

# Positional Bias in Long-Document Ranking: Impact, Assessment, and Mitigation

Anonymous ACL submission

## Abstract

We tested over 20 Transformer models for ranking of long documents (including recent *LongP* models trained with FlashAttention and RankGPT models “powered” by OpenAI and Anthropic cloud APIs). We compared them with a simple *FirstP* baselines, which applied the *same* model to the truncated input (at most 512 tokens). On MS MARCO, TREC DL, and Robust04 no long-document model outperformed *FirstP* by more than 5% (on average). We hypothesized that the lack of improvement by long-context models is not due to inherent model limitations, but due to benchmark positional bias (most relevant passages tend to occur early in documents), **which is known to exist in MS MARCO**. To further confirm this we analyzed positional relevance distributions across four *long-document* corpora (with six query sets), and observed the same early-position bias. **Surprisingly, we also found bias in several BEIR collections, which are typically categorized as *short-document* datasets.** We then introduced a new diagnostic dataset, MS MARCO FarRelevant, where relevant spans were deliberately placed beyond the first 512 tokens. On this dataset, many long-context models—including RankGPT—failed to generalize and performed near the random baseline, suggesting overfitting to positional bias. Finally, we experimented with de-biasing the training data, but the success of this approach was mixed. Our findings (1) highlight the need for careful benchmark design in evaluating long-context models for document ranking, (2) identify model types that are more robust to positional bias, and (3) motivate further work on approaches to de-bias training data. We release our code and data to support further research.<sup>1</sup>

## 1 Introduction

Various advances in Transformer architectures—including sparse attention (Zaheer et al., 2020; Belt-

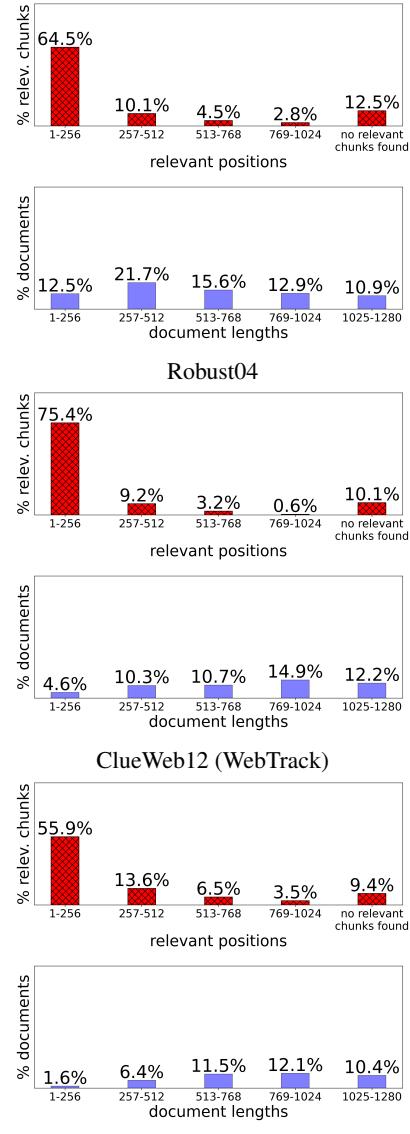


Figure 1: **Illustration of positional relevance bias for three document collections.** We show a distribution of first relevant passage positions (red bars) vs. relevant document lengths (blue bars). Lengths and offsets are measured in the number of subword tokens (BERT-base tokenizer). Best viewed in color. See more results in Figures 7 and 8 in Appendix B.1)

<sup>1</sup>[https://anonymous.4open.science/r/long\\_doc\\_rank\\_model\\_analysis\\_v2-78E9/](https://anonymous.4open.science/r/long_doc_rank_model_analysis_v2-78E9/).

042 agy et al., 2020) and FlashAttention (Dao et al.,  
043 2022)—have motivated a growing interest in long-  
044 document ranking and retrieval. However, de-  
045 spite the ability of these models to process sub-  
046 substantially more text, on *popular retrieval bench-*  
047 *marks*, the improvements of these models over  
048 simpler truncation-based approaches remain sur-  
049 prisingly modest (Dai and Callan, 2019; Gao and  
050 Callan, 2022; Coelho et al., 2024). A widely used  
051 truncation-based *FirstP* baseline (Dai and Callan,  
052 2019)—where models score only the first 512 to-  
053 kens of each document—often performs competi-  
054 tively, or sometimes even better, than long-context  
055 counterparts (see, e.g., Table 1).

056 Despite anecdotal knowledge about the presence  
057 of this phenomenon in the MS MARCO documen-  
058 t-retrieval collection among TREC Deep Learning  
059 track participants and some early reports (Hofstät-  
060 ter et al., 2020b, 2021a) the available evidence  
061 has been scattered and incomplete. In particu-  
062 lar, despite the track’s five-year history, none of  
063 the track’s overview papers mentioned this issue  
064 (Craswell et al., 2020, 2021a,b, 2022, 2023).

065 Moreover, it remains unclear whether these limi-  
066 tations stem from model deficiencies or from char-  
067 acteristics of the benchmarks themselves. In this  
068 paper, we initially hypothesized that both factors—  
069 model robustness and benchmark design—may be  
070 responsible for the limited gains of long-document  
071 models over *FirstP* baselines. However, our find-  
072 ings suggest that benchmark design, particularly  
073 positional relevance bias, is the dominating factor.

074 To verify our research hypotheses, we first con-  
075 ducted a large-scale, systematic study of over  
076 20 Transformer-based ranking models (Devlin  
077 et al., 2019; Vaswani et al., 2017) for long-  
078 document retrieval. This was done using three  
079 popular document collections—MS MARCO Doc-  
080 uments v1/v2 (Craswell et al., 2021a) and Rob-  
081 bust04 (Clarke et al., 2004)—along with diverse  
082 query sets (both large and small) several Trans-  
083 former backbones and multiple training seeds. In  
084 addition to locally trained models, we also assessed  
085 a listwise LLM ranker RankGPT (Sun et al., 2023)  
086 “powered” by OpenAI (OpenAI, 2023) and An-  
087 thropic (Anthropic, 2024) cloud APIs. Despite  
088 the increased context capacity of long-document  
089 models, we found that none of them consistently  
090 outperformed their *FirstP* baselines by more than  
091 5% on average.

092 Next, we estimated positional relevance bias

093 across five document collections (including MS  
094 MARCO v2) and more than six query sets. As  
095 can be seen in Fig. 1, in the vast majority of cases  
096 the first relevant passage occurred within the initial  
097 512 tokens. In contrast, the distribution of **relevant**  
098 passage positions is more uniform and has a long  
099 tail. This confirms the presence of positional bias  
100 not only in MS MARCO, but also in other TREC  
101 collections (a complete set of plots is provided in  
102 Fig. 7 of Appendix § B.4).

103 Our initial exploration prompted two *broad* re-  
104 search questions:

- **RQ1:** How robust are long-document models  
105 to the positional-bias of relevant passages?
- **RQ2:** To what extent has the research com-  
106 munity advanced the state of long-document  
107 ranking models? Specifically, do current ap-  
108 proaches yield substantial improvements over  
109 *FirstP* baselines? Considering that all existing  
110 long-document models are at least 2× slower  
111 than their respective *FirstP* counterparts (see  
112 Figure 3, § A.4), it is reasonable to question  
113 the practicality of such models and to consider  
114 whether *FirstP* variants might be preferable in  
115 real-world applications.

116 To answer these questions, we constructed a new  
117 *diagnostic* synthetic collection MS MARCO Far-  
118 Relevant where relevant passages were not present  
119 among the first 512 tokens. On this dataset, many  
120 long-context models—including RankGPT (Sun  
121 et al., 2023)—failed to generalize and performed  
122 at a random baseline level, suggesting overfitting  
123 to positional bias. Poor performance of several  
124 models on our new synthetic collection prompted  
125 another important question **RQ3:** Can de-biasing  
126 training data mitigate model overfitting to pos-  
127 i-tional bias? We addressed this question by eval-  
128 uating an existing de-biasing approach (Hofstätter  
129 et al., 2021a).

130 Our paper makes the following contributions:

- We re-examined the issue of positional relevance bias, gathered extensive evidence con-  
131 firming its presence in both long- and short-  
132 document datasets, showed that it negatively  
133 affects all models including recent LLM-  
134 based rankers, and evaluated the robustness of  
135 ranking models against this bias using a new  
136 diagnostic dataset MS MARCO FarRelevant.

- Our work highlights the need for careful benchmark design in evaluating long-context models for document ranking, which do not mask the benefits of long-context models and identifies model types that are more robust to positional bias;
- We perform an extensive reproduction study of over 20 ranking models using two established benchmark collections for long-document retrieval and ranking;
- We experiment with an existing approach to de-biasing training data and motivate further research in this area.

We release our code and data to support further research.<sup>2</sup>

## 2 Related Work

*Neural Ranking* models have been a widely studied topic in recent years (Guo et al., 2019), though the success of early approaches was debated (Lin, 2019). This changed with the introduction of BERT, a bi-directional encoder-only Transformer model (Devlin et al., 2019), which significantly outperformed previous methods in both NLP (Devlin et al., 2019) and information retrieval (IR) tasks (Nogueira et al., 2019; Craswell et al., 2021a).

Several Transformer-based models, such as ELECTRA (Clark et al., 2020) and DEBERTA (He et al., 2021), have improved upon BERT through different training strategies and datasets. However, due to their architectural similarities, we—following Lin et al. (2021)—refer to these collectively as BERT models.

Despite their strong performance, neural models are vulnerable to distribution shifts, often relying on superficial features and exhibiting various *biases*. They do not consistently outperform BM25 on out-of-domain data (Thakur et al., 2021; Mokri et al., 2021), can be misled by minor text modifications and distractor sentences (MacAvaney et al., 2022), or reformulated queries (Penha et al., 2022). They also struggle to effectively utilize information located in the middle of long input contexts (Liu et al., 2024).

A seemingly similar yet distinct issue of positional relevance bias has been identified in information retrieval settings, particularly within the

MS MARCO document-retrieval collection (Hofstätter et al., 2020b; Coelho et al., 2024). Some studies have reported strong performance of *FirstP* baselines on long-document retrieval collections, interpreting this as evidence of benchmark-induced positional bias (Zhu et al., 2024; Rau et al., 2024).

Because positional relevance bias can “obscure” benefits of long-document ranking models, (Rau et al., 2024) proposed to compare these models with *RandP* baselines, which score a randomly selected passage. However, we believe this approach is problematic for two reasons. First, since a *RandP* model often fails to score an entire relevant passage, it artificially underestimates model accuracy, making comparisons with *RandP* unfair. Second, this approach does not address the core problem of biased benchmarks, which still allow models to exploit the shortcut of focusing only on the document’s initial portion.

However, strong performance of *FirstP* baseline is only indirect evidence, potentially resulting from implementation bugs, suboptimal training methods, or model-inherent biases. (Coelho et al., 2024) found that embedding models trained on MS MARCO “dwell” in the beginning and are somewhat less effective when relevant information is present elsewhere in a document. The study used only two embedding models and has additional limitations: (1) it is unclear if the bias is fully attributable to data (2) the models were not tested under conditions of extreme positional bias, (3) no mitigation strategy was evaluated.

To address the issue with existing benchmarks, (Zhu et al., 2024) proposed a *LongEmbed* benchmark with two synthetic tasks where relevant mini-passages were scattered uniformly across documents whose lengths varied from 256 to 32768 tokens. However, as discussed in §B.3 of the Appendix, these synthetic sets are quite unnatural and lack diversity. Furthermore, Zhu et al. (2024) provide only small query sets and no in-domain training data, making it difficult to assess the upper performance bound that models can achieve on this dataset. As another important limitation Zhu et al. (2024), only explored training-free extensions of positional encoding and did not investigate methods to de-bias training data. In contrast, Hofstätter et al. (2021a) proposed to de-bias training data using a simple-yet-effective approach. However, they did *not* evaluate it on challenging long-document datasets.

<sup>2</sup>[https://anonymous.4open.science/r/long\\_doc\\_rank\\_model\\_analysis\\_v2-78E9/](https://anonymous.4open.science/r/long_doc_rank_model_analysis_v2-78E9/).

Due to the quadratic complexity of the Transformer’s attention mechanism (Vaswani et al., 2017; Bahdanau et al., 2015), early Transformer models restricted input length to a maximum of 512 (subword) tokens. Until around 2022, two main strategies were used to process long documents: (1) localizing attention and (2) splitting documents into smaller, independently processed chunks. Attention-localization methods apply a limited-span (sliding window) attention with selective global attention. Given the vast number of such approaches (see Tay et al. 2020), evaluating all of them is impractical. Therefore, we focus on two popular models: Longformer (Beltagy et al., 2020) and BigBird (Zaheer et al., 2020). More recently, it has also become feasible to train long-context models with (an IO-efficient) FlashAttention algorithm without sparsifying attention (Dao et al., 2022). In our work we use two such models: JINA (Günther et al., 2023) and MOSAIC (Portes et al., 2023).

In summary, the methods to tackle longer documents are divided into *LongP* methods—where longer document lengths are “natively” supported and *SplitP* methods—where the longer document cannot be processed as a whole and needs to be processed in chunks. The results of each chunk are aggregated together using various aggregation techniques, including a computation of a maximum or a weighted-sum prediction score (Yilmaz et al., 2019; Dai and Callan, 2019; MacAvaney et al., 2019). This includes MaxP (Dai and Callan, 2019), AvgP, SumP (MacAvaney et al., 2019), as well as PARADE Avg and PARADE Max models (MacAvaney et al., 2019). **MaxP is an important baseline that computes relevance scores for each chunk independently and takes their maximum.**

Some *SplitP* approaches aggregate using simple neural networks. This includes all CEDR (MacAvaney et al., 2019) models, the Neural Model 1 (Boytsov and Kolter, 2021), and the PARADE Attention model (Li et al., 2024). In contrast, PARADE Transformer (Li et al., 2024) models’ aggregator network is an additional Transformer model. Due to space constraints, a detailed description of document-splitting (*SplitP*) approaches is provided in the Appendix § C.

The recent success of decoder-only models—commonly known as LLMs—has led to a new generation of cross-encoding *LongP* models that natively support longer contexts. First, a pre-trained cross-encoding decoder-only model can be directly

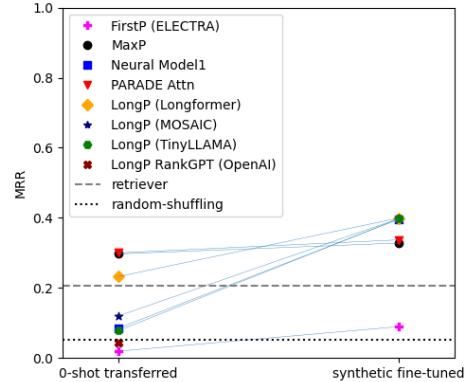


Figure 2: Zero-shot vs. fine-tuned performance on MS MARCO FarRelevant. **This figure shows results for a representative set of models.**

fine-tuned in a ranking or embedding task (Ma et al., 2023). Second, the RankGPT approach (Sun et al., 2023) formulates document ranking as a generation task: The model is prompted with a list of documents and an instruction to generate their ranking—an ability made possible through instruction-tuning and/or alignment (Wei et al., 2022; Ouyang et al., 2022). When the combined length of concatenated documents exceeds the input context size, RankGPT employs an overlapping sliding window strategy, followed by aggregation of the results.

