Fast and Effective Early Termination for Simple Ranking Functions

Jinrui Gou
New York University
New York, US
jg6226@nyu.edu

Antonio Mallia
Pinecone
New York, US
antonio@pinecone.io

Yifan Liu†
University of California, Los Angeles
Los Angeles, US
bmmliu@ucla.edu

Minghao Shao
New York University
New York, US
ms12416@nyu.edu

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

Abstract

Web search engines often perform an initial candidate generation phase using a fast and simple ranking function, followed by subsequent reranking with more expensive rankers. Such simple ranking functions usually compute the score of a document as the sum of term-wise impact scores, and they include traditional baselines such as BM25 and Query Likelihood, as well as some recently proposed learned sparse models based on document expansion and learned impact scores. In this paper, we explore extremely fast and highly effective early termination techniques for such simple ranking functions. Our extensive experiments with a number of different ranking functions show that our methods achieve very fast response times on MSMarco V1 and V2 data while maintaining retrieval quality close to that of a safe and much slower baseline.

CCS Concepts

• **Information systems** → **Information retrieval**.

Keywords

query processing, early termination, learned sparse indexing

ACM Reference Format:
Jinrui Gou, Antonio Mallia, Yifan Liu†, Minghao Shao, and Torsten Suel. 2025. Fast and Effective Early Termination for Simple Ranking Functions. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '25), July 13–18, 2025, Padua, Italy*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3726302.3730197

1 Introduction

A lot of research has focused on the efficiency of search engine query processing, and in particular the processing of disjunctive queries over inverted index structures. This has led to many highly optimized safe and unsafe early termination algorithms that minimize the postings in the index that are accessed and scored to return results. Recent breakthroughs on transformers and other deep neural networks have resulted in new *learned sparse models*, such as Splade [9–11, 14], DeepImpact [21], uniCoil [15], or Tilde [44], that significantly outperform rankers such as BM25 and query likelihood and provide strong baselines for candidate generation in state-of-the-art search systems. These new models directly map to the quantized inverted index structures used in many engines.

However, due to the different impact score distributions in these new models, many well-known optimized top-k algorithms such as MaxScore [37], WAND [3], and BMW [7, 23, 24] perform poorly in this new scenario [13, 20, 21]. This has motivated recent work on finding more suitable algorithms for these models, including Guided Traversal (GT) [22, 33, 34], Clipping [18], and Seismic [4].

In this paper, we explore a method for fast early termination on a class of *simple ranking functions* that covers traditional rankers such as Query Likelihood and BM25, including when document expansion techniques such as DocT5Query are applied, as well as newer learned sparse models such as DeepImpact, DeeperImpact, and Tilde. Our method follows the approach in [39], where a prefix index is built that allows score-at-a-time access on terms and term pairs, with subsequent lookups into a second-tier standard inverted index to resolve unknown term scores. Our experiments show that the method achieves a retrieval quality close to an exhaustive computation across a number of simple ranking functions while achieving much lower response times than the state-of-the-art.

2 Background and Related Work

Inverted Indexes and Simple Ranking Functions: We assume a collection of documents $D = \{d_0, \ldots, d_{n-1}\}$. Documents may have already been expanded with additional terms using document expansion techniques such as DocT5Query [31] or Tilde [44]. A quantized inverted index for D contains an inverted list for each term in the collection. An inverted list for term t consists of postings, where each posting contains a document ID (docID) and an impact score that models the relevance of the document to term t. Impact scores can be computed in arbitrary ways at indexing time, including inference via contextualized language models (LMs), usually quantized to one byte of precision for efficiency.

†Work done while the author was at New York University.

SIGIR '25, Padua, Italy

ACM ISBN 979-8-4007-1592-1/2025/07
https://doi.org/10.1145/3726302.3730197

We focus on simple ranking functions [29], where the score of a document d with respect to a query q is defined as $\sum_{t \in q} is(t, d)$ where $is(t, d)$ is the impact score of document d stored in the inverted list for term t, We note that this definition includes ranking functions such as BM25 [35] and query likelihood [43], as well as recent methods based on learned sparse indexing.

