

Accelerating Learned Sparse Indexes Via Term Impact Decomposition*

Discussion Paper

Joel Mackenzie¹, Antonio Mallia², Alistair Moffat³ and Matthias Petri²

¹University of Queensland, Australia

²Amazon Alexa

³University of Melbourne, Australia

Abstract

Novel inverted index-based *learned sparse ranking models* provide more effective, but less efficient, retrieval performance compared to traditional ranking models like BM25. In this paper, we introduce a technique we call *postings clipping* to improve the query efficiency of learned representations. Our technique amplifies the benefit of dynamic pruning query processing techniques by accounting for changes in term importance distributions of learned ranking models. The new clipping mechanism accelerates top- k retrieval without any loss in effectiveness.

1. Introduction

Sparse term importance representations such as DeepImpact [2] and uniCOIL [3, 4] have enabled the use of effective transformer-based text representations that can match the effectiveness of recent dense text representations [5, 6] while still being supported by *inverted indexes* and their query operations. This is of importance as inverted indexes have been optimized to provide search functionality in distributed settings at web-scale through 40 years of research, providing a variety of time, space, and retrieval quality tradeoffs; while also supporting efficient updates, advanced querying modes such as phrase matching or filtering, and good scalability, all of which are crucial in real-world settings [7, 8].

One of the key indexing techniques that enables efficient top- k query processing is storing additional metadata about index term importance scores (also referred to as *impacts*), seeking to facilitate the bypassing of the majority of the matching documents, and thus allow faster retrieval than would be possible via exhaustive disjunctive processing. For example, dynamic pruning algorithms such as MaxScore [9] and WAND [10] store the index-wide maximum impact of each term; at query-time, these impacts can be used to rapidly estimate document scores, allowing documents that have no prospect of entering the current min-heap of k results to be bypassed.

IIR2023: 13th Italian Information Retrieval Workshop, 8th - 9th June 2023, Pisa, Italy


*This work is an extended abstract of [1].

This work was in part supported by the Australian Research Council (Project DP200103136).

✉ joel.mackenzie@uq.edu.au (J. Mackenzie); malliaam@amazon.com (A. Mallia); ammoffat@unimelb.edu.au (A. Moffat); mkp@amazon.com (M. Petri)



© 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

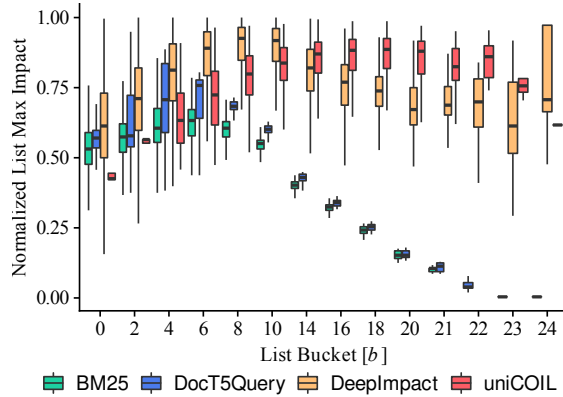


Figure 1: Normalized maximum list impact distribution stratified by list length buckets $b \in [2^b, 2^{b+1})$.

Traditional similarity models such as BM25 guarantee that frequent terms have low importance scores, a symbiotic relationship that allows fast query processing. On the other hand, recent transformer-based learned term importance techniques such as DeepImpact [2] are not constrained by term occurrence frequency when assigning importance scores to terms in documents.

Indeed, learned representations assign high term importance to even very frequent terms, whereas BM25 always assigns low importance to such terms. This divergent behavior substantially reduces the ability of MaxScore and WAND to bypass low-impact documents during querying, with both techniques relying on maximum list-wise impact scores to prune the search space. Figure 1 highlights the pervasive nature of this issue.

We present a new form of impact decomposition that we call *postings clipping* and adapt dynamic pruning mechanisms to enable efficient query processing for learned term importance schemes. When integrated into the retrieval engine, postings clipping allows faster top- k term-based querying, with negligible increases to index storage costs, and no effect on result quality.

2. Postings Clipping

Several authors have proposed explicitly or implicitly splitting postings lists into two (or more) parts, a high-impact segment $\mathcal{H}(t)$ and a low-impact segment $\mathcal{L}(t)$ to facilitate efficient processing; see, for example, Strohman and Croft [11], Ding and Suel [12], Daoud et al. [13], Daoud et al. [14], Kane and Tompa [15] and Mackenzie et al. [16].

Our proposal – denoted *postings clipping* – is illustrated in Figure 2. Rather than partitioning the set of postings in \mathcal{I}_t across $\mathcal{L}(t)$ and $\mathcal{H}(t)$, every posting remains in $\mathcal{L}(t)$, and we “clip” the high-impact postings by slicing them into two parts, and forming a *posting pair*. The base part remains in $\mathcal{L}(t)$ as a posting with an impact equal to $U_{\mathcal{L}(t)}$, the maximum score contribution permitted in $\mathcal{L}(t)$; and the second component of the pair becomes a new posting in $\mathcal{H}(t)$, to account for the “trimmed” part of the original impact value, and retain the same total.

This arrangement has the singular advantage of no additional upper bounds management

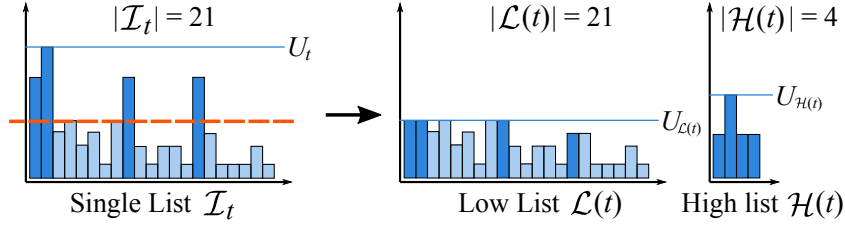


Figure 2: *Postings clipping* involves trimming the impact scores in the low-impact list, and creating new postings in a separate list $\mathcal{H}(t)$ to account for the remainder.

(smart bounds) being needed, nor equivalent run-time manipulation of score estimates. Such adjustments in the list splitting approach of Kane and Tompa [15] to correct for the constraint that no document can appear in both $\mathcal{L}(t)$ and $\mathcal{H}(t)$, and hence that $U_{\mathcal{L}(t)} + U_{\mathcal{H}(t)}$ is an over-estimate (by an addend of $U_{\mathcal{L}(t)}$) of t 's true upper bound U_t . With postings clipping, $U_{\mathcal{H}(t)}$ is instead set to the maximum *residual* amount across all of t 's postings, and hence we have $U_t = U_{\mathcal{L}(t)} + U_{\mathcal{H}(t)}$. In turn, that means that when queries are being processed the lists $\mathcal{L}(t)$ and $\mathcal{H}(t)$ can be treated as if they were derived from completely independent terms, with all interactions between them handled by the underlying processing logic, be that MaxScore, WAND, or BMW. That is, while there are more total postings to be stored and processed, the change from list splitting with smart bounds to postings clipping substantially simplifies the query-time processing logic.

As an orthogonal enhancement, *priming* can be applied whenever any high-impact list contains k or more postings, $|\mathcal{H}(t)| \geq k$. If that holds, then

$$\theta_0 = \max\{U_{\mathcal{L}(t)} \mid t \in Q \wedge |\mathcal{H}(t)| \geq k\} \quad (1)$$

can be used as an initial value for the heap bound, without risking top- k integrity. In combination, the result is that – as we demonstrate in Section 3 – quite dramatic reductions in query processing times for learned sparse retrieval models can be achieved.

3. Experiments

We now describe experiments that quantify the benefits arising from the postings clipping approach. Our experiments make use of both MSMARCO-v1 (8.8 million passages) and MSMARCO-v2 (138.4 million passages) collections, four representative ranking algorithms, and the PISA [17] query processing system which was recently shown to outperform the commonly used Anserini system for document-at-a-time retrieval over learned sparse indexes [18].

Index Size Since clipping is applied only to postings with more than 256 elements, and even then only adds 1/64 as many new postings, the space overhead compared to the default index is negligible. For instance, the largest overhead of 600 MiB to the ≈ 33 GiB index for the uniCOIL model on MSMARCO-v2 represents an increase of only 1.8%.

Query Speed Table 1 presents query processing times recorded for the MSMARCO-v1 collection and DeepImpact retrieval, with response latency measured as average milliseconds per

Table 1

Query processing times, all in average milliseconds per query, for the MSMARCO-v1 collection and the DeepImpact retrieval model. Similar relativities were also observed in regard to median query times, and 90% and 99% tail latency query times.

Method	$k = 10$	$k = 1000$
MaxScore baseline	8.1	18.8
+ postings clipping	1.6	5.9
WAND baseline	14.9	34.0
+ postings clipping	2.7	10.8
VBMW baseline	4.2	12.2
+ postings clipping	3.3	9.7

query, and with the three blocks of values corresponding to dynamic query pruning approaches.

The last row in each block shows the combination of postings clipping, with $1/64$ postings taken into $\mathcal{H}(t)$, in conjunction with priming (and length-based ordering for MaxScore). The fastest query time in each of the six sections is highlighted in blue.

As can be seen, for DeepImpact retrieval, the fastest querying is achieved by MaxScore pruning with postings clipping. That combination takes less than half the time of standard MaxScore processing. The gains from posting clipping are less for WAND and VBMW, in part because both algorithms exhibit greater sensitivity to doubling the number of query terms.

Table 2

Query processing times, all in average milliseconds per query, for the MSMARCO-v2 collection.

Method	BM25		DocT5Query		DeepImpact		uniCOIL	
	$k = 10$	1000	$k = 10$	1000	$k = 10$	1000	$k = 10$	1000
MaxScore baseline	11.0	38.7	8.8	28.2	828.0	1170.4	164.9	267.9
+ postings clipping	10.5	30.8	8.7	26.2	50.6	108.2	46.5	114.6

Table 2 then applies all four retrieval models to the large MSMARCO-v2 collection. The first row shows “standard” retrieval, applying an inverted index and MaxScore dynamic pruning method; and then the second row compares that baseline against what can be achieved by postings clipping, priming using the same $U_{\mathcal{L}(t)}$ information that arises from the clipping. The best time in each column is shown in blue.

4. Conclusion

To keep up with increasingly large volumes of data, search practitioners require sophisticated structures and processing algorithms, so that response times can remain plausible. In this paper, we have demonstrated the speed benefits that arise through the use of a novel technique we call postings clipping. We have established new benchmarks for querying speed, with minimal costs overheads, for both shallow $k = 10$ and deep $k = 1000$ retrieval.

References

- [1] J. Mackenzie, A. Mallia, A. Moffat, M. Petri, Accelerating learned sparse indexes via term impact decomposition, in: *Findings of the Association for Computational Linguistics: EMNLP*, 2022, pp. 2830–2842. URL: <https://aclanthology.org/2022.findings-emnlp.205>.
- [2] A. Mallia, O. Khattab, N. Tonello, T. Suel, Learning passage impacts for inverted indexes, in: