Learning Passage Impacts for Inverted Indexes

Antonio Mallia
antonio.mallia@nyu.edu
New York University

Omar Khattab
okhattab@stanford.edu
Stanford University

Torsten Suel
torsten.suel@nyu.edu
New York University

Nicola Tonellotto
nicola.tonellotto@unipi.it
University of Pisa

ABSTRACT

Neural information retrieval systems typically use a cascading pipeline, in which a first-stage model retrieves a candidate set of documents and one or more subsequent stages re-rank this set using contextualized language models such as BERT. In this paper, we propose DeepImpact, a new document term-weighting scheme suitable for efficient retrieval using a standard inverted index. Compared to existing methods, DeepImpact improves impact-score modeling and tackles the vocabulary-mismatch problem. In particular, DeepImpact leverages DocT5Query to enrich the document collection and, using a contextualized language model, directly estimates the semantic importance of tokens in a document, producing a single-value representation for each token in each document. Our experiments show that DeepImpact significantly outperforms prior first-stage retrieval approaches by up to 17% on effectiveness metrics w.r.t. DocT5Query, and, when deployed in a re-ranking scenario, can reach the same effectiveness of state-of-the-art approaches with up to 5.1× speedup in efficiency.

CCS CONCEPTS

• **Information systems** → **Information retrieval**; **Information retrieval query processing**; **Retrieval models and ranking**.

KEYWORDS

Query processing; Inverted index; Neural IR; Term Weighting;

ACM Reference Format:
Antonio Mallia, Omar Khattab, Torsten Suel, and Nicola Tonellotto. 2021. Learning Passage Impacts for Inverted Indexes. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '21), July 11–15, 2021, Virtual Event, Canada.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3404835.3463030

1 INTRODUCTION

Modern search engines employ complex, machine-learned ranking functions to retrieve the most relevant documents for a query. Recently, the development of pre-trained contextualized language models such as BERT [6] has resulted in impressive benefits in search effectiveness, at the cost of expensive query processing times, which can make their deployment in production scenarios challenging. Nogueira and Cho [18] and MacAvaney et al. [14] showed the superior performance of BERT in term of effectiveness for passage and document re-ranking tasks, respectively, by fine-tuning the pre-trained transformer network to distinguish between relevant and non-relevant query–document pairs. However, several recent studies [7, 14] have shown that this can have very high computational cost, even if re-ranking just the top 1000 results. Other studies [9, 12, 13] proposed methods with lower computational cost but typically some loss in retrieval quality. BERT's Transformer encoder is composed of many neural layers performing expensive processing to compute the query-document relevance signals. Different solutions have been proposed to address this performance bottleneck, based on the pre-computation of query-document representations produced by BERT. EPIC [13] proposes to build on top of BERT a new ranking model trained to generate query and document representations in a given fixed-length vector space, equal to the size of the lexicon. Document representations are pre-computed, while query representations are computed at retrieval time, and then used to obtain a ranking score by computing a similarity between the two representations.

PreTTR [12] and ColBERT [9] experimentally show that the query-document interactions in most layers of BERT have little impact on the final effectiveness. This leads them to pre-compute document representations at indexing time, which are used at query processing time to compute the query-document interaction only in a final layer. While PreTTR still relies upon a first-stage candidate generation based on BM25, ColBERT investigates the ability of the pre-computed document representations to identify relevant documents among *all* documents in the index. Due to space/time requirements of the document representation, ColBERT leverages approximate nearest neighbor (ANN) search applied to dense representations as a first-stage retrieval system, followed by an exact re-ranking stage, while similar approaches using exact nearest neighbor search [23] can perform processing in a single stage.

Following a different paradigm, Dai and Callan [4] investigated the use of the contextual word representations from BERT to generate more effective document term weights for bag-of-words retrieval. DeepCT [4], for passages, and HDCT [5], for documents, estimate a term's context-specific importance in each passage/document, by projecting each word's BERT representation into a single term weight. These term weights are then transformed into term frequency-like integer values that can be stored in an inverted index to be used with classical retrieval models. A main limitation of DeepCT that we address in this work is that it is trained as a *per-token regression task*, in which a ground truth term weight for every word is needed, and which does not permit the individual impact scores to co-adapt for the downstream objective of identifying relevant documents.

