARCO FarRelevant dataset was created as follows: Assume that C_t is the number of tokens in the passage:

- Select randomly a document length between $512 + C_t$ and 1431;

[6] Note that both PARADE average and AvgP vanilla ranker use the same approach to aggregate contextualized embeddings of [CLS] tokens, but they differ in their approach to select document chunks. In particular, AveP uses non-overlapping chunks while PARADE average relies on the sliding window approach.

[7] `https://microsoft.github.io/MSMARCO-Document-Ranking-Submissions/leaderboard/`

- Using rejection sampling, obtain K_1 non-relevant samples such that their *total* length exceeds 512, but the length of $K_1 - 1$ first samples is at most 512.

- Using the same approach, sample another $K_2 + 1$ samples such that the total length of K_2 samples is at most $1431 - C_t$, but the total length of $K_2 + 1$ samples exceeds this value.

- Discard the last sampled passage and randomly mix the remaining K_2 non-relevant passages with a single relevant passage.

- Finally, append the resulting string to the concatenation of the first K_1 non-relevant passages.

B.2 Detailed Training and Evaluation Setup

B.2.1 General Setup

In our work, a ranker is applied to the output of the first-stage retrieval model, also known as a candidate-generator. Depending on the experiment and the dataset we use different candidate generators: for MS MARCO v1 and Robust04 we used a BM25 ranker (Robertson, 2004). In that, for MS MARCO v1 it was applied to documents expanded using the doc2query approach (Nogueira and Lin, 2019). For MS MARCO v2, we used a hybrid retriever where candidate records are first produced using a k-NN search and subsequently re-ranked using a linear fusion of BM25 scores and the cosine similarity between query and document embeddings. Embeddings were generated using ANCE (Xiong et al., 2021).

Depending on the collection we computed a subset of the following metrics: the mean reciprocal rank (MRR), the non-discounted cumulative gain at rank k (NDCG@K) (Järvelin and Kekäläinen, 2002), the mean average precision (MAP), and precision at rank (P@K), $k \in \{10, 20\}$. Due to space constraints, we included results with MAP and P@K only in the Appendix (see § B.5). Note that for test sets with sparse labels (MS MARCO development set and MS MARCO FarRelevant) we computed only MRR.

All experiments were carried out using the an **anonymous** retrieval toolkit framework, which employed Lucene and an **anonymous** toolkit for k-NN search to provide retrieval capabilities. Deep learning support was provided via PyTorch (Paszke et al., 2019) and HuggingFace Transformers library (Wolf et al., 2019). The instructions to reproduce our key results are publicly available in the on-line appendix.[8]

B.2.2 Model Traning

A ranker was trained to distinguish between positive examples (known relevant documents) and hard negative examples (documents not marked as relevant) sampled from the set of top-k candidates returned by the candidate generator. We used $k = 100$ for MS MARCO and MS MARCO FarRelevant and $k = 1000$ for Robust04 (based on preliminary experiments).

Each model was trained using *three* seeds. All models except MOSAIC were trained using half-precision. MOSAIC models were trained using full-precision. MOSAIC training was unstable (even though we used the full precision) and often resulted in close-to-zero performance. For this reason we continued training with *more* seeds until we obtained three models with reasonable performance. This seed selection strategy could potentially have biased (up) MOSAIC results. To compute statistical significance, we averaged query-specific metric values over these seeds.

All MS MARCO models were trained from scratch. Then these models were fine-tuned on Robust04. Note that except for the aggregation Transformers and TinyLLAMA, we use a *base*, i.e., a 12-layer Transformer (Vaswani et al., 2017) models. TinyLLAMA has 22 layers and about 1B parameters. BERT-base is more practical then a 24-layer BERT-large and performs at par with BERT-large on MS MARCO and Robust04 (Hofstätter et al., 2020a; Li et al., 2024). In our own experiments, we see that large (24 and more layers) model perform much better on the MS MARCO Passage collection, but we were not able to outperform 12-layer models on the MS MARCO Documents collection. Note that Longformer (Beltagy et al., 2020), BigBird (Zaheer et al., 2020), and DEBERTA base (He et al., 2021), JINA (**?**), and MOSAIC (Portes et al., 2023) all have 12 layers, but a larger embedding matrix.