### 3 Experiments

#### 3.1 Data

Our primary datasets, used for both training and evaluation, consist of several realistic collections (along with their respective query sets) and synthetic data. All datasets are in *English*. Document and query statistics are provided in Appendix § B.1 Tables 9 and 10.

The realistic datasets include two versions of the MS MARCO Documents collection (v1 and v2), MS MARCO Passages collection (v1), (Bajaj et al., 2016; Craswell et al., 2020, 2021b), Robust04 (Voorhees, 2004), and the NQ BEIR, which is a Natural Question (Kwiatkowski et al., 2019) subset incorporated into the BEIR benchmark (Thakur et al., 2021).

Following Hofstätter et al. (2021a), we created a de-biased version of MS MARCO by randomly splitting documents at word boundaries and then concatenating the shuffled segments. This debiasing process is only partial, as shorter documents remain more frequent. To address this imbal-

Retriever / Ranker	MS MARCO	TREC DL	title	Robust04	Avg. gain over FirstP
	dev	(2019-2021)		description	
	MRR	NDCG@10	NDCG@20		
<b>BM25</b> retriever (if different from BM25)	<b>0.274</b> 0.312	<b>0.545</b> 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) <b>zero-shot</b>	0.380	0.641	0.438	0.429	-
FirstP RankGPT (GPT-4o-mini)	-	<b>0.708</b>	-	0.562	-
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 <sup>a</sup> (+1.9%)	0.488 <sup>a</sup> (+2.6%)	0.544 <sup>a</sup> (+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 <sup>a</sup> (-3.2%)	0.671 (-0.1%)	0.535 (+0.2%)	0.609 <sup>a</sup> (+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 <sup>a</sup> (-2.3%)	0.629 (-0.5%)	0.466 (-2.0%)	0.533 (+1.3%)	-0.9%
CEDR-KNRM	0.379 <sup>a</sup> (-3.8%)	0.630 (-0.3%)	0.483 (+1.7%)	0.535 (+1.7%)	-0.2%
CEDR-PACRR	0.395 (+0.3%)	0.643 <sup>a</sup> (+1.6%)	0.496 <sup>a</sup> (+4.3%)	0.549 <sup>a</sup> (+4.2%)	+2.6%
Neural Model1	0.398 (+0.9%)	0.650 <sup>a</sup> (+2.8%)	0.484 (+1.8%)	0.537 (+1.9%)	+1.8%
PARADE Attn	0.416 <sup>a</sup> (+5.5%)	0.652 <sup>a</sup> (+3.1%)	0.503 <sup>a</sup> (+5.7%)	0.556 <sup>a</sup> (+5.6%)	<b>+5.0%</b>
PARADE Attn (ELECTRA)	0.431 <sup>a</sup> (+3.3%)	0.680 <sup>a</sup> (+2.7%)	0.523 <sup>a</sup> (+6.4%)	0.581 <sup>a</sup> (+5.3%)	+4.4%
PARADE Attn (DEBERTA)	0.422 <sup>a</sup> (+1.6%)	0.688 <sup>a</sup> (+2.4%)	<b>0.549<sup>a</sup></b> (+2.9%)	<b>0.615<sup>a</sup></b> (+3.2%)	+2.5%
PARADE Avg	0.392 (-0.6%)	0.646 <sup>a</sup> (+2.1%)	0.483 (+1.5%)	0.534 (+1.5%)	+1.1%
PARADE Max	0.405 <sup>a</sup> (+2.7%)	0.655 <sup>a</sup> (+3.5%)	0.489 <sup>a</sup> (+2.8%)	0.548 <sup>a</sup> (+4.0%)	+3.3%
PARADE Transf-RAND-L2	0.419 <sup>a</sup> (+6.3%)	0.655 <sup>a</sup> (+3.6%)	0.488 <sup>a</sup> (+2.8%)	0.548 <sup>a</sup> (+4.1%)	+4.2%
PARADE Transf-RAND-L2 (ELECTRA)	<b>0.433<sup>a</sup></b> (+3.9%)	0.670 (+1.2%)	0.523 <sup>a</sup> (+6.3%)	0.574 <sup>a</sup> (+3.9%)	+3.8%
PARADE Transf-PRETR-L6	0.402 <sup>a</sup> (+1.9%)	0.643 (+1.6%)	0.494 <sup>a</sup> (+4.0%)	0.554 <sup>a</sup> (+5.1%)	+3.2%
LongP (Longformer)	0.412 <sup>a</sup> (+1.9%)	0.668 <sup>a</sup> (+3.9%)	0.500 <sup>a</sup> (+3.6%)	0.568 <sup>a</sup> (+5.1%)	+3.6%
LongP (Big-Bird)	0.397 <sup>a</sup> (-2.9%)	0.651 (-0.7%)	0.452 <sup>a</sup> (-10.9%)	0.477 <sup>a</sup> (-14.9%)	-7.3%
LongP (JINA)	0.416 <sup>a</sup> (-1.5%)	0.665 <sup>a</sup> (+1.7%)	0.503 <sup>a</sup> (+2.9%)	0.558 <sup>a</sup> (+4.9%)	+2.0%
LongP (MOSAIC)	0.421 (-0.4%)	0.664 <sup>a</sup> (+3.3%)	0.456 (+0.6%)	0.570 <sup>a</sup> (+6.0%)	+2.4%
LongP (TinyLLAMA)	0.402 <sup>a</sup> (+1.7%)	0.608 (-1.1%)	0.452 <sup>a</sup> (+4.8%)	0.505 <sup>a</sup> (+6.7%)	+3.0%
LongP (E5-4K) <b>zero-shot</b>	0.353 <sup>a</sup> (-7.1%)	0.649 (+1.3%)	0.439 (+0.1%)	0.434 (+1.1%)	-1.1%
LongP RankGPT (GPT-4o-mini)	-	0.706 (-0.3%)	-	0.562 (+0.0%)	-0.1%

In each column we show a relative gain with respect model’s respective *FirstP* baseline: The last column shows the average relative gain over *FirstP* baselines. 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 **a**. *p*-value threshold is 0.01 for an MS MARCO development collection and 0.05 otherwise.

Table 1: Comparison between long-document models and respective *FirstP* (truncation) baselines. Results for MS MARCO, TREC DL, and Robust04.

ance, we experimented with oversampling longer documents, but this approach did not yield improvements.

Our synthetic data consists of two subsets from LongEmbed (Zhu et al., 2024) and our newly created MS MARCO FarRelevant collection. All these can be considered a variant of the needle-in-the-haystack tests, where an informational “nugget” is randomly embedded within unrelated text (Saad-Falcon et al., 2024; Zhu et al., 2024; Liu et al., 2024). We use two LongEmbed subsets: Needle and Passkey: Each has 800 question-document pairs with document lengths varying from (approximately) 256 to 32768 tokens.

MS MARCO FarRelevant was created by randomly mixing relevant and non-relevant pas-

sages from the MS MARCO *Passage* collection (Craswell et al., 2020) in such a way that (1) each document contained exactly one relevant passage, (2) this passage did not start before token 512, and (3) the document length was at most 1431 tokens with (see an algorithm in the Appendix § B.2). It has about 0.5 million documents with an average length of 1.1K tokens. Due to MS MARCO datasets having a **non-commercial license**, MS MARCO FarRelevant has the same licensing restriction. In Appendix § B.3, we present dataset examples and argue that—while all these collections share the limitation of not resembling natural documents—MS MARCO FarRelevant offers greater diversity and serves as a more suitable benchmark for evaluating text retrieval systems.

Ranker	MS MARCO dev		TREC DL (2019-2021)		FarRelevant		LongEmbed	
	MRR	NDCG@10	MRR	MRR	zero-shot transf.	fine-tuned	Needle	Paskey
BM25	0.274	0.545	0.207	0.207	0.305	0.339		
Original MS MARCO training set								
FirstP (ELECTRA)	0.417	0.662	0.019	0.089	0.205	0.235		
MaxP (ELECTRA)	0.414	0.659	0.328	0.349	<b>0.331</b>	<b>0.338</b>		
PARADE Attn (ELECTRA)	0.431	<b>0.680</b>	<b>0.338</b>	0.354	0.270	0.334		
PARADE Transf-RAND-L2 (ELECTRA)	<b>0.433</b>	0.670	0.229	<b>0.432</b>	0.321	0.333		
CEDR-KNRM	0.379	0.630	0.055	0.382	0.129	0.166		
De-biased MS MARCO (Hofstätter et al., 2021a)								
MaxP (ELECTRA)	0.377 <sup>a</sup> (-9.1%)	0.665 (+0.8%)	0.321 (-2.1%)	0.349	0.316 (-4.6%)	0.325 (-3.9%)		
PARADE Attn (ELECTRA)	0.390 <sup>a</sup> (-9.4%)	0.653 <sup>a</sup> (-3.9%)	0.326 (-3.6%)	0.354	0.251 (-7.1%)	0.330 (-1.0%)		
PARADE Transf-RAND-L2 (ELECTRA)	0.410 <sup>a</sup> (-5.4%)	0.677 (+1.0%)	0.328 <sup>a</sup> (+43.5%)	<b>0.432</b>	0.259 <sup>a</sup> (-19.4%)	0.331 (-0.7%)		
CEDR-KNRM	0.269 <sup>a</sup> (-29.0%)	0.503 <sup>a</sup> (-20.2%)	0.202 <sup>a</sup> (+268.8%)	0.382	0.121 (-5.8%)	0.181 (+8.6%)		

Except *FirstP* we train each model using the original and the de-biased MS MARCO. For each model trained on the de-biased dataset, we compute a gain (or loss) compared to the same model trained on the original training set. Statistical significance of the differences are denoted using the superscript superscript **a** (*p*-value threshold is 0.05 for TREC DL and 0.01 for other collections). Best numbers are in **bold**: Results are averaged over three seeds.

Table 2: De-biasing effectiveness. Performance of (selected) rankers trained on original and de-biased MS MARCO.

Robust04 is another relatively small dataset containing 0.5 million documents, comprising a mix of news articles and government records, some of which are quite lengthy. However, it includes only a limited number of queries (250), making it a challenging benchmark for training models in low-data scenarios. Each query consists of a title and description: the title expresses a concise information need, while the description provides a more detailed request, often written in proper English prose. We use Robust04 in a cross-validation setting with folds created by [Huston and Croft \(2014\)](#) provided via IR-datasets ([MacAvaney et al., 2021](#)).<sup>3</sup>

MS MARCO v1 was created from the MS MARCO reading comprehension dataset ([Bajaj et al., 2016](#)) and consist of two *related* collections: MS MARCO Passages and MS MARCO 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., 2021b](#)). 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 judgments were created for the passage variant of the dataset. Then, document-level relevance labels were created

<sup>3</sup>In that we do not train Robust04 models from scratch, but rather fine-tune models trained on MS MARCO Documents.

by transferring passage-level relevance to original documents from which passages were extracted.

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 judgments—separated into training and developments subsets—there is a small number (98) of queries that have “dense” judgments provided by NIST assessors for TREC 2019 and 2020 deep learning (DL) tracks ([Craswell et al., 2021a](#)).

The MS MARCO v2 collection was created for the TREC 2021 Deep Learning (DL) track ([Craswell et al., 2021b](#)). It is an expanded version of MS MARCO v1 and incorporates a subset of sparse relevance judgments from MS MARCO v1. In the training set, newly added documents lack both positive and negative judgments, introducing a bias where many relevant documents are mistakenly considered non-relevant. As a result, we do not train on v2 data and only use it for testing.

### 3.2 Setup

We focus cross-encoding rankers, which process queries concatenated with documents ([Nogueira and Cho, 2019](#)). This includes various *SplitP* and *LongP* models discussed in § 2 and in the Appendix § C. As a reference point we also tested a bi-encoder embedding 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 are based on BERT models (bi-directional encoder-only Transformer) with 100M–200M parameters (see Table 11). Additionally, we evaluated two types of LLM rankers: (1) a fine-tuned TinyLLAMA model which delivers strong performance relative to its compact size (Zhang et al., 2024) and (2) generative black-box LLMs. For (2), we used OpenAI’s GPT-4o-mini (OpenAI, 2023) and Anthropic’s Claude Haiku-3 (Anthropic, 2024), both of which support at least a 128K-token input context.

We trained each model using *three* seeds, except the bi-encoder model E5 (Zhu et al., 2024) and RankGPT (Sun et al., 2023), which were evaluated only in the zero-shot mode. Due to the high evaluation cost (**more than \$1 per one thousand query-document pairs**),<sup>4</sup> we also did not test RankGPT on some query sets, in particular, excluding MS MARCO dev set, as it is quite large.

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 § A.1. 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-based metrics are presented in the Appendix (see § A.5).

### 3.3 Results

**Realistic Datasets.** Our main experimental results for MS MARCO, TREC DL 2019–2021, and Robust04 are presented in Table 1. Fig. 2 and Table 4 (in the Appendix § A.5) show results for MS MARCO FarRelevant. In the Appendix (see A.3) we also 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).<sup>5</sup>

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 § A.3 in the Appendix).

To ease understanding and simplify presentation, we display key results for a representative sample of models in Fig. 3 and Fig. 2 (in § 1). Moreover, in Table 1 we present only a single aggregate number for all TREC DL query sets, which is obtained by combining all the queries and respective relevance judgments (i.e., we post an overall average rather than an average over the mean values for 2019, 2020, and 2020). More detailed results, including both OpenAI and Anthropic RankGPT, are available in Appendix A.5, specifically in Tables 7 and 8.

From Fig. 3 and Table 1 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 sometimes match or underperform their *FirstP* baselines on one or more dataset and some of these differences are statistically significant. In particular, this is true for RankGPT (Sun et al., 2023), CEDR-DRMM, CEDR-KNRM (MacAvaney et al., 2019), JINA (Günther et al., 2023) and MOSAIC (Portes et al., 2023).

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 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 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 perform better).

Based on our analysis of positions of first rele-

<sup>4</sup><https://openai.com/api/pricing/>

<sup>5</sup>Note, however, that TRANSF-PRETR-L2 has only four attention heads.

vance passages, we hypothesize 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 Figure 1 and Figure 7 in Appendix B.4).

**Synthetic Data.** To further support this hypothesis, we carried out two sets of experiments using our new MS MARCO FarRelevant collection, where a relevant passage did not start until token 512. We carried out both the zero-shot experiment (evaluation of the model trained on MS MARCO) as well fine-tuning experiment using 50K in-domain queries (from the MS MARCO FarRelevant).

