Contextualized Offline Relevance Weighting for Efficient and Effective Neural Retrieval

Xuanang Chen[†]
University of Chinese Academy of Sciences, Beijing, China
chenxuanang19@mails.ucas.ac.cn

Ben He[†]
University of Chinese Academy of Sciences, Beijing, China
benhe@ucas.ac.cn

Kai Hui[*]
Amazon Alexa
Berlin, Germany
kaihuibj@amazon.com

Yiran Wang
University of Chinese Academy of Sciences, Beijing, China
wangyiran20@mails.ucas.ac.cn

Le Sun
Institute of Software, Chinese Academy of Sciences, Beijing, China
sunle@iscas.ac.cn

Yingfei Sun[†]
University of Chinese Academy of Sciences, Beijing, China
yfsun@ucas.ac.cn

ABSTRACT

Online search latency is a major bottleneck in deploying large-scale pre-trained language models, e.g. BERT, in retrieval applications. Inspired by the recent advances in transformer-based document expansion technique, we propose to trade offline relevance weighting for online retrieval efficiency by utilizing the powerful BERT ranker to weight the neighbour documents collected by generated pseudo-queries for each document. In the online retrieval stage, the traditional query-document matching is reduced to the much less expensive query to pseudo-query matching, and a document rank list is quickly recalled according to the pre-computed neighbour documents. Extensive experiments on the standard MS MARCO dataset with both passage and document ranking tasks demonstrate promising results of our method in terms of both online efficiency and effectiveness.

CCS CONCEPTS

• **Information systems → Retrieval models and ranking**.

KEYWORDS

Neural IR; Efficiency; Offline Relevance Weighting; BERT

ACM Reference Format:
Xuanang Chen, Ben He, Kai Hui, Yiran Wang, Le Sun, and Yingfei Sun. 2021. Contextualized Offline Relevance Weighting for Efficient and Effective Neural Retrieval. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '21), July 11–15, 2021, Virtual Event, Canada.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3404835.3463073

[*]This work was done before joining Amazon.
[†]Corresponding author

SIGIR '21, July 11–15, 2021, Virtual Event, Canada.

ACM ISBN 978-1-4503-8037-9/21/07…$15.00
https://doi.org/10.1145/3404835.3463073

1 INTRODUCTION

The applications of the pre-trained contextualized models, like BERT [7], T5 [20], and ELECTRA [2], have advanced the state-of-the-art of information retrieval. Though being effective [14, 16, 17], these large-scale models suffer from high latency [10]. As a result, such models are mainly used to re-rank a shallow-pool (e.g. 1,000 documents) recalled by a faster retrieval model, like BM25 [21], as the trade-off between the effectiveness and the latency requirements. More recently, there are efforts to exploit the strength of BERT also in the first retrieval stage, hoping to further boost the end-to-end retrieval effectiveness without introducing extra online computation cost. One direction is to optimize the index using BERT or seq2seq models before searching the index with a lexicon matching model, like BM25, by either employing BERT-based term-frequency [5] or expanding documents with potential queries [18, 19]. Another direction is to pre-compute the document encoding offline and replace the expensive cross-attentions with some lightweight relevance weighting mechanisms [11, 12].

Inspired by these works, we propose a novel approach to transfer significant amount of computation offline, decomposing the expensive online query-document matching to two parts, namely, the online query to pseudo-query matching, and the offline pseudo-query to document relevance weighting. Specifically, given a query in the online system, some seed documents are first fetched, whose pre-computed neighbour documents are also collected as candidates for re-ranking. Inspired by doc2query [19], the neighbour documents of a document are recalled offline using the pseudo-queries generated from this document. For a candidate document (a neighbour of one seed document), we use the query to pseudo-query similarity (pseudo-query is from this seed document) and the pre-computed pseudo-query to document relevance to estimate the final relevance of this candidate document. Therein, during online retrieval, the expensive cross-attentions on the query-document pair is replaced with the cheaper computation of query to pseudo-query similarity, given that the computational complexity of transformer-based model is quadratic to the sequence length. Evaluation at the TREC 2019 & 2020 Deep Learning Track demonstrates that the proposed method achieves superior effectiveness with better latency.