One training epoch consisted in iterating over all queries and sampling one positive and one negative example with a subsequent computation of a pairwise margin loss. We used the minibatch size one with gradient accumulation over 16 steps. The learning rates are provided in the model configura-

[8] https://anonymous.4open.science/r/long_doc_rank_model_analysis_v2-78E9/

Table 7: Comparison of Long-context Models to Respective FirstP baselines and Prior Art.

Model	MS MARCO dev	TREC DL 2019	TREC DL 2020	TREC DL 2021	Robust04 title	Robust04 description
	MRR	**NDCG@10**			**NDCG@20**	
Prior work (FirstP, MaxP), Zhang et al. (Zhang et al., 2021)						
FirstP (BERT)	–	–	–	–	0.449	0.510
MaxP (BERT)	–	–	–	–	0.477 (+6.2%)	0.530 (+3.9%)
MaxP (ELECTRA)	–	–	–	–	0.523	0.574
Prior work (PARADE) Li et al. (Li et al., 2024)						
PARADE Attn (ELECTRA)	–	–	–	–	0.527	0.587
PARADE Max (ELECTRA)	–	0.679	0.613	–	0.544	0.602
PARADE Transf-RAND (ELECTRA)	–	0.650	0.601	–	**0.566**	0.613
Our results						
FirstP (BERT)	0.394	0.631	0.598	0.660	0.475	0.527
MaxP (BERT)	0.392 (−0.4%)	0.648 (+2.6%)	0.615 (+2.8%)	0.665 (+0.8%)	0.488^a (+2.6%)	0.544^a (+3.3%)
PARADE Attn	0.416^a (+5.5%)	0.647 (+2.5%)	0.626^a (+4.6%)	0.677 (+2.5%)	0.503^a (+5.7%)	0.556^a (+5.6%)
FirstP (ELECTRA)	0.417	0.652	0.642	0.686	0.492	0.552
MaxP (ELECTRA)	0.414 (−0.6%)	0.659 (+1.0%)	0.630 (−1.9%)	0.683 (−0.5%)	0.502 (+2.0%)	0.563 (+2.1%)
PARADE Attn (ELECTRA)	**0.431**^{a} (+3.3%)	0.675^a (+3.5%)	0.653 (+1.8%)	0.705 (+2.8%)	0.523^a (+6.4%)	0.581^a (+5.3%)
FirstP (DEBERTA)	0.415	0.675	0.629	0.702	0.534	0.596
MaxP (DEBERTA)	0.402 (−3.2%)	0.679 (+0.6%)	0.620 (−1.4%)	0.705 (+0.4%)	0.535 (+0.2%)	0.609 (+2.2%)
PARADE Attn (DEBERTA)	0.422^a (+1.6%)	**0.685** (+1.4%)	**0.659**^{a} (+4.8%)	**0.713** (+1.4%)	0.549^a (+2.9%)	**0.615**^{a} (+3.2%)
FirstP (Longformer)	0.404	0.657	0.616	0.654	0.483	0.540
LongP (Longformer)	0.412^a (+1.9%)	0.676^a (+2.9%)	0.628 (+2.0%)	0.693^a (+6.0%)	0.500^a (+3.6%)	0.568^a (+5.1%)
FirstP (Big-Bird)	0.408	0.663	0.620	0.679	0.507	0.560
LongP (Big-Bird)	0.397^a (−2.9%)	0.655 (−1.1%)	0.618 (−0.3%)	0.675 (−0.5%)	0.452^a (−10.9%)	0.477^a (−14.9%)
FirstP (JINA)	0.422	0.658	0.618	0.679	0.488	0.532
LongP (JINA)	0.416^a (−1.5%)	0.670^a (+1.8%)	0.632 (+2.1%)	0.689 (+1.4%)	0.503^a (+2.9%)	0.558^a (+4.9%)
FirstP (MOSAIC)	0.423	0.654	0.607	0.662	0.453	0.538
LongP (MOSAIC)	0.421 (−0.4%)	0.660 (+0.9%)	0.630^a (+3.7%)	0.694^a (+4.9%)	0.456 (+0.6%)	0.570^a (+6.0%)

In each column we show a relative gain over model's respective *FirstP* baseline: The last column shows the average relative gain over *FirstP*. Best numbers are in **bold**: Our results are averaged over three seeds (but not necessarily prior art).
Statistical significant differences with respect to this baseline are denoted using the superscript superscript **a**. p-value threshold is 0.01 for an MS MARCO development collection and 0.05 otherwise.

tion files (in the on-line repository).[9] We used the AdamW optimizer (Loshchilov and Hutter, 2017) and a constant learning rate with a 20% linear warm-up (Mosbach et al., 2020).

We have learned that—unlike neural *retrievers*—cross-encoding rankers (Nogueira and Cho, 2019) are relatively insensitive to learning rates, their schedules, and the choice of loss functions. We were sometimes able to achieve better results using multiple negatives per query and a listwise margin loss (or cross-entropy). However, the gains were small and not consistent compared to a simple pairwise margin loss used in our work (in fact, using a listwise loss function sometimes lead to overfitting). Note again that this is different from neural *retrievers* where training is difficult without using a listwise loss and/or batch-negatives (Karpukhin et al., 2020; Xiong et al., 2021; Qu et al., 2021; Zerveas et al., 2021; Formal et al., 2021).

For MS MARCO, all models except PARADE-Transf-Pretr-LATEIR-L6 and PARADE-Transf-RAND-L2 were trained for one epoch: Further training did not improve (and sometimes degraded) accuracy. However, PARADE-Transf-RAND-L2 and PARADE-Transf-Pretr-LATEIR-L6 required two-to-three epochs to reach the maximum accuracy. In the case of Robust04, each model was finetuned for 100 epochs, but all epochs were short, so the overall training and evaluation time was comparable to that of fine-tuning on MS MARCO for a single epoch.

To reproduce our main results, we estimate that one needs about 6400 GPU hours: 6000 hours using NVIDIA A10 (or RTX 3090) with 24 GB of memory and 400 hours using NVIDIA A6000 with 48 GB of memory. A6000 was required only for TinyLLAMA.

From our experience models trained on MS MARCO v2 performed worse on TREC 2021 queries compared to models trained on MS MARCO v1. This may indicate that models somehow learn to distinguish between original MS MARCO v1 documents and newly added ones (which did not have positive judgements in the

[9] https://anonymous.4open.science/r/long_doc_rank_model_analysis_v2-78E9/.

training sets). As a result, these models are biased and tend to not rank these new documents well even when they are considered to be relevant by NIST assessors. For this reason, we used MS MARCO v2 data in a zero-shot transfer mode where ranking models trained on MS MARCO v1 are evaluated on TREC DL 2021 queries.

B.2.3 Miscellaneous Notes

To enable efficient training and evaluation of the large number of models, documents were truncated to have at most 1431 BERT tokens. In § B.3 (see Table 8) we show that for our top-performing model PARADE Attention (Li et al., 2024) using a larger number of chunks only marginally improves outcomes. Depending on a dataset, the highest accuracy is achieved using either three or four chunks.

For *SplitP* approaches, queries were padded to 32 BERT tokens with padding being masked out during training (longer queries were truncated). For *SplitP* models with greedy partitioning into disjoint chunks, long document were split into at most three chunks containing 477 document tokens (each concatenated with up to 32 query tokens plus three special tokens).