Results for key models are shown in Fig. 2 and more detailed results can be found in Table 4 of the Appendix A.5. The *FirstP* models performed roughly at the random-baseline level in both zero-shot and fine-tuning modes (**RQ1**). Because of this our main baselines here are 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).

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 1). Both GPT-4o-mini and Claude Haiku-3 RankGPT perform at the random-baseline level as well! As a sanity check, and to verify if more accurate and expensive LLMs could do better, we assessed performance of GPT-4o for a sample of 100 queries. The respective RankGPT ([Sun et al., 2023](#)) ranker was still not better than a random baseline (**RO1**).

Simple aggregation models including MaxP and PARADE Attention had good zero-shot accuracy, but benefited little from fine-tuning on MS MARCO FarRelevant (**RQ1**); 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 (**RQ1** and **RQ2**).

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 (**RO1** and **RO2**)

With exception of RankGPT (Sun et al., 2023)

on TREC DL 2019-2021, 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**).

**Bias Mitigation.** To address **RQ3**, we trained four representative models on a de-biased version of MS MARCO (see Appendix § A.1.3 for selection rationale), using the de-biasing approach by Hofstätter et al. (2021a), and tested them on MS MARCO FarRelevant as well as on Needle/Passkey subsets of LongEmbed (Zhu et al., 2024). We also evaluated four models fine-tuned on MS MARCO FarRelevant on TREC DL 2019-2020 query sets. Due to the substantial NDCG@10 drop (0.1–0.15) observed for PARADE Transformer and CEDR-KNRM, we concluded that fine-tuning on purely synthetic data is not viable and did not pursue it further.

According to Table 2, de-biasing improved performance of CEDR-KNRM and PARADE Transformer on MS MARCO FarRelevant. Yet, it mostly caused performance degradation on the original MS MARCO dataset and on LongEmbed subsets. It did not benefit the MaxP and PARADE Attention models, which were the most robust to positional bias.

We further tested de-biased models on short-document collections using MS MARCO Passage collection and a subset of seven BEIR collections (Thakur et al., 2021). According to Table 3 in Appendix A, for three out of four models de-biasing has either positive or small negative effect (at most 1% degradation). In particular, on BEIR NQ subset PARADE Transformer and CEDR-KNRM improved by 2.5% and 11%, respectively. Despite the positive average improvement, we also observe substantial degradation for several datasets.

For example, effectiveness of CEDR-KNRM decreases by 12.4% and 4.6% on SciFact and SciDocs, respectively. One possible reason is that short-document collections can have relevance position bias as well (see Figure 8 in Appendix B.1). Thus, training on the original (biased) MS MARCO collection teaches the model to rely on a helpful shortcut—focusing more on the beginning of a document—whereas training on de-biased data removes this beneficial behavior.

619 In summary, de-biasing results are promising,  
620 but they also suggest that mitigating positional bias  
621 remains a challenging problem (**RQ3**).  
622

623 **Key Findings.** Based on our results, we make  
the following key observations:

- We confirmed presence of positional relevance bias in both long- and short-document datasets.
- Not only positional relevance bias diminished benefits of supporting longer document contexts, but it also leads to model overfitting to the bias and performing poorly in a zero-shot setting when the distribution of relevant passages changed substantially;
- In that, we found that de-biasing training data had a mixed success: Although it improved the effectiveness of some models on some long-document ranking tasks without substantial degradation on short-document collections, it degraded performance in several cases especially for MaxP and PARADE Attention models, which were already intrinsically more robust to relevance position bias (which can be due to many short-document collections having positional relevance bias as well).
- It is also worth highlighting the consistently strong performance of PARADE models on both standard long-document collections and MS MARCO FarRelevant. The best PARADE models substantially outperformed the best *LongP* models in both zero-shot and fine-tuning settings, although the specific models leading in each setting may differ.

## 652 4 Conclusion

653 In this work, we revisited the problem of positional  
654 relevance bias in long-document retrieval and pre-  
655 sented extensive evidence of its *widespread* pres-  
656 ence across existing benchmarks. Using both real  
657 and synthetic datasets—including our new *diagno-  
658 stic* dataset, MS MARCO FarRelevant—we eval-  
659 uated effectiveness of over 20 ranking models as  
660 well as their robustness to relevance positional bias.  
661

662 Our findings highlight the importance of bench-  
663 mark design that does not obscure the benefits of  
664 long-context modeling. We identified model fami-  
665 lies (i.e., PARADE Attention and MaxP) that are  
666 more robust to positional bias, and confirmed the

667 strong performance of PARADE models (Li et al.,  
668 2024), which remain competitive even against re-  
669 cent long-context architectures.  
670

671 Finally, our de-biasing experiments yielded lim-  
672 ited and/or inconsistent gains, motivating further  
673 research into more effective mitigation strategies,  
including combining de-biasing with training on  
well-designed synthetic data.  
674

## 674 5 Limitations

675 Our paper has several limitations related primarily  
676 to the choice of datasets, models, and the strength  
677 of evidence for the positional bias of relevant pas-  
678 sages.  
679

680 First of all, our evaluation uses only cross-  
681 encoding ranking models. With an exception of  
682 E5-4K model, which is used in the zero-shot rank-  
683 ing mode, we do not train or evaluate bi-encoding  
684 models (typically used to create query and docu-  
685 ment embeddings for the first-stage retrieval). We  
686 nonetheless believe that—given a large number  
687 of proposals for long-document ranking—a repro-  
688 duction and evaluation of cross-encoding long-  
689 document rankers is a sufficiently important topic  
690 that alone warrants a publication.  
691

692 Moreover, as we explain below, we also use  
693 cross-encoding rankers as a tool to detect and ex-  
694 pose bias in the position of relevant information. In  
695 that, cross-encoders are easier to train using stan-  
696 dard (rather than high-memory) GPUs with mini-  
697 batch size one and gradient accumulation. They  
698 also typically require only one epoch to converge  
699 (only a few models need two or three epochs). In  
700 contrast, bi-encoders are trained using large batches  
701 with in-batch negatives for multiple epochs (e.g.,  
702 Karpukhin et al. (2020) report using at least 40  
703 epochs).  
704

705 Second, a bulk of our ranking experiments  
706 uses only two *English* document collections: MS  
707 MARCO Documents v1 and v2 (Craswell et al.,  
708 2021b) and Robust04 (Clarke et al., 2004). How-  
709 ever, we have to restrict the choice of datasets to  
710 make multi-seed evaluations of 20+ models feasi-  
711 ble. Yet, to corroborate existence of positional bias  
712 we used two additional popular long-document col-  
713 lections: Gov2 (Allan et al., 2008) and ClueWeb12  
714 (Collins-Thompson et al., 2013a). For the study  
715 on robustness of model to positional bias and its  
mitigation, we used two additional synthetic col-  
lections: MS MARCO FarRelevant and two sub-  
sets from LongEmbed (Zhu et al., 2024) together  
716

716 with short-document collections MS MARCO Pas-  
717 sages (v1) (Craswell et al., 2020) and BEIR NQ  
718 (Kwiatkowski et al., 2019; Thakur et al., 2021).

719 One could argue that the limited improvements  
720 over *FirstP* baselines result from the models’ in-  
721 ability to handle long contexts. To address this  
722 concern, we trained and evaluated a diverse set  
723 of cross-encoding ranking models, including both  
724 split-and-aggregate models and models explicitly  
725 designed for long input sequences. Additionally,  
726 we assessed cloud-based RankGPT rankers, which  
727 have shown strong performance in recent research  
728 (Sun et al., 2023).

729 However, we can still test only a limited number  
730 of models: One might always argue that there are  
731 untested architectures that would outperform *FirstP*  
732 baselines by a much larger margin. To demonstrate  
733 that selected models can, in principle, benefit from  
734 long contexts and decisively outperform simple  
735 baselines such as *FirstP* and even *MaxP* models we  
736 trained and/or evaluated them on a synthetic collec-  
737 tion MS MARCO FarRelevant, which can be seen  
738 as a challenging version of a needle-in-the-haystack  
739 test. This is still a limiting experiment, because  
740 synthetic collections—with documents composed  
741 from randomly selected passages—are imperfect  
742 proxies for real-life datasets. In Appendix § B.3  
743 we discuss this limitation in detail and, neverthe-  
744 less, argue that MS MARCO FarRelevant is a more  
745 suitable synthetic benchmark for evaluating text  
746 retrieval systems compared to LongEmbed subsets  
747 Needle and Passkey (Zhu et al., 2024).

748 In summary, we provided three types of evidence  
749 for positional bias of relevant passages: strong  
750 performance of *FirstP* models on standard collec-  
751 tions, direct estimation of the distribution of re-  
752 relevant passages using substring matching and LLM  
753 relevance judges (Upadhyay et al., 2024), as well  
754 as experimentation with the synthetic collection  
755 MS MARCO FarRelevant where relevant passages  
756 distribution was not skewed towards the beginning  
757 of a document. Each experiment provided imper-  
758 fect/limited evidence on its own, but together they  
759 strongly supported the existence of relevance posi-  
760 tion bias.

761 While our analysis confirms a strong early-  
762 position relevance bias across multiple retrieval  
763 benchmarks, we acknowledge that this pattern may  
764 not generalize to all domains. For example, prior  
765 work has shown that in scientific abstracts, both  
766 the first and last sentences tend to be crucial (Ruch

767 et al., 2006). Investigating positional relevance pat-  
768 terns in such domains is an important direction for  
769 future work.

770 In our experiments with Robust04 and MS  
771 MARCO, we truncated documents to a maximum  
772 of 1431 BERT tokens. However, this constraint  
773 did not hinder our ability to address key research  
774 questions. As detailed in Appendix § A.2, using  
775 larger inputs led to only marginal improvements.

776 Notably, when models trained on MS MARCO  
777 were applied to MS MARCO FarRelevant in a  
778 zero-shot setting, we observed a significant drop in  
779 MRR (at least 17%) across many models. Several  
780 models—including RankGPT (Sun et al., 2023)—  
781 even failed to outperform a random-shuffling base-  
782 line, despite MS MARCO FarRelevant documents  
783 containing fewer than 1500 tokens.

784 Interestingly, despite this token limitation, sev-  
785 eral long-context models significantly outper-  
786 formed both *FirstP* and *MaxP* baselines by over  
787 20%. This suggests that even datasets with rel-  
788 atively short documents can serve as a mean-  
789 ingful benchmark for distinguishing strong and  
790 weak long-context models—unlike the original MS  
791 MARCO, where all models have close accuracy.

## 6 Ethics Statement

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

800 In terms of the environmental impact, our com-  
801 putational requirements are rather modest, because  
802 we only fine-tuned our models rather than trained  
803 them from scratch. These models were also rather  
804 small by modern standards. Except 1B-parameter  
805 TinyLLAMA (Zhang et al., 2024), each model has  
806 about 100M parameters (see Table 11 for details).  
807 Despite training and testing 20+ models with three  
808 seeds, we estimate to have spent only about 6400  
809 GPU hours for our main experiments. 96% of the  
810 time we used NVIDIA A10 (or similarly-powerful)  
811 RTX 3090 GPUs and 4% of the time we used  
812 NVIDIA A6000.

813 We believe this is roughly equivalent to train-  
814 ing a single 1B-parameter TinyLLAMA model,  
815 which required about 3400 GPU hours using a

817	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). <sup>6</sup>	Charles L. A. Clarke, Nick Craswell, and Ian Soboroff. 2004. Overview of the TREC 2004 terabyte track. In <i>TREC</i> , volume 500-261 of <i>NIST Special Publication</i> . National Institute of Standards and Technology (NIST).	865 866 867 868 869
821	<b>References</b>	João Coelho, Bruno Martins, João Magalhães, Jamie Callan, and Chenyan Xiong. 2024. Dwell in the beginning: How language models embed long doc- uments for dense retrieval. In <i>ACL (Short Papers)</i> , pages 370–377. Association for Computational Lin- guistics.	870 871 872 873 874 875
822	James Allan, Javed A. Aslam, Virgil Pavlu, Evangelos Kanoulas, and Ben Carterette. 2008. Million query track 2008 overview. In <i>TREC</i> , volume 500-277 of <i>NIST Special Publication</i> . National Institute of Standards and Technology (NIST).	Kevyn Collins-Thompson, Paul N. Bennett, Fernando Diaz, Charlie Clarke, and Ellen M. Voorhees. 2013a. TREC 2013 web track overview. In <i>TREC</i> , volume 500-302 of <i>NIST Special Publication</i> . National Insti- tute of Standards and Technology (NIST).	876 877 878 879 880
823	Anthropic. 2024. <a href="#">The claude 3 model family: Opus, sonnet, haiku</a> .	Kevyn Collins-Thompson, Paul N. Bennett, Fernando Diaz, Charlie Clarke, and Ellen M. Voorhees. 2013b. TREC 2013 web track overview. In <i>TREC</i> , volume 500-302 of <i>NIST Special Publication</i> . National Insti- tute of Standards and Technology (NIST).	881 882 883 884 885
824	Negar Arabzadeh and Charles LA Clarke. 2025. Bench- marking ILM-based relevance judgment methods. <i>arXiv preprint arXiv:2504.12558</i> .	Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. <i>J. Mach. Learn. Res.</i> , 12:2493–2537.	886 887 888 889
825	Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Ben- gio. 2015. Neural machine translation by jointly learning to align and translate. In <i>3rd International Conference on Learning Representations, ICLR 2015</i> .	Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2021a. Overview of the TREC 2020 deep learning track. <i>CoRR</i> , abs/2102.07662.	890 891 892
826	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 ma- chine reading comprehension dataset. <i>arXiv preprint arXiv:1611.09268</i> .	Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Jimmy Lin. 2021b. Overview of the TREC 2021 deep learning track.	893 894 895
827	Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. <i>CoRR</i> , abs/2004.05150.	Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, Jimmy Lin, Ellen M. Voorhees, and Ian Soboroff. 2022. <a href="#">Overview of the TREC 2022 deep learning track</a> . In <i>Proceedings of the Thirty-First Text REtrieval Conference, TREC 2022, online, November 15-19, 2022</i> , volume 500-338 of <i>NIST Special Publication</i> . National Institute of Standards and Tech- nology (NIST).	896 897 898 899 900 901 902 903
828	Adam Berger and John Lafferty. 1999. Information re- trieval as statistical translation. In <i>Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval</i> , pages 222–229.	Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. 2020. Overview of the TREC 2019 deep learning track. <i>CoRR</i> , abs/2003.07820.	904 905 906 907
829	Leonid Boytsov and Zico Kolter. 2021. Exploring clas- sic and neural lexical translation models for infor- mation retrieval: Interpretability, effectiveness, and efficiency benefits. In <i>ECIR (1)</i> , volume 12656 of <i>Lecture Notes in Computer Science</i> , pages 63–78. Springer.	Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Hos- sein A. Rahmani, Daniel Campos, Jimmy Lin, Ellen M. Voorhees, and Ian Soboroff. 2023. Overview of the TREC 2023 deep learning track. In <i>TREC</i> , volume 500-xxx of <i>NIST Special Publica- tion</i> . National Institute of Standards and Technology (NIST).	908 909 910 911 912 913 914
830	Peter F. Brown, Stephen Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estima- tion. <i>Computational Linguistics</i> , 19(2):263–311.	Zhuyun Dai and Jamie Callan. 2019. Deeper text un- derstanding for IR with contextual neural language modeling. In <i>SIGIR</i> , pages 985–988. ACM.	915 916 917
831	Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: pre- training text encoders as discriminators rather than generators. In <i>ICLR</i> . OpenReview.net.		