By storing new integer values as term frequencies in the inverted index, DeepCT and HDCT enrich a document's bag-of-words representation with additional document-level context information, to match queries more accurately. Using a different approach,

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

ACM ISBN 978-1-4503-8037-9/21/07…$15.00
https://doi.org/10.1145/3404835.3463030

Nogueira and Lin [19] propose DocT5Query, a document expansion strategy to enrich each document with additional terms able to improve the retrieval effectiveness of documents w.r.t. queries for which they are relevant. DocT5Query trains a sequence-to-sequence model to predict queries potentially relevant to a given document, and appends these queries to the documents before indexing. As another way of expanding documents, the very recent SparTerm [1] method predicts an importance score for every term in the vocabulary and uses a gating mechanism to only keep a sparse subset of those, using them to learn an *end-to-end* score for relevant and non-relevant documents. However, this only increases the MRR@10 of DocT5Query from 0.277 to 0.279.

We propose DeepImpact, a more effective approach for learning a relevance score contribution for term-document pairs that can also be stored in a classical inverted index. DeepImpact improves impact-score modeling and tackles the vocabulary-mismatch problem [25] between queries and documents. Instead of learning *independent* term-level scores without taking into account the term co-occurrences in the document, as in DeepCT, or relying on unchanged BM25 scoring, as in DocT5Query, DeepImpact directly optimizes the *sum* of query term impacts to maximize the score difference between relevant and non-relevant passages for the query. In other words, while DeepCT learns the term frequency component of existing IR models, e.g., BM25, *in this work we aim at learning the final term impact jointly across all query terms occurring in a passage*. In this way, our proposed model learns richer interaction patterns among the impacts, when compared to training each impact in isolation. To address vocabulary mismatch, DeepImpact leverages DocT5Query to enrich every document with new terms likely to occur in queries for which the document is relevant. Using a contextualized language model, it directly estimates the semantic importance of tokens in a document, producing a single-value representation for each token in each document that can be stored in an inverted index for efficient retrieval. Our experiments show that DeepImpact significantly outperforms prior first-stage retrieval approaches by up to 17% on effectiveness metrics w.r.t. DocT5Query. When deployed in a re-ranking scenario, it reaches the same effectiveness as state-of-the-art approaches up to 5.1× faster.

In summary, this paper makes the following contributions:

- We propose DeepImpact, a more effective scheme for *jointly learning* term impacts over *expanded documents*.
- We evaluate DeepImpact on the MS MARCO passage ranking task. We find that DeepImpact can improve ranking effectiveness for passage ranking versus prior first-stage retrieval approaches and is competitive when compared to complex systems based on ANN search, while exhibiting much lower computational costs.
- We evaluate DeepImpact as a first-stage model in a re-ranking pipeline, and show that this pipeline matches or outperforms strong baseline approaches, while being highly efficient.

2 DEEP IMPACT FRAMEWORK

Document Expansion. In our approach, we leverage DocT5Query document expansion to enrich the original document collection with expansion terms. As noted by Nogueira et al. [21], document expansion can be seen as a two-fold approach. By adding terms that are already part of the document, it rewrites their frequencies, similar to DeepCT. Furthermore, it injects into the passage new terms, originally not part of the document, in order to address the term mismatch problem. We refer to the two as *Rewrite* and *Inject*, respectively. Table 1 summarizes the effect of DocT5Query when applied to the MSMARCO passage ranking collection, and isolates the two contributions. While *Rewrite* alone achieves stronger MRR@10 than *Inject*, the latter achieves higher recall. Using both significantly outperforms either one on both measures. Indeed, *Inject* is important for capturing additional results, but *Rewrite* is needed to then properly weight the injected terms. However, the comparison of *Rewrite* vs. DeepCT indicates that DocT5Query is still sub-optimal in determining the right frequencies, and resulting impact scores, for the terms.

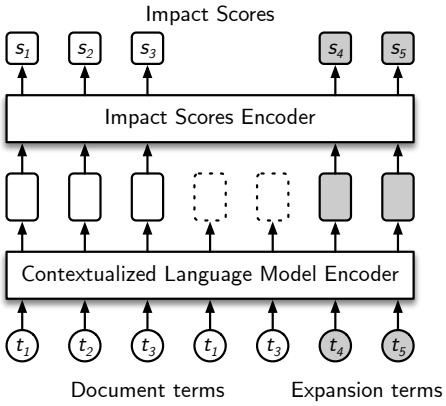

Figure 1: Neural network architecture of DeepImpact.