We evaluated 20+ models, but we had to exclude two LongT5 variants (Guo et al., 2022) and RoFormer (with ROPE embeddings) (Su et al., 2024) due to poor convergence and/or crashes. Specifically, even after 10 epochs of training LongT5 models were $\approx 10\%$ less accurate than BERT-base *FirstP* trained for one epoch. Given the uncertainty regarding the possible convergence of models as well as the need to train these for three epochs, we have to abandon this experiment as overly expensive. RoFormer models were failing due to CUDA errors when the context length exceeded 512: We were not able to resolve this issue.

B.3 Varying the Number of Chunks

To understand if truncating input to have at most 1431 BERT tokens negatively affected model performance, we carried out an ablation study where one of the top-performing models was trained and evaluated using inputs of varying maximum lengths. To this end we used PARADE Attention with the number of input chunks varying from one to six. In that the same number of chunks was used during training and evaluation, i.e., we had to carry out six experiments. Similar to our main experiments, we trained each model using three seeds. We carried out this ablation experiment using our MS MARCO and Robust04 datasets.

The results are presented in Table 8: We can see that—depending on the dataset—three or four input chunks are optimal. However, the additional gains over the *FirstP* baseline are at most 0.6% when averaged over all test sets.

Gao and Callan 2022 carried out a similar ablation using ClueWeb09: Increasing the number of input chunks from three to six lead to only about 2.3% relative improvement in NDCG@20. However, even this modest gain could have been slightly inflated due to model not being trained *directly* on shorter inputs. Indeed, truncation of an input for a six-chunk model to one chunk is potentially less effective than training and evaluating the model using one-chunk data.

B.4 Reproducibility and Backbone Selection for *SplitP* Models

To understand if using BERT-base as backbone model for various *SplitP* (i.e., chunk-and-aggregate) approaches diminished models' ability to process and understand long contexts, we carried out a focused comparison of several backbone models, including ELECTRA (Clark et al., 2020) and DEBERTA (He et al., 2021). To this end, we used two methods: PARADE (Li et al., 2024) Attention and *MaxP*. PARADE Attention model achieved the largest average gain over *FirstP* in our main experiments (see Table 4), whereas *MaxP* models were extensively benchmarked in the past (Li et al., 2024; Dai and Callan, 2019; Zhang et al., 2021). Although prior work found ELECTRA to be a better backbone model in terms of *absolute* accuracy (Li et al., 2024; Zhang et al., 2021), we found no systematic evaluation of the relationship between a backbone model and achievable *FirstP* gains.

Results in Tables 7 and 4 confirm overall superiority of both ELECTRA and DEBERTA over BERT-base. In that, DEBERTA models are nearly always more effective compared to ELECTRA models with biggest differences on Robust04. However, their *relative* effectiveness with respect to their respective *FirstP* baselines does not exceed that of BERT-base. The same is true for *LongP* models. Except Longformer they performed equally or worse compared to *FirstP* on 8 test sets out of 18. Moreover, all *LongP* models achieved lower average gains over *FirstP* (see the last column in Table 4). We conclude that to measure capabilities of chunk-and-aggregate model to un-

Table 8: Effectiveness of the PARADE Attention Model for Different Input Truncation Thresholds

Retriever / Ranker	MS MARCO dev	TREC DL (2019-2021)	Robust04 title	Robust04 description	Avg. gain Over FirstP
	MRR	**NDCG@10**	**NDCG@20**		
Retriever	0.312	0.629	0.428	0.402	–
PARADE Attn (1 chunk)	0.401	0.637	0.476	0.527	–
PARADE Attn (2 chunks)	0.408^a (+1.8%)	0.653^a (+2.7%)	0.499^a (+4.9%)	0.544^a (+3.3%)	+3.2%
PARADE Attn (3 chunks)	0.406^a (+1.3%)	0.648^a (+1.7%)	**0.505**^{a} (+6.1%)	0.557^a (+5.7%)	+3.7%
PARADE Attn (4 chunks)	**0.412**^{a} (+2.9%)	**0.654**^{a} (+2.7%)	0.504^a (+5.9%)	**0.558**^{a} (+5.9%)	**+4.3%**
PARADE Attn (5 chunks)	0.409^a (+2.0%)	0.652^a (+2.4%)	0.502^a (+5.6%)	0.556^a (+5.5%)	+3.9%
PARADE Attn (6 chunks)	0.411^a (+2.4%)	0.653^a (+2.6%)	0.504^a (+5.9%)	0.554^a (+5.2%)	+4.0%

derstand and incorporate long context, it appears to be *beneficial* to use BERT-base instead of ELECTRA or DEBERTA.