The contributions of this work are threefold: 1) We propose a neural retrieval framework based on contextualized offline document recall, replacing the expensive query-document cross-attentions

with cheaper query to pseudo-queries similarity matching; 2) Compared to several competitive models, including BM25+RM3, DeepCT and doc2query, our method shows superior effectiveness; and 3) Our method has comparable latency with ColBERT, while consuming markedly less disk space for storage. Some related data is available at https://github.com/cxa-unique/offline-relevance-weighting.

2 RELATED WORK

Transformer-based ranking models. Transformer-based ranking models have recently emerged as a promising research direction. For instance, BERT is applied to the MS MARCO passage retrieval task by fine-tuning [16], then extended to the long document retrieval task [6]. Besides, the T5 model is used to generate target words from documents/passages to indicate the relevance [17]. We refer readers to [25] for a more comprehensive survey.

Index optimization. DeepCT [5] uses BERT to produce the term frequency component of BM25 during indexing. doc2query [19] uses transformer [22] to generate queries before appending them in the end of the documents, then it relies on a BM25 index for retrieval from the expanded documents. It is latter improved by using the more powerful T5 model for the query generation [18].

Speeding up BERT ranker. Recently, there are ongoing efforts to compress BERT using knowledge distillation [1, 8, 9]. Besides, early exiting is also an efficient method wherein examples can exit from the early layers of BERT if the prediction threshold is reached [23]. ColBERT [12] separately encodes query and document, and employs a MaxSim-based late interaction mechanism. While ColBERT shows efficiency superiority compared to BERT, the huge storage cost of document embeddings remains a challenge [25].

3 METHOD

3.1 Overview

Framework. The proposed retrieval framework includes the offline computation and the online retrieval. There are two components in the offline system, namely, *doc2query* and *doc2doc*. Specifically, for an input document d, $doc2query(d)$ generates its pseudo-queries and $doc2doc(d)$ seeks its neighbour documents from the same corpus. In the online system, given a query, the lexicon matching function BM25 [21] is first used to obtain a set of initial documents coined as *seed documents*. All pre-computed neighbour documents of these seed documents will be recalled, ending up with a set of candidate documents for re-ranking. Thereafter, the relevance of a candidate document is evaluated using the pseudo-queries of some seed documents whose neighbour document set includes this candidate document. Akin to MaxP [6], for a long document, we segment it into several short passages before applying our method, and the maximum relevance score among all passages is used.

Notation. Given a corpus C, each document is denoted as $d \in C$. The two components are coined as functions taking individual d as input. As mentioned, they are denoted as $doc2query(d)$ and $doc2doc(d)$ as in Eq. 1 and 2. The outputs of $doc2query(d)$ are m pseudo-queries, which are denoted as $\mathcal{Q}_d = \{\overline{q}_1, \overline{q}_2, \cdots, \overline{q}_m\}$.

$$\mathcal{Q}_d = doc2query(d) \tag{1}$$

$doc2doc(d)$ in Eq. 2 fetches the neighbour documents of d from the corpus C, and the output is a set of n neighbour documents of d, coined as $\mathcal{D}_d = \{d_1, d_2, \cdots, d_n\}$.

$$\mathcal{D}_d = doc2doc(d) \tag{2}$$

We further denote the relevance between a (pseudo-)query and a document pair as $rel(q, d)$; and the similarity between an input query q and a pseudo-query $\overline{q}$ from $doc2query(d)$ as $sim(q, \overline{q})$. The output of these functions is a scalar in $[0, 1]$, indicating the degree of relevance or similarity between the two input text pieces. Finally, we denote $\mathcal{R}_q = [d_1, d_2, \cdots, d_r]$ as the returned document ranking from our retrieval framework when given a query q.