This motivates our approach, DeepImpact, where we first use the *Inject* step of DocT5Query to add new terms, and then directly learn the right impact scores for both old and newly injected terms.

Table 1: Different contributions to effectiveness metrics on the MSMARCO passage ranking collection.

	BM25	DeepCT	DocT5Query		
			Cumulative	*Rewrite*	*Inject*
MRR@10	0.188	0.244	0.278	0.215	0.194
Recall	0.858	0.910	0.947	0.878	0.912

Neural Network Architecture. Figure 1 depicts DeepImpact's architecture. DeepImpact feeds a contextual LM encoder the original document terms (in white) and the injected expansion terms (in gray), separating both by a `[SEP]` separator token. The LM encoder produces an embedding for each input token. The first occurrence of each *unique* term is provided as input to the impact score encoder, which is a two-layer MLP with ReLU activations. This produces a single-value score for each unique term in the document, representing its impact. Given a query q, we model the score of document d as simply the sum of impacts for the intersection of terms in q and d.

Network Training. We train our model using triples sampled from the official MS-MARCO training dataset, consisting of a query, a relevant passage, and a presumed non-relevant passage per sample.

We expand each passage using the DocT5Query as discussed. The model converts each document into a list of scores, corresponding to the document terms matching the query. These scores are then summed up, obtaining an accumulated query-document score. For each triple, two scores for the corresponding two documents are computed. The model is optimized via pairwise softmax cross-entropy loss over the document scores. We use BERT-base as the contextual LM. We set the maximum sequence length to 160 tokens. Losses are back-propagated through the whole DeepImpact neural model with a learning rate of 3×10^{-6} with the Adam optimizer. We used batches of 32 triples and train for 100,000 iterations.

Impact Scores Computation. Following the training phase, DeepImpact can leverage the learned term-weighting scheme to predict the semantic importance of each token of the documents offline. Each document is represented as a list of term-score pairs, which are converted into an inverted index. The index can then be searched using efficient query processing strategies. We infer the scores using three digits of precision, and we do not perform any scaling.

Quantization and Query Processing. We predict real-valued document-term impact scores, but as storing a floating point value per posting would blow up the space requirements of the inverted index, we decided to store impacts in a quantized form. The quantized impact scores belong to the range of $[1, 2^b - 1]$, where b is the number of bits used to store each value. We experimented with $b = 8$ using linear quantization, and did not notice any loss in precision w.r.t. the original scores. Since we quantized all the scores in the index in the same way, to compute a query-document score at query processing we can just sum up all the quantized scores of the document terms matching the query.

3 EXPERIMENTAL RESULTS

In this section, we analyze the performance of the proposed method using a standard test collection and query logs.

Hardware. To evaluate the latency, we use a single core of a machine with four Intel Xeon Platinum 8268 CPUs and 369 GB of RAM, running Linux 4.18. To run ColBERT a GPU is required, and we used an NVIDIA RTX8000 with 48GB of memory.

Dataset and query logs. We conduct our experiments on the MS-MARCO passage ranking [17] dataset. To evaluate query processing effectiveness and efficiency, we compare with existing methods using the MSMARCO Dev Queries,[1] and we test all methods on the TREC 2019 [3] and TREC 2020 [2] queries from the TREC Deep Learning passage ranking track.

Baselines. We perform two different sets of experiments. Our initial experiment aims at comparing the performance of DeepImpact as a first-stage ranker, processing queries on inverted indexes but without complex reranking. In this experiment we compare our proposed DeepImpact with the classical BM25 relevance model over the unmodified collection, and state-of-the-art solutions dealing with inverted indexes, namely DeepCT, and BM25 over a collection expanded with DocT5Query. We do not compare with DeepCT over the collection expanded with DocT5Query, since that would involve training a new DeepCT model from scratch to learn how to weigh expanded documents. Our second set of experiments compares DeepImpact in a re-ranking setting. First, the top 1000 documents retrieved by DeepImpact are re-ranked by EPIC and ColBERT and compared to ColBERT end-to-end (E2E) where the candidates are generated using ANN search. Finally, we look at first-stage recall and re-ranking-stage MRR@10 when applying ColBERT at several first-stage cutoffs to different candidate generation methods.