We also use Table 7 to compare with prior art. We generally reproduce prior art, in particular, experiments by Li et al. 2024, who invented PARADE models. Our ELECTRA-based models achieve higher NDCG@10 on TREC DL 2019-2020 and PARADE Attention models come very close, but they are about 3-5% worse compared to their PARADE Transformer on Robust04. At the same time, our DEBERTA-based PARADE Attention model achieves similar NDCG@20 scores.

Note that one should not expect identical results due to differences in training regimes and candidate generators. In particular, in the case of Robust04, Li et al. 2024 use RM3 (BM25 with a pseudo-relevance feedback (Jaleel et al., 2004)), which is more effective than BM25 (Robertson, 2004) (which we use on Robust04).

Another important comparison point is Robust04 results by Zhang et al. 2021 who were able to reproduce original *MaxP* results by Dai and Callan 2019, which used BERT-base as a backbone. In addition, they experimented with ELECTRA models "pre-finetuned" on MS MARCO. When comparing BERT-base results, Zhang et al. 2021 have the maximum relative gain of 6.6% compared to ours 3.3%. However, in absolute terms we got higher NDCG@20 for both *FirstP* and *MaxP*. Their MaxP (ELECTRA) has comparable performance to ours on TREC DL 2019-2020, but it is 2-4% better on Robust04. In turn, our MaxP (DEBERTA) is better by 2-6%. Although we do not always match prior art using the same backbone models, we generally match or outperform prior results, which, we believe, boosts the trustworthiness of our experiments.

Table 9: Ranking Performance on MS MARCO and TREC DL.