3.2 Offline System: Preparation of Pseudo-Queries and Neighbour Documents

***doc2query*(d): Pseudo-queries generation** produces short text pieces for each document $d \in C$. These pseudo-queries are relevant to the document and may contain some keywords of the document. Akin to [18, 19], for each document d, $\mathcal{Q}_d = doc2query(d)$ generates and stores m pseudo-queries by a seq2seq model.

***doc2doc*(d): Seeking the nearest neighbours of document d in the corpus.** For individual d, using its pseudo-queries $\mathcal{Q}_d$, we further fetch its neighbour documents that include some similar relevant information. In particular, each pseudo-query $\overline{q} \in \mathcal{Q}_d$ is used to recall a fixed number of documents from the corpus using BM25. Putting all these documents together, we then select n documents to form the neighbour document set of d, which is denoted as $\mathcal{D}_d = doc2doc(d)$, according to the recall frequency and BM25 scores. Note that, by definition, a document is always included in its own neighbour document set, namely, $d \in \mathcal{D}_d$.

Pre-computation of relevance scores using BERT. The last step in the offline system is the relevance weighting of pairs between pseudo-queries and neighbour documents from document $d \in C$. In particular, for each $\overline{q}_j \in \mathcal{Q}_d$ and each $d_i \in \mathcal{D}_d$, a BERT-based ranker is used to evaluate their relevance, as in Eq. 3, which is stored for the uses in the online system.

$$rel(\overline{q}_j, d_i) = \text{BERT}(\overline{q}_j, d_i) \tag{3}$$

3.3 Online System: Neighbour Document Re-ranking using Pseudo-Queries

Creation of the candidate document set for re-ranking. In the online system, given a query q, the lexicon matching function BM25 is first used to get top-s ranked documents to form a seed document set, which is denoted as $\mathcal{S}_q = \{d_1, d_2, \cdots, d_s\}$. For each seed document $d \in \mathcal{S}_q$, the pre-computed $doc2doc(d)$ is used to fetch its neighbour document set $\mathcal{D}_d$. The union of neighbour document sets of all seed documents is the final candidate document set with r documents, which is denoted as $\mathcal{R}_q = \{d_1, d_2, \cdots, d_r\}$. As $d \in \mathcal{D}_d$, $\mathcal{R}_q$ contains all seed documents in $\mathcal{S}_q$, namely, $r \geq s$.

Relevance for the candidate documents. For each candidate document $d_i \in \mathcal{R}_q$, picking one seed document $d \in \mathcal{S}_q$ whose neighbour document set includes d_i, namely, $d_i \in \mathcal{D}_d$, the relevance of d_i relative to q can be approximated using the pre-computed relevance between d_i and each $\overline{q}_j \in \mathcal{Q}_d$. As in Eq. 4, for $\overline{q}_j \in \mathcal{Q}_d$, we can calculate the product between $sim(q, \overline{q}_j)$ and $rel(\overline{q}_j, d_i)$, ultimately using the maximum value among m products as the $rel(q, d_i)$. It can be seen that the online computation is reduced from $rel(q, d_i)$ to $sim(q, \overline{q}_j)$ that will be introduced in Eq. 5. Note

Table 1: Retrieval results of passage ranking task at TREC Deep Learning Track. Statistical significant difference relative to docTTTTTquery at p-value < 0.01, 0.05, and 0.1 are denoted as $***$, $**$, $*$, respectively. s is the number of seed documents.

Method	TREC 2019 DL Passage Ranking Task (43 queries)						TREC 2020 DL Passage Ranking Task (54 queries)					
	MRR	NDCG@10	MAP	R@100	R@500	R@1000	MRR	NDCG@10	MAP	R@100	R@500	R@1000
BM25	0.8245	0.5058	0.3773	0.4531	0.6816	0.7389	0.6585	0.4796	0.2856	0.5599	0.7278	0.7863
BM25+RM3	0.8167	0.5180	0.4270	0.4761	0.7237	0.7882	0.6360	0.4821	0.3019	0.6046	0.7864	0.8217
DeepCT	0.8178	0.5140	0.3866	0.4755	0.6705	0.7329	0.6522	0.5259	0.3248	0.6221	0.7625	0.8056
doc2query	0.8573	0.5333	0.3942	0.4707	0.6738	0.7414	0.6661	0.5258	0.3186	0.5953	0.7604	0.8088
docTTTTTquery	0.8884	0.6417	0.4625	0.5141	0.7561	0.8020	0.7326	0.6187	0.4074	0.7044	0.8107	0.8452
Ours (s = 30)	**0.9155**	0.6894**	0.5037***	0.5513*	0.7726	0.8155	0.7749*	0.6517**	0.4480***	0.7256	0.8352*	0.8598
Ours (s = 50)	**0.9155**	**0.6924****	**0.5100*****	**0.5574****	**0.7789**	**0.8160**	**0.7899****	**0.6637****	**0.4530*****	**0.7276**	**0.8388***	**0.8613**

Table 2: Retrieval results of document ranking task at TREC Deep Learning Track. Statistical significant difference relative to docTTTTTquery at p-value < 0.01, 0.05, and 0.1 are denoted as $***$, $**$, $*$, respectively. s is the number of seed documents.

Method	TREC 2019 DL Document Ranking Task (43 queries)						TREC 2020 DL Document Ranking Task (45 queries)					
	MRR	NDCG@10	MAP	R@100	R@500	R@1000	MRR	NDCG@10	MAP	R@100	R@500	R@1000
BM25	0.8046	0.5190	0.3309	0.3948	0.6157	0.6966	0.8521	0.5271	0.4043	0.6110	0.7641	0.8085
BM25+RM3	0.7718	0.5169	0.3870	0.4189	0.6687	**0.7504**	0.8541	0.5248	0.4263	0.6392	0.7784	0.8260
docTTTTTquery	0.9097	0.6131	0.3641	0.4054	0.6314	0.7029	0.9278	0.5963	0.4434	0.6258	0.7697	0.8048
Ours (s = 30)	0.9277	0.6533***	0.4046***	0.4369***	0.6805***	0.7423***	**0.9389**	**0.6065**	0.4724***	0.6630***	0.8081**	0.8400***
Ours (s = 50)	**0.9382**	**0.6546*****	**0.4125*****	**0.4449*****	**0.6870*****	0.7479***	**0.9389**	0.6064	**0.4731*****	**0.6664*****	**0.8089****	**0.8414*****

that d_i can be given several relevance scores as it may be included in the neighbour document set of more than one $d \in \mathcal{S}_q$, wherein the maximum is used as the final relevance score.

$$rel(q, d_i) = \max_{\overline{q}_j \in Q_d} sim(q, \overline{q}_j) \times rel(\overline{q}_j, d_i) \tag{4}$$

Online computation: The similarity between the query and the pseudo-query. For each seed document $d \in \mathcal{S}_q$, as mentioned, the similarity scores used in Eq. 4 between query q and pseudo-queries $\overline{q}_j \in Q_d$ are computed by a BERT-based ranker as in Eq. 5.

$$sim(q, \overline{q}_j) = \text{BERT}(q, \overline{q}_j) \tag{5}$$

As mentioned, the computation of the similarity between query and pseudo-query is relatively cheaper as it avoids the direct uses of BERT on the concatenated sequences of query and documents.

Interpolation with exact matching signals. As the above relevance only considers the contextualized semantic matching, similar to exiting work [26], exact matching signals from lexicon model BM25 are further added as in Eq. 6.

$$\widehat{rel}(q, d_i) = \alpha \times rel(q, d_i) + (1 - \alpha) \times \text{BM25}(q, d_i) \tag{6}$$

where α trade-offs the weight between the semantic relevance score and the BM25 score (normalized into $[0, 1]$). After getting relevance scores $\widehat{rel}(q, d_i)$ for each $d_i \in \mathcal{R}_q$, $\mathcal{R}_q$ can be ranked accordingly.