Implementations. We use Anserini [24] to generate the inverted indexes of the collections. We then export the Anserini indexes using the CIFF common index file format [10], and process them with PISA [16] using the MaxScore query processing algorithm [22]. We use the BM25 scoring method provided by Anserini. For DeepCT, we used the source code and data[2] provided by Mackenzie et al. [15]. For DocT5Query we use the predicted queries available online,[3] using 40 concatenated predictions for each passage in the corpus, as recommended by Nogueira and Lin [19]. We use the EPIC implementation in OpenNIR [11] and the official pretrained model[4]. We use the ColBERT implementation[5] provided by Khattab and Zaharia [9], trained for 200k iterations. Both training and indexing tasks of DeepImpact are implemented in Python. After the quantization step, the documents are indexed directly by PISA. Query processing efficiency is measured using PISA for all baselines. Query processing is performed using MaxScore to retrieve the top 1000 documents. Our source code is publicly available.[6]

Metrics. To measure effectiveness, we use the official metrics for each query set, mean reciprocal rank (MRR@10) for MSMARCO queries, and normalised discounted cumulative gain (NDCG@10) as well as mean average precision (MAP) for TREC queries, following [8]. We also report recall on the first stage and MRR@10 on the re-ranking stage at different cutoff values. Finally, we compute the mean response time (MRT) for every query processing strategy, in ms. We conduct Bonferroni corrected pairwise t-tests, and report significance with $p < 0.05$.

Overall comparison. Our first experiment aims to show the early-stage effectiveness improvements that DeepImpact achieves when compared to prior work. The results are presented in Table 2, which shows effectiveness and efficiency for the three query logs on MS MARCO. We retrieve the top 1000 documents for each query, without re-ranking, and report the values of NDCG@10, MRR@10, and MAP, as well as MRT.

DeepImpact significantly outperforms all methods and is statistically significantly better than other strategies for all effectiveness metrics on the MSMARCO Dev Queries. For the TREC 2019 and TREC 2020 queries, DeepImpact is always better than the competitors, with statistically significant improvements on NDCG@10 and MAP in some cases. Statistical significance on the latter two query traces is limited by their relatively small number of queries.

We also see that DeepImpact mean response time exceeds the time reported for other methods. We trace this to the query processing strategy: the distribution of scores induced by BM25, used in BM25, DeepCT, and DocT5Query is exploited more efficiently by

[1] We have made a submission to the official leaderboard and obtained an MRR@10 of 0.318 on the "eval" queries.

[2] https://github.com/jmmackenzie/term-weighting-efficiency
[3] https://github.com/castorini/docTTTTTquery
[4] https://github.com/Georgetown-IR-Lab/epic-neural-ir
[5] https://github.com/stanford-futuredata/ColBERT
[6] https://github.com/DI4IR/SIGIR2021

the MaxScore algorithm, whereas DeepImpact learns new scores, whose distribution is not efficiently exploited by MaxScore. We performed additional experiments using disjunctive query processing without optimizations, omitted for space limitations. These experiments show DeepImpact to be in line with the speed of the other approaches. Optimizing the query processing speed of DeepImpact is an interesting open problem for future research.

Table 2: Effectiveness metrics and mean response time (MRT, in ms) for first-stage methods, on MSMARCO Dev Queries, TREC 2019 queries, and TREC 2020 queries. The symbol ▽ denotes a significant difference viz. DeepImpact

Strategy	NDCG@10	MRR@10	MAP	MRT
MSMARCO Dev Queries				
BM25	0.235▽	0.188▽	0.196▽	13.24
DeepCT	0.298▽	0.244▽	0.252▽	10.91
DocT5Query	0.338▽	0.278▽	0.286▽	12.62
DeepImpact	**0.385**	**0.326**	**0.332**	58.64
TREC 2019				
BM25	0.497▽	0.683	0.290▽	10.27
DeepCT	0.578▽	0.714	0.329▽	11.02
DocT5Query	0.648	0.799	0.405	11.76
DeepImpact	**0.695**	**0.863**	**0.456**	51.23
TREC 2020				
BM25	0.483▽	0.659▽	0.286▽	14.67
DeepCT	0.550▽	0.705	0.349▽	12.00
DocT5Query	0.619	0.742	0.408	15.51
DeepImpact	**0.651**	**0.820**	**0.426**	58.00

Table 3: Effectiveness metrics and mean response time (MRT, in ms) using several re-ranking techniques on MSMARCO Dev Queries, TREC 2019 queries, and TREC 2020 queries. ▽ denotes a significant difference viz. ColBERT E2E

Strategy	NDCG@10	MRR@10	MRT
MSMARCO Dev Queries			
DeepImpact + EPIC	0.367▽	0.303▽	194.64
DeepImpact + ColBERT	**0.425**	**0.362**	81.00
ColBERT E2E	0.424	0.361	380.97
TREC 2019			
DeepImpact + EPIC	0.711	**0.880**	191.23
DeepImpact + ColBERT	**0.722**	0.826	73.29
ColBERT E2E	0.694	0.826	370.98
TREC 2020			
DeepImpact + EPIC	0.646	0.773	196.00
DeepImpact + ColBERT	**0.691**	**0.781**	79.84
ColBERT E2E	0.676	0.776	364.82

Re-ranking evaluation. Table 3 shows the effect of re-ranking the top 1000 candidates produced by DeepImpact using two complex re-rankers, EPIC and ColBERT. The table also shows, as a comparison, the performance obtained by ColBERT when used end-to-end by employing ANN search as the first-stage retrieval mechanism. DeepImpact followed by a ColBERT re-ranker obtains higher effectiveness values than ColBERT E2E on all query sets. Moreover, DeepImpact + ColBERT exhibits a $4.4\times - 5.1\times$ speedup w.r.t. ColBERT E2E.

First-stage cutoff evaluation. DeepImpact is able to achieve statistically significant higher recall than all the compared methods (with one single exception at cutoff 1000). In particular, Table 4 shows that the gap with the other methods is greater with smaller cutoff values, which reduces the re-ranking cost and thus could enable the use of more complex pairwise ranking models, such as DuoBERT [20]. In re-ranking, DeepImpact outperforms all other methods at cutoff 10. Moreover, it outperforms DeepCT on all cutoff values except 1000, and it is comparable with DocT5Query.

Table 4: First-stage recall and re-rank-stage MRR@10 using ColBERT at several first-stage cutoffs for different candidate generation methods w.r.t. MSMARCO Dev Queries. The symbol ▽ denotes a significant difference viz. DeepImpact

k	BM25	DeepCT	DocT5Query	DeepImpact
Recall (first stage)				
10	0.394▽	0.484▽	0.542▽	**0.584**
20	0.483▽	0.577▽	0.649▽	**0.680**
200	0.739▽	0.816▽	0.869▽	**0.882**
1000	0.858▽	0.910▽	0.947	**0.948**
MRR@10 (re-rank stage)				
10	0.270▽	0.302▽	0.341▽	**0.350**
20	0.299▽	0.322▽	0.355	**0.357**
200	0.343▽	0.353▽	**0.361**	**0.361**
1000	0.355▽	0.360	**0.362**	**0.362**

4 CONCLUSIONS AND FUTURE WORK

In this paper, we introduced DeepImpact, a new first-stage retrieval method that leverages a combination of a traditional inverted indexes and contextualized language models for efficient retrieval. By estimating semantic importance, DeepImpact produces a single-value impact score for each tokens of a document collection. Our results show that DeepImpact outperforms every inverted-index based baseline, in some cases even matching the effectiveness of more complex neural retrieval approaches such as ColBERT. Furthermore, when ColBERT is used to re-rank candidates retrieved by DeepImpact instead of approximate nearest neighbor, we find a dramatic reduction of query processing latency, and a more modest improvement in effectiveness of the whole pipeline. Future work will focus on further enhancing the underlying model. First, we would like to experiment with more relaxed matching conditions, instead of exact match, between the query-document terms. Second, we believe that we could improve further term expansion with more sophisticated techniques. Finally, we plan to investigate how changing the distribution of impact scores affects query processing algorithms such as MaxScore, and how we can address this issue.

Acknowledgments: This research was partially supported by NSF Grant IIS-1718680 and CAREER grant CNS-1651570, affiliate members and other supporters of the Stanford DAWN project—Ant Financial, Facebook, Google, Infosys, NEC, and VMware—as well as Cisco and SAP, the Italian Ministry of Education and Research (MIUR) in the framework of the CrossLab project (Departments of Excellence), and by the University of Pisa in the framework of the AUTENS project (Sustainable Energy Autarky). We would like to thank Matei Zaharia for insightful discussions and feedback. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] Yang Bai, Xiaoguang Li, Gang Wang, Chaoliang Zhang, Lifeng Shang, Jun Xu, Zhaowei Wang, Fangshan Wang, and Qun Liu. 2020. SparTerm: Learning Term-based Sparse Representation for Fast Text Retrieval. *arXiv preprint arXiv:2010.00768* (2020).

[2] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2021. Overview of the TREC 2020 deep learning track. *Preprint arXiv:2102.07662* (2021).

[3] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M Voorhees. 2020. Overview of the trec 2019 deep learning track. *Preprint arXiv:2003.07820* (2020).

[4] Zhuyun Dai and Jamie Callan. 2019. Context-aware sentence/passage term importance estimation for first stage retrieval. *Preprint arXiv:1910.10687* (2019).

[5] Zhuyun Dai and Jamie Callan. 2020. Context-aware document term weighting for ad-hoc search. In *Proc. WWW*. 1897–1907.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *Preprint arXiv:1810.04805* (2018).

[7] Sebastian Hofstätter and Allan Hanbury. 2019. Let's measure run time! Extending the IR replicability infrastructure to include performance aspects. In *OSIRRC@SIGIR*.

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

[9] Omar Khattab and Matei Zaharia. 2020. ColBERT: Efficient and effective passage search via contextualized late interaction over BERT. In *Proc. SIGIR*. 39–48.

[10] Jimmy Lin, Joel Mackenzie, Chris Kamphuis, Craig Macdonald, Antonio Mallia, Michał Siedlaczek, Andrew Trotman, and Arjen de Vries. 2020. Supporting interoperability between open-source search engines with the common index file format. In *Proc. SIGIR*. 2149–2152.

[11] Sean MacAvaney. 2020. OpenNIR: A Complete Neural Ad-Hoc Ranking Pipeline. In *WSDM 2020*.

[12] Sean MacAvaney, Franco Maria Nardini, Raffaele Perego, Nicola Tonellotto, Nazli Goharian, and Ophir Frieder. 2020. Efficient Document Re-Ranking for Transformers by Precomputing Term Representations. In *Proc. SIGIR*. 49–58.

[13] Sean MacAvaney, Franco Maria Nardini, Raffaele Perego, Nicola Tonellotto, Nazli Goharian, and Ophir Frieder. 2020. Expansion via Prediction of Importance with Contextualization. In *Proc. SIGIR*. 1573–1576.

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

[15] J. Mackenzie, Z. Dai, L. Gallagher, and J. Callan. 2020. Efficiency Implications of Term Weighting for Passage Retrieval. In *Proc. SIGIR*. 1821–1824.

[16] Antonio Mallia, Michal Siedlaczek, Joel Mackenzie, and Torsten Suel. 2019. PISA: performant indexes and search for academia. *OSIRRC@SIGIR* (2019).

[17] 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. (2016).

[18] Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage Re-ranking with BERT. arXiv:Preprint arXiv:1901.04085

[19] Rodrigo Nogueira and Jimmy Lin. 2019. From doc2query to docTTTTTquery. *Online preprint* (2019).

[20] Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. 2019. Multi-stage document ranking with BERT. *arXiv preprint arXiv:1910.14424* (2019).

[21] Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. 2019. Document expansion by query prediction. *Preprint arXiv:1904.08375* (2019).

[22] Howard Turtle and James Flood. 1995. Query evaluation: strategies and optimizations. *Information Processing & Management* 31, 6 (1995), 831–850.

[23] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. *Preprint arXiv:2007.00808* (2020).

[24] Peilin Yang, Hui Fang, and Jimmy Lin. 2017. Anserini: Enabling the use of Lucene for information retrieval research. In *Proc. SIGIR*. 1253–1256.

[25] Le Zhao. 2012. *Modeling and solving term mismatch for full-text retrieval.* Ph.D. Dissertation. Carnegie Mellon University.