

# Expansion via Prediction of Importance with Contextualization

Sean MacAvaney  
IR Lab, Georgetown University, USA  
sean@ir.cs.georgetown.edu

Franco Maria Nardini  
ISTI-CNR, Pisa, Italy  
francomaria.nardini@isti.cnr.it

Raffaele Perego  
ISTI-CNR, Pisa, Italy  
raffaele.perego@isti.cnr.it

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

Nazli Goharian  
IR Lab, Georgetown University, USA  
nazli@ir.cs.georgetown.edu

Ophir Frieder  
IR Lab, Georgetown University, USA  
ophir@ir.cs.georgetown.edu

## ABSTRACT

The identification of relevance with little textual context is a primary challenge in passage retrieval. We address this problem with a representation-based ranking approach that: (1) explicitly models the importance of each term using a contextualized language model; (2) performs passage expansion by propagating the importance to similar terms; and (3) grounds the representations in the lexicon, making them interpretable. Passage representations can be pre-computed at index time to reduce query-time latency. We call our approach EPIC (Expansion via Prediction of Importance with Contextualization). We show that EPIC significantly outperforms prior importance-modeling and document expansion approaches. We also observe that the performance is additive with the current leading first-stage retrieval methods, further narrowing the gap between inexpensive and cost-prohibitive passage ranking approaches. Specifically, EPIC achieves a MRR@10 of 0.304 on the MS-MARCO passage ranking dataset with 78ms average query latency on commodity hardware. We also find that the latency is further reduced to 68ms by pruning document representations, with virtually no difference in effectiveness.

## ACM Reference Format:

Sean MacAvaney, Franco Maria Nardini, Raffaele Perego, Nicola Tonellotto, Nazli Goharian, and Ophir Frieder. 2020. Expansion via Prediction of Importance with Contextualization. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20)*, July 25–30, 2020, Virtual Event, China. ACM, New York, NY, USA, 5 pages. <https://doi.org/10.1145/3397271.3401262>

## 1 INTRODUCTION

Passage retrieval is fundamentally burdened by short passages. While document retrieval systems can rely on signals such as term frequency to estimate the importance of a given term in a document, passages usually do not have this benefit. Consequently, traditional retrieval approaches often perform poorly at passage retrieval. Supervised deep learning approaches—in particular, those that make use of pretrained contextualized language models—have successfully overcome this limitation by making use of general language

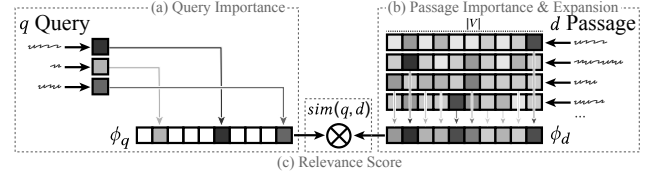


Figure 1: Overview of EPIC.

characteristics [1, 6]. However, these approaches have a substantial computational burden, which can make them impractical [7, 14].

We propose a new approach for passage retrieval that performs modeling of term importance (i.e., salience) and expansion over a contextualized language model to build query and document representations. We call this approach **EPIC** (Expansion via Prediction of Importance with Contextualization). At query time, EPIC can be employed as an inexpensive re-ranking method because document representations can be pre-computed at index time. EPIC improves upon the prior state of the art on the MS-MARCO passage ranking dataset by substantially narrowing the effectiveness gap between practical approaches with subsecond retrieval times and those that are considerably more expensive, e.g., those using BERT as a re-ranker. Furthermore, the proposed representations are interpretable because the dimensions of the representation directly correspond to the terms in the lexicon. An overview is shown in Fig. 1.

Neural re-ranking approaches can generally be characterized as either representation-based or interaction-based [5]. Representation-based models, like ours, build representations of a query and passage independently and then compare these representations to calculate a relevance score. These are beneficial because one can compute document representations at index time to reduce the query-time cost. Interaction-based models combine signals from the query and the document at query time to compute the relevance score [13]. The Duet model [12] aims to achieve low query-time latency by combining signals from both a representation-based and an interaction-based model. However, this approach substantially under-performs the latest pure interaction-based approaches such as the one in [13]. TK [8] attempts to bridge this performance gap by using a smaller transformer network, but still utilizes an interaction-based approach which itself adds considerable computational overhead. Finally, other interesting proposals have investigated alternative approaches for offloading computational cost to index time. Doc2query [15] and docTTTTTquery [14] add important context to otherwise short documents by using a sequence-to-sequence model to predict additional terms to add to the document. DeepCT-Index [2] models an importance score for each term

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

SIGIR '20, July 25–30, 2020, Virtual Event, China

© 2020 Association for Computing Machinery.