4 EXPERIMENTS

4.1 Experiment Settings

Dataset and metrics. MS MARCO [15] is used as the main dataset in our experiments with both passage and document retrieval tasks. The test queries from TREC 2019 and 2020 Deep Learning (DL) Track [3, 4], together with the manual multi-graded judgments according to NIST assessors, are used, allowing the calculation of precision and recall at different pool depths. We report the official evaluation metrics MRR, NDCG@10 and MAP at TREC DL Track. Besides, we also report the retrieval Recall metric R@100, R@500 and R@1000 to examine the quality of candidate set. Statistical significance for paired two-tailed t-test is reported.

Models in comparisons. We compare our method against the following traditional and neural baselines: **BM25** [21] is a classical unsupervised probabilistic ranking model based on exact term matching. On top of that, **BM25+RM3** applies query expansion based on the relevance language model [13]. **DeepCT** [5] replaces the term frequency in BM25 using BERT-based prediction during offline indexing. **doc2query** [19] and **docTTTTTquery** [18] expand documents with queries generated by a seq2seq model, and rely on BM25 for retrieval over expanded documents. We use Anserini's [24] implementations of all above models (including the BM25 used in our method) with default parameter settings.

Implementation details. The pseudo-query to document relevance and query to pseudo-query similarity are both generated using a BERT-Base model, whose checkpoint is fine-tuned on MS MARCO Passage dataset and released in [14]. The maximum sequence length of (pseudo-)query and document are 64 and 448, respectively. In the *offline* system, for each document, the first five predicted pseudo-queries (m = 5) from docTTTTTquery [18] are used. Each pseudo-query recalls top-1,000 documents using BM25, before further selecting 1,000 documents as the neighbour documents (n = 1,000). In the *online* system, given a query, we experiment with 30 and 50 seed documents using BM25 (namely, s = 30 and 50). As for the interpolation in Eq. 6, α is set to 0.9, and d_i in BM25 is expanded using 40 and 10 pseudo-queries for passage and document ranking tasks, respectively. For document ranking task, we segment each document using a sliding window of ten sentences and stride of five sentences as in docTTTTTquery with per-passage expansion [18]. For the ColBERT (over BERT-Base) re-ranker, all training and inference settings follow the ones in [12].

Table 3: Results of passage retrieval after being re-ranked (rr.) by BERT-Base or ColBERT over top-20/1000 passages. s is the number of seed documents. NDCG@10 is reported.

Method	TREC 2019 DL Track	TREC 2020 DL Track
BM25 rr. BERT-Base	0.6235 / 0.6945	0.5814 / 0.6677
BM25 rr. ColBERT	0.6276 / 0.7164	0.5943 / 0.6913
BM25+RM3 rr. BERT-Base	0.6014 / 0.6963	0.5664 / 0.6608
BM25+RM3 rr. ColBERT	0.5998 / 0.7281	0.5900 / 0.6814
DeepCT rr. BERT-Base	0.6348 / 0.6929	0.5855 / 0.6883
DeepCT rr. ColBERT	0.6534 / 0.7128	0.5970 / **0.7050**
doc2query rr. BERT-Base	0.6478 / 0.6949	0.6049 / 0.6793
doc2query rr. ColBERT	0.6501 / 0.7233	0.6205 / 0.6920
docTTTTTquery rr. BERT-Base	0.7269 / 0.7100	0.6820 / 0.6784
docTTTTTquery rr. ColBERT	0.7399 / 0.7328	0.6931 / 0.6935
Ours (s = 30) rr. BERT-Base	0.7237 / 0.6990	0.6808 / 0.6749
Ours (s = 30) rr. ColBERT	0.7454 / 0.7244	0.7086 / 0.6842
Ours (s = 50) rr. BERT-Base	0.7271 / 0.7043	0.6860 / 0.6791
Ours (s = 50) rr. ColBERT	**0.7556** / **0.7340**	**0.7129** / 0.6916

Table 4: Efficiency of passage retrieval. The 43 queries in TREC 2019 DL Track are used. Each method retrieves top-1000 passages per query. s is the number of seed documents.