Learned Sparse Models: Recent work has proposed methods to create inverted index structures using pre-trained LMs; see [2, 6, 11, 12, 14, 15, 21, 31, 32, 42, 44]. One common approach performs document expansion to address the vocabulary mismatch [2, 6, 12, 21, 31, 32, 44]. The impact scores in the resulting index are then computed via traditional ranking formulas such as BM25 or inferred using a specially trained neural network [2, 21, 44], resulting in a simple ranking function as defined above. Another approach [11, 14, 15] maps both queries and documents into a high-dimensional but sparse space that can then be indexed using inverted lists; this leads to query-dependent term weights and is thus not a simple ranking problem. We defer this extension to future work.

Fast Early Termination Algorithms: There are many optimized algorithms for disjunctive top-k query processing on inverted indexes, where the goal is to return the highest scoring k documents containing at least one of the query terms. Safe methods, which always return the same top-k results as an exhaustive evaluation, include MaxScore [37], WAND [3], block-max methods [7, 23, 27, 28], and the FA and TA algorithms widely used in databases [8]. Many unsafe methods, not guaranteed to return the same results but often faster than safe ones, have also been proposed; of particular interest in our context are score-at-a-time methods that sort and access postings in inverted lists by their term impacts [1, 5, 17, 19, 36, 39]. As observed in [18, 20, 21], many existing early termination techniques are not efficient on the types of score distributions arising in learned sparse models, motivating our search for a better approach.

Overall, the most closely related previous approaches are the FA algorithm of Fagin [8], the candidate generation framework in [39], and the JASS algorithm in [17, 36]. In particular, our work here uses the same basic approach as [39], but simplifies it and applies it to the new ranking functions arising from learned sparse models. The approach in [39] in turn builds on top of the FA algorithm [8], but adds sorted access to pairs of terms, a different access order, and a budget-based cutoff in contrast to the safe termination condition used in FA. Finally, our work is related to the score-at-a-time JASS method and basically suggests that JASS could be significantly improved by adding pairs of terms and random lookups.

3 Description of our Approach: SPRAWL

We now describe our method in detail. As discussed, we adapt an approach in [39], which also builds a two-tier index consisting of a prefix index with postings for certain terms and term pairs sorted by impact score, and a standard inverted index as the second tier. At query time, a certain number of postings from the prefix are accessed and accumulated, and then lookups are issued for the missing term scores of the top results. Our method, called *Sorted PRefix Access With Lookups* (SPRAWL), works as follows:

At Indexing Time: Build a prefix index by selecting certain terms and term pairs based on a training query log, deciding for each how many postings p to store in its prefix, and issuing a top-p conjunctive query for the term or term pair. The resulting up to p postings are then stored in the corresponding prefix structure in order of descending total impact score, where for a term pair posting we store the docID and both term impact scores.

At Query Time: Assign an access budget ab and a lookup budget lb to query q, and determine the prefix structures relevant to q, i.e., for terms or term pairs contained in q. Then process q as follows:

(1) Select a total of ab postings from the relevant prefix structures based on highest total impact score (i.e., the sum of two impact scores for a term pair posting), using a heap.
(2) Aggregate the selected postings by docID to obtain partial document scores, using a hash table.
(3) Perform lookups into the inverted index for the missing term scores of the lb documents with the highest partial scores. Lookups into each inverted list are performed in sorted order by docID to optimize performance.
(4) The top results are returned as query results.

The above method follows [39], but with some changes. Instead of training a posting quality model as in [39], we select prefix structures at indexing time based on a training log, and select postings at query time based on impact scores, inspired by recent score-at-a-time methods such as JASS [17, 36]. Aggregation of postings is done using a simple hash table instead of a sorting-based approach. Preliminary experiments showed that these changes had almost no impact on performance, while significantly simplifying the method.

4 Experimental Results

Data. We evaluate our approach on two widely used MS MARCO passage collections. The first dataset, referred to as MS MARCO V1, consists of 8.8 million passages, while the second dataset, MS MARCO V2, contains 138 million passages.

Ranking Models. To assess the performance of our approach, we compare several ranking functions that employ different scoring mechanisms and expansion strategies:

(1) **BM25** based on the PISA [25] implementation with parameters $k_1 = 0.9$ and $b = 0.4$.
(2) **DocT5Query**: we use the predicted queries available online, using 40 concatenated predictions for each passage, as recommended by [31]; documents are scored with BM25.
(3) **DeepImpact**: using DocT5Query document expansions and learned impact scores as in [21].
(4) **DeeperImpact**: using Llama 2 document expansions and improvements to impact score prediction as in [2].
(5) **DeeperImpact Tilde**: using Tilde [44] instead of Llama 2.

Due to space constraints, we only present a subset of our results. In most figures and tables, we use *DeepImpact* as a default baseline for comparison. Note that our approach currently does not support methods such as *Splade* [11] or *uniCOIL* [15], as these use query term weights, which our framework does not support.

Evaluation Measures. Our evaluations use multiple metrics to cover effectiveness and efficiency aspects. We focus on nDCG@10, recall@1000, and Rank-Biased Overlap (RBO) [30, 40] to capture different facets of ranking performance, providing insights into how well our approach ranks the most relevant passages at the top and how comprehensively it retrieves relevant passages overall. For efficiency, we measure the mean response time (MRT) to quantify average query latency, and report speed-ups relative to a strong

common baseline, the safe MaxScore implementation in the PISA toolkit [25], which was also used in previous work on Clipping [18] and Guided Traversal [22], thus allowing for a fair comparison.

Index Construction: We use Anserini [41] to create indexes and export them into Common Index File Format (CIFF) [16]. The compressed inverted index for all approaches was build in PISA [25]. Indexes for MaxScore are compressed with SIMD-BP128 following [26], while for SPRAWL the lookup index uses Elias-Fano [38]. We explored various prefix configurations by adjusting the types of prefixes constructed and the maximum construction depth. The default settings include a smaller prefix for MS MARCO V1 and a large one for MS MARCO V2, with prefix sizes slightly larger than inverted index size for V1, and smaller than the index for V2. The sizes of inverted indexes and prefixes vary depending on the document expansion performed. Detailed configuration settings and implementations are available at https://github.com/pisa-engine/sprawl.

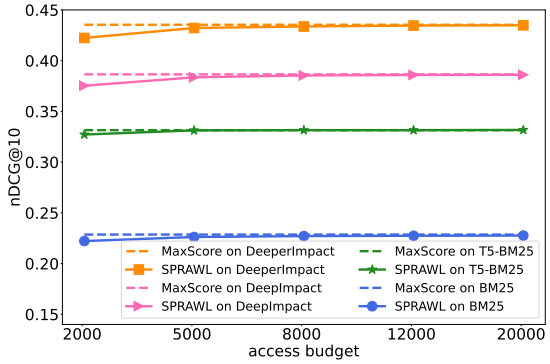

Figure 1: nDCG@10 for 6980 dev queries on different ranking functions, for MS MARCO V1 and with full lookups.

Performance for different Rankers: Figures 1, 2, and 3 show the nDCG@10, recall@1000, and RBO measures, respectively, for four different rankers, BM25, BM25 with DocT5Query expansions, DeepImpact, and DeeperImpact, for different values of the access budget *ab* on MS MARCO V1 with full lookups. We also show retrieval performances for exhaustive methods as horizontal lines. Behavior is fairly similar for all four ranking functions, with performance very close to the exhaustive baseline for larger values of *ab*. For nDCG@10, even small access budgets around 5000 almost match the baseline, while for recall@1000, larger budgets are needed. Overall our approach works well across a range of simple ranking functions, especially works well for the type of learned impact score distributions, such as DeepImpact, where techniques such as MaxScore and WAND struggle [20].

Multi-Term Structures: One main difference between our approach and others such as JASS [17, 36] and Fagin's algorithm [8] is the use of pairwise score-at-a-time access structures. In Figure 4 we justify this choice by showing results for nDCG@10 on DeepImpact, for three cases: (1) only single-term prefix structures, (2) single-term and pairwise structures, and (3) structures for up to three terms. To do this, we created a very large prefix index containing sufficiently deep prefixes for all terms, pairs of terms, and triples that appear in the training trace; thus, in case (3) we allow much more space for the prefix than in (1) and (2). Our results show that pairwise prefixes give a significant boost over single terms, particularly for moderate access budgets, and that triples gives only

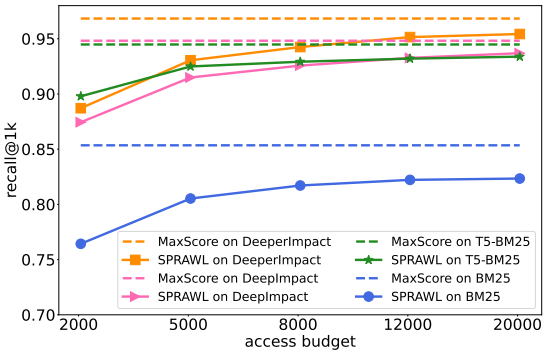

Figure 2: recall@1000 for 6980 dev queries on different ranking functions, for MS MARCO V1 and with full lookups.

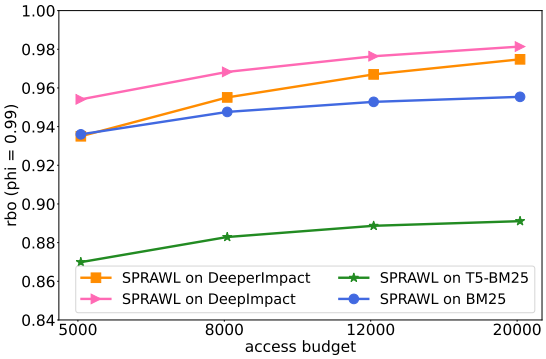

Figure 3: RBO vs. exhaustive methods for 6980 dev queries on different ranking functions, for MS MARCO V1 and with full lookups.

minuscule benefits even with much higher space use. Results were similar for other ranking functions and retrieval measures.

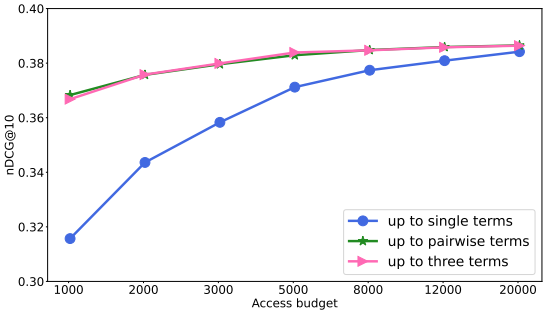

Figure 4: nDCG@10 for 6980 dev queries on DeepImpact, for MS MARCO V1 with different types of prefix structures.

Limiting Lookups: An important difference to score-at-a-time methods such as JASS is the use of lookups to resolve missing scores. Figure 5 shows the effect on nDCG@10 of varying the number of lookups that are performed, from no lookups, to lookups for the 50% docIDs with the highest partial scores, to full lookups, for DeepImpact. We see that lookups are crucial for good retrieval quality, and that full lookups are best. Again, we observed similar behavior for other ranking function and retrieval measures.

Impact of Query Length: Next, we look at the performance of our approach for different query lengths. Table 1 compares our method to MaxScore under DeepImpact ranking. As we see, our method achieves basically identical retrieval quality for queries up

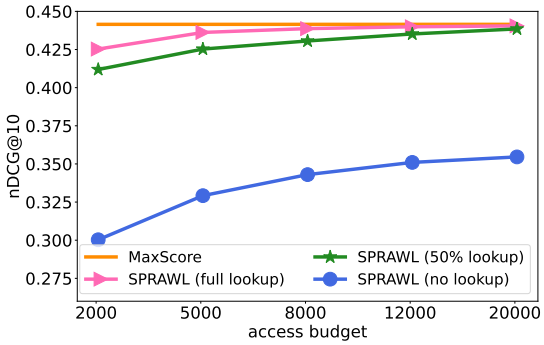

Figure 5: nDCG@10 for 6980 dev queries on DeepImpact, for MS MARCO V1 with different budgets for lookups.

Table 1: nDCG@10, Recall@1k, and Recall@100 for different query lengths, with DeepImpact as ranking functions.

