Revisiting Document Expansion and Filtering for Effective First-Stage Retrieval

Watheq Mansour
The University of Queensland
Brisbane, Australia
w.mansour@uq.edu.au

Shengyao Zhuang
CSIRO
Brisbane, Australia
shengyao.zhuang@csiro.au

Guido Zuccon
The University of Queensland
Brisbane, Australia
g.zuccon@uq.edu.au

Joel Mackenzie
The University of Queensland
Brisbane, Australia
joel.mackenzie@uq.edu.au

ABSTRACT

Document expansion is a technique that aims to reduce the likelihood of term mismatch by augmenting documents with related terms or queries. Doc2Query *minus minus* (Doc2Query--) represents an extension to the expansion process that uses a neural model to identify and remove expansions that may not be relevant to the given document, thereby increasing the quality of the ranking while simultaneously reducing the amount of augmented data. In this work, we conduct a detailed reproducibility study of Doc2Query-- to better understand the trade-offs inherent to document expansion and filtering mechanisms. After successfully reproducing the best-performing method from the Doc2Query-- family, we show that filtering actually harms recall-based metrics on various test collections. Next, we explore whether the two-stage "generate-then-filter" process can be replaced with a single generation phase via reinforcement learning. Finally, we extend our experimentation to learned sparse retrieval models and demonstrate that filtering is not helpful when term weights can be learned. Overall, our work provides a deeper understanding of the behaviour and characteristics of common document expansion mechanisms, and paves the way for developing more efficient yet effective augmentation models.

CCS CONCEPTS

• **Information systems** → **Retrieval models and ranking**; **Document representation**; **Retrieval effectiveness**.

KEYWORDS

Document expansion, query filtering, reproducibility

ACM Reference Format:
Watheq Mansour, Shengyao Zhuang, Guido Zuccon, and Joel Mackenzie. 2024. Revisiting Document Expansion and Filtering for Effective First-Stage Retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '24), July 14–18, 2024, Washington, DC, USA*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3626772.3657850

SIGIR '24, July 14–18, 2024, Washington, DC, USA

ACM ISBN 979-8-4007-0431-4/24/07…$15.00
https://doi.org/10.1145/3626772.3657850

1 INTRODUCTION

The *vocabulary mismatch problem* — where the terms provided by a user query do not match the terms used in some relevant documents — is a key problem in Information Retrieval (IR). Decades of research have explored methods for alleviating vocabulary mismatch, with the most common family of techniques being *query expansion* methods, where queries are supplemented with additional terms to improve both recall (for retrieval) and precision (for ranking) [1, 5]. Recently, however, there has been an interest in *document expansion*, where each document is augmented with additional terms or queries as an alternative method for alleviating term mismatch [58, 59].

The most common *document expansion* techniques generate *queries* that may be answered by a given document, which are then augmented with the original document text prior to indexing. Intuitively, document expansion can help to combat vocabulary mismatch by incorporating terms that would not otherwise be present in the text. Document expansion models have also been shown to increase effectiveness by *boosting* the relative term frequency of relevant terms within a given passage [17, 40, 59].

As a concrete example of document expansion, consider Table 1, which shows a passage and a series of expansion queries from the T5-based [63] Doc2Query model [58]. Without expansion, a user query such as "`which font type for math`" would only lexically match on the terms "`font`" and "`math`" (appearing three times, and once, respectively). However, after appending $N = 80$ expanded queries to the passage, those two terms have their frequencies increased to 64 and 23, respectively. In addition, the terms "`which`", "`type`", and "`of`" — which were not in the original passage at all — now have term frequencies of 3, 6, and 9. Although some of these terms are stopwords, it is easy to see the advantage of document expansion. However, document expansion has some obvious disadvantages, too. Firstly, methods like Doc2Query that rely on sequence-to-sequence models like T5 [63] require a significant amount of additional offline computation, as each document must have an expensive series of inferences to compute the expansion queries [66, 87]. Secondly, these expansion queries naturally add to the size of the index, resulting in a larger space occupancy and, in some cases, increased retrieval latency [27, 51, 58, 59].

Recently, Gospodinov et al. [27] offered a novel perspective on a less obvious issue with document expansion: document expansion models may *hallucinate* queries that are not actually answered

by the given document. To resolve this issue, they proposed an approach called Doc2Query *minus minus* (Doc2Query--) that scores each document/expansion pair using a cross-encoder or bi-encoder model and filters out those with low scores, aiming to reduce the amount of hallucinated or otherwise non-relevant expansions.

In this work, we conduct a detailed reproducibility study of Doc2Query--. Beyond our interest in exploring this interesting research area and validating the work of Gospodinov et al. [27], our motivation for conducting this study is fourfold:

(1) **Holistic Evaluation:** The original work focused on end-to-end retrieval, where the final results lists were evaluated using shallow precision-oriented metrics like NDCG@10 or Reciprocal Rank. However, lexical retrieval systems are commonly used as first-stage retrieval systems, generating a large set of candidate documents as input to a more expensive ranking model [40, 77]. In such re-ranking pipelines, the first stage ought to maximize recall. As such, we are motivated to revisit the evaluation from a recall-oriented perspective to provide a more holistic view of the trade-offs inherent in document expansion and filtering.
(2) **Lexical Matching:** The rationale that led Gospodinov et al. to explore filtering mechanisms is to remove poor queries that were generated by sequence-to-sequence models[1] like T5 [63], as these models are prone to *hallucination*. However, the retrieval systems used in the original work are based on *lexical matching*. As such, the main risk present from such hallucinations is that a non-relevant document may be retrieved for a (user) query that matches the (hallucinated) expansion query. On the other hand, it is possible that these non-relevant queries could actually be *helpful* for boosting the relevance signal in their target passages, even if the query itself is not relevant or correct. Thus, we are motivated to further understand where the improvements arise due to filtering and to explore how "bad" expansions impact the effectiveness of lexical ranking models.
(3) **Learned Impact Scores:** While the risk of including expansion terms that do not accurately reflect the content of the document must be acknowledged, current state-of-the-art lexical models apply neural networks to determine the appropriate weight to give to each term, known as *learned sparse retrieval* [56]. We hypothesize that learned sparse retrieval models may already implicitly filter non-relevant terms by assigning them a low weight, which motivates us to experiment with document expansion and filtering in the context of these models.
(4) **Generating Better Queries:** Another promising avenue for exploration is the combination of the generation and filtering phases into a single process; by generating better queries in the first place, filtering may not be necessary at all. To this end, we experiment with reinforcement learning to train a query generator that is biased towards generating queries with high filter scores.

Based on these focus areas, our main contributions are as follows:

[1]Those typically used for document expansion.

Table 1: Original passage (top), best five expansions after de-duplication (middle), and worst five expansions (bottom) for `MSMARCO-v1` passage `1320289` (Doc2Query expansion) according to the ELECTRA model.

If you are into typesetting mathematics, you might also be interested in the Latin Modern Math fonts in the OpenType format. The Family. The Latin Modern fonts are derived from the famous Computer Modern fonts designed by Donald E. Knuth and described in Volume E of his Computers & Typesetting series.
who designed latin modern math fonts, who designed the latin modern font, who designed the modern fonts, latin modern fonts, who made latin modern font
who is latin font?, what font is called computer modern slang, what is the latin text font, what is lm modern font, where did the word math come from

- We conduct a reproducibility and generalizability study of Doc2Query-- and provide a deeper analysis of the importance of document expansion and filtering on both precision and recall-oriented retrieval settings, including in both retrieval and re-ranking contexts;
- We explore a quality-aware query generation model in an effort to avoid the expensive two-stage generate-then-filter process; and
- We extend the Doc2Query-- framework to learned sparse retrieval models to determine whether filtering can provide additive improvements over learned impact scores.

Overall, our reproducibility study provides a more holistic view of document expansion and filtering mechanisms, their efficiency and effectiveness characteristics, and their application to wider contexts beyond end-to-end BM25-based passage retrieval. Our experimental code and resources are available: https://github.com/175edda-sps/d2qminus-repro.

2 BACKGROUND

With the advent of large-scale pre-trained language models, the field of Information Retrieval has seen a drastic evolution in terms of search and ranking architectures. In particular, embedding-based retrieval techniques and neural re-ranking strategies have been shown to capture similarity beyond simple term matching systems, resulting in improvements in various search contexts [24, 30, 33, 57]. We refer the reader to the survey of Lin et al. [40] for a more detailed overview of these techniques.

Lexical Matching and Vocabulary Mismatch. Despite advances in neural ranking, traditional lexical inverted index-based retrieval systems are still widely used in the *neural retrieval age* due to their simplicity, scalability, and their utility in finding documents matching given terms [73, 88]. For example, inverted index-based systems can be augmented with neural scoring functions to allow for much more effective retrieval while maintaining similar efficiency to traditional statistical models [51, 56]. Similarly, *hybrid search* techniques that combine both lexical and semantic matching are typically more effective than relying on a single model [36, 37, 53, 78].

Vocabulary mismatch is a common problem for classic lexical-based models, where the query terms provided by a user do not match the terms used to describe the same concept in a relevant document [23, 83]. Ultimately, this means that a relevant document may not be retrieved as it may not be matched at all during query processing, harming effectiveness. A well-studied solution to this problem is to expand the user query with additional terms to minimize the rate of term mismatch [1]. However, query expansion can be a risky prospect, as parameters can vary widely from query to query, and issuing a poorly expanded query can significantly harm effectiveness [8, 11, 29]. We refer to the survey of Azad and Deepak [5] for a more detailed overview of query expansion techniques.

Document Expansion. Another technique for reducing term mismatch — the one we focus on in this work — is known as document expansion. The idea of document expansion is to augment a given *document* with additional relevant terms to reduce term mismatch. Given a collection of documents, $\mathcal{D}$, and a document expansion method, E, E is applied to each document $D \in \mathcal{D}$ to generate N queries $D \mapsto Q^N$. Typically, these resultant queries are simply *appended* to the corresponding document text to create a new corpus $\mathcal{D}'$, which can then be indexed and searched as usual. Previous work has shown that the choice of E can have a significant impact on the effectiveness of document expansion [58, 59].

Historically, document expansion has been based on simple heuristics. For example, expanding a document with inbound anchor text or terms from URLs has been explored [12, 20, 80]. Historical queries have also been used as a source of expansion terms [60, 67], under the assumption that a click-graph can provide implicit feedback on queries that should retrieve a given document [16, 31]. Another approach applies statistical language modelling for expanding documents with new, relevant terms [19, 70].

More recently, neural models have been applied to automatically generate queries for a given passage. For example, the Doc2Query method of Nogueira and Lin [58] trained a simple sequence-to-sequence transformer to generate queries using top-k random sampling [21], resulting in modest effectiveness improvements over the BM25 baseline. However, replacing this simple sequence-to-sequence transformer with the much larger T5 model [63] — known as Doc2Query-T5 or DocT5Query — yields much larger improvements [58]. More recently, Weller et al. [79] applied ChatGPT to generate document expansions, but it is unclear whether these expansions outperform the T5-based Doc2Query approach.

Document Filtering. The focus of our reproducibility study is the work of Gospodinov et al. [27], who observed that the queries generated by a given expansion model *may not be relevant* to the given document D, which may harm the effectiveness of retrieval. Their solution, Doc2Query--, adds a filtering mechanism that uses a relevance model to map a query/document pair to a relevance score $\mathcal{S} = Q \times D \mapsto \mathbb{R}$. It is assumed that higher values of $\mathcal{S}$ are more relevant. Next, this scoring function is applied to *every* query/document pair in $\mathcal{D}$. Finally, a threshold t can be used to retain only the queries surpassing the threshold:

$$\mathcal{D}' = \{D + \{Q \mid \mathcal{S}(Q, D) \geq t\} \mid D \in \mathcal{D}\} \quad (1)$$

Clearly, this formulation is model agnostic since it depends only on applying a global threshold t to the distribution of $\mathcal{S}$ values (referred to as a *global filter*). The choice of model then depends on operational constraints such as the offline computational effort, assuming that more expensive scoring models typically perform better. To this end, the original work explored three models for computing query/document scores; two cross-encoders (ELECTRA [10, 61] and MonoT5 [57]) and a bi-encoder (TCT-ColBERT[69]).

3 EXPERIMENTAL SETUP

Datasets and Measures. Following the experimental setup of Gospodinov et al. [27], we used the MSMARCO-v1 passage corpus [7] as the basis of our study. We experiment with three query sets, namely the Dev set consisting of 6,980 queries (sparsely judged, averaging 1.1 judgments per query), and both the 2019 and 2020 TREC Deep Learning Track collections with deeper judgments (43 and 54 queries, 215 and 211 judgments per query, respectively) [13, 15]. We evaluate runs using the official metrics (RR@10 for Dev, NDCG@10 for the TREC collections); we also evaluate *recall* at various cut-offs.

Beyond MSMARCO-v1, we extend our analysis to a subset of the BEIR benchmark, computing NDCG@10 and Recall@100 following common practice for BEIR [72]. In particular, we experiment with DBPedia [28], TREC COVID [76], Robust04 [75], Touché [9], and Quora, with 38.2, 493.5, 69.9, 19, 1.6 judgments per query, in turn (see [72] for detailed statistics). The selection of this subset was based on availability and computing resources.

Significance testing is conducted using a two-tailed pairwise t-test with a Bonferroni correction; we report significance at $p < 0.05$. We note that caution must be applied when interpreting significance tests on the Dev collection, as the test is overpowered due to the large number of topics [65]. We also note that the Dev and Touché collections are only sparsely judged, which may lead to unstable comparisons [3, 49].

Environment and Tools. All experiments were carried out on a Linux workstation with a 32-core AMD Ryzen Threadripper Pro 5975WX operating at 3.6 GHz with 512 GiB of RAM and two Nvidia RTX A5500 GPUs. Index sizes are computed in GiB, and latency is reported as the mean response time over the large Dev query set.

Following the same setup of Gospodinov et al. [27], we conduct our experiments using PyTerrier toolkit [43, 46, 47] and PISA [54]. We also use Anserini [81] and CIFF [39] for processing, indexing, and exchanging raw JSON data between systems. Following best practice [48, 55], query processing experiments are conducted using the MaxScore algorithm [74] on SIMD-BP128 compressed indexes [35] after document reordering [18, 50].

Models. The T5-based Doc2Query model [58] is used for document expansion throughout the experiments presented. In particular, the publicly available $N = 80$ expansion queries are used for MSMARCO-v1 [42]; similarly, for BEIR, we made use of the $N = 20$ queries made available by Thakur et al. [71]. We score the expansion queries with the ELECTRA[2] cross-encoder model [10, 61] as it shows the top effectiveness in the work of Gospodinov et al. [27]. Unless stated otherwise, we use BM25 for lexical retrieval after tuning the k_1 and b parameters on Dev using grid search in the range

[2] crystina-z/monoELECTRA_LCE_nneg31

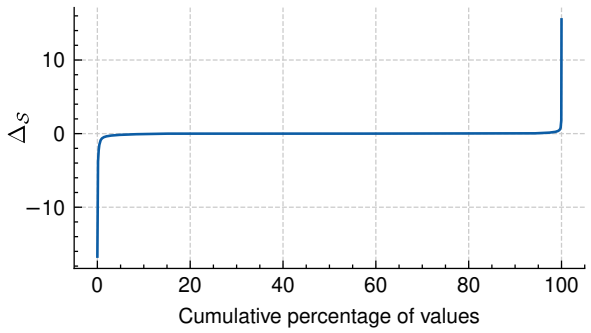


Figure 1: The distribution of ≈ 700 million per-query score differences between the reference and reproduced Doc2Query-- (with the ELECTRA scorer) on MSMARCO-v1.

$[0.5, 2.5]$ with a 0.25 step, and $[0, 1]$ with a 0.1 step, respectively; we used PISA's instantiation of BM25 [32, 64].

4 REPRODUCING DOC2QUERY--

This section outlines our main reproducibility experiments and subsequent analysis.

Experiment: Scoring the Expansion Queries. Our first experiment aims to reproduce the *scoring phase* of Doc2Query--. After loading the MSMARCO-v1 corpus and the associated $N = 80$ (per-document) expansion queries, we apply the ELECTRA model to compute a relevance score for each passage/query pair. Using a single GPU, this process took approximately 300 hours, or about 1.75× longer than reported by the reference work. This can be attributed to our GPU having lower memory throughput, and our experimental server being used for other tasks during the experiment.

Although we use the same model as the reference work, minor differences in score predictions are highly likely due to differences in the experimental hardware. To measure this, we compute the difference between our generated scores and the reference scores on a per-query basis (denoted Δ_S). Figure 1 shows the resulting distribution. Examining the absolute differences shows that, while the majority of score differences are small (79% with $|\Delta_S| \leq 0.01$ and 10% with $0.01 < |\Delta_S| \leq 0.1$), there are some notable differences (9% with $0.1 < |\Delta_S| \leq 1.0$, and 1% with $|\Delta_S| > 1.0$). The cause of these differences was found to be related to *text encoding differences* between the input passages we downloaded, and the passages used by the Doc2Query-- authors. In particular, they used the ir_datasets tool [45] to load MSMARCO-v1 which automatically fixes encoding errors, whereas we loaded the raw passages directly without fixing these encoding errors. As prior work has noted that seemingly minor changes can indeed lead to large effectiveness differences [34, 41, 82, 85], we continue to use our reproduced scores in the following analysis to ensure any such effects are observed.

Experiment: End-to-End Retrieval. The next step in our reproducibility study involves *filtering* to retain only the highest scoring queries, and then running end-to-end retrieval experiments to measure the downstream effectiveness. In the reference work, a global cut-off was tuned by measuring the RR@10 scores on the MSMARCO-v1 Dev set. Doc2Query-- with ELECTRA was found to be the strongest model with a global cut-off of 30% (that is, the top scoring 30% of Doc2Query expanded queries were retained, and the rest filtered out). We follow this and remove queries that are below the 70 th percentile, append the remaining queries to their corresponding passages, and then index the corpus accordingly.

We also explore whether a local filter (Doc2Query--LF) — applied by retaining the highest scoring t% of expansion queries on a *per-passage basis* (not on the whole corpus) — can yield similar effectiveness to the global filter. We are interested in this approach since a local filter would be much easier to deploy in practice with dynamic corpora; there is no need to compute the collection-wide scores, allowing filtering to be applied on a document-by-document basis. To this end, and following the experimental setup used by the original authors, we tuned the value of t to maximize RR@10 on the Dev set and found that the best value of t was at 90% (meaning only the lowest scoring 10% of queries — 8 out of 80 — were removed for each passage). Table 2 shows the results of our reproduction pipeline with respect to the original work, along with the local filtering approach.

Our first observation is that the reproduced results are relatively close to those reported by Gospodinov et al. [27], even with the noted differences in the computed filtering scores. For example, we observe differences in RR@10 (Dev) of at most 0.001, and in NDCG@10 (DL19 and DL20) of 0.031 and 0.008, all within 3.3% of the original work. Interestingly, we find that our results are often slightly better than the original work. Beyond the differences in the filter scores, we believe that our parameter tuning grid-search may have explored a wider number of values, thus providing small benefits; the original grid search ranges were not reported. For these early precision metrics, the findings from the reference work are successfully reproduced — Doc2Query-- can significantly improve the effectiveness of Doc2Query.

However, a different conclusion is reached when considering the deeper recall-based metrics, where Doc2Query-- actually *degrades* the effectiveness of the unfiltered Doc2Query approach. We revisit this observation in the next experiment.

Additionally, the local filtering (Doc2Query--LF) is noticeably worse than the global filtering (Doc2Query--) for all precision metrics. Therefore, we apply the global filtering for the remainder of the experiments.

Finally, the last two columns of Table 2 report the efficiency characteristics of each system. While it is difficult to directly compare latency values with the reference work due to differences in hardware and algorithmic configurations, we observe a speedup of around 22% (from 10.7 to 8.8 ms), which is comparable to the 30% reduction observed by Gospodinov et al. [27].[3] Similarly, our reproduced Doc2Query-- index reduces the size of the Doc2Query index from 1.47 to 0.94 GiB, a reduction of 56% (compared to their reported 48% reduction). The most likely cause of this difference is due to variance in the data (different encoding as previously mentioned) and configuration options (such as the index block size and index reordering). Overall, the efficiency trends observed when

[3]Note that in Table 2, the original Doc2Query-- ("Run supplied [27]") efficiency numbers are copied directly from their paper [27].

Table 2: Reproduced effectiveness measurements for Doc2Query-- with the ELECTRA filter (top 30%) as compared to the BM25 and unfiltered Doc2Query baselines. The first set of Doc2Query-- entries represent the original results, and the second set represent our independently reproduced results. We also report mean response time (MRT) and index size. Significant differences with respect to the baseline system (third line) are marked with vertical arrows (the upward arrow means significantly better, and the downward significantly worse).

System	Notes	RR@10	NDCG@10		Recall@100			Recall@1000			MRT	Size
		Dev	DL19	DL20	Dev	DL19	DL20	Dev	DL19	DL20	ms	GiB
BM25	Tuned	$\downarrow$0.187	$\downarrow$0.508	$\downarrow$0.470	$\downarrow$0.671	0.511	0.581	$\downarrow$0.858	0.752	0.795	6.9	0.74
Doc2Query	$N = 80$, Tuned	$\downarrow$0.279	0.622	0.601	$\uparrow$0.831	0.590	0.703	$\uparrow$0.950	0.814	0.843	10.7	1.47
Doc2Query--	Run supplied [27]	0.323	0.669	0.614	0.819	0.572	0.669	0.934	0.792	0.816	23.0	0.95
Doc2Query--	Index supplied [27]	0.323	0.669	0.614	0.819	0.572	0.669	0.934	0.792	0.816	9.2	0.90
Doc2Query--	Default params, ours	0.322	0.672	0.616	$\downarrow$0.816	0.573	0.664	0.934	0.793	0.813	9.3	0.94
Doc2Query--	Tuned, ours	0.322	0.689	0.619	$\uparrow$0.821	0.584	0.675	0.935	0.799	$\uparrow$0.828	8.8	0.94
Doc2Query--LF	Tuned, ours	$\downarrow$0.282	0.633	0.612	$\uparrow$0.830	0.591	0.695	$\uparrow$0.950	0.811	0.839	10.7	1.33

Table 3: NDCG@10 after re-ranking different first-stage runs using MonoT5 at two different cut-offs. No significant differences were observed between any of the systems. Note that Doc2Query-- represents our reproduced, tuned system.

System	$k = 100$		$k = 1000$	
	DL19	DL20	DL19	DL20
BM25	0.712	0.682	0.744	0.704
Doc2Query	0.749	0.714	0.740	0.705
Doc2Query--	0.745	0.718	0.740	0.703

comparing Doc2Query and Doc2Query-- follow those reported in the original work.

Experiment: Re-ranking. Given that the models reported in Table 2 are likely to be deployed as an efficient first stage in a multi-stage *retrieve and re-rank* pipeline [40, 77], the reduced recall of Doc2Query-- raises concerns regarding its fitness for such pipelines. To understand how much recall matters in this scenario, we re-rank a subset of the systems from Table 2 using the strong MonoT5 model [57][4] for both $k = 100$ and $k = 1000$, and report the outcomes in Table 3. Interestingly, we find that there are no significant differences across the first-stage systems, even where the recall gap is quite evident in the initial lists. We attribute this to finding to the notion of *saturation* — so long as the re-ranker has access to "enough" good documents, adding more does not significantly improve the re-ranking effectiveness. However, MonoT5 may also exhibit some bias towards BM25 ranked lists since these formed the basis of its training regime [26]. We leave a more thorough exploration to future work, where we will investigate the effect of parameter choices, training regimes, and first-stage recall on the effectiveness of large language model (LLM) based re-rankers.

Experiment: New vs Copied Terms. In the first Doc2Query work [58], the authors analysed the expansion terms and found that 33% of expansion terms were not present in the original document (*new* terms), and the remaining 67% were already present in the document (*copied* terms). However, they did not attempt to quantify where the effectiveness improvements over the default passages arise; it may be that the new terms are more important due to their ability to reduce vocabulary mismatch, or that the copied terms are useful in increasing the weight of important terms within given passages [17]. Lin et al. [40] bridged this gap by evaluating the effectiveness of Doc2Query when the expansion incorporates new terms only, copied terms only, both new and copied terms (the default setting), and all expansion terms in lieu of the original document content. They found that both new *and* copied terms contribute to the overall effectiveness of Doc2Query.

Table 4: Ablation study on the effect of *new* and *copied* terms with respect to Doc2Query and Doc2Query-- on the Dev queries.

Configuration	RR@10	R@1000	Expansion %
Original text	0.187	0.858	0%
w/ d2q new only	0.174	0.875	24%
w/ d2q-- new only	0.214	0.894	13%
w/ d2q copied only	0.233	0.908	76%
w/ d2q-- copied only	0.276	0.904	87%
w/ d2q new + copied	0.279	0.950	100%
w/ d2q-- new + copied	0.322	0.934	100%
d2q expansion only	0.230	0.906	100%
d2q-- expansion only	0.281	0.847	100%

Our next experiment extends this study to Doc2Query-- given that the overall term distribution is likely to have changed. We lower-cased the expansion queries and original text, removed stopwords, and then applied stemming[5] to categorize terms into the new and copied sets. We experimented with three combinations: original text + new terms only, original text + copied terms only, and all expansion terms (both new and copied) *without* the original passage text. We also report the original text + both new *and* copied terms, thus representing the default Doc2Query or Doc2Query-- systems. Beyond measuring effectiveness, we also computed the

[4]See the `llm-rankers` library [84]: https://github.com/ielab/llm-rankers

[5]We used the same stopwords list and stemming model as Doc2Query--.

percentage of expansion terms included in each setting with respect to the entire set of expansion terms.

Table 4 reports the results, where four important observations can be noted. The first one is that Doc2Query-- has a higher percentage of copied terms at 87% (compared to 76% for Doc2Query), indicating that the filtering model may prefer copied terms over new terms. Second, Doc2Query scored a higher recall in all combinations (as compared to Doc2Query-- under the same setting) except for *new terms only*, which indicates that the lower rate of new terms may indeed harm recall. Third, the new or copied term sets alone are not adequate to achieve the best effectiveness — confirming the findings of Lin et al. [40] — and so finding the best combination of these sets is the main challenge for optimizing document expansion models. Fourth, it is evident from the *expansion only* rows that the content retained by Doc2Query-- is much more effective than the expansions generated by Doc2Query, indicating that the original document content may be less important in the context of Doc2Query--; again, this can be attributed to the higher rate of copied terms which simply boost the existing document content rather than introducing new, unseen terms.

Experiment: Term Filtering Ablation. Based on our previous observations, both new and copied terms are important components for Doc2Query--. However, we also know that filtering expansion queries with low ELECTRA scores can harm recall, as the overall proportion of new terms is lower for Doc2Query-- as compared to Doc2Query, presumably increasing vocabulary mismatch. As such, we are interested in understanding how the quality and composition of the expansion set impact result quality. For example, if keeping the top 30% of the queries yields the best effectiveness, what would happen if we kept the bottom 70% of the queries instead? Can these queries help improve effectiveness at all? To measure this, we run Doc2Query-- with a global threshold $t \in \{10, 30, 50, 70, 90\}$, and examine the effect of keeping the top (highest scoring) t% or the bottom (lowest scoring) t% of expansion queries (achieved by switching the $\geq$ sign with a $<$ sign in Equation 1).

Figure 2 illustrates our findings. It is clear that the proportion of the new and copied terms (Figure 2a) varies drastically between the term sets we explored. In particular, the ELECTRA filter favours expansion queries that have a very high term overlap with the original passage, as the rate of copied terms increases as the retention threshold increases. Figure 2b and Figure 2c show the effect of these expansion sets on both Recall@1000 and NDCG@10, respectively. Although removing low scoring expansion queries brings noticeable enhancements to NDCG@10, it causes some decline in the recall. Looking at Figure 2b, it is evident that the recall of Doc2Query-- suffers with smaller expansion sets, irrespective of whether they are good or bad, suggesting that *new* terms are indeed more important for maximizing recall as compared with *copied* terms, confirming the findings in Table 4. And while this experiment also demonstrates that including "bad" queries may not necessarily harm recall, there is still a delicate balance between the quality of the expansion terms and the effectiveness of the system.

5 EXTENSIONS

In the previous section, we examined Doc2Query-- with a focus on reproducibility. In this section, we extend our experimentation beyond the original work of Gospodinov et al. [27].

5.1 Testing Zero-Shot Generalization

Since Doc2Query-- was only tested on MSMARCO-v1, we opt to test whether it generalizes to other datasets. To this end, we score and evaluate Doc2Query-- in a zero-shot manner on five datasets from the BEIR benchmark: DBPedia, COVID, Robust04, Touché, and Quora. Given the zero-shot nature of BEIR, we applied default BM25 parameters ($k_1 = 0.9$ and $b = 0.4$) and selected two global thresholds for Doc2Query-- (top $t = 30$% and $t = 50$%).[6]

Table 5 draws a comparison between these systems with respect to Doc2Query. From a recall perspective, Doc2Query-- ($t = 30$%) is significantly worse than Doc2Query on all datasets but Touché. From a precision perspective, Doc2Query-- is only significantly worse on DBPedia and Quora with respect to NDCG@10, and outperforms Doc2Query on COVID at $t = 50$%. This general decline suggests that Doc2Query-- is not a good option under the zero-shot setup. Further investigation of the reasons behind this decline is left for future work; however, we do have a few hypotheses. Firstly, the zero-shot setting means that parameters cannot be tuned, making it difficult to assess whether the system is properly configured. Secondly, it may be the case that $N = 20$ queries are insufficient for filtering, and more queries would be required to see improved effectiveness. Thirdly, the truncation of long text documents during the expansion and filtering process means that the cross-encoder model might not have access to the full context of the document, biasing the expansion to the beginning of the documents. We believe these aspects are worth further investigation in future work.

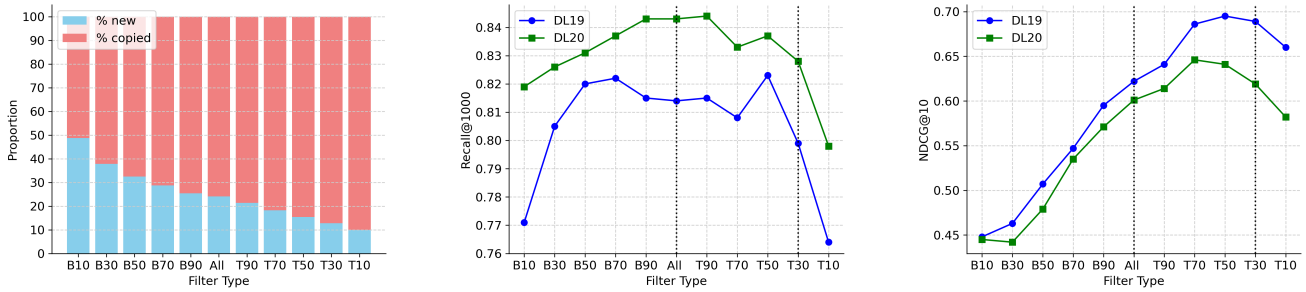
5.2 Better Queries via Reinforcement Learning

Doc2Query-- utilizes the ELECTRA cross-encoder in a two-stage pipeline. This process involves initially generating a substantial number of queries and subsequently evaluating them using ELECTRA. As both generation and filtering are computationally expensive, a natural question arises: can we streamline this process by directly generating queries with high ELECTRA scores?

To address this question, we employ Reinforcement Learning (RL) to train a Doc2Query model that optimizes the ELECTRA score directly. Specifically, we utilize the Proximal Policy Optimization (PPO) algorithm [68], with ELECTRA serving as the reward model. The policy network is initialized with the original Doc2Query model checkpoint. During each training step, passages are randomly sampled from the MSMARCO-v1 passage dataset. The policy network generates queries based on these passages, and the reward model assigns scores to the generated queries. The training objective is to adapt the policy network to maximize the reward provided by the ELECTRA model.

Figure 3a shows the ELECTRA score distribution of queries generated by our RL-trained Doc2Query model (RLGen) as compared to the original Doc2Query model. The figure clearly indicates a positive shift in the score distribution after training Doc2Query

[6]Note that we select the less aggressive $t = 50$% threshold since we have only $N = 20$ expansion queries for each document in the BEIR collection.



(a) Proportion of new and copied terms. **(b)** Recall@1000 as the global threshold varies. **(c)** NDCG@10 the global threshold varies.

Figure 2: The distribution of the expansion terms (left) and effectiveness analysis of the bottom (B) and top (T) thresholds on DL19 and DL20 with respect to Recall@1000 (middle) and NDCG@10 (right). "T30" represents the reported Doc2Query-- system from the previous experiments, and "All " represents the Doc2Query baseline; both are marked with vertical dotted lines.

Table 5: Effectiveness of two baselines (BM25 and Doc2Query) and two Doc2Query-- configurations across five datasets of BEIR benchmark. Significant differences with respect to the baseline (Doc2Query) are marked with vertical arrows.

System	NDCG@10					Recall@100				
	DBPedia	COVID	Robust04	Touché	Quora	DBPedia	COVID	Robust04	Touché	Quora
BM25	$\downarrow$0.320	$\downarrow$0.571	$\downarrow$0.407	$\uparrow$0.448	$\uparrow$0.780	$\downarrow$0.472	$\downarrow$0.107	$\downarrow$0.367	$\uparrow$0.581	0.972
Doc2Query	0.347	0.668	0.430	0.289	0.769	0.499	0.126	0.392	0.543	0.973
Doc2Query-- (t = 30%)	$\downarrow$0.314	0.664	0.421	$\uparrow$0.361	$\downarrow$0.660	$\downarrow$0.459	$\downarrow$0.118	$\downarrow$0.380	0.570	$\downarrow$0.939
Doc2Query-- (t = 50%)	$\downarrow$0.333	0.683	0.423	$\uparrow$0.342	$\downarrow$0.721	$\downarrow$0.462	0.123	0.386	0.564	$\downarrow$0.955

with the RL algorithm, thereby demonstrating the accomplishment of the intended objective. Figure 3b and Figure 3c demonstrate that the RLGen tends to produce longer queries[7] and, interestingly, incorporates a higher percentage of copied terms and fewer new terms compared to Doc2Query.

While RLGen generates queries with higher ELECTRA scores, the results in Table 6 suggest that this gain does not necessarily translate to improved retrieval effectiveness. In fact, the RLGen model is significantly less effective than Doc2Query-- across all datasets and metrics. These unexpected results indicate that simply optimizing for the ELECTRA score may lead to a sub-optimal policy. In particular, the queries from RLGen typically copy term sequences from the input passage rather than expanding new terms, confirming our earlier observation of ELECTRA preferring queries with copied terms. In short, higher ELECTRA scores do not necessarily translate to higher effectiveness. In future work, it would be interesting to see if a diversity-aware RL policy could lead to a more effective single-stage document expansion model, allowing filtering to be bypassed entirely. We are also interested in exploring the effect of leveraging the final retrieval effectiveness as a reward instead of the estimated relevance.

5.3 Query Filtering for Learned Sparse Retrieval

Finally, we turn our attention to learned sparse retrieval (LSR) methods. In particular, we are interested in determining whether filtering low-scoring expanded queries can lead to more effective LSR methods — or, in other words, whether the same gains observed from filtering on traditional BM25 rankers translate to learned ranking models. Prior work has demonstrated the importance of document expansion in the context of LSR and, in particular, augmenting documents with new relevant terms [38, 42, 52], but the effect of including non-relevant expansion terms has not been studied.

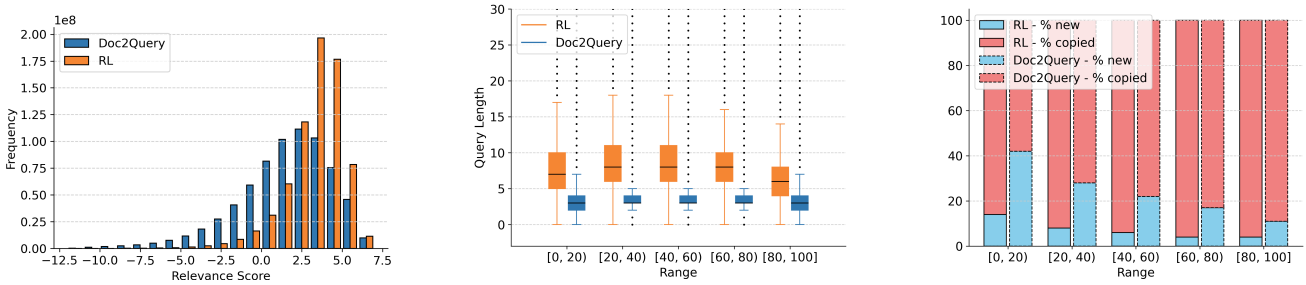
To measure how filtering affects LSR, we employ two LSR methods that have been shown to benefit from document expansion, namely DeepImpact [54] and UniCOIL [25, 38]. First, we reproduce the original results from both works by applying the $N = 80$ Doc2Query generated queries to each document in the original corpus and then conducting inference on these augmented documents with the original models;[8] the resulting data is then indexed via PISA and the top $k = 1000$ documents are retrieved across the same collections and settings as in Section 4. As baselines, we reused existing indexes from a recent study on LSR methods [51]. Table 7 shows the results. Firstly, we find that our reproduced DeepImpact system outperforms the results reported by Mallia et al. [52]. Our assumption is that this is due to the original work applying $N = 40$ expanded queries per passage, whereas we apply $N = 80$ following earlier experiments. Consequently, processing latency and index size are slightly higher in our reproduced systems. We observed similar trends for UniCOIL, although the overall effectiveness was not consistently improved by including more expansion queries.

[7] Lengths can be quite long (up to around 50-60 terms) but are generally short. We trimmed Figure 3b at $y = 30$ for readability.

[8] https://github.com/luyug/COIL/ + `castorini/unicoil-msmarco-passage` and https://github.com/terrierteam/pyterrier_deepimpact

Table 6: Efficiency and effectiveness comparison between Doc2Query-- and the RLGen model. RLGen is significantly worse than Doc2Query-- across most collections and metrics.

System	Notes	RR@10	NDCG@10		Recall@100			Recall@1000			MRT	Size
		Dev	DL19	DL20	Dev	DL19	DL20	Dev	DL19	DL20	ms	GiB
Doc2Query--	Tuned, ours	0.322	0.689	0.619	0.821	0.584	0.675	0.935	0.799	0.828	8.8	0.94
RLGen	$N = 80$, tuned, ours	↓0.235	↓0.534	↓0.557	↓0.733	↓0.525	↓0.627	↓0.897	0.760	0.816	9.4	1.48



(a) Score distribution. **(b)** Length of expansion queries. **(c)** Proportion of new and copied terms.

Figure 3: ELECTRA score distribution for Doc2Query and RLGen (left). A detailed comparison is shown for query length (middle) and proportion of new and copied terms (right) for each model across fixed-range (20%) buckets of ELECTRA scores.

Table 7: Effectiveness of learned sparse retrieval systems, including those incorporating Doc2Query--. For significance testing, both DeepImpact and UniCOIL with $N = 80$ are used as the baselines for the respective system-level comparisons.

System	Notes	RR@10	NDCG@10		Recall@100			Recall@1000			MRT	Size
		Dev	DL19	DL20	Dev	DL19	DL20	Dev	DL19	DL20	ms	GiB
Doc2Query--	Tuned, ours	0.322	0.689	0.619	0.821	0.584	0.675	0.935	0.799	0.828	8.8	0.94
DeepImpact	$N = 40$, index supplied [51]	↓0.327	0.696	0.652	↓0.842	0.597	0.695	0.948	0.806	0.834	17.4	1.53
DeepImpact	$N = 80$, ours	0.341	0.706	0.658	0.855	0.585	0.696	0.955	0.811	0.837	19.2	1.77
DeepImpact--	$N = 80, t = 70\%$, ours	0.341	0.700	0.655	0.851	0.589	0.694	0.952	0.799	0.830	15.7	1.42
UniCOIL	$N = 40$, index supplied [51]	0.352	0.701	0.675	↓0.861	0.607	↓0.701	↓0.958	0.829	0.843	26.0	1.33
UniCOIL	$N = 80$, ours	0.350	0.696	0.675	0.868	0.605	0.711	0.964	0.839	0.850	33.6	1.71
UniCOIL--	$N = 80, t = 70\%$, ours	0.353	0.687	0.680	0.870	0.600	0.717	0.961	0.832	0.844	28.4	1.41

In any case, our results are comparable to those reported in the original work, so we regard our reproduction of these baselines as successful.

Next, we run the same end-to-end pipeline, but modify the corpus augmentation step to only include the top t% highest scoring queries (from ELECTRA) with $t \in \{10, 30, 50, 70, 90\}$. This allows us to isolate the effect of including different levels of filtering on the effectiveness of learned sparse retrieval. We only report the best-performing value of t ($t = 70\%$) after tuning it on the Dev queries. Evidently, the filtered LSR systems (DeepImpact-- and UniCOIL--) are empirically similar to their unfiltered counterparts; however, they can provide slightly better efficiency characteristics (18–22% less time, and 18–25% less space, respectively). Interestingly, this indicates that in the LSR scenario, filtering does not provide an *additive* effect [4]. One possible explanation is that LSR models are already implicitly incorporating filtering by assigning low-impact scores to non-relevant terms within each document. However, more experiments would be required to either confirm or refute this hypothesis. Furthermore, it is worth noting that similar efficiency and effectiveness can be achieved by both baseline LSR systems with $N = 40$ queries per document, raising further questions about the utility of filtering in the LSR scenario. In future work, it would be worth exploring the complex efficiency-vs-effectiveness trade-offs between various levels of query generation and query filtering in the context of LSR mechanisms.

6 SUMMARY OF FINDINGS

Finally, we summarize the key findings from our reproducibility study, opportunities for future work, and some limitations.

Reproducibility. Overall, our independent study successfully reproduced the key findings of the Doc2Query-- work. In particular, using the ELECTRA model to filter expansion queries from the T5-based Doc2Query model yields significant improvements in early precision metrics on the MSMARCO-v1 corpus. Furthermore, Doc2Query-- improves both the space and time efficiency of top-k retrieval. While some experimental differences were observed, these were primarily due to differences in the data preparation process and in the configuration of systems.

Practicality. A range of our experiments focused on aspects related to the practicality of deploying systems such as Doc2Query and Doc2Query--. For example, we explored whether local filtering can be applied to Doc2Query-- instead of the global filter. On end-to-end retrieval experiments, this reduced the effectiveness of Doc2Query--. However, in the context of re-ranking, it was found that all models yielded comparable effectiveness, suggesting that the best choice of the first-stage retrieval system depends on the context in which it is deployed. If the first stage is the only stage, then using Doc2Query-- can be a good option, depending on the collection and the data available for training and tuning. If re-ranking is applied, then there is a complex trade-off between aspects such as the index size, the value of k, and, in turn, the first- and second-stage latency. In particular, care must be taken to ensure a sufficient recall base is supplied to the downstream re-ranker while aiming to keep efficiency under control. Exploring this trade-off space in greater detail was out of the scope of this study.

Reinforcement Learning. Another practicality improvement we measured was the use of reinforcement learning to yield better queries in the hope of avoiding the filtering process altogether. Unfortunately, our failed approach demonstrates that directly optimising for increased ELECTRA scores results in biased queries that actually reduce effectiveness. A promising avenue of future work is to investigate how to incorporate the notion of diversity into the RL policy, ensuring that a mix of new and copied terms can be generated. Another option is to investigate alternative formulations of the policy optimization framework [62, 68].

Learned Sparse Retrieval. Applying Doc2Query-- to the learned sparse retrieval regime demonstrated that query filtering does not increase effectiveness when term impacts are learned. Interestingly, it may be that Doc2Query-- and learned sparse retrieval models are two sides of the same coin. For example, our experiments confirmed that both Doc2Query and Doc2Query-- improve effectiveness through a delicate combination of term expansion and term re-weighting. Similarly, learned sparse retrieval systems are sensitive to the inclusion of expansion terms and re-weight terms within a document based on their estimated relevance. This may explain why filtering has no effect on LSR methods; they are already filtering non-relevant terms by assigning them low weights. In future work, we plan to explore this idea in more detail to better understand whether these different approaches are, in essence, exploiting the same underlying qualities to improve effectiveness.

Limitations. Throughout our study, we have made a best endeavour to remain faithful to the original experimentation and settings from prior work. However, occasionally, we have diverged from those settings. For example, experiments on BEIR applied only $N = 20$ expansion queries due to computational constraints. Similarly, we only explored the best-performing filtering mechanism from the original Doc2Query-- paper, ELECTRA, and did not reproduce the remaining models. Another key limitation is that the models explored in this work were trained on MSMARCO-v1 given the availability of large-scale training data [14]. It is plausible that better effectiveness would be observed in the zero-shot retrieval setting, for instance, if more diverse training data is available.

Overall, document expansion and filtering methods are computationally expensive since they typically involve inference on cross-encoders, limiting the number of experiments that can be conducted in a reasonable amount of time and computational resources. However, cross-encoders are not the only viable approach for neural document expansion. One alternative is to use a masked language model (instead of a sequence-to-sequence transformer) to generate a series of unigrams or tokens that can be appended to the given document [22, 44, 86, 87]. While these methods are typically much more efficient than their sequence-to-sequence counterparts [66], they do not generate syntactically correct queries [27, 51]. Ultimately, this means that a cross-encoder may not be able to accurately score expansion tokens, as they are typically trained on natural language sequences; nevertheless, this is indeed an important avenue for future exploration. Interestingly, as discussed above, learned sparse retrieval models could be considered as a viable filtering mechanism for bag-of-words based expansion models, given their ability to evaluate relevance on a term-by-term basis. We look forward to examining this connection more deeply in future work.

Finally, the quality vs effectiveness trade-offs between different document expansion methods — including cheap, traditional, approaches [6], and expensive LLM-based methods such as ChatGPT [2, 79] — has not been widely explored. Adding the notion of filtering to this complex space could yield better operational trade-offs than those already explored.

7 CONCLUSIONS

In this reproducibility study, we provide a comprehensive analysis of Doc2Query-- across various contexts, including first-stage retrieval, second-stage re-ranking, zero-shot out-of-domain retrieval, and learned sparse retrieval. We also proposed and evaluated simple yet practical improvements to the original Doc2Query-- approach, including local filtering and reinforcement learning-based query generation.

Our empirical evaluation on MSMARCO-v1 successfully reproduced the findings of Gospodinov et al. [27], demonstrating that Doc2Query-- offers both efficiency and effectiveness improvements over Doc2Query. We also found that while Doc2Query-- harms recall over Doc2Query, this has no negative effect on re-ranking effectiveness. Extending our analysis to a subset of the BEIR benchmark shows mixed effectiveness in the zero-shot setting, with Doc2Query generally outperforming Doc2Query--. Finally, applying Doc2Query-- to learned sparse retrieval systems demonstrated modest efficiency improvements, but the large effectiveness gains observed on BM25 did not translate to these methods.

Acknowledgments. We thank Sean MacAvaney, Xueguang Ma, and Nandan Thakur for providing clarifications and/or data that helped with this study.

REFERENCES

[1] Nasreen Abdul-Jaleel, James Allan, W Bruce Croft, Fernando Diaz, Leah Larkey, Xiaoyan Li, Mark D Smucker, and Courtney Wade. 2004. UMass at TREC 2004: Novelty and HARD. In *Proc. TREC*.

[2] Marwah Alaofi, Luke Gallagher, Mark Sanderson, Falk Scholer, and Paul Thomas. 2023. Can Generative LLMs Create Query Variants for Test Collections? An Exploratory Study. In *Proc. SIGIR*. 1869–1873.

[3] Negar Arabzadeh, Alexandra Vtyurina, Xinyi Yan, and Charles L. A. Clarke. 2022. Shallow pooling for sparse labels. *Inf. Retr.* 25, 4 (2022), 365–385.

[4] Timothy G. Armstrong, Alistair Moffat, William Webber, and Justin Zobel. 2009. Improvements That Don't Add up: Ad-hoc Retrieval Results since 1998. In *Proc. CIKM*. 601–610.

[5] Hiteshwar Kumar Azad and Akshay Deepak. 2019. Query expansion techniques for information retrieval: A survey. *Inf. Proc. & Man.* 56, 5 (2019), 1698–1735.

[6] Leif Azzopardi, Maarten de Rijke, and Krisztian Balog. 2007. Building simulated queries for known-item topics: An analysis using six European languages. In *Proc. SIGIR*. 455–462.

[7] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. 2018. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. *arXiv:1611.09268v3* (2018).

[8] Bodo Billerbeck and Justin Zobel. 2004. Questioning query expansion: An examination of behaviour and parameters. In *Proc. ADC*. 69–76.

[9] Alexander Bondarenko, Maik Fröbe, Meriem Beloucif, Lukas Gienapp, Yamen Ajjour, Alexander Panchenko, Chris Biemann, Benno Stein, Henning Wachsmuth, Martin Potthast, and Matthias Hagen. 2020. Overview of Touché 2020: Argument Retrieval. In *Proc. CLEF Assoc.*

[10] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *Proc. ICLR*.

[11] Kevyn Collins-Thompson. 2009. Reducing the risk of query expansion via robust constrained optimization. In *Proc. CIKM*. 837–846.

[12] Nick Craswell, David Hawking, and Stephen Robertson. 2001. Effective site finding using link anchor information. In *Proc. SIGIR*. 250–257.

[13] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2021. Overview of the TREC 2020 deep learning track. In *Proc. TREC*.

[14] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Jimmy Lin. 2021. MS MARCO: Benchmarking Ranking Models in the Large-Data Regime. In *Proc. SIGIR*. 1566–1576.

[15] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. 2020. Overview of the TREC 2019 deep learning track. (2020).

[16] Nick Craswell and Martin Szummer. 2007. Random Walks on the Click Graph. In *Proc. SIGIR*. 239–246.

[17] Zhuyun Dai and Jamie Callan. 2020. Context-Aware Document Term Weighting for Ad-Hoc Search. In *Proc. WWW*. 1897–1907.

[18] Laxman Dhulipala, Igor Kabiljo, Brian Karrer, Giuseppe Ottaviano, Sergey Pupyrev, and Alon Shalita. 2016. Compressing Graphs and Indexes with Recursive Graph Bisection. In *Proc. KDD*. 1535–1544.

[19] Miles Efron, Peter Organisciak, and Katrina Fenlon. 2012. Improving retrieval of short texts through document expansion. In *Proc. SIGIR*. 911–920.

[20] Nadav Eiron and Kevin S. McCurley. 2003. Analysis of anchor text for web search. In *Proc. SIGIR*. 459–460.

[21] Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical Neural Story Generation. In *Proc. ACL*. 889–898.

[22] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking. In *Proc. SIGIR*. 2288–2292.

[23] George W. Furnas, Thomas K. Landauer, Louis M. Gomez, and Susan T. Dumais. 1987. The vocabulary problem in human-system communication. *Commun. ACM* 30, 11 (1987), 964–971.

[24] Yukang Gan, Yixiao Ge, Chang Zhou, Shupeng Su, Zhouchuan Xu, Xuyuan Xu, Quanchao Hui, Xiang Chen, Yexin Wang, and Ying Shan. 2023. Binary Embedding-based Retrieval at Tencent. In *Proc. KDD*. 4056–4067.

[25] Luyu Gao, Zhuyun Dai, and Jamie Callan. 2021. COIL: Revisit Exact Lexical Match in Information Retrieval with Contextualized Inverted List. In *Proc. NAACL*. 3030–3042.

[26] Luyu Gao, Zhuyun Dai, and Jamie Callan. 2021. Rethink Training of BERT Rerankers in Multi-stage Retrieval Pipeline. In *Proc. ECIR*. 280–286.

[27] Mitko Gospodinov, Sean MacAvaney, and Craig Macdonald. 2023. Doc2Query–: When Less is More. In *Proc. ECIR*. 414–422.

[28] Faegheh Hasibi, Fedor Nikolaev, Chenyan Xiong, Krisztian Balog, Svein Erik Bratsberg, Alexander Kotov, and Jamie Callan. 2017. DBpedia-entity v2: A test collection for entity search. In *Proc. SIGIR*. 1265–1268.

[29] Ben He and Iadh Ounis. 2009. Studying Query Expansion Effectiveness. In *Proc. ECIR*. 611–619.

[30] Jui-Ting Huang, Ashish Sharma, Shuying Sun, Li Xia, David Zhang, Philip Pronin, Janani Padmanabhan, Giuseppe Ottaviano, and Linjun Yang. 2020. Embedding-based Retrieval in Facebook Search. In *Proc. KDD*. 2553–2561.

[31] Shan Jiang, Yuening Hu, Changsung Kang, Tim Daly, Dawei Yin, Yi Chang, and Chengxiang Zhai. 2016. Learning Query and Document Relevance from a Web-scale Click Graph. In *Proc. SIGIR*. 185–194.

[32] Chris Kamphuis, Arjen P. de Vries, Leonid Boytsov, and Jimmy Lin. 2020. Which BM25 do you mean? A large-scale reproducibility study of scoring variants. In *Proc. ECIR*. 28–34.

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

[34] Carlos Lassance and Stephane Clinchant. 2023. The Tale of Two MSMARCO - and Their Unfair Comparisons. In *Proc. SIGIR*. 2431–2435.

[35] Daniel Lemire and Leonid Boytsov. 2015. Decoding billions of integers per second through vectorization. *Soft. Prac. & Exp.* 41, 1 (2015), 1–29.

[36] Jurek Leonhardt, Henrik Müller, Koustav Rudra, Megha Khosla, Abhijit Anand, and Avishek Anand. 2023. Efficient Neural Ranking using Forward Indexes and Lightweight Encoders. *ACM Transactions on Information Systems* (2023). Just Accepted.

[37] Hang Li, Shuai Wang, Shengyao Zhuang, Ahmed Mourad, Xueguang Ma, Jimmy Lin, and Guido Zuccon. 2022. To Interpolate or not to Interpolate: PRF, Dense and Sparse Retrievers. In *Proc. SIGIR*. 2495–2500.

[38] Jimmy Lin and Xueguang Ma. 2021. A Few Brief Notes on DeepImpact, COIL, and a Conceptual Framework for Information Retrieval Techniques. *arXiv:2106.14807* (2021).

[39] 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.

[40] Jimmy Lin, Rodrigo Nogueira, and Andrew Yates. 2021. *Pretrained Transformers for Text Ranking: BERT and Beyond.* Morgan & Claypool Publishers.

[41] Jimmy Lin and Peilin Yang. 2019. The Impact of Score Ties on Repeatability in Document Ranking. In *Proc. SIGIR*. 1125–1128.

[42] Xueguang Ma, Ronak Pradeep, Rodrigo Nogueira, and Jimmy Lin. 2022. Document Expansion Baselines and Learned Sparse Lexical Representations for MSMARCO V1 and V2. In *Proc. SIGIR*. 3187–3197.

[43] Sean MacAvaney and Craig Macdonald. 2022. A Python Interface to PISA!. In *Proc. SIGIR*. 3339–3344.

[44] 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.

[45] Sean MacAvaney, Andrew Yates, Sergey Feldman, Doug Downey, Arman Cohan, and Nazli Goharian. 2021. Simplified Data Wrangling with ir_datasets. In *Proc. SIGIR*. 2429–2436.

[46] Craig Macdonald and Nicola Tonellotto. 2020. Declarative Experimentation in Information Retrieval using PyTerrier. In *Proc. ICTIR*.

[47] Craig Macdonald, Nicola Tonellotto, Sean MacAvaney, and Iadh Ounis. 2021. PyTerrier: Declarative Experimentation in Python from BM25 to Dense Retrieval. In *Proc. CIKM*. 4526–4533.

[48] Joel Mackenzie and Alistair Moffat. 2020. Examining the additivity of top-k query processing innovations. In *Proc. CIKM*. 1085–1094.

[49] Joel Mackenzie, Matthias Petri, and Alistair Moffat. 2021. A Sensitivity Analysis of the MSMARCO Passage Collection. *arXiv preprint arXiv:2112.03396* (2021).

[50] Joel Mackenzie, Matthias Petri, and Alistair Moffat. 2022. Tradeoff options for bipartite graph partitioning. *IEEE Trans. Know. & Data Eng.* (2022), 8644–8657.

[51] Joel Mackenzie, Andrew Trotman, and Jimmy Lin. 2023. Efficient document-at-a-time and score-at-a-time query evaluation for learned sparse representations. *ACM Transactions on Information Systems* 41, 4 (2023), 1–28.

[52] Antonio Mallia, Omar Khattab, Torsten Suel, and Nicola Tonellotto. 2021. Learning Passage Impacts for Inverted Indexes. In *Proc. SIGIR*. 1723–1727.

[53] Antonio Mallia, Joel Mackenzie, Torsten Suel, and Nicola Tonellotto. 2022. Faster Learned Sparse Retrieval with Guided Traversal. In *Proc. SIGIR*. 1901–1905.

[54] Antonio Mallia, Michał Siedlaczek, Joel Mackenzie, and Torsten Suel. 2019. PISA: Performant Indexes and Search for Academia. In *Proc. OSIRRC at SIGIR*. 50–56.

[55] Antonio Mallia, Michał Siedlaczek, and Torsten Suel. 2019. An Experimental Study of Index Compression and DAAT Query Processing Methods. In *Proc. ECIR*. 353–368.

[56] Thong Nguyen, Sean MacAvaney, and Andrew Yates. 2023. A Unified Framework for Learned Sparse Retrieval. In *Proc. ECIR*. 101–116.

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

[58] Rodrigo Nogueira and Jimmy Lin. 2019. From doc2query to docTTTTTquery. (2019). https://cs.uwaterloo.ca/~jimmylin/publications/Nogueira_Lin_2019_docTTTTTquery-latest.pdf.

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

[60] Jeremy Pickens, Matthew Cooper, and Gene Golovchinsky. 2010. Reverted indexing for feedback and expansion. In *Proc. CIKM*. 1049–1058.

[61] Ronak Pradeep, Yuqi Liu, Xinyu Zhang, Yilin Li, Andrew Yates, and Jimmy Lin. 2022. Squeezing Water from a Stone: A Bag of Tricks for Further Improving Cross-Encoder Effectiveness for Reranking. In *Proc. ECIR*. 655–670.
[62] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. In *Proc. NeurIPS*.
[63] 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, 140 (2020), 1–67.
[64] Stephen Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Found. Trnd. Inf. Retr.* 3, 4 (2009), 333–389.
[65] Tetsuya Sakai. 2016. Statistical Significance, Power, and Sample Sizes: A Systematic Review of SIGIR and TOIS, 2006-2015. In *Proc. SIGIR*. 5–14.
[66] Harrisen Scells, Shengyao Zhuang, and Guido Zuccon. 2022. Reduce, Reuse, Recycle: Green Information Retrieval Research. In *Proc. SIGIR*. 2825–2837.
[67] Falk Scholer, Hugh E. Williams, and Andrew Turpin. 2004. Query association surrogates for Web search. *J. Assoc. Inf. Sci. Technol.* 55, 7 (2004), 637–650.
[68] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).
[69] Jheng-Hong Yang Sheng-Chieh Lin and Jimmy Lin. 2021. In-Batch Negatives for Knowledge Distillation with Tightly-Coupled Teachers for Dense Retrieval. In *Proc. Wkshp. on Representation Learning for NLP*. 163–173.
[70] Tao Tao, Xuanhui Wang, Qiaozhu Mei, and ChengXiang Zhai. 2006. Language Model Information Retrieval with Document Expansion. In *Proc. HLT-NAACL*. 407–414.
[71] Nandan Thakur, Nils Reimers, and Jimmy Lin. 2023. Injecting Domain Adaptation with Learning-to-hash for Effective and Efficient Zero-shot Dense Retrieval. In *Proc. ReNeuIR at SIGIR 2023*.
[72] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models. In *Proc. NeurIPS*.
[73] Nicola Tonellotto, Craig Macdonald, and Iadh Ounis. 2018. Efficient Query Processing for Scalable Web Search. *Found. Trnd. Inf. Retr.* 12, 4–5 (2018), 319–500.
[74] Howard R. Turtle and James Flood. 1995. Query Evaluation: Strategies and Optimizations. *Inf. Proc. & Man.* 31, 6 (1995), 831–850.
[75] Ellen M. Voorhees. 2004. Overview of the TREC 2004 Robust Retrieval Track. In *Proc. TREC*.
[76] Ellen M. Voorhees, Tasmeer Alam, Steven Bedrick, Dina Demner-Fushman, William R. Hersh, Kyle Lo, Kirk Roberts, Ian Soboroff, and Lucy Lu Wang. 2021. TREC-COVID: Constructing a pandemic information retrieval test collection. *SIGIR Forum* 54, 1 (2021), 1–12.
[77] Lidan Wang, Jimmy Lin, and Donald Metzler. 2011. A Cascade Ranking Model for Efficient Ranked Retrieval. In *Proc. SIGIR*. 105–114.
[78] Shuai Wang, Shengyao Zhuang, and Guido Zuccon. 2021. BERT-based Dense Retrievers Require Interpolation with BM25 for Effective Passage Retrieval. In *Proc. ICTIR*. 317–324.
[79] Orion Weller, Kyle Lo, David Wadden, Dawn Lawrie, Benjamin Van Durme, Arman Cohan, and Luca Soldaini. 2023. When do Generative Query and Document Expansions Fail? A Comprehensive Study Across Methods, Retrievers, and Datasets. *arXiv preprint arXiv:2309.08541* (2023).
[80] Thijs Westerveld, Wessel Kraaij, and Djoerd Hiemstra. 2002. Retrieving Web Pages using Content, Links, URLs and Anchors. In *Proc. TREC*. 663–672.
[81] Peilin Yang, Hui Fang, and Jimmy Lin. 2018. Anserini: Reproducible Ranking Baselines Using Lucene. *J. Data Inf. Qual.* 10, 4 (2018).
[82] Ziying Yang, Alistair Moffat, and Andrew Turpin. 2016. How Precise Does Document Scoring Need to Be?. In *Proc. Asia Info. Retri. Soc. Conf.* 279–291.
[83] Le Zhao and Jamie Callan. 2010. Term necessity prediction. In *Proc. CIKM*. 259–268.
[84] Shengyao Zhuang, Honglei Zhuang, Bevan Koopman, and Guido Zuccon. 2023. A Setwise Approach for Effective and Highly Efficient Zero-shot Ranking with Large Language Models. *arXiv preprint arXiv:2310.09497* (2023).
[85] Shengyao Zhuang and Guido Zuccon. 2021. Dealing with Typos for BERT-based Passage Retrieval and Ranking. In *Proc. EMNLP*. 2836–2842.
[86] Shengyao Zhuang and Guido Zuccon. 2021. TILDE: Term Independent Likelihood moDEl for Passage Re-ranking. In *Proc. SIGIR*. 1483–1492.
[87] Shengyao Zhuang and Guido Zuccon. 2022. Fast Passage Re-ranking with Contextualized Exact Term Matching and Efficient Passage Expansion. In *Proc. ReNeuIR at SIGIR*.
[88] Justin Zobel and Alistair Moffat. 2006. Inverted files for text search engines. *ACM Comp. Surv.* 38, 2 (2006), 6.1–6.56.