Method	CPU (ms/query)	GPU (ms/query)	Latency (ms/query)
BM25	93	-	93
BM25+RM3	186	-	186
DeepCT	-	-	93 (est.)
doc2query	116	-	116
docTTTTTquery	139	-	139
ColBERT	-	-	687 (est.)
BERT-Base (re-rank)	-	11777	11777
Ours (s = 30)	120	363	483
Ours (s = 50)	174	605	779

4.2 Results

Our method returns significantly better candidate ranking. For passage ranking task, as shown in Table 1, our method significantly outperforms the best baseline, namely, docTTTTTquery, in both tracks in terms of NDCG@10 and MAP. Meanwhile, our method also outperforms all baselines in terms of MRR and Recall. For document ranking task, as shown in Table 2, our method performs significantly better than all baselines in terms of MAP and Recall. Our method significantly outperforms all baselines at TREC 2019 DL Track in terms of NDCG@10, and performs better than all baselines at both DL tracks in terms of MRR.

Only a small number of candidates need to be re-ranked using our method. In Table 3, the passage re-ranking results in early precision measured by NDCG@10 are reported. When top-20 passages are re-ranked, the candidates retrieved by our method and then re-ranked by ColBERT can get best NDCG@10 at both DL tracks, and even outperform all other baselines that re-rank top-1000 passages. This observation is encouraging as only a small set of candidates are needed for achieving competitive performance with improved online efficiency using our method.

4.3 Analysis

Online efficiency on MS MARCO. The efficiency of different models is summarized in Table 4. The latency excludes the time of data pre-processing and reading indexes or other data into memory.

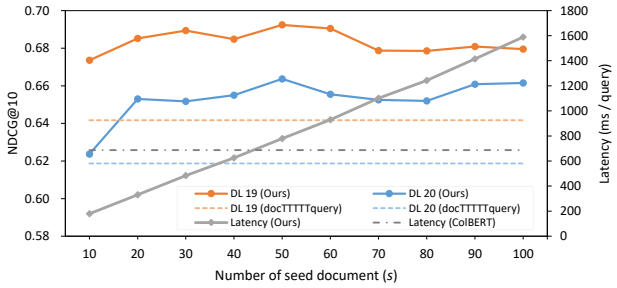

Figure 1: Influence of the number of seed document on passage ranking task at TREC DL Track.

For the latency of BM25(+RM3), doc2query and docTTTTTquery, the Anserini's implementations are used. Besides, DeepCT's latency is estimated as in [12] and ColBERT's latency is estimated proportionally to BM25's latency (both are denoted by est.). The latency of BERT-Base (re-rank 1,000 candidates) is significantly higher. In our method, when 30 seed documents are used, a query only needs to match 30×5=150 pseudo-queries, wherein the latency is lower than ColBERT. Figure 1 shows the influence of the number of seed documents. Our method performs best when $s = 50$, and the latency is lower than or close to ColBERT if s is between 10 and 50. Note that while full ranking with ColBERT can hopefully achieve better effectiveness, its latency linearly grows with the collection size.

The offline computing cost. For MS MARCO Passage dataset (about 8.8 M passages), it takes about 6,631 days to complete the offline relevance weighting for pseudo-query to neighbour passage pairs (5,000 pairs/passage) by BERT-Base on one Titan RTX 24G GPU with sequence length of 512. But, this can be drastically speed up by reducing sequence length, parallel computing and using TPUs (a single TPU v3 is 20-30× faster from our experience). For example, it would take less than 1 day to finish the offline computation by using 100 TPUs v3 with sequence length of 256.

The space footprint. In our method, a document has five pseudo-queries (1 byte/letter, about 34 letters/pseudo-query) and $m \times n = 5,000$ relevance scores (4 bytes/score), adding up to 20,170 (1×) bytes. Assuming a document on average includes 200 tokens and is represented using 128-dimensional vectors (4 bytes/dimension), ColBERT requires 102,400 (5×) bytes for one document. Thus, the space footprint of our method is about 0.2× of ColBERT.

5 CONCLUSIONS

In this paper, we propose a neural retrieval framework based on contextualized offline document recall, wherein the neighbour documents of selected seed documents are quickly re-ranked using pre-computed relevance and pseudo-queries. Experiments at TREC 2019 and 2020 DL Track demonstrate that our method achieves not only superior effectiveness and recall rates, but also lowers online latency and disk storage cost. For the future work, we plan to further investigate how to accelerate the offline computation.

ACKNOWLEDGMENTS

This research work is supported by the National Key Research and Development Program of China under Grant No.2017YFB1002104.

REFERENCES

[1] Xuanang Chen, Ben He, Kai Hui, Le Sun, and Yingfei Sun. 2021. Simplified TinyBERT: Knowledge Distillation for Document Retrieval. In *ECIR (2) (Lecture Notes in Computer Science, Vol. 12657)*. Springer, 241–248.

[2] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. In *ICLR*. OpenReview.net.

[3] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2021. Overview of the TREC 2020 deep learning track. *CoRR* abs/2102.07662 (2021).

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

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

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

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT (1)*. Association for Computational Linguistics, 4171–4186.

[8] Luyu Gao, Zhuyun Dai, and Jamie Callan. 2020. Understanding BERT Rankers Under Distillation. In *ICTIR*. ACM, 149–152.

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

[10] Sebastian Hofstätter and Allan Hanbury. 2019. Let's measure run time! Extending the IR replicability infrastructure to include performance aspects. In *OSIRRC@SIGIR (CEUR Workshop Proceedings, Vol. 2409)*. CEUR-WS.org, 12–16.

[11] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *EMNLP (1)*. Association for Computational Linguistics, 6769–6781.

[12] Omar Khattab and Matei Zaharia. 2020. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. In *SIGIR*. ACM, 39–48.

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

[14] Canjia Li, Andrew Yates, Sean MacAvaney, Ben He, and Yingfei Sun. 2020. PARADE: Passage Representation Aggregation for Document Reranking. *CoRR* abs/2008.09093 (2020).

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

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

[17] Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document Ranking with a Pretrained Sequence-to-Sequence Model. In *EMNLP (Findings)*. Association for Computational Linguistics, 708–718.

[18] Rodrigo Nogueira, Jimmy Lin, and AI Epistemic. 2019. From doc2query to docTTTTTquery. *Online preprint* (2019).

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

[20] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.* 21 (2020), 140:1–140:67.

[21] Stephen E. Robertson, Steve Walker, Micheline Hancock-Beaulieu, Mike Gatford, and A. Payne. 1995. Okapi at TREC-4. In *TREC (NIST Special Publication, Vol. 500-236)*. National Institute of Standards and Technology (NIST).

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

[23] Ji Xin, Rodrigo Nogueira, Yaoliang Yu, and Jimmy Lin. 2020. Early Exiting BERT for Efficient Document Ranking. In *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing*. Association for Computational Linguistics, 83–88.

[24] Peilin Yang, Hui Fang, and Jimmy Lin. 2017. Anserini: Enabling the Use of Lucene for Information Retrieval Research. In *SIGIR*. ACM, 1253–1256.

[25] Andrew Yates, Rodrigo Nogueira, and Jimmy Lin. 2021. Pretrained Transformers for Text Ranking: BERT and Beyond. In *WSDM*. ACM, 1154–1156.

[26] Zeynep Akkalyoncu Yilmaz, Wei Yang, Haotian Zhang, and Jimmy Lin. 2019. Cross-Domain Modeling of Sentence-Level Evidence for Document Retrieval. In *EMNLP/IJCNLP (1)*. Association for Computational Linguistics, 3488–3494.