Model	MS MARCO dev	TREC DL 2019-2021		
	MRR	NDCG@10	P@10	MAP
Retriever	0.312	0.629	0.720	0.321
FirstP (BERT)	0.394	0.632	0.712	0.311
FirstP (Longformer)	0.404	0.643	0.722	0.317
FirstP (ELECTRA)	0.417	0.662	0.734	0.320
FirstP (DEBERTA)	0.415	0.672	0.741	0.327
FirstP (Big-Bird)	0.408	0.656	0.727	0.321
FirstP (JINA)	0.422	0.654	0.728	0.320
FirstP (MOSAIC)	0.423	0.643	0.726	0.316
FirstP (TinyLLAMA)	0.395	0.615	0.692	0.301
FirstP (E5-4K) **zero-shot**	0.380	0.641	0.722	0.317
AvgP	0.389 (-1.3%)	0.642 $(+1.5\%)$	0.717 $(+0.7\%)$	0.317^a $(+2.0\%)$
MaxP	0.392 (-0.4%)	0.644^a $(+1.9\%)$	0.723 $(+1.5\%)$	0.322^a $(+3.7\%)$
MaxP (ELECTRA)	0.414 (-0.6%)	0.659 (-0.5%)	0.745 $(+1.5\%)$	0.326 $(+2.1\%)$
MaxP (DEBERTA)	0.402^a (-3.2%)	0.671 (-0.1%)	0.746 $(+0.7\%)$	0.335^a $(+2.5\%)$
SumP	0.390 (-1.0%)	0.639 $(+1.0\%)$	0.715 $(+0.4\%)$	0.319^a $(+2.6\%)$
CEDR-DRMM	0.385^a (-2.3%)	0.629 (-0.5%)	0.708 (-0.5%)	0.313 $(+0.6\%)$
CEDR-KNRM	0.379^a (-3.8%)	0.630 (-0.3%)	0.711 (-0.1%)	0.313 $(+0.8\%)$
CEDR-PACRR	0.395 $(+0.3\%)$	0.643^a $(+1.6\%)$	0.719 $(+0.9\%)$	0.320^a $(+2.9\%)$
Neural Model1	0.398 $(+0.9\%)$	0.650^a $(+2.8\%)$	0.723^a $(+1.5\%)$	0.323^a $(+3.9\%)$
PARADE Attn	0.416^a $(+5.5\%)$	0.652^a $(+3.1\%)$	0.728^a $(+2.2\%)$	0.324^a $(+4.2\%)$
PARADE Attn (ELECTRA)	0.431^a $(+3.3\%)$	0.680^a $(+2.7\%)$	0.763^a $(+3.9\%)$	0.335^a $(+4.9\%)$
PARADE Attn (DEBERTA)	0.422^a $(+1.6\%)$	**0.688**^{a} $(+2.4\%)$	**0.763**^{a} $(+3.0\%)$	**0.339**^{a} $(+3.9\%)$
PARADE Avg	0.392 (-0.6%)	0.646^a $(+2.1\%)$	0.715 $(+0.4\%)$	0.317^a $(+2.1\%)$
PARADE Max	0.405^a $(+2.7\%)$	0.655^a $(+3.5\%)$	0.733^a $(+2.9\%)$	0.324^a $(+4.1\%)$
PARADE Transf-RAND-L2	0.419^a $(+6.3\%)$	0.655^a $(+3.6\%)$	0.734^a $(+3.1\%)$	0.326^a $(+5.0\%)$
PARADE Transf-RAND-L2 (ELECTRA)	**0.433**^{a} $(+3.9\%)$	0.670 $(+1.2\%)$	0.747 $(+1.8\%)$	0.327 $(+2.2\%)$
PARADE Transf-PRETR-L6	0.402^a $(+1.9\%)$	0.643 $(+1.6\%)$	0.717 $(+0.8\%)$	0.322^a $(+3.6\%)$
PARADE Transf-PRETR-LATEIR-L6	0.398 $(+1.1\%)$	0.626 (-0.9%)	0.707 (-0.7%)	0.307 (-1.1%)
LongP (Longformer)	0.412^a $(+1.9\%)$	0.668^a $(+3.9\%)$	0.752^a $(+4.1\%)$	0.333^a $(+5.1\%)$
LongP (Big-Bird)	0.397^a (-2.9%)	0.651 (-0.7%)	0.726 (-0.2%)	0.322 $(+0.3\%)$
LongP (JINA)	0.416^a (-1.5%)	0.665^a $(+1.7\%)$	0.742^a $(+2.0\%)$	0.328^a $(+2.4\%)$
LongP (MOSAIC)	0.421 (-0.4%)	0.664^a $(+3.3\%)$	0.740^a $(+1.9\%)$	0.327^a $(+3.7\%)$
LongP (TinyLLAMA)	0.402^a $(+1.7\%)$	0.608 (-1.1%)	0.692 $(+0.0\%)$	0.306 $(+1.6\%)$
LongP (E5-4K) **zero-shot**	0.353^a (-7.1%)	0.649 $(+1.3\%)$	0.724 $(+0.3\%)$	0.323 $(+1.8\%)$

In each column we show a relative gain with respect model's respective *FirstP* baseline: The last column shows the average relative gain of *FirstP*. Best numbers are in **bold**: Results are averaged over three seeds. Unless specified explicitly, the backbone is **BERT-base**.
Statistical significant differences with respect to this baseline are denoted using the superscript superscript **a**. p-value threshold is 0.01 for an MS MARCO development collection and 0.05 otherwise.
E5-models were used only in the zero-shot model, i.e., without fine-tuning.

Table 10: Ranking Performance on Robust04.