Query Length	2-	3	4	5,6	7+	total
# Queries	797	1715	1980	1984	504	6980
SPRAWL$_{10k,5k}$						
nDCG@10	0.4559	0.4051	0.3773	0.3555	0.3411	0.3843
Recall@1k	0.9322	0.9417	0.9215	0.9018	0.8681	0.9183
Recall@100	0.9021	0.8569	0.8214	0.7932	0.751	0.8263
MaxScore						
nDCG@10	0.4562	0.4052	0.3787	0.3593	0.3508	0.3865
Recall@1k	0.9385	0.9512	0.9477	0.9516	0.9415	0.9482
Recall@100	0.9034	0.8613	0.8345	0.8194	0.7966	0.8419

Table 2: nDCG@10 and recall@1k for BM25 and DeeperImpact with Tilde expansions on MS MARCO V2, using full lookups.

		BM25			Deeper-Tilde		
		DL21	DL22	DL23	DL21	DL22	DL23
nDCG@10	MaxScore	0.4285	0.2045	0.2056	0.5253	0.4104	0.3418
	SPRAWL$_{10k}$	0.4087	0.2036	0.1961	0.5038	0.4059	0.3247
	SPRAWL$_{20k}$	0.4262	0.2036	0.1945	0.5027	0.4074	0.3277
	SPRAWL$_{50k}$	0.4303	0.2043	0.1954	0.5142	0.4079	0.3328
Recall@1k	MaxScore	0.5782	0.2741	0.3653	0.6035	0.3906	0.4195
	SPRAWL$_{10k}$	0.4461	0.2143	0.2754	0.4452	0.3149	0.3165
	SPRAWL$_{20k}$	0.4938	0.2381	0.3033	0.4836	0.3374	0.3345
	SPRAWL$_{50k}$	0.5257	0.2479	0.3303	0.5235	0.3627	0.3648

to length 4, while there is a slight decrease in quality versus MaxScore for longer queries. This is to be expected, as early termination techniques tend to decrease in retrieval quality or efficiency for longer queries, due to the increased dimensionality of the problem.

Scaling up to MS MARCO V2: Next, we evaluate performance on the much larger MS MARCO V2 passages. We look at two ranking functions, BM25 and a version of DeeperImpact that uses Tilde instead of Llama 2 for document expansion, chosen because LLama 2 would have been very expensive to run on this larger data sets, while Tilde achieves almost the same quality at much lower costs.

The results in Table 2 for DL21, DL22, and DL23 queries show that SPRAWL scales well to larger collections. Of course, *ab* and *lb* need to be increased, resulting in increased MRT, as is also the case for the MaxScore baseline. Again results are better for nDCG@10 than for recall@1000, where there is more of a gap to the baseline.

Efficiency and Comparison to Previous Work: Next, we compare actual running times, first for different settings of SPRAWL and then compared to existing state-of-the-art methods.

In Figure 6 we see the MRT of our method for the four ranking functions on MS MARCO V1, under varying access budgets *ab* with full lookups. MRT ranges from less than 0.5ms to 2.5*ms* for

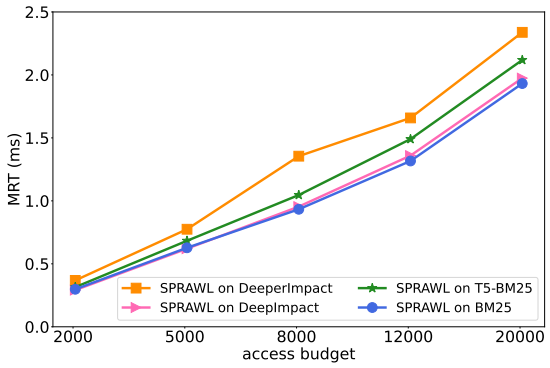

Figure 6: MRT for 6980 dev queries on different ranking functions, for MS MARCO V1 and with full lookups.

Table 3: Comparison of SPRAWL to previous optimized methods, for DeepImpact on Marco v1, using full lookups for SPRAWL.

	MRR@10	MRT (ms)	Speedup
MaxScore	0.3273	13.63	-
Clipping	0.3273	-	3.2
GT	0.3260	-	4.1
SPRAWL$_{2k}$	0.3190	0.29	47.0
SPRAWL$_{5k}$	0.3253	0.63	21.6

Table 4: Comparison of SPRAWL to previous optimized methods, for BM25 on Marco V2, using full lookups for SPRAWL.

	MRR@10	MRT (ms)	Speedup
MaxScore	0.0548	35.8	-
Clipping	0.0548	-	1.26
SPRAWL$_{20k}$	0.0523	3.50	10.2
SPRAWL$_{50k}$	0.0537	7.33	4.9

this range of *ab*, while achieving very good retrieval quality. We use about 40ns to process a posting from a prefix – slightly more for term-pair postings and slightly less for single terms – while a lookup into an inverted list takes between 30 and 80ns, depending on the distance between accesses into a list. Resolving all missing scores for one document requires on average between 1.5 and 2 lookups as there may be several missing scores. Overall, in methods with full lookups, up to two thirds of time is spent on lookups.

We now compare to Clipping and Guided Traversal on both DeepImpact and BM25 ranking models. Table 3 shows results for MS MARCO v1 using DeepImpact, while Table 4 shows a comparison with Clipping on MS MARCO v2 using BM25. Both tables presents speedups relative to the same PISA MaxScore implementation. We observe that SPRAWL achieves up to an order-of-magnitude faster retrieval while introducing only minimal reduction in quality.

5 Concluding Remarks

In this paper, we have proposed and evaluated an early-termination method called SPRAWL for simple ranking functions that achieves a very good tradeoff between efficiency and retrieval effectiveness. There are several possibly improvements and extensions that we plan to pursue. In particular, we plan to extend the method to models such as Splade and uniCOIL that use query term weights.

References

[1] Vo Ngoc Anh and Alistair Moffat. 2002. Impact transformation: effective and efficient web retrieval. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[2] Soyuj Basnet, Jerry Gou, Antonio Mallia, and Torsten Suel. 2024. DeeperImpact: Optimizing Sparse Learned Index Structures. *arXiv preprint arXiv:2405.17093* (2024).

[3] Andrei Z. Broder, David Carmel, Michael Herscovici, Aya Soffer, and Jason Zien. 2003. Efficient query evaluation using a two-level retrieval process. In *Proc. CIKM*.

[4] Sebastian Bruch, Franco Maria Nardini, Cosimo Rulli, and Rossano Venturini. 2024. Efficient inverted indexes for approximate retrieval over learned sparse representations. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 152–162.

[5] Matt Crane, J Shane Culpepper, Jimmy Lin, Joel Mackenzie, and Andrew Trotman. 2017. A comparison of document-at-a-time and score-at-a-time query evaluation. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. 201–210.

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

[7] Shuai Ding and Torsten Suel. 2011. Faster Top-k Document Retrieval Using Block-max Indexes. In *Proc. SIGIR*.

[8] R. Fagin, A. Lotem, and M. Naor. 2001. Optimal Aggregation Algorithms for Middleware. In *Proc. SIGMOD*.

[9] Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE v2: Sparse lexical and expansion model for information retrieval. *arXiv preprint arXiv:2109.10086* (2021).

[10] Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. 2022. From distillation to hard negative sampling: Making sparse neural ir models more effective. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*. 2353–2359.

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

[12] Mitko Gospodinov, Sean MacAvaney, and Craig Macdonald. 2023. Doc2Query–: when less is more. In *European Conference on Information Retrieval*. Springer, 414–422.

[13] Carlos Lassance and Stéphane Clinchant. 2022. An efficiency study for SPLADE models. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2220–2226.

[14] Carlos Lassance, Hervé Déjean, Thibault Formal, and Stéphane Clinchant. 2024. SPLADE-v3: New baselines for SPLADE. *arXiv preprint arXiv:2403.06789* (2024).

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

[16] 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 *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2149–2152.

[17] Jimmy Lin and Andrew Trotman. 2015. Anytime ranking for impact-ordered indexes. In *Proceedings of the 2015 International Conference on The Theory of Information Retrieval*. 301–304.

[18] Joel Mackenzie, Antonio Mallia, Alistair Moffat, and Matthias Petri. 2022. Accelerating learned sparse indexes via term impact decomposition. In *Proc. EMNLP*.

[19] Joel Mackenzie, Matthias Petri, and Luke Gallagher. 2022. IOQP: A simple Impact-Ordered Query Processor written in Rust. In *CEUR Workshop Proceedings*, Vol. 3480. Rheinisch-Westfaelische Technische Hochschule Aachen, 22–34.

[20] 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 Trans. Inf. Syst.* (2023).

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

[22] Antonio Mallia, Joel Mackenzie, Torsten Suel, and Nicola Tonellotto. 2022. Faster learned sparse retrieval with guided traversal. In *Proc. SIGIR*.

[23] A. Mallia, G. Ottaviano, E. Porciani, N. Tonellotto, and R. Venturini. 2017. Faster BlockMax WAND with Variable-Sized Blocks. In *Proc. SIGIR*.

[24] Antonio Mallia and Elia Porciani. 2019. Faster BlockMax WAND with longer skipping. In *Advances in Information Retrieval: 41st European Conference on IR Research, ECIR 2019, Cologne, Germany, April 14–18, 2019, Proceedings, Part I 41*. Springer, 771–778.

[25] Antonio Mallia, Michal Siedlaczek, Joel Mackenzie, and Torsten Suel. 2019. PISA: Performant indexes and search for academia. In *Proc. OSIRRC@SIGIR*.

[26] Antonio Mallia, Michał Siedlaczek, and Torsten Suel. 2019. An experimental study of index compression and DAAT query processing methods. In *Advances in Information Retrieval: 41st European Conference on IR Research, ECIR 2019, Cologne, Germany, April 14–18, 2019, Proceedings, Part I 41*. Springer, 353–368.

[27] Antonio Mallia, Michał Siedlaczek, and Torsten Suel. 2021. Fast Disjunctive Candidate Generation Using Live Block Filtering. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*.

[28] Antonio Mallia, Torsten Suel, and Nicola Tonellotto. 2024. Faster learned sparse retrieval with block-max pruning. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2411–2415.

[29] Bhaskar Mitra, Corby Rosset, David Hawking, Nick Craswell, Fernando Diaz, and Emine Yilmaz. 2019. Incorporating query term independence assumption for efficient retrieval and ranking using deep neural networks. *arXiv preprint arXiv:1907.03693* (2019).

[30] Alistair Moffat, Joel Mackenzie, Antonio Mallia, and Matthias Petri. 2024. Rank-Biased Quality Measurement for Sets and Rankings. In *Proceedings of the 2024 Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region*. 135–144.

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

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

[33] Yifan Qiao, Yingrui Yang, Haixin Lin, Tianbo Xiong, Xiyue Wang, and Tao Yang. 2022. Dual Skipping Guidance for Document Retrieval with Learned Sparse Representations. *arXiv preprint arXiv:2204.11154* (2022).

[34] Yifan Qiao, Yingrui Yang, Haixin Lin, and Tao Yang. 2023. Optimizing guided traversal for fast learned sparse retrieval. In *Proceedings of the ACM Web Conference 2023*. 3375–3385.

[35] Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends® in Information Retrieval* 3, 4 (2009), 333–389.

[36] Andrew Trotman and Matt Crane. 2019. Micro-and macro-optimizations of SaaT search. *Software: Practice and Experience* 49, 5 (2019), 942–950.

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

[38] Sebastiano Vigna. 2013. Quasi-succinct indices. In *Proc. WSDM*.

[39] Qi. Wang, C. Dimopoulos, and T. Suel. 2016. Fast First-Phase Candidate Generation for Cascading Rankers. In *Proc. SIGIR*.

[40] William Webber, Alistair Moffat, and Justin Zobel. 2010. A similarity measure for indefinite rankings. *ACM Trans. Inf. Syst.* (2010).

[41] Peilin Yang, Hui Fang, and Jimmy Lin. 2017. Anserini: Enabling the use of lucene for information retrieval research. In *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval*. 1253–1256.

[42] Puxuan Yu, Antonio Mallia, and Matthias Petri. 2024. Improved Learned Sparse Retrieval with Corpus-Specific Vocabularies. In *European Conference on Information Retrieval*. Springer, 181–194.

[43] ChengXiang Zhai and John Lafferty. 2009. *Statistical Language Models for Information Retrieval*. Morgan & Claypool Publishers.

[44] Shengyao Zhuang and Guido Zuccon. 2021. TILDE: Term independent likelihood moDEl for passage re-ranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1483–1492.