<sup>6</sup>[https://github.com/microsoft/Llama-2-Onnx/  
blob/main/MODEL-CARD-META-LLAMA-2.md](https://github.com/microsoft/Llama-2-Onnx/blob/main/MODEL-CARD-META-LLAMA-2.md)

918	Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. In <i>NeurIPS</i> .	974
919		975
920		
921		
922	Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. pages 4171–4186.	976
923		977
924		978
925		979
926	Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE v2: Sparse lexical and expansion model for information retrieval. <i>CoRR</i> , abs/2109.10086.	980
927		
928		
929		
930	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 <i>Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining</i> , WSDM ’22, page 298–306, New York, NY, USA. Association for Computing Machinery.	981
931		982
932		983
933		984
934		985
935		
936		
937		
938	Luyu Gao and Jamie Callan. 2022. Long document re-ranking with modular re-ranker. In <i>Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval</i> , SIGIR ’22, page 2371–2376, New York, NY, USA. Association for Computing Machinery.	986
939		987
940		988
941		989
942		990
943		
944	Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. 2016. A deep relevance matching model for ad-hoc retrieval. In <i>CIKM</i> , pages 55–64. ACM.	991
945		992
946		993
947	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. <i>Information Processing &amp; Management</i> , page 102067.	994
948		
949		
950		
951		
952	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 <i>Findings of the Association for Computational Linguistics: NAACL 2022</i> , pages 724–736, Seattle, United States. Association for Computational Linguistics.	1000
953		1001
954		1002
955		1003
956		1004
957		1005
958		
959	Michael Günther, Jackmin Ong, Isabelle Mohr, Alaedine Abdessalem, Tanguy Abel, Mohammad Kalim Akram, Susana Guzman, Georgios Mastropas, Saba Sturua, Bo Wang, Maximilian Werk, Nan Wang, and Han Xiao. 2023. Jina embeddings 2: 8192-token general-purpose text embeddings for long documents.	1006
960		1007
961		1008
962		
963		
964		
965		
966	Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing.	1009
967		1010
968		1011
969		1012
970	Sebastian Hofstätter, Aldo Lipani, Sophia Althammer, Markus Zlabinger, and Allan Hanbury. 2021a. Mitigating the position bias of transformer models in passage re-ranking. In <i>ECIR (1)</i> , volume 12656 of <i>Lecture Notes in Computer Science</i> , pages 238–253. Springer.	1013
971		1014
972		
973		
974	Sebastian Hofstätter, Bhaskar Mitra, Hamed Zamani, Nick Craswell, and Allan Hanbury. 2021b. Intra-document cascading: Learning to select passages for neural document ranking. In <i>SIGIR</i> , pages 1349–1358. ACM.	1015
975		1016
976		1017
977		1018
978		
979		
980		
981	Sebastian Hofstätter, Markus Zlabinger, and Allan Hanbury. 2020a. Interpretable & time-budget-constrained contextualization for re-ranking. In <i>ECAI</i> , volume 325 of <i>Frontiers in Artificial Intelligence and Applications</i> , pages 513–520. IOS Press.	1019
982		1020
983		1021
984		1022
985		1023
986	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 <i>CIKM</i> , pages 3031–3038. ACM.	1024
987		1025
988		1026
989		1027
990		
991	Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. 2018. Co-pacrr: A context-aware neural IR model for ad-hoc retrieval. In <i>WSDM</i> , pages 279–287. ACM.	1028
992		1029
993		1030
994		
995	Samuel Huston and W Bruce Croft. 2014. A comparison of retrieval models using term dependencies. In <i>Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management</i> , pages 111–120.	1031
996		1032
997		1033
998		1034
999		1035
1000	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 <i>TREC</i> , volume 500–261 of <i>NIST Special Publication</i> . National Institute of Standards and Technology (NIST).	1036
1001		1037
1002		1038
1003		1039
1004		1040
1005		
1006	Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. <i>ACM Trans. Inf. Syst.</i> , 20(4):422–446.	1041
1007		1042
1008		1043
1009	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 <i>EMNLP (1)</i> , pages 6769–6781. Association for Computational Linguistics.	1044
1010		1045
1011		1046
1012		1047
1013		1048
1014		
1015	Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over BERT. In <i>SIGIR</i> , pages 39–48. ACM.	1049
1016		1050
1017		1051
1018		
1019	Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: a benchmark for question answering research. <i>Trans. Assoc. Comput. Linguistics</i> , 7:452–466.	1052
1020		1053
1021		1054
1022		1055
1023		1056
1024		1057
1025		1058
1026		1059
1027		

1028	Canjia Li, Andrew Yates, Sean MacAvaney, Ben He, and Yingfei Sun. 2024. PARADE: passage representation aggregation for document reranking. <i>ACM Trans. Inf. Syst.</i> , 42(2):36:1–36:26.	1082
1029		1083
1030		1084
1031		
1032	Jimmy Lin. 2019. The neural hype and comparisons against weak baselines. In <i>ACM SIGIR Forum</i> , volume 52, pages 40–51. ACM New York, NY, USA.	1085
1033		1086
1034		1087
1035	Jimmy Lin, Rodrigo Nogueira, and Andrew Yates. 2021. <i>Pretrained Transformers for Text Ranking: BERT and Beyond</i> . Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.	1088
1036		1089
1037		
1038		
1039	Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. <i>Trans. Assoc. Comput. Linguistics</i> , 12:157–173.	1090
1040		1091
1041		1092
1042		1093
1043		1094
1044	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. <i>CoRR</i> , abs/1907.11692.	1095
1045		1096
1046		1097
1047		
1048		
1049	Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. <i>arXiv preprint arXiv:1711.05101</i> .	1098
1050		1099
1051		1100
1052	Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. 2023. Fine-tuning llama for multi-stage text retrieval. <i>CoRR</i> , abs/2310.08319.	1101
1053		1102
1054		1103
1055	Sean MacAvaney, Sergey Feldman, Nazli Goharian, Doug Downey, and Arman Cohan. 2022. ABNIRML: analyzing the behavior of neural IR models. <i>Trans. Assoc. Comput. Linguistics</i> , 10:224–239.	1104
1056		
1057		
1058		
1059	Sean MacAvaney, Andrew Yates, Arman Cohan, and Nazli Goharian. 2019. CEDR: contextualized embeddings for document ranking. In <i>SIGIR</i> , pages 1101–1104. ACM.	1105
1060		1106
1061		1107
1062		1108
1063	Sean MacAvaney, Andrew Yates, Sergey Feldman, Doug Downey, Arman Cohan, and Nazli Goharian. 2021. Simplified data wrangling with ir-datasets. In <i>SIGIR</i> .	1109
1064		
1065		
1066		
1067	Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In <i>NIPS</i> , pages 3111–3119.	1110
1068		1111
1069		1112
1070		1113
1071	Iurii Mokrii, Leonid Boytsov, and Pavel Braslavski. 2021. <i>A Systematic Evaluation of Transfer Learning and Pseudo-Labeling with BERT-Based Ranking Models</i> , page 2081–2085. Association for Computing Machinery, New York, NY, USA.	1114
1072		
1073		
1074		
1075		
1076	Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. 2020. On the stability of fine-tuning BERT: misconceptions, explanations, and strong baselines. <i>CoRR</i> , abs/2006.04884.	1115
1077		1116
1078		1117
1079		1118
1080	Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with BERT. <i>CoRR</i> , abs/1901.04085.	1119
1081		1120
1082	Rodrigo Nogueira and Jimmy Lin. 2019. From doc2query to docTTTTTquery. <i>MS MARCO passage retrieval task publication</i> .	1121
1083		1122
1084		1123
1085	Jimmy Lin, Wei Yang, Jimmy Lin, and Kyunghyun Cho. 2019. Document expansion by query prediction. <i>CoRR</i> , abs/1904.08375.	1124
1086		
1087		
1088	OpenAI. 2023. <i>GPT-4 technical report</i> . <i>CoRR</i> , abs/2303.08774.	1125
1089		
1090	Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In <i>NeurIPS</i> .	1126
1091		1127
1092		1128
1093		1129
1094		1130
1095		1131
1096		
1097		
1098	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 <i>Advances in neural information processing systems</i> , pages 8026–8037.	1125
1099		1126
1100		1127
1101		1128
1102		1129
1103		1130
1104		1131
1105	Gustavo Penha, Arthur Câmara, and Claudia Hauff. 2022. Evaluating the robustness of retrieval pipelines with query variation generators. In <i>ECIR (1)</i> , volume 13185 of <i>Lecture Notes in Computer Science</i> , pages 397–412. Springer.	1125
1106		1126
1107		1127
1108		1128
1109		1129
1110	Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In <i>Proceedings of NAACL-HLT</i> , pages 2227–2237.	1125
1111		1126
1112		1127
1113		1128
1114		1129
1115	Jacob Portes, Alexander R Trott, Sam Havens, DANIEL KING, Abhinav Venigalla, Moin Nadeem, Nikhil Sardana, Daya Khudia, and Jonathan Frankle. 2023. <i>MosaicBERT: A bidirectional encoder optimized for fast pretraining</i> . In <i>Thirty-seventh Conference on Neural Information Processing Systems</i> .	1125
1116		1126
1117		1127
1118		1128
1119		1129
1120		1130
1121	Ofir Press, Noah Smith, and Mike Lewis. 2022. <i>Train short, test long: Attention with linear biases enables input length extrapolation</i> . In <i>International Conference on Learning Representations</i> .	1125
1122		1126
1123		1127
1124		1128
1125	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 <i>NAACL-HLT</i> , pages 5835–5847. Association for Computational Linguistics.	1125
1126		1126
1127		1127
1128		1128
1129		1129
1130		1130
1131		1131
1132	David Rau, Mostafa Dehghani, and Jaap Kamps. 2024. Revisiting bag of words document representations for efficient ranking with transformers. <i>ACM Trans. Inf. Syst.</i> , 42(5):114:1–114:27.	1125
1133		1126
1134		1127
1135		1128

1136	Stephen Robertson. 2004. Understanding inverse document frequency: on theoretical arguments for IDF. <i>Journal of Documentation</i> , 60(5):503–520.	1187
1137		1188
1138		1189
1139	Patrick Ruch, Imad Tbahriti, Julien Gobeill, and Alan R. Aronson. 2006. Argumentative feedback: A linguistically-motivated term expansion for information retrieval. In <i>ACL</i> . The Association for Computer Linguistics.	1190
1140		1191
1141		1192
1142		1193
1143		1194
1144	Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierrette 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. <i>ArXiv</i> , abs/1910.03771.	1195
1145		
1146		
1147	Alexander M Rush. 2018. The annotated transformer. In <i>Proceedings of workshop for NLP open source software (NLP-OSS)</i> , pages 52–60.	1196
1148		1197
1149		1198
1150		1199
1151	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. <i>CoRR</i> , abs/2402.07440.	1200
1152		1201
1153		1202
1154		1203
1155	Jianlin Su, Murtadha H. M. Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. <i>Neurocomputing</i> , 568:127063.	1204
1156		1205
1157		1206
1158	Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017. End-to-end neural ad-hoc ranking with kernel pooling. In <i>SIGIR</i> , pages 55–64. ACM.	1207
1159		1208
1160		1209
1161		1210
1162		1211
1163	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 <i>ICLR</i> . OpenReview.net.	1212
1164		1213
1165		1214
1166		1215
1167	Zhichao Xu. 2024. Rankmamba: Benchmarking mamba’s document ranking performance in the era of transformers. <i>arXiv preprint arXiv:2403.18276</i> .	1216
1168		1217
1169		1218
1170		1219
1171	Zeynep Akkalyoncu Yilmaz, Shengjin Wang, Wei Yang, Haotian Zhang, and Jimmy Lin. 2019. Applying BERT to document retrieval with birch. In <i>EMNLP/IJCNLP (3)</i> , pages 19–24. Association for Computational Linguistics.	1220
1172		1221
1173		1222
1174		1223
1175		1224
1176	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 <i>NeurIPS</i> .	1225
1177		1226
1178		1227
1179		1228
1180	George Zerveas, Navid Rekabsaz, Daniel Cohen, and Carsten Eickhoff. 2021. Coder: An efficient framework for improving retrieval through contextualized document embedding reranking. <i>ArXiv</i> , abs/2112.08766.	1229
1181		1230
1182	Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. 2024. Tinyllama: An open-source small language model. <i>CoRR</i> , abs/2401.02385.	1231
1183		1232
1184		1233
1185		1234
1186	Xinyu Zhang, Andrew Yates, and Jimmy Lin. 2021. Comparing score aggregation approaches for document retrieval with pretrained transformers. In <i>ECIR (2)</i> , volume 12657 of <i>Lecture Notes in Computer Science</i> , pages 150–163. Springer.	1235
1187		1236
1188		1237
1189		
1190		
1191		
1192		
1193		
1194		
1195		

1243 Dawei Zhu, Liang Wang, Nan Yang, Yifan Song, Wen-  
1244 hao Wu, Furu Wei, and Sujian Li. 2024. Longembed:  
1245 Extending embedding models for long context re-  
1246 trieval. pages 802–816.

1247 Lixin Zou, Shengqiang Zhang, Hengyi Cai, Dehong Ma,  
1248 Suqi Cheng, Shuaiqiang Wang, Daiting Shi, Zhicong  
1249 Cheng, and Dawei Yin. 2021. Pre-trained language  
1250 model based ranking in baidu search. In *KDD*, pages  
1251 4014–4022. ACM.