Model	NDCG@20	P@20	MAP	NDCG@20	P@20	MAP
Retriever	0.428	0.365	0.255	0.402	0.334	0.240
FirstP (BERT)	0.475	0.405	0.277	0.527	0.447	0.303
FirstP (Longformer)	0.483	0.413	0.277	0.540	0.454	0.307
FirstP (ELECTRA)	0.492	0.424	0.294	0.552	0.465	0.320
FirstP (DEBERTA)	0.534	0.459	0.319	0.596	0.503	0.350
FirstP (Big-Bird)	0.507	0.435	0.300	0.560	0.473	0.325
FirstP (JINA)	0.488	0.421	0.287	0.532	0.450	0.305
FirstP (MOSAIC)	0.453	0.390	0.266	0.538	0.455	0.310
FirstP (TinyLLAMA)	0.431	0.370	0.246	0.473	0.398	0.262
FirstP (E5-4K)	0.438	0.371	0.247	0.429	0.355	0.234
AvgP	0.478 $(+0.5\%)$	0.411 $(+1.6\%)$	0.292^a $(+5.4\%)$	0.531 $(+0.9\%)$	0.451 $(+1.0\%)$	0.324^a $(+6.7\%)$
MaxP	0.488^a $(+2.6\%)$	0.425^a $(+5.1\%)$	0.306^a $(+10.6\%)$	0.544^a $(+3.3\%)$	0.467^a $(+4.5\%)$	0.338^a $(+11.5\%)$
MaxP (ELECTRA)	0.502 $(+2.0\%)$	0.441^a $(+3.9\%)$	0.319^a $(+8.3\%)$	0.563 $(+2.1\%)$	0.483^a $(+4.0\%)$	0.350^a $(+9.3\%)$
MaxP (DEBERTA)	0.535 $(+0.2\%)$	0.464 $(+1.2\%)$	0.340^a $(+6.7\%)$	0.609^a $(+2.2\%)$	0.519^a $(+3.2\%)$	0.378^a $(+7.9\%)$
SumP	0.486 $(+2.2\%)$	0.418^a $(+3.4\%)$	0.305^a $(+10.2\%)$	0.538 $(+2.1\%)$	0.461^a $(+3.1\%)$	0.337^a $(+11.1\%)$
CEDR-DRMM	0.466 (-2.0%)	0.403 (-0.4%)	0.287^a $(+3.8\%)$	0.533 $(+1.3\%)$	0.458^a $(+2.5\%)$	0.326^a $(+7.6\%)$
CEDR-KNRM	0.483 $(+1.7\%)$	0.413 $(+1.9\%)$	0.291^a $(+5.1\%)$	0.535 $(+1.7\%)$	0.456 $(+2.0\%)$	0.324^a $(+6.8\%)$
CEDR-PACRR	0.496^a $(+4.3\%)$	0.426^a $(+5.3\%)$	0.307^a $(+11.0\%)$	0.549^a $(+4.2\%)$	0.466^a $(+4.4\%)$	0.337^a $(+11.2\%)$
Neural Model1	0.484 $(+1.8\%)$	0.417^a $(+3.1\%)$	0.298^a $(+7.7\%)$	0.537 $(+1.9\%)$	0.459^a $(+2.6\%)$	0.330^a $(+8.8\%)$
PARADE Attn	0.503^a $(+5.7\%)$	0.433^a $(+6.9\%)$	0.311^a $(+12.4\%)$	0.556^a $(+5.6\%)$	0.476^a $(+6.5\%)$	0.344^a $(+13.3\%)$
PARADE Attn (ELECTRA)	0.523^a $(+6.4\%)$	0.456^a $(+7.4\%)$	0.329^a $(+11.7\%)$	0.581^a $(+5.3\%)$	0.495^a $(+6.5\%)$	0.358^a $(+11.9\%)$
PARADE Attn (DEBERTA)	**0.549**^{a} $(+2.9\%)$	**0.475**^{a} $(+3.6\%)$	**0.346**^{a} $(+8.7\%)$	**0.615**^{a} $(+3.2\%)$	**0.522**^{a} $(+3.8\%)$	**0.383**^{a} $(+9.4\%)$
PARADE Avg	0.483 $(+1.5\%)$	0.412 $(+1.8\%)$	0.291^a $(+5.2\%)$	0.534 $(+1.5\%)$	0.457 $(+2.4\%)$	0.318^a $(+4.7\%)$
PARADE Max	0.489^a $(+2.8\%)$	0.420^a $(+3.8\%)$	0.306^a $(+10.8\%)$	0.548^a $(+4.0\%)$	0.470^a $(+5.3\%)$	0.337^a $(+11.0\%)$
PARADE Transf-RAND-L2	0.488^a $(+2.8\%)$	0.423^a $(+4.6\%)$	0.303^a $(+9.7\%)$	0.548^a $(+4.1\%)$	0.469^a $(+5.0\%)$	0.338^a $(+11.6\%)$
PAR. Transf-RAND-L2 (ELECTRA)	0.523^a $(+6.3\%)$	0.454^a $(+6.9\%)$	0.330^a $(+12.2\%)$	0.574^a $(+3.9\%)$	0.488^a $(+5.0\%)$	0.354^a $(+10.6\%)$
PARADE Transf-PRETR-L6	0.494^a $(+4.0\%)$	0.426^a $(+5.3\%)$	0.308^a $(+11.5\%)$	0.554^a $(+5.1\%)$	0.474^a $(+6.1\%)$	0.346^a $(+14.1\%)$
PAR. Transf-PRETR-LATEIR-L6	0.450^a (-5.2%)	0.389^a (-3.9%)	0.277 $(+0.3\%)$	0.501^a (-4.9%)	0.423^a (-5.3%)	0.302 (-0.5%)
LongP (Longformer)	0.500^a $(+3.6\%)$	0.435^a $(+5.3\%)$	0.309^a $(+11.5\%)$	0.568^a $(+5.1\%)$	0.482^a $(+6.1\%)$	0.347^a $(+12.9\%)$
LongP (Big-Bird)	0.452^a (-10.9%)	0.389^a (-10.7%)	0.274^a (-8.8%)	0.477^a (-14.9%)	0.400^a (-15.5%)	0.279^a (-14.2%)
LongP (JINA)	0.503^a $(+2.9\%)$	0.434^a $(+3.1\%)$	0.309^a $(+7.5\%)$	0.558^a $(+4.9\%)$	0.473^a $(+5.2\%)$	0.335^a $(+9.7\%)$
LongP (MOSAIC)	0.456 $(+0.6\%)$	0.393 $(+0.8\%)$	0.280^a $(+5.3\%)$	0.570^a $(+6.0\%)$	0.484^a $(+6.3\%)$	0.350^a $(+13.0\%)$
LongP (TinyLLAMA)	0.452^a $(+4.8\%)$	0.396^a $(+6.9\%)$	0.267^a $(+8.7\%)$	0.505^a $(+6.7\%)$	0.428^a $(+7.6\%)$	0.297^a $(+13.3\%)$
LongP (E5-4K)	0.439 $(+0.1\%)$	0.375 $(+1.0\%)$	0.250 $(+1.3\%)$	0.434 $(+1.1\%)$	0.360 $(+1.6\%)$	0.241^a $(+2.9\%)$

In each column we show a relative gain with respect model's respective *FirstP* baseline: The last column shows the average relative gain of *FirstP*. Best numbers are in **bold**: Results are averaged over three seeds. Unless specified explicitly, the backbone is **BERT-base**.
Statistical significant differences with respect to this baseline are denoted using the superscript superscript **a**. p-value threshold is 0.05.
E5-models were used only in the zero-shot model, i.e., without fine-tuning.

B.5 Additional Accuracy Metrics

In this section we show results for additional effectiveness metrics. MS MARCO and TREC DL results are shown in Table 9. Robust04 datasets are presented and Table 10.