ACM ISBN 978-1-4503-8016-4/20/07...\$15.00

<https://doi.org/10.1145/3397271.3401262>

in the document and replaces the term frequency values in the inverted index with these scores. Unlike these approaches, EPIC models query/document term importance and performs document expansion. We found that it can be employed as an inexpensive yet effective re-ranking model; the impact on query time latency can be as low as an additional 5ms per query (for a total of 68ms).

In summary, the novel contributions presented are:

- We propose a new representation-based ranking model that is grounded in the lexicon.
- We show that this model can improve ranking effectiveness for passage ranking, with a minimal impact on query-time latency.
- We show that the model yields interpretable representations of both the query and the document.
- We show that latency and storage requirements of our approach can be reduced by pruning the document representations.
- For reproducibility, our code is integrated into OpenNIR [10], with instructions and trained models available at: <https://github.com/Georgetown-IR-Lab/epic-neural-ir>.

## 2 MODEL

**Overview and notation.** Our model follows the representation-focused neural ranking paradigm. That is, we train a model to generate a query and document<sup>1</sup> representation in a given fixed-length vector space, and produce a ranking score by computing a similarity score between the two representations.

Assume that queries and documents are composed by sequences of terms taken from a vocabulary  $V$ . Any sequence of terms, either a query or a document, is firstly represented as a sequence of vectors using a contextualized language model like BERT [3]. More formally, let  $f : V^n \rightarrow \mathbb{R}^{n \times e}$  denote such a function associating an input sequence  $s$  of  $n$  terms  $t_1, \dots, t_n$  to their  $n$  embeddings  $f_1(s), \dots, f_n(s)$ , where  $f_i(s) \in \mathbb{R}^e$  and  $e$  is the size of the embedding. So, a  $n$ -term query  $q$  is represented with the  $n$  embeddings  $f_1(q), \dots, f_n(q)$ , and a  $m$ -term document  $d$  is represented with  $m$  embeddings  $f_1(d), \dots, f_m(d)$ . Given the embeddings for queries and documents, we now illustrate the process for constructing query representations, document representations, the final query-document similarity score.

**Query representation.** A query  $q$  is represented as a sparse vector  $\phi_q \in \mathbb{R}^{|V|}$  (Fig. 1 (a)). The elements of  $\phi_q$  that correspond to terms not in the query are set to 0. For each term  $t_i$  appearing in the  $t_1, \dots, t_n$  terms of the query  $q$ , the corresponding element  $\phi_q(t_i)$  is equal to the importance  $w_q(t_i)$  of the term w.r.t. the query

$$w_q(t_i) = \ln \left( 1 + \text{softplus}(\theta_1^\top f_i(q)) \right), \quad (1)$$

where  $\theta_1 \in \mathbb{R}^e$  is a vector of learned parameters. The  $\text{softplus}(\cdot)$  function is defined as  $\text{softplus}(x) = \ln(1 + e^x)$ . The use of  $\text{softplus}$  ensures that no terms have a negative importance score, while imposing no upper bound. The logarithm prevents individual terms from dominating. When a term appears more than once in a query, the corresponding value of  $\phi_q$  sums up all contributions. The elements of the query representation encode the importance of the terms w.r.t. the query. This approach allows the query representation model to learn to assign higher weights to the query terms that

are most important to match given the textual context. Note that the number of elements in the representation is equal to the number of query terms; thus the query processing time is proportional to the number of query terms [17].