## A Experiments: Additional Information, Ablations, and Detailed Results

### A.1 Detailed Training and Evaluation Setup

#### A.1.1 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 § A.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.<sup>7</sup>

<sup>7</sup>[https://anonymous.4open.science/r/long\\_doc\\_rank\\_model\\_analysis\\_v2-78E9/](https://anonymous.4open.science/r/long_doc_rank_model_analysis_v2-78E9/)

#### A.1.2 Model Training

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 Far-Relevant 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 than 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 (Günther et al., 2023), 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 configuration files (in the on-line repository).<sup>8</sup> We used the AdamW optimizer (Loshchilov and Hutter, 2017) and a constant learning rate with a 20% linear

<sup>8</sup>[https://anonymous.4open.science/r/long\\_doc\\_rank\\_model\\_analysis\\_v2-78E9/](https://anonymous.4open.science/r/long_doc_rank_model_analysis_v2-78E9/)

Ranker	TREC DL	NQ	BEIR (without NQ)						
	(2019-2020)		Touche	COVID	NFC	DBP	SciFact	SciDocs	average
	NDCG@10	NDCG@10	NDCG@10						
BM25	0.519	0.325	0.336	0.677	0.326	0.339	0.655	0.160	0.417
Original MS MARCO training set									
MaxP (ELECTRA)	0.715	0.514	0.314	0.744	0.312	0.404	<b>0.659</b>	0.153	<b>0.477</b>
PARADE Attn (ELECTRA)	0.710	0.496	<b>0.356</b>	0.743	0.266	0.386	0.606	0.140	0.463
PARADE Transf-RAND-L2 (ELECTRA)	0.703	0.474	0.293	0.728	0.297	0.380	0.658	0.154	0.461
CEDR-KNRM	0.599	0.318	0.242	0.710	0.268	0.305	0.478	0.127	0.381
De-biased MS MARCO (Hofstätter et al., 2021a)									
MaxP (ELECTRA)	<b>0.716</b> (+0.1%)	<b>0.516</b> (+0.6%)	0.309	0.742	0.296 <sup>a</sup>	<b>0.408</b>	0.646	0.150	0.473 (-0.8%)
PARADE Attn (ELECTRA)	0.675 <sup>a</sup> (-4.9%)	0.427 <sup>a</sup> (-14.0%)	0.335	0.742	0.226 <sup>a</sup>	0.353 <sup>a</sup>	0.522 <sup>a</sup>	0.122 <sup>a</sup>	0.425 (-8.2%)
PARADE Transf-RAND-L2 (ELECTRA)	0.706 (+0.4%)	0.485 <sup>a</sup> (+2.5%)	0.299	<b>0.753<sup>a</sup></b>	0.288 <sup>a</sup>	0.393 <sup>a</sup>	0.643	0.153	0.465 (+0.9%)
CEDR-KNRM	0.604 (+0.7%)	0.353 <sup>a</sup> (+11.0%)	0.231	0.683 <sup>a</sup>	0.265	0.340 <sup>a</sup>	0.419 <sup>a</sup>	0.122 <sup>a</sup>	0.377 (-1.0%)

We train each model using the original and the de-biased MS MARCO. For each model trained on the de-biased dataset, we compute a gain (or loss) compared to the same model trained on the original training set. Statistical significance of the differences are denoted using the superscript superscript **a** (except TREC DL and TREC COVID that have  $p$ -value threshold of 0.05, the  $p$ -value threshold is 0.01). Best numbers are in **bold**: Results are averaged over three seeds.

Table 3: De-biasing impact on short-document collection performance. The table shows effectiveness of (selected) rankers trained on original and de-biased MS MARCO (and tested on short-document collections).

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. For training using de-biased MS MARCO, we used only one epoch. 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 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.

### A.1.3 Miscellaneous Notes

To enable efficient training and evaluation of the large number of models, for Robust04 and original MS MARCO documents were truncated to have at most 1431 BERT tokens. In § A.2 (see Table 6) 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. However, for training on de-biased MS MARCO, the truncation threshold is much higher: 8109 tokens.

For *SplitP* approaches, queries were padded to 32 BERT tokens with padding being masked out during training (longer queries were truncated). For

1340  
1341  
1342  
1343  
1344  
1345  
1346  
1347  
1348  
1349  
1350  
1351  
1352  
1353  
1354  
1355  
1356  
1357  
1358  
1359  
1360  
1361  
1362  
1363  
1364  
1365  
1366  
1367  
1368  
1369  
1370  
1371  
1372

1373  
1374  
1375  
1376  
1377  
1378  
1379  
1380  
1381  
1382  
1383  
1384  
1385  
1386  
1387  
1388  
1389  
1390  
1391  
1392  
1393  
1394  
1395  
1396  
1397  
1398  
1399  
1400  
1401  
1402  
1403  
1404

Retriever / Ranker	zero-shot transferred	fine-tuned
Random shuffling of top-100	0.052	0.052
<b>Retriever (BM25)</b>	0.207	0.207
FirstP (BERT)	0.016 <sup>b</sup>	0.090 <sup>b</sup>
FirstP (Longformer)	0.017 <sup>b</sup>	0.091 <sup>b</sup>
FirstP (ELECTRA)	0.019 <sup>b</sup>	0.089 <sup>b</sup>
FirstP (Big-Bird)	0.021 <sup>b</sup>	0.089 <sup>b</sup>
FirstP (JINA)	0.018 <sup>b</sup>	0.088 <sup>b</sup>
FirstP (MOSAIC)	0.018 <sup>b</sup>	0.089 <sup>b</sup>
FirstP (TinyLLAMA)	0.020 <sup>b</sup>	0.079 <sup>b</sup>
FirstP (E5-4K)	0.015 <sup>ab</sup>	–
AvgP	0.154 <sup>ab</sup> (−48.1%)	0.365 <sup>ab</sup> (+11.4%)
MaxP	0.297 <sup>b</sup>	0.328 <sup>b</sup>
MaxP (ELECTRA)	0.328 <sup>b</sup>	0.349 <sup>b</sup>
MaxP (DEBERTA)	0.298 <sup>b</sup>	0.332 <sup>b</sup>
SumP	0.211 <sup>ab</sup> (−28.8%)	0.327 <sup>b</sup> (−0.4%)
CEDR-DRMM	0.157 <sup>ab</sup> (−47.3%)	0.372 <sup>ab</sup> (+13.3%)
CEDR-KNRM	0.055 <sup>ab</sup> (−81.5%)	0.382 <sup>a</sup> (+16.4%)
CEDR-PACRR	0.209 <sup>ab</sup> (−29.6%)	0.393 <sup>a</sup> (+19.9%)
Neural Model1	0.085 <sup>ab</sup> (−71.3%)	0.396 <sup>a</sup> (+20.6%)
PARADE Attn	0.300 <sup>b</sup> (+1.0%)	0.337 <sup>b</sup> (+2.8%)
PARADE Attn (ELECTRA)	<b>0.338<sup>b</sup></b> (+3.3%)	0.354 <sup>b</sup> (+1.6%)
PARADE Attn (DEBERTA)	0.307 <sup>b</sup> (+3.2%)	0.343 <sup>b</sup> (+3.4%)
PARADE Avg	0.274 <sup>ab</sup> (−7.6%)	0.322 <sup>b</sup> (−1.7%)
PARADE Max	0.291 <sup>b</sup> (−2.1%)	0.330 <sup>b</sup> (+0.6%)
PARADE Transf-RAND-L2	0.243 <sup>a</sup> (−18.2%)	0.419 <sup>ab</sup> (+27.7%)
P. Transf-RAND-L2 (ELECTRA)	0.229 <sup>a</sup> (−30.2%)	<b>0.432<sup>ab</sup></b> (+23.8%)
PARADE Transf-PRETR-L6	0.267 <sup>ab</sup> (−10.0%)	0.413 <sup>a</sup> (+26.0%)
P. Transf-PRETR-LATEIR-L6	0.244 <sup>a</sup> (−18.0%)	0.358 <sup>ab</sup> (+9.2%)
LongP (Longformer)	0.233 <sup>a</sup> (−21.7%)	0.399 <sup>a</sup> (+21.7%)
LongP (Big-Bird)	0.126 <sup>ab</sup> (−57.4%)	0.401 <sup>a</sup> (+22.1%)
LongP (JINA)	0.069 <sup>ab</sup> (−76.9%)	0.372 <sup>ab</sup> (+13.4%)
LongP (MOSAIC)	0.120 <sup>ab</sup> (−59.6%)	0.397 <sup>a</sup> (+21.2%)
LongP (TinyLLAMA)	0.078 <sup>ab</sup> (−73.6%)	0.397 <sup>a</sup> (+21.1%)
LongP (E5-4K)	0.057 <sup>ab</sup> (−80.7%)	N/A (zero-shot only)
LongP RankGPT (GPT-4o-mini)	0.043 <sup>b</sup>	N/A (zero-shot only)
LongP RankGPT (Claude-3-haiku)	0.051 <sup>b</sup>	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). 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  $< 0.01$ ).

Table 4: Comparison between long-document models and respective *FirstP* (truncation) baselines. Results on MS MARCO FarRelevant.

*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.

For bias-mitigation experiments, for several reasons, we used only a subset of models. First, we had to exclude all *LongP* models since none of them supported a context longer than 8192 tokens. In contrast, in this experiment we trained our chunk-and-aggregate tokens up to the length of 8109 and then extrapolated to rank documents up to 32768 tokens long. Second, we chose representative models with vastly different generalization properties. *MaxP* and PARADE Attention models performed well on MS MARCO FarRelevant in the zero-shot setting, but did not benefit much from in-domain fine-tuning. PARADE Transformer MRR dropped from 0.433 to 0.229 in the zero-shot setting, but increased up to 0.432 after in-domain fine-tuning. CEDR-KNRM also benefited a lot from fine-tuning on MS MARCO FarRelevant, but its zero-shot performance was at the level of random-baseline.

## A.2 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.

Model	MS MARCO dev	2019		TREC DL 2020		title	Robust04 description
		MRR		NDCG@10	2021		
BM25	0.274	0.548	0.538	0.549	0.428	0.402	
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	-	<b>0.566</b>	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 <sup>a</sup> (+2.6%)	0.544 <sup>a</sup> (+3.3%)	
PARADE Attn	0.416 <sup>a</sup> (+5.5%)	0.647 (+2.5%)	0.626 <sup>a</sup> (+4.6%)	0.677 (+2.5%)	0.503 <sup>a</sup> (+5.7%)	0.556 <sup>a</sup> (+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)	<b>0.431<sup>a</sup> (+3.3%)</b>	0.675 <sup>a</sup> (+3.5%)	0.653 (+1.8%)	0.705 (+2.8%)	0.523 <sup>a</sup> (+6.4%)	0.581 <sup>a</sup> (+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 <sup>a</sup> (+1.6%)	<b>0.685</b> (+1.4%)	<b>0.659<sup>a</sup></b> (+4.8%)	<b>0.713</b> (+1.4%)	0.549 <sup>a</sup> (+2.9%)	<b>0.615<sup>a</sup></b> (+3.2%)	
FirstP (Longformer)	0.404	0.657	0.616	0.654	0.483	0.540	
LongP (Longformer)	0.412 <sup>a</sup> (+1.9%)	0.676 <sup>a</sup> (+2.9%)	0.628 (+2.0%)	0.693 <sup>a</sup> (+6.0%)	0.500 <sup>a</sup> (+3.6%)	0.568 <sup>a</sup> (+5.1%)	
FirstP (Big-Bird)	0.408	0.663	0.620	0.679	0.507	0.560	
LongP (Big-Bird)	0.397 <sup>a</sup> (-2.9%)	0.655 (-1.1%)	0.618 (-0.3%)	0.675 (-0.5%)	0.452 <sup>a</sup> (-10.9%)	0.477 <sup>a</sup> (-14.9%)	
FirstP (JINA)	0.422	0.658	0.618	0.679	0.488	0.532	
LongP (JINA)	0.416 <sup>a</sup> (-1.5%)	0.670 <sup>a</sup> (+1.8%)	0.632 (+2.1%)	0.689 (+1.4%)	0.503 <sup>a</sup> (+2.9%)	0.558 <sup>a</sup> (+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 <sup>a</sup> (+3.7%)	0.694 <sup>a</sup> (+4.9%)	0.456 (+0.6%)	0.570 <sup>a</sup> (+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.

Table 5: Comparison between long-document models and respective *FirstP* (truncation) baselines as well as to *prior art*. Results for MS MARCO, TREC DL, and Robust04.

The results are presented in Table 6: 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.

### A.3 Backbone Selection for *SplitP* Models and Prior Art Comparison

#### A.3.1 Choice of a Backbone

To understand if using BERT-base as back-

bone 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 1), 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 5 and 1 confirm overall superiority of both ELECTRA and DEBERTA over BERT-base. In that, DEBERTA models are nearly always more effective compared to ELECTRA

1455	The results are presented in Table 6: We can	1475
1456	see that—depending on the dataset—three or four	1476
1457	input chunks are optimal. However, the additional	1477
1458	gains over the <i>FirstP</i> baseline are at most 0.6%	1478
1459	when averaged over all test sets.	1479
1460	Gao and Callan 2022 carried out a similar abla-	1480
1461	tion using ClueWeb09: Increasing the number of	1481
1462	input chunks from three to six lead to only about	1482
1463	2.3% relative improvement in NDCG@20. How-	1483
1464	ever, even this modest gain could have been slightly	1484
1465	inflated due to model not being trained <i>directly</i> on	1485
1466	shorter inputs. Indeed, truncation of an input for	1486
1467	a six-chunk model to one chunk is potentially less	1487
1468	effective than training and evaluating the model	1488
1469	using one-chunk data.	1489
1470	<b>A.3 Backbone Selection for <i>SplitP</i> Models and</b>	1490
1471	<b>Prior Art Comparison</b>	1491
1472	<b>A.3.1 Choice of a Backbone</b>	1492
1473	To understand if using BERT-base as back-	1493
1474	bone model for various <i>SplitP</i> (i.e., chunk-and-	1494
1475	aggregate) approaches diminished models’ ability	1495
1476	to process and understand long contexts, we carried	
1477	out a focused comparison of several backbone mod-	
1478	els, including ELECTRA (Clark et al., 2020) and	
1479	DEBERTA (He et al., 2021). To this end, we used	
1480	two methods: PARADE (Li et al., 2024) Attention	
1481	and <i>MaxP</i> . PARADE Attention model achieved	
1482	the largest average gain over <i>FirstP</i> in our main	
1483	experiments (see Table 1), whereas <i>MaxP</i> models	
1484	were extensively benchmarked in the past (Li et al.,	
1485	2024; Dai and Callan, 2019; Zhang et al., 2021).	
1486	Although prior work found ELECTRA to be a bet-	
1487	ter backbone model in terms of <i>absolute</i> accuracy	
1488	(Li et al., 2024; Zhang et al., 2021), we found no	
1489	systematic evaluation of the relationship between a	
1490	backbone model and achievable <i>FirstP</i> gains.	
1491	Results in Tables 5 and 1 confirm overall su-	
1492	periority of both ELECTRA and DEBERTA over	
1493	BERT-base. In that, DEBERTA models are nearly	
1494	always more effective compared to ELECTRA	

Retriever / Ranker	MS MARCO dev	TREC DL (2019-2021)	title	Robust04 description	Avg. gain Over FirstP
	MRR	NDCG@10		NDCG@20	
BM25	0.274	0.545	0.428	0.402	-
Retriever (if different from BM25)	0.312	0.629	-	-	-
PARADE Attn (1 chunk)	0.401	0.637	0.476	0.527	-
PARADE Attn (2 chunks)	0.408 <sup>a</sup> (+1.8%)	0.653 <sup>a</sup> (+2.7%)	0.499 <sup>a</sup> (+4.9%)	0.544 <sup>a</sup> (+3.3%)	+3.2%
PARADE Attn (3 chunks)	0.406 <sup>a</sup> (+1.3%)	0.648 <sup>a</sup> (+1.7%)	<b>0.505<sup>a</sup></b> (+6.1%)	0.557 <sup>a</sup> (+5.7%)	+3.7%
PARADE Attn (4 chunks)	<b>0.412<sup>a</sup></b> (+2.9%)	<b>0.654<sup>a</sup></b> (+2.7%)	0.504 <sup>a</sup> (+5.9%)	<b>0.558<sup>a</sup></b> (+5.9%)	<b>+4.3%</b>
PARADE Attn (5 chunks)	0.409 <sup>a</sup> (+2.0%)	0.652 <sup>a</sup> (+2.4%)	0.502 <sup>a</sup> (+5.6%)	0.556 <sup>a</sup> (+5.5%)	+3.9%
PARADE Attn (6 chunks)	0.411 <sup>a</sup> (+2.4%)	0.653 <sup>a</sup> (+2.6%)	0.504 <sup>a</sup> (+5.9%)	0.554 <sup>a</sup> (+5.2%)	+4.0%

Table 6: Effectiveness of the PARADE Attention model for different input truncation thresholds. **Results for MS MARCO, TREC DL, and Robust04.**

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 1). We conclude that to measure capabilities of chunk-and-aggregate model to understand and incorporate long context, it appears to be *beneficial* to use BERT-base instead of ELECTRA or DEBERTA.

Finally, we would like to note that on standard benchmarks Big-Bird’s (Zaheer et al., 2020) *FirstP* version *always* outperforms its *LongP* version (sometimes by as much as 10-15%), which seems to be puzzling. We noticed, however, that for shorter inputs, the model turns off sparse attention and prints the respective warning. Thus, we hypothesize that it is the use of sparse attention that causes this degradation. In contrast, the sparse attention implementation of the Longformer (Beltagy et al., 2020) does not exhibit such a degradation (although with Longformer, not all attention is sparse: query-to-document attention is full). Despite Big-Bird underperforms on standard benchmarks, it still does well on MS MARCO FarRelevant after fine-tuning (see Table 4).

#### A.4 Efficiency Evaluation

From the efficiency-effectiveness plot in Fig. 3, 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

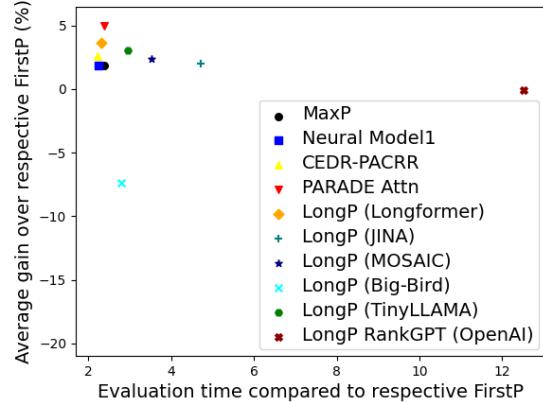


Figure 3: **Efficiency of long-document models vs respective (truncation) *FirstP* baselines.** The figure shows an average relative gain (in %) vs. relative increase in run-time compared to *respective FirstP* baselines on MS MARCO, TREC DL 2019-2021, and Robust04 (for a representative subset of models). Except LongP RankGPT, *LongP* models truncate documents to be at most 1431 tokens. There is no truncation for RankGPT.

Model	MS MARCO dev		TREC DL 2019-2021	
	MRR	NDCG@10	P@10	MAP
<b>BM25</b>	<b>0.274</b>	<b>0.545</b>	<b>0.636</b>	<b>0.282</b>
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)	0.380	0.641	0.722	0.317
FirstP RankGPT (OpenAI)	—	<b>0.708</b>	<b>0.790</b>	<b>0.352</b>
FirstP RankGPT (Anthropic)	—	0.703	0.776	0.347
AvgP	0.389 (-1.3%)	0.642 (+1.5%)	0.717 (+0.7%)	0.317 <sup>a</sup> (+2.0%)
MaxP	0.392 (-0.4%)	0.644 <sup>a</sup> (+1.9%)	0.723 (+1.5%)	0.322 <sup>a</sup> (+3.7%)
MaxP (ELECTRA)	0.414 (-0.6%)	0.659 (-0.5%)	0.745 (+1.5%)	0.326 (+2.1%)
MaxP (DEBERTA)	0.402 <sup>a</sup> (-3.2%)	0.671 (-0.1%)	0.746 (+0.7%)	0.335 <sup>a</sup> (+2.5%)
SumP	0.390 (-1.0%)	0.639 (+1.0%)	0.715 (+0.4%)	0.319 <sup>a</sup> (+2.6%)
CEDR-DRMM	0.385 <sup>a</sup> (-2.3%)	0.629 (-0.5%)	0.708 (-0.5%)	0.313 (+0.6%)
CEDR-KNRM	0.379 <sup>a</sup> (-3.8%)	0.630 (-0.3%)	0.711 (-0.1%)	0.313 (+0.8%)
CEDR-PACRR	0.395 (+0.3%)	0.643 <sup>a</sup> (+1.6%)	0.719 (+0.9%)	0.320 <sup>a</sup> (+2.9%)
Neural Model1	0.398 (+0.9%)	0.650 <sup>a</sup> (+2.8%)	0.723 <sup>a</sup> (+1.5%)	0.323 <sup>a</sup> (+3.9%)
PARADE Attn	0.416 <sup>a</sup> (+5.5%)	0.652 <sup>a</sup> (+3.1%)	0.728 <sup>a</sup> (+2.2%)	0.324 <sup>a</sup> (+4.2%)
PARADE Attn (ELECTRA)	0.431 <sup>a</sup> (+3.3%)	0.680 <sup>a</sup> (+2.7%)	0.763 <sup>a</sup> (+3.9%)	0.335 <sup>a</sup> (+4.9%)
PARADE Attn (DEBERTA)	0.422 <sup>a</sup> (+1.6%)	0.688 <sup>a</sup> (+2.4%)	0.763 <sup>a</sup> (+3.0%)	0.339 <sup>a</sup> (+3.9%)
PARADE Avg	0.392 (-0.6%)	0.646 <sup>a</sup> (+2.1%)	0.715 (+0.4%)	0.317 <sup>a</sup> (+2.1%)
PARADE Max	0.405 <sup>a</sup> (+2.7%)	0.655 <sup>a</sup> (+3.5%)	0.733 <sup>a</sup> (+2.9%)	0.324 <sup>a</sup> (+4.1%)
PARADE Transf-RAND-L2	0.419 <sup>a</sup> (+6.3%)	0.655 <sup>a</sup> (+3.6%)	0.734 <sup>a</sup> (+3.1%)	0.326 <sup>a</sup> (+5.0%)
PARADE Transf-RAND-L2 (ELECTRA)	<b>0.433<sup>a</sup> (+3.9%)</b>	0.670 (+1.2%)	0.747 (+1.8%)	0.327 (+2.2%)
PARADE Transf-PRETR-L6	0.402 <sup>a</sup> (+1.9%)	0.643 (+1.6%)	0.717 (+0.8%)	0.322 <sup>a</sup> (+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 <sup>a</sup> (+1.9%)	0.668 <sup>a</sup> (+3.9%)	0.752 <sup>a</sup> (+4.1%)	0.333 <sup>a</sup> (+5.1%)
LongP (Big-Bird)	0.397 <sup>a</sup> (-2.9%)	0.651 (-0.7%)	0.726 (-0.2%)	0.322 (+0.3%)
LongP (JINA)	0.416 <sup>a</sup> (-1.5%)	0.665 <sup>a</sup> (+1.7%)	0.742 <sup>a</sup> (+2.0%)	0.328 <sup>a</sup> (+2.4%)
LongP (MOSAIC)	0.421 (-0.4%)	0.664 <sup>a</sup> (+3.3%)	0.740 <sup>a</sup> (+1.9%)	0.327 <sup>a</sup> (+3.7%)
LongP (TinyLLAMA)	0.402 <sup>a</sup> (+1.7%)	0.608 (-1.1%)	0.692 (+0.0%)	0.306 (+1.6%)
LongP (E5)	0.353 <sup>a</sup> (-7.1%)	0.649 (+1.3%)	0.724 (+0.3%)	0.323 (+1.8%)
LongP RankGPT (OpenAI)	—	0.706 (-0.3%)	0.783 (-1.0%)	0.350 (-0.7%)
LongP RankGPT (Anthropic)	—	0.707 (+0.5%)	0.780 (+0.5%)	0.348 (+0.4%)

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 7: Comparison between long-document models and respective FirstP (truncation) baselines. Results for MS MARCO and TREC DL.

Model	NDCG@20	P@20	MAP	NDCG@20	P@20	MAP
<b>Retriever (BM25)</b>	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
FirstP RankGPT (OpenAI)	—	—	—	0.562	0.456	0.280
FirstP RankGPT (Anthropic)	—	—	—	0.541	0.446	0.268
AvgP	0.478 (+0.5%)	0.411 (+1.6%)	0.292 <sup>a</sup> (+5.4%)	0.531 (+0.9%)	0.451 (+1.0%)	0.324 <sup>a</sup> (+6.7%)
MaxP	0.488 <sup>a</sup> (+2.6%)	0.425 <sup>a</sup> (+5.1%)	0.306 <sup>a</sup> (+10.6%)	0.544 <sup>a</sup> (+3.3%)	0.467 <sup>a</sup> (+4.5%)	0.338 <sup>a</sup> (+11.5%)
MaxP (ELECTRA)	0.502 (+2.0%)	0.441 <sup>a</sup> (+3.9%)	0.319 <sup>a</sup> (+8.3%)	0.563 (+2.1%)	0.483 <sup>a</sup> (+4.0%)	0.350 <sup>a</sup> (+9.3%)
MaxP (DEBERTA)	0.535 (+0.2%)	0.464 (+1.2%)	0.340 <sup>a</sup> (+6.7%)	0.609 <sup>a</sup> (+2.2%)	0.519 <sup>a</sup> (+3.2%)	0.378 <sup>a</sup> (+7.9%)
SumP	0.486 (+2.2%)	0.418 <sup>a</sup> (+3.4%)	0.305 <sup>a</sup> (+10.2%)	0.538 (+2.1%)	0.461 <sup>a</sup> (+3.1%)	0.337 <sup>a</sup> (+11.1%)
CEDR-DRMM	0.466 (-2.0%)	0.403 (-0.4%)	0.287 <sup>a</sup> (+3.8%)	0.533 (+1.3%)	0.458 <sup>a</sup> (+2.5%)	0.326 <sup>a</sup> (+7.6%)
CEDR-KNRM	0.483 (+1.7%)	0.413 (+1.9%)	0.291 <sup>a</sup> (+5.1%)	0.535 (+1.7%)	0.456 (+2.0%)	0.324 <sup>a</sup> (+6.8%)
CEDR-PACRR	0.496 <sup>a</sup> (+4.3%)	0.426 <sup>a</sup> (+5.3%)	0.307 <sup>a</sup> (+11.0%)	0.549 <sup>a</sup> (+4.2%)	0.466 <sup>a</sup> (+4.4%)	0.337 <sup>a</sup> (+11.2%)
Neural Model1	0.484 (+1.8%)	0.417 <sup>a</sup> (+3.1%)	0.298 <sup>a</sup> (+7.7%)	0.537 (+1.9%)	0.459 <sup>a</sup> (+2.6%)	0.330 <sup>a</sup> (+8.8%)
PARADE Attn	0.503 <sup>a</sup> (+5.7%)	0.433 <sup>a</sup> (+6.9%)	0.311 <sup>a</sup> (+12.4%)	0.556 <sup>a</sup> (+5.6%)	0.476 <sup>a</sup> (+6.5%)	0.344 <sup>a</sup> (+13.3%)
PARADE Attn (ELECTRA)	0.523 <sup>a</sup> (+6.4%)	0.456 <sup>a</sup> (+7.4%)	0.329 <sup>a</sup> (+11.7%)	0.581 <sup>a</sup> (+5.3%)	0.495 <sup>a</sup> (+6.5%)	0.358 <sup>a</sup> (+11.9%)
PARADE Attn (DEBERTA)	<b>0.549<sup>a</sup> (+2.9%)</b>	<b>0.475<sup>a</sup> (+3.6%)</b>	<b>0.346<sup>a</sup> (+8.7%)</b>	<b>0.615<sup>a</sup> (+3.2%)</b>	<b>0.522<sup>a</sup> (+3.8%)</b>	<b>0.383<sup>a</sup> (+9.4%)</b>
PARADE Avg	0.483 (+1.5%)	0.412 (+1.8%)	0.291 <sup>a</sup> (+5.2%)	0.534 (+1.5%)	0.457 (+2.4%)	0.318 <sup>a</sup> (+4.7%)
PARADE Max	0.489 <sup>a</sup> (+2.8%)	0.420 <sup>a</sup> (+3.8%)	0.306 <sup>a</sup> (+10.8%)	0.548 <sup>a</sup> (+4.0%)	0.470 <sup>a</sup> (+5.3%)	0.337 <sup>a</sup> (+11.0%)
PARADE Transf-RAND-L2	0.488 <sup>a</sup> (+2.8%)	0.423 <sup>a</sup> (+4.6%)	0.303 <sup>a</sup> (+9.7%)	0.548 <sup>a</sup> (+4.1%)	0.469 <sup>a</sup> (+5.0%)	0.338 <sup>a</sup> (+11.6%)
PAR. Transf-RAND-L2 (ELECTRA)	0.523 <sup>a</sup> (+6.3%)	0.454 <sup>a</sup> (+6.9%)	0.330 <sup>a</sup> (+12.2%)	0.574 <sup>a</sup> (+3.9%)	0.488 <sup>a</sup> (+5.0%)	0.354 <sup>a</sup> (+10.6%)
PARADE Transf-PRETR-L6	0.494 <sup>a</sup> (+4.0%)	0.426 <sup>a</sup> (+5.3%)	0.308 <sup>a</sup> (+11.5%)	0.554 <sup>a</sup> (+5.1%)	0.474 <sup>a</sup> (+6.1%)	0.346 <sup>a</sup> (+14.1%)
PAR. Transf-PRETR-LATEIR-L6	0.450 <sup>a</sup> (-5.2%)	0.389 <sup>a</sup> (-3.9%)	0.277 (+0.3%)	0.501 <sup>a</sup> (-4.9%)	0.423 <sup>a</sup> (-5.3%)	0.302 (-0.5%)
LongP (Longformer)	0.500 <sup>a</sup> (+3.6%)	0.435 <sup>a</sup> (+5.3%)	0.309 <sup>a</sup> (+11.5%)	0.568 <sup>a</sup> (+5.1%)	0.482 <sup>a</sup> (+6.1%)	0.347 <sup>a</sup> (+12.9%)
LongP (Big-Bird)	0.452 <sup>a</sup> (-10.9%)	0.389 <sup>a</sup> (-10.7%)	0.274 <sup>a</sup> (-8.8%)	0.477 <sup>a</sup> (-14.9%)	0.400 <sup>a</sup> (-15.5%)	0.279 <sup>a</sup> (-14.2%)
LongP (JINA)	0.503 <sup>a</sup> (+2.9%)	0.434 <sup>a</sup> (+3.1%)	0.309 <sup>a</sup> (+7.5%)	0.558 <sup>a</sup> (+4.9%)	0.473 <sup>a</sup> (+5.2%)	0.335 <sup>a</sup> (+9.7%)
LongP (MOSAIC)	0.456 (+0.6%)	0.393 (+0.8%)	0.280 <sup>a</sup> (+5.3%)	0.570 <sup>a</sup> (+6.0%)	0.484 <sup>a</sup> (+6.3%)	0.350 <sup>a</sup> (+13.0%)
LongP (TinyLLAMA)	0.452 <sup>a</sup> (+4.8%)	0.396 <sup>a</sup> (+6.9%)	0.267 <sup>a</sup> (+8.7%)	0.505 <sup>a</sup> (+6.7%)	0.428 <sup>a</sup> (+7.6%)	0.297 <sup>a</sup> (+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 <sup>a</sup> (+2.9%)
LongP RankGPT (OpenAI)	—	—	—	0.562 (+0.0%)	0.456 (-0.0%)	0.281 (+0.3%)
LongP RankGPT (Anthropic)	—	—	—	0.538 (-0.6%)	0.445 (-0.2%)	0.268 (-0.0%)

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.

Table 8: Comparison between long-document models and respective FirstP (truncation) baselines. Results for Robust04.

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.

#### A.4.1 Comparison to Prior Art

We also use Table 5 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.

### A.5 Additional Experimental Results

In this section we provide links for additional experimental results. In particular, we compute additional effectiveness metrics for MS MARCO, TREC DL, and Robust04. MS MARCO and TREC DL results are shown in Table 7. Robust04 datasets are presented and Table 8. Furthermore, we provide detailed results for MS MARCO FarRelevant in Table 4. Evaluation results of rankers trained on de-biased data and tested on short-document

collections can be found in Table 3.

## B Additional Dataset Details

### B.1 Summary Dataset Statistics

data set	# of documents	average # of BERT tokens per document
Long-document collections		
MS MARCO doc. v1	3.2M	1.4K
MS MARCO doc. v2	12M	2K
Robust04	0.5M	0.6K
MS MARCO FarRelevant	0.53M	1.1K
Needle (LongEmbed)	0.8K	variable-length
Passkey (LongEmbed)	0.8K	variable-length
Short-document collections		
MS MARCO pass. v1	8.8M	75
Natural Questions (BEIR)	2.7M	107

LONGEMBED subsets each have 16 subsets of documents whose lengths vary from (approximately) 256 to 32768 tokens.

Table 9: Document Statistics

	# of queries	avg. # of BERT tokens	avg. # of pos. judgements
MS MARCO doc. v1			
MS MARCO doc. train	352K	7	1
MS MARCO doc. 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
LongEmbed			
Needle	800	13.0	1
Passkey	800	9.7	1
Natural Questions (BEIR)			
all queries	3452	9.9	1.2
MS MARCO pass. v1			
TREC DL 2019	43	7	95.4
TREC DL 2020	54	7.2	66.8

Table 10: Query Statistics

Summary query and dataset statistics is given in Tables 9 and 10. Please, note that in the case of MS MARCO FarRelevant, we created about 500K training and 7K testing queries, but to reduce experimentation cost we ended up using only 50K and 1K, respectively.

1592           **B.2 MS MARCO FarRelevant Creation**  
1593           **Algorithm**

1594           The MS MARCO FarRelevant dataset was created  
1595           as follows: Assume that  $C_t$  is the number of tokens  
1596           in the passage:

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

1603           **B.3 Comparison of FarRelevant and**  
1604           **Synthetic Data from LongEmbed**

1605           Our synthetic data consists of two subsets Needle  
1606           and Passkey from LongEmbed collection ([Zhu et al., 2024](#)) and our newly created MS MARCO  
1607           FarRelevant dataset. The datasets can be seen  
1608           as a variant of a needle-in-the-haystack benchmark  
1609           ([Saad-Falcon et al., 2024; Zhu et al., 2024](#)) and they share a common limitation: the resulting documents  
1610           are constructed by combining pieces of text  
1611           in a purely mechanical fashion and do not represent natural documents. In this section, we describe  
1612           Needle and Passkey in more detail and compare them to our proposed collection MS MARCO Far-Relevant,  
1613           which—we believe—offers several practical advantages despite also being synthetic.

1614           The needle subset is created by taking a single  
1615           document (a Paul Graham essay on taste<sup>9</sup>), truncating it to generate 16 buckets of varying lengths  
1616           (from 256 to 32,768 tokens), and inserting a single answer-bearing sentence at a random location.  
1617           An example query-document pair can be found in  
1618           Figure 4. While this design ensures precise control over document length (doubling per bucket), it  
1619           introduces several problems:

9<https://www.paulgraham.com/goodtaste.html>

1. **Extremely low document diversity:** All examples are truncated variants of the same original text, which severely limits the variability in document style and content.

2. **Artificial signal separation:** The inserted answer sentence differs substantially from the background text, making it easy for models to identify it.

The Passkey subset is similar in structure and also uses length-bucketed documents, but instead of a single sentence, it inserts a three-sentence passkey definition into a synthetic background context (see Figure 5). However, the background is even less natural than Needle’s subset background, being composed of unrelated or nonsensical declarative statements (e.g., “The sky is blue. The sun is yellow. Here we go. There and back again.”). This leads to even greater distributional mismatch between signal and context.

Critically, both Needle and Passkey were designed primarily to test *answer extraction* or retrieval of small, highly localized answer-bearing spans. They are not well suited to studying retrieval or ranking of entire passages or documents, especially when relevance is more distributed or contextual.

Our MS MARCO FarRelevant (see Figure 6 for an example) is designed to be textually similar to MS MARCO Documents but with different positional biases for relevant passages. We believe it offers a more robust testbed for long-context *document ranking*. While it is also synthetic in construction, it avoids many of the pitfalls noted above. Each document is created by concatenating multiple passages, which are typically *meaningful* and *complete*, only one of which is relevant to the query. The remaining passages serve as distractors but are independently coherent. Although these documents do not exist in the wild, they are much more diverse in content and style than those in Needle or Passkey despite being typically much shorter. Furthermore, each individual passage is semantically complete and belongs to a real corpus, namely, MS MARCO Passages.

**B.4 Positional Bias Identification**

To assess positional bias, we used a combination of approximate string matching and LLM-based judging, which was recently shown to highly correlate with human judgments ([Upadhyay et al.,](#)

Aaron Swartz created a scraped feed of the essays page. November 2021 (This essay is derived from a talk at the Cambridge Union.) When I was a kid, I'd have said there wasn't. My father told me so. Some people like some things, and other people like other things, and who's to say who's right? It seemed so obvious that there was no such thing as good taste that it was only through indirect evidence that I realized my father was wrong.

**The novel "The Lord of the Rings" was written by Tolkien and published in the mid-20th century.** And that's what I'm going to give you here: a proof by reductio ad absurdum. If we start from the premise that there's no such thing as good taste, we end up with conclusions that are obviously false, and therefore the premise must be wrong. We'd better start by saying what good taste is. There's a narrow sense in which it refers to aesthetic judgements and a broader one in which it refers to preferences of any kind. The strongest proof would be to show that taste exists in the

Figure 4: A sample relevant document for the Needle collection. The query/question is: "Who wrote the novel "The Lord of the Rings" and when was it published?". The answer-bearing sentence is marked by bold font.

The grass is green. The sky is blue. The sun is yellow. Here we go. There and back again. The grass is green. The sky is blue. The sun is yellow. Here we go. There and back again. The grass is green. The sky is blue. The sun is yellow. Here we go. There and back again. The grass is green. The sky is blue. The sun is yellow. Here we go. There and back again. Joyce Jenkins's pass key is **44349**. Remember it. **44349** is the pass key for Joyce Jenkins. The grass is green. The sky is blue. The sun is yellow. Here we go. There and back again. The grass is green. The sky is blue. The sun is yellow. Here we go. There and back again. The grass is green. The sky is blue. The sun is yellow. Here we go. There and back again. The grass is green. The sky is blue. The sun is yellow. Here we go. There and back again. The grass is green. The sky is blue. The sun is yellow. Here we go.

Figure 5: A sample relevant document for the Passkey collection. The query/question is: "what is the passkey for Joyce Jenkins?". The answer-bearing sentence is marked by bold font.

Andhra Pradesh Airports make an easy access for tourists visiting the state. This huge state has many airports, which serves the needs of both tourists and residents commuting to different parts in its large expanse. However, Hyderabad Airport is the major as well as the only international airport of Andhra Pradesh. Hyderabad, a major IT hub of India, boasts of the sixth busiest airport in India. Keeping the rush of passengers in mind, the Government is planning to establish another airport in Hyderabad.

In contrast, traditional English Longbow shooters step into the bow, exerting force with both the bow arm and the string hand arm simultaneously, especially when using bows having draw weights from 100 lbs to over 175 lbs. Heavily stacked traditional bows (recurves, long bows, and the like) are released immediately upon reaching full draw at maximum weight, whereas compound bows reach their maximum weight around the last inch and a half, dropping holding weight significantly at full draw. The Oreo Biscuit was first developed and produced by the National Biscuit Company (today known as Nabisco) in 1912 at its Chelsea, Manhattan factory in the current-day Chelsea Market complex, located on Ninth Avenue between 15th and 16th Streets. Today, this same block of Ninth Avenue is known as Oreo Way..

It's possible to take a day trip to the Bahamas by ferry. In some cases, less than it would cost to fly. The high-speed Balearia Bahamas Express travels from Fort Lauderdale to the city of Freeport on Grand Bahama, one of many Bahamian islands with British roots. It's roughly the distance from Philadelphia to New York. Prepare for a long day. In some cases, less than it would cost to fly. The high-speed Balearia Bahamas Express travels from Fort Lauderdale to the city of Freeport on Grand Bahama, one of many Bahamian islands with British roots. It's roughly the distance from Philadelphia to New York.

I was 17 when I took Bactrim for a UTI, I was a month away from turning 18 and was given this because another antibiotic would have more side effects I was told. I took it for 5 days and became horribly sick from a nasty cold and was told to stop Bactrim and to take Z Pac instead for 5 days.

Cases When Medicare Does NOT Automatically Start for You Medicare will NOT automatically start when you turn 65 if you're not receiving Social Security Benefits or Railroad Retirement Benefits for at least 4 months prior to your 65th birthday. You wanted to know how you can feel on your belly that you're pregnant. And actually this a pretty hard thing to do. There are better ways to find out if you're pregnant or not. For example, if you ever miss a period, the best thing to do is to take a home pregnancy test, because that's the first sign of pregnancy. And if it's positive, ...here are better ways to find out if you're pregnant or not. For example, if you ever miss a period, the best thing to do is to take a home pregnancy test, because that's the first sign of pregnancy.

1 Whisk light soy sauce, dark soy sauce, red wine vinegar, chili oil, ginger, sugar, garlic, and green onion together in a bowl; pour into a sealable container, seal, and refrigerate 1 hour. See how to make a simple sweet-and-sour peach sauce. See how easy and delicious it is to make horseradish sauce from scratch.

Here you'll find a number of Kentucky facts including the state history at a glance; Kentucky state facts such as the location of the state capital, city populations, geography and natural resources; Information on Kentucky' government, symbols and traditions; and even a list of famous Kentuckians.

Confidence votes 193. Replacing a car window usually cost around \$300 give or take based upon where you live, what options your car glass needs and what type of car you drive. Additionally, you can save money if you can repair your car glass instead of completely replacing it. However, if the window is completely shattered this will not be an option.

**Flying time from Chicago, IL to Cairo, Egypt. The total flight duration from Chicago, IL to Cairo, Egypt is 12 hours, 47 minutes. This assumes an average flight speed for a commercial airliner of 500 mph, which is equivalent to 805 km/h or 434 knots. It also adds an extra 30 minutes for take-off and landing. Your exact time may vary depending on wind speeds.**

Figure 6: A sample relevant document for the MS MARCO FarRelevant collection. The query/question is: "how long is the flight from chicago to cairo". The answer-bearing passage is marked by bold font.

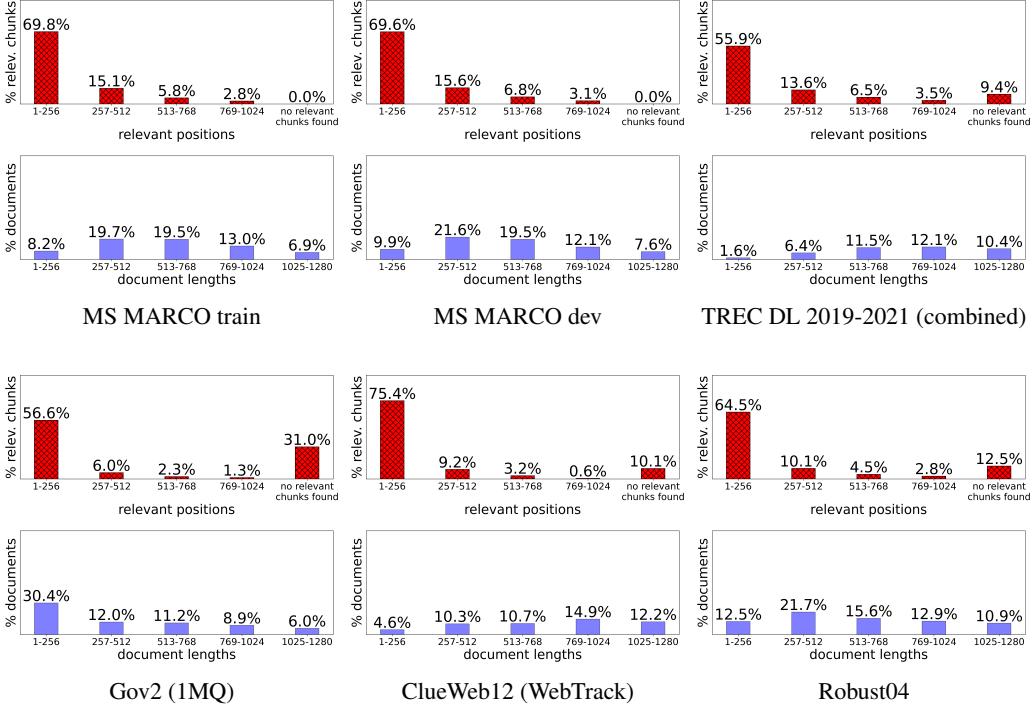


Figure 7: Illustration of positional relevance bias for four long-document collections and six query sets. We show a distribution of first relevant passage positions (red bars) vs. relevant document lengths (blue bars). Positions and lengths are measured in the number of subword tokens (BERT-base tokenizer). Best viewed in color.

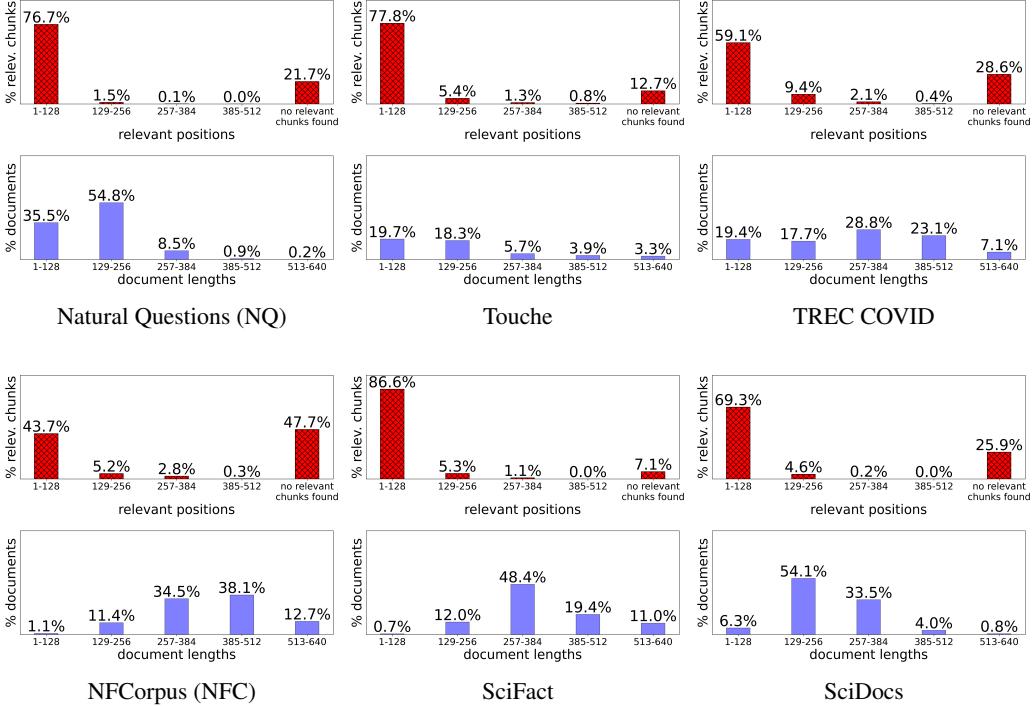


Figure 8: Illustration of positional relevance bias for six BEIR collections, all of which are typically categorized as *short-document* datasets. We show a distribution of first relevant passage positions (red bars) vs. relevant document lengths (blue bars). Positions and lengths are measured in the number of subword tokens (BERT-base tokenizer). Best viewed in color.

1687 2024; Arabzadeh and Clarke, 2025). The resulting  
1688 distributions can be found in Figure 7.

1689 Approximate string matching was used for MS  
1690 MARCO training and development (*dev*) sets, both  
1691 of which have sparse labels. Although initially MS  
1692 MARCO passages were exact substrings of MS  
1693 MARCO documents, document and passage texts  
1694 were collected at different times this lead to some  
1695 content divergence (Craswell et al., 2021a) that  
1696 made exact mapping virtually impossible. There  
1697 have been prior and contemporaneous attempts to  
1698 recover initial positions, but only with a limited  
1699 success. In particular, (Coelho et al., 2024) used  
1700 exact matching and found only about one thousand  
1701 matching passages. Hofst tter et al. (2020b) used  
1702 matching of answer words—rather than passage  
1703 text itself—and were able to match only 32% of  
1704 the passages. Both of these attempts match only a  
1705 small-to-modest fraction of passages while being a  
1706 subject to biases.

1707 Our approximate string matching combines  
1708 approximate substring matching with longest-  
1709 substring matching and incorporates efficiency  
1710 heuristics to identify initial candidate sets. Can-  
1711 didate sets were constructed using two approaches:  
1712 selecting all relevant passages and documents (for a  
1713 given query) and retrieving top-5 documents using  
1714 relevant passages as queries. To assess reliability,  
1715 we manually inspected a subset of the matched pas-  
1716 sages and found the procedure to be sufficiently  
1717 accurate. We then applied this approach to two sets  
1718 of queries:

- A set of all 5193 queries from the *dev* set;
- A (random and uniform) sample of 5000 train-  
ing queries.

1722 In both cases, we were able to find matches for  
1723 about 85% of the queries.

1724 For queries sets with “dense” relevance judg-  
1725 ments produced by TREC NIST assessors we used  
1726 an LLM judge. This included TREC 2019, 2020,  
1727 2021 TREC DL queries (Craswell et al., 2021b),  
1728 Robust04 (Clarke et al., 2004), Gov2 with 2007,  
1729 2008 Million Query Track queries (Allan et al.,  
1730 2008), and ClueWeb12 with 2012 and 2013 Web  
1731 Track queries (Collins-Thompson et al., 2013b).  
1732 We also used an LLM judge for BEIR datasets  
1733 (Thakur et al., 2021). It is noteworthy that Hof-  
1734 st tter et al. (2020b) employed crowd-workers to  
1735 identify the distribution of relevance chunks. They  
1736 found similar evidence of the relevance position

bias, but their study was limited only to TREC DL  
1737 2019 query set.

To avoid potential positional biases in LLM-judging, we divided each document into non-overlapping chunks and judged each chunk separately. Chunking was preserving sentence boundary while ensuring each chunk size was close to having 256 tokens for long-document collections and 128 tokens for BEIR collections (Thakur et al., 2021). For efficiency reasons, we only considered at most 36 chunks (about 9K tokens) and 500-2000 positive query-document pairs per query set. A chunk was considered to be relevant if it received a positive grade from the LLM-judge. The LLM-judge model was GPT4-mini (OpenAI, 2023).

For most collections, there was only a small fraction of documents where the LLM-judge found no relevant query-document pairs. One exception is the Gov2 collection with 2007, 2008 Million Query Track queries where this happened in about 30% of the cases. Despite this gap, Gov2 still had a substantial positional bias with about 57% of the cases where the first chunk was deemed to be relevant.

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

Table 11: Number of Model Parameters

## C 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 11). TinyLLAMA is a so-called LLM-ranker: a “causal” decoder-only Transformer that has about 1B parameters. Moreover, we focus only on the pure-Transformer architectures leaving hybrid architectures such as RankMamba for future work (Xu, 2024).

We assume that an input text is split into small chunks of texts called *tokens*. Although tokens can

1776 be complete English words, Transformer models  
1777 usually split text into sub-word units (Wu et al.,  
1778 2016).

1779 The length of a document  $d$ —denoted as  $|d|$ —  
1780 is measured in the number of tokens. Because  
1781 neural networks cannot operate directly on text, a  
1782 sequence of tokens  $t_1 t_2 \dots t_n$  is first converted to  
1783 a sequences of  $d$ -dimensional embedding vectors  
1784  $w_1 w_2 \dots w_n$  by an *embedding* network. These em-  
1785 beddings are context-independent, i.e., each token  
1786 is always mapped to the same vector (Collobert  
1787 et al., 2011; Mikolov et al., 2013).

1788 For a detailed description of Transformer mod-  
1789 els, please see the annotated Transformer guide  
1790 (Rush, 2018) as well as the recent survey by Lin  
1791 et al. (Lin, 2019), which focuses on the use of  
1792 BERT-style cross-encoding models for ranking and  
1793 retrieval. For this paper, it is necessary to know  
1794 only the following basic facts:

- 1795 • BERT is an encoder-only model, which con-  
1796 verts a sequence of tokens  $t_1 t_2 \dots t_n$  to a se-  
1797 quence of  $d$ -dimensional vectors  $w_1 w_2 \dots w_n$ .  
1798 These vectors—which are token representa-  
1799 tions from the *last* model layer—are com-  
1800 monly referred to as contextualized token em-  
1801 beddings (Peters et al., 2018);
- 1802 • BERT operates on word pieces (Wu et al.,  
1803 2016) rather than on complete words;
- 1804 • The vocabulary includes two special tokens:  
1805 [CLS] (an aggregator) and [SEP] (a sepa-  
1806 tor);
- 1807 • Using a *pooled* representation of token vectors  
1808  $w_1 w_2 \dots w_n$ , a linear layer is used to produce  
1809 a ranking score. A nearly universal pooling  
1810 approach in cross-encoding rankers is to use  
1811 the vector of the [CLS] token, i.e.,  $w_1$ . How-  
1812 ever, we learned that some models (e.g., JINA  
1813 (Günther et al., 2023)) converge well *only* with  
1814 mean pooling, i.e., they use  $\frac{1}{n} \sum_{i=1}^n w_i$ .

1815 A “vanilla” BERT ranker (dubbed as monoBERT  
1816 by Lin et al. (Lin, 2019)) uses a single fully-connect  
1817 layer  $F$  as a prediction head, which converts the  
1818 last-layer representation of the [CLS] token (i.e., a  
1819 contextualized embedding of [CLS]) into a scalar  
1820 (Nogueira and Cho, 2019). It makes a prediction  
1821 based on the following sequence of tokens: [CLS]  
1822  $q$  [SEP]  $d$  [SEP], where  $q$  is a query and  $d$  is a  
1823 document.

1824 An alternative approach is to aggregate con-  
1825 textualized embeddings of regular tokens using a  
1826 shallow neural network (MacAvaney et al., 2019;  
1827 Boytsov and Kolter, 2021; Khattab and Zaharia,  
1828 2020) (possibly together with the contextualized  
1829 embedding of [CLS]). This was first proposed by  
1830 MacAvaney et al. (MacAvaney et al., 2019) who  
1831 also found that incorporating [CLS] improves per-  
1832 formance. However, Boytsov and Kolter proposed  
1833 a shallow aggregating network that does not use the  
1834 output of the [CLS] token and achieved the same  
1835 accuracy on MS MARCO datasets (Boytsov and  
1836 Kolter, 2021).

1837 Replacing the standard BERT model in the  
1838 vanilla BERT ranker with a BERT variant that “na-  
1839 tively” supports longer documents (e.g., Big-Bird  
1840 (Zaheer et al., 2020)) is, perhaps, the simplest way  
1841 to deal with long documents. We collectively call  
1842 these models as LongP models. For a typical BERT  
1843 model, however, long documents and queries need  
1844 to be split or truncated so that the overall num-  
1845 ber of tokens does not exceed 512. In the *FirstP*  
1846 mode, we process only the first chunk and ignore  
1847 the truncated text. In the *SplitP* mode, each chunk  
1848 is processed separately and the results are aggre-  
1849 gated. In the remaining of this section, we discuss  
1850 these approaches in detail.

## C.1 LongP models

1851 In our work, we benchmark both sparse-attention  
1852 and full-attention models. Sparse attention LongP  
1853 models include two popular options: Longformer  
1854 (Beltagy et al., 2020) and Big-Bird (Zaheer et al.,  
1855 2020). In that, we use the same approach to  
1856 score documents as with the vanilla BERT ranker,  
1857 namely, concatenating queries with documents and  
1858 making a prediction based on the contextualized  
1859 embedding of the [CLS] token (Nogueira and Cho,  
1860 2019). Both Big-Bird and Longformer use a com-  
1861 bination of the local, “scattered” (our terminol-  
1862 ogy), and global attention. The local attention utilizes a  
1863 sliding window of a constant length where each to-  
1864 ken attends to each other token within this window.  
1865 In the case of the global attention, certain tokens  
1866 can attend to *all* other tokens and vice-versa. In  
1867 Big-Bird, only special tokens such as [CLS] can  
1868 attend globally. In Longformer, the user have to  
1869 select such tokens explicitly. Following Beltagy  
1870 et al. (Beltagy et al., 2020), who applied this tech-  
1871 nique to question-answering, we “place” global  
1872 attention only on query tokens. Unlike the global  
1873

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 TinyLLAMA 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].

## C.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. Because our primary focus is accuracy and we aim to understand the limits of long-document models, we exclude from evaluation several *SplitP* models, e.g., by Hofstätter et al. (2021b); Zou et al. (2021),

which achieve better efficiency-effectiveness trade-offs by pre-selecting certain document parts and feeding only selected parts into a BERT ranker.

**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:  $d = d_1 d_2 \dots d_m$ ;
- 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  $w_1 w_2 \dots w_n$  (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  $t_1 t_2 \dots t_n$ ;
- splitting a tokenized document  $d$  into  $m$  possibly overlapping chunks  $d_i = t_{i \cdot s} t_{i \cdot s+1} \dots t_{i \cdot s+w-1}$ : 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  $cls_i$ .<sup>10</sup> 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:  $w_1 w_2 \dots w_m = \text{softmax}(C \cdot cls_1, C \cdot cls_2, \dots, C \cdot cls_m)$ . These weights are used to compute the aggregated representation as  $\sum_{i=1}^m w_i cls_i$

<sup>10</sup>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, AvgP uses non-overlapping chunks while PARADE average relies on the sliding window approach.

PARADE Transformer models combine [CLS]-vectors  $cls_i$  with an additional *aggregator* transformer model  $\text{AggregTransf}()$ . 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  $\text{AggregTransf}()$ . The last-layer representation of the first vector is passed through a linear layer  $F$  to produce a relevance score:

$$F(\text{AggregTransf}(C, cls_1, cls_2, \dots, cls_m)[0]) \quad (1)$$

An aggregator Transformer can be either pre-trained 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  $\text{AggregTransf}$ , 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.<sup>11</sup>. 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

<sup>11</sup><https://microsoft.github.io/MSMARCO-Document-Ranking-Submissions/leaderboard/>

2059  
2060 an aggregator Transformer. In total, the model uses  
2061 three backbone encoder-only Transformers: All of  
2062 these Transformers are initialized using pretrained  
models.

2063 [Fu et al. \(2022\)](#) proposed a multi-view  
2064 interactions-based ranking model (MIR). They im-  
2065 plement inter-passage interactions via a multi-view  
2066 attention mechanism, which enables information  
2067 propagation at token, sentence, and passage levels.  
2068 Due to the computational complexity, they restrict  
2069 these interactions to a set of salient/pivot tokens.  
2070 However, the paper does not provide enough de-  
2071 tails regarding the choices of these tokens. There is  
2072 no software available and authors did not respond  
2073 to our clarification requests. Thus, this model is  
2074 also excluded from our evaluation.