**Document representation.** A document  $d$  is represented as a dense vector  $\phi_d \in \mathbb{R}^{|V|}$  (Fig. 1 (b)). Firstly, to perform document expansion, each  $e$ -dimensional term embedding  $f_j(d)$  is projected into a  $|V|$ -dimensional vector space, i.e.,  $\psi_j : f_j(d) \mapsto \Theta_2 f_j(d)$ , where  $\Theta_2 \in \mathbb{R}^{|V| \times e}$  is a matrix of learned parameters. Note that  $\psi_j \in \mathbb{R}^{|V|}$ , and let  $\psi_j(\tau)$  denote the entry of this vector corresponding to term  $\tau \in V$ . Secondly, the importance  $w_d(t_j)$  of the terms w.r.t. the document is computed as in Eq (1):

$$w_d(t_j) = \ln \left( 1 + \text{softplus}(\theta_3^\top f_j(d)) \right), \quad (2)$$

where  $\theta_3 \in \mathbb{R}^e$  is a vector of learned parameters. Thirdly, we compute a factor representing the overall quality  $c(d)$  of the document

$$c(d) = \text{sigmoid}(\theta_4^\top d_{[\text{CLS}]}), \quad (3)$$

where  $\theta_4 \in \mathbb{R}^e$  is a vector of learned parameters, and  $d_{[\text{CLS}]} \in \mathbb{R}^e$  is the embedding produced by the contextualized language model's classification mechanism. We find that this factor helps give poor-quality documents lower values overall. The  $\text{sigmoid}(\cdot)$  function is defined as:  $\text{sigmoid}(x) = \frac{1}{1+e^{-x}}$ . Finally, for each term  $\tau$  appearing in the vocabulary, the corresponding element of the document representation  $\phi_d(\tau)$  is defined as:

$$\phi_d(\tau) = c(d) \max_{t_j \in d} (w_d(t_j) \psi_j(\tau)). \quad (4)$$

This step takes the maximum score for each term in the vocabulary generated by any term in the document. Since they do not rely on the query, these representations can be computed at index time.

**Similarity measure.** We use the dot product to compute the similarity between the query and document vectors (Fig. 1 (c)), i.e.,

$$\text{sim}(q, d) = \phi_q^\top \phi_d = \sum_{\tau \in V} \phi_q(\tau) \phi_d(\tau). \quad (5)$$

## 3 EXPERIMENTAL EVALUATION

We conduct experiments using the MS-MARCO passage ranking dataset (full ranking setting).<sup>2</sup> This dataset consists of approximately 1 million natural-language questions gathered from a query log (average length: 7.5 terms, stddev: 3.1), and 8.8 million candidate answer passages (avg length: 73.1, stddev: 28.4). The dataset is shallowly-annotated. Annotators were asked to write a natural-language answer to the given question using a set of candidate answers from a commercial search engine. The annotators were asked to indicate which (if any) of the passages contributed to their answers, which are then treated as relevant to the question. This results in 0.7 judgments per query on average (1.1 judgments per query of the 62% that have an answer). Thus, this dataset has a lot of variation in queries, making it suitable for training neural ranking methods. Although this dataset is limited by the method of construction, the performance on these shallow judgments correlate well with those conducted on a deeply-judged subset [1].

<sup>1</sup>For ease of notation, we refer to passages as documents.

<sup>2</sup><https://microsoft.github.io/msmarco/>

**Table 1: Effectiveness and efficiency of our approach compared to a variety of baselines. The values in *italics* represent a good trade-off between effectiveness and query latency. The value marked with \* was reported by [14].**

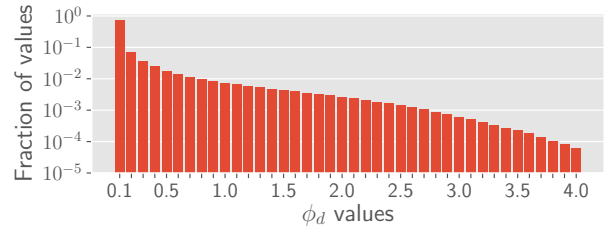
Method	MS-Marco Dev MRR@10	Latency ms/query
<b>Single-Stage Ranking</b>		
BM25 (from Anserini [18])	0.198	21
doc2query [15]	0.218	48
DeepCT-Index [2]	0.243	15
docTTTTTquery [14]	0.277	63
<b>Representation-based Re-Ranking</b>		
EPIC + BM25 (ours)	0.273	106
pruned $r = 2000$	0.273	104
pruned $r = 1000$	0.272	48
EPIC + docTTTTTquery (ours)	0.304	78
pruned $r = 2000$	0.304	77
pruned $r = 1000$	0.303	68
<b>Other Re-Ranking</b>		
Duet (v2, ensemble) [12]	0.252	440
BM25 + TK (1 layer) [8]	0.303	445
BM25 + TK (3 layers) [8]	0.314	640
BM25 + BERT (large) [13]	0.365	3,500*

**Training.** We train our model using the official MS-MARCO sequence of training triples (query, relevant passage, presumed non-relevant passage) using cross-entropy loss. We use BERT-base [3] as the contextualized language model, as it was shown to be an effective foundation for various ranking techniques [2, 11, 13]. We set the dimensionality  $|V|$  of our representations to the size of the BERT-base word-piece vocabulary ( $d=30,522$ ). The embedding size is instead  $e = 768$ .  $\Theta_2$  is initialized to the pre-trained masked language model prediction matrix; all other added parameters are randomly initialized. Errors are back-propagated through the entire BERT model with a learning rate of  $2 \times 10^{-5}$  with the Adam optimizer [9]. We train in batches of 16 triples using gradient accumulation, and we evaluate the model on a validation set of 200 random queries from the development set every 512 triples. The optimal training iteration and re-ranking cutoff threshold is selected using this validation set. We roll back to the top-performing model after 20 consecutive iterations (training iteration 42) without improvement to Mean Reciprocal Rank at 10 (MRR@10).

**Baselines and Evaluation.** We test our approach by re-ranking the results from several first-stage rankers. We report the performance using MRR@10, the official evaluation metric, on the MS-MARCO passage ranking Dev set. We measure significance using a paired t-test at  $p < 0.01$ . We compare the performance of our approach with the following baselines:

- BM25 retrieval from a Porter-stemmed Anserini [18] index using default settings.<sup>3</sup>
- DeepCT-Index [2], a model which predicts document term importance scores, and replaces the term frequency values with these importance scores for first-stage retrieval.

<sup>3</sup>We observe that the default settings outperform the BM25 results reported elsewhere and on the official leaderboard (e.g., [15]).



**Figure 2: Frequencies of values (log scale) appearing in the document representations. In this figure, values are rounded up to the nearest decimal value.**

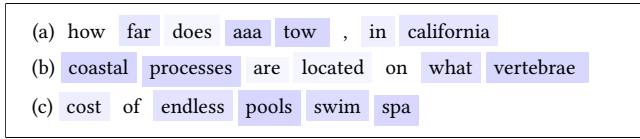
- doc2query [15] and docTTTTTquery [14], document expansion approaches which predict additional terms to add to the document via a sequence-to-sequence transformer model. These terms are then indexed and used for retrieval using BM25. The docTTTTTquery model uses a pre-trained T5 model [16].
- Duet [12], a hybrid representation- and interaction-focused model. We include the top Duet variant on the MS-MARCO leaderboard (version 2, ensemble) to compare with another model that utilizes query and document representations.
- TK [8], a contextualized interaction-based model, focused on minimizing query time. We report results from [8] with the optimal re-ranking threshold and measure end-to-end latency on our hardware.
- BERT Large [13], an expensive contextualized language model-based re-ranker. This approach differs from ours in that it models the query and passage jointly at query time, and uses the model’s classification mechanism for ranking.

We also measure query latency over the entire retrieval and re-ranking process. The experiments were conducted on commodity hardware equipped with an AMD Ryzen 3.9GHz processor, 64GiB DDR4 memory, a GeForce GTX 1080ti GPU, and a SSD drive. We report the latency of each method as the average execution time (in milliseconds) of 1000 queries from the Dev set after an initial 1000 queries is used to warm up the cache. First-stage retrieval is conducted with Anserini [18].

**Ranking effectiveness.** We report the effectiveness of our approach in terms of MMR@10 in Table 1. When re-ranking BM25 results, our approach substantially outperforms doc2query and DeepCT-Index. Moreover, it performs comparably to docTTTTTquery (0.273 compared to 0.277, no statistically significant difference). More importantly, we observe that the improvements of our approach and docTTTTTquery are additive as we achieve a MRR@10 of 0.304 when used in combination. This is a statistically significant improvement, and substantially narrows the gap between approaches with low query-time latency and those that trade off latency of effectiveness (e.g., BERT Large).

To test whether EPIC is effective on other passage ranking tasks as well, we test on the TREC CAR passage ranking benchmark [4]. When trained and evaluated on the 2017 dataset (automatic judgments) with BM25, the MRR increases from 0.235 to 0.353. This also outperforms the DeepCT performance reported by [2] of 0.332.

**Effect of document representation pruning.** For document vectors, we observe that the vast majority of values are very low (approximately 74% have a value of 0.1 or below, see Fig. 2). This suggests that many of the values can be pruned with little impact



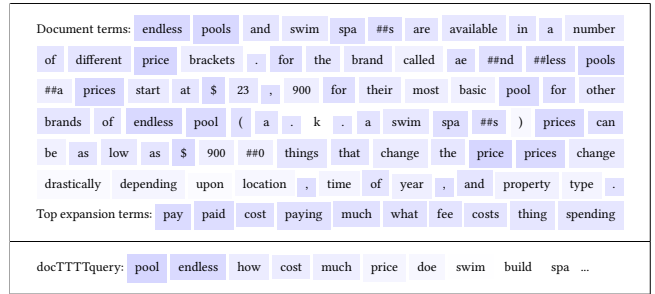
**Figure 3: Relative importance scores of sample queries. Darker colors correspond to higher weights in the query representation.**

on the overall performance. This is desirable because pruning can substantially reduce the storage required for the document representations. To test this, we apply our method keeping only the top  $r$  values for each document. We show the effectiveness and efficiency of  $r = 2000$  (reduces vocabulary by 93.4%) and  $r = 1000$  (96.7%) in Table 1. We observe that the vectors can be pruned to  $r = 1000$  with virtually no difference in ranking effectiveness (differences not statistically significant). We also tested with lower values of  $r$ , but found that the effectiveness drops off considerably by  $r = 100$  (0.241 and 0.285 for BM25 and docTTTTTquery, respectively).

**Ranking efficiency.** We find that EPIC can be implemented with a minimal impact on query-time latency. On average, the computation of the query representation takes 18ms on GPU and 51ms on CPU. Since this initial stage retrieval does not use our query representation, it is computed in parallel with the initial retrieval, which reduces the impact on latency. The similarity measure consistently takes approximately 1ms per query (both on CPU and GPU), with the remainder of the time spent retrieving document representations from disk. Interestingly, we observe that the latency of EPIC BM25 is higher than EPIC docTTTTTquery. This is because when re-ranking docTTTTTquery results, a lower re-ranking cutoff threshold is needed than for BM25. This further underscores the importance of using an effective first-stage ranker. When using pruning at  $r = 1000$ , the computational overhead can be substantially reduced. Specifically, we find that EPIC only adds a 5ms overhead per query to docTTTTTquery, while yielding a significant improvement in effectiveness. With pruning at  $r = 1000$ , EPIC BM25 performs comparably with docTTTTTquery with a  $1.3\times$  speedup.

**Cost of pre-computing.** We find that document vectors can be pre-computed for the MS-MARCO collection in approximately 14 hours on a single commodity GPU (GeForce GTX 1080ti). This is considerably less expensive than docTTTTTquery, which takes approximately 40 hours on a Google TPU (v3). When stored as half-precision (16-bit) floating point values, the vector for each document uses approximately 60KiB, regardless of the document length. This results in a total storage burden of approximately 500GiB for the entire collection. Pruning the collection to  $r = 1000$  (which has minimal impact on ranking effectiveness) reduces the storage burden of each document to 3.9KiB (using 16-bit integer indices) and total storage to 34 GiB.

**Interpretability of representations.** A benefit of our approach is that the dimensions of the representation correspond to terms in the lexicon, allowing the representations to be easily inspected. In Fig. 3, we present the relative scores for sample queries from MS-MARCO. We observe that the model is generally able to pick up on the terms that match intuitions of term importance. For instance, (a) gives highest scores to *california*, *aaa* (American Automobile



**Figure 4: Relative representation values of terms that appear in a sample document. Alongside terms that appeared in the passage, the top ‘expansion’ terms are also shown. For reference, the most frequent terms produced by docTTTTTquery are also given, weighted by term frequency.**

Association), and *tow*. These three terms are good candidates for a keyword-based query with the same query intent. This approach does not necessarily just remove stop words; in (b) *what* is assigned a relatively high score. We provide an example of document vector importance scores in Fig. 4. Because the document vector is dense, the figure only shows the terms that appear directly in the document and other top-scoring terms. Notice that terms related to *price*, *endless*, *pool(s)*, and *cost* are assigned the highest scores. In this case, the expansion of the term *cost* was critical for properly scoring this document, as a relevant query is *cost of endless pools/spas*. Although the terms that docTTTTTquery generate for this document are similar, the continuous values generated by EPIC paid off in a higher MRR@10 score for the query “*cost of endless pools/swim spa*” (a relevant question for this passage).

## 4 CONCLUSION

We demonstrated an effective and inexpensive technique for re-ranking passages based on lexicon-grounded representations generated from contextualized language models. This work advances the art by further approaching fully BERT-based re-ranking performance, while providing low query-time latency and easy interpretability of representations. We also find that pruning can be an effective technique for reducing query latency and the size of the pre-computed passage representations without sacrificing effectiveness. Future work can investigate how well this approach generalizes to document retrieval.

## ACKNOWLEDGMENTS

Work partially supported by the ARCS Foundation. Work partially supported by the Italian Ministry of Education and Research (MIUR) in the framework of the CrossLab project (Departments of Excellence). Work partially supported by the BIGDATAGRAPHES project funded by the EU Horizon 2020 research and innovation programme under grant agreement No. 780751, and by the OK-INSAD project funded by the Italian Ministry of Education and Research (MIUR) under grant agreement No. ARS01\_00917.

## REFERENCES

- [1] Nick Craswell, Bhaskar Mitra, and Daniel Campos. 2019. Overview of the TREC 2019 Deep Learning Track. In *TREC*.



- [2] Zhuyun Dai and Jamie Callan. 2019. Context-aware sentence/passage term importance estimation for first stage retrieval. *arXiv* (2019).
- [3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*.
- [4] Laura Dietz, Ben Gamari, Jeff Dalton, and Nick Craswell. 2017. TREC Complex Answer Retrieval Overview. In *TREC*.
- [5] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. 2016. A Deep Relevance Matching Model for Ad-hoc Retrieval. In *CIKM*.
- [6] Helia Hashemi, Mohammad Aliannejadi, Hamed Zamani, and W. Bruce Croft. 2019. ANTIQUE: A Non-Factoid Question Answering Benchmark. *arXiv* (2019).
- [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, Markus Zlabinger, and Allan Hanbury. 2020. Interpretable & Time-Budget-Constrained Contextualization for Re-Ranking. *arXiv* (2020).
- [9] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.
- [10] Sean MacAvaney. 2020. OpenNIR: A Complete Neural Ad-Hoc Ranking Pipeline. In *WSDM*.
- [11] Sean MacAvaney, Andrew Yates, Arman Cohan, and Nazli Goharian. 2019. CEDR: Contextualized Embeddings for Document Ranking. In *SIGIR*.
- [12] Bhaskar Mitra and Nick Craswell. 2019. An updated duet model for passage re-ranking. *arXiv* (2019).
- [13] Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage Re-ranking with BERT. *arXiv* (2019).
- [14] Rodrigo Nogueira and Jimmy Lin. 2019. From doc2query to docTTTTTquery. (2019). [https://cs.uwaterloo.ca/~jimmylin/publications/Nogueira\\_Lin\\_2019\\_docTTTTTquery-v2.pdf](https://cs.uwaterloo.ca/~jimmylin/publications/Nogueira_Lin_2019_docTTTTTquery-v2.pdf)
- [15] Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. 2019. Document expansion by query prediction. *arXiv* (2019).
- [16] Colin Raffel, Noam Shazeer, Adam Kaleo Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *arXiv* (2019).
- [17] Nicola Tonello, Craig Macdonald, and Iadh Ounis. 2018. Efficient Query Processing for Scalable Web Search. *Foundations and Trends in Information Retrieval* 12, 4–5 (2018), 319–492. <http://dx.doi.org/10.1561/15000000057>
- [18] Peilin Yang, Hui Fang, and Jimmy Lin. 2018. Anserini: Reproducible Ranking Baselines Using Lucene. *J. Data and Information Quality* 10 (2018), 16:1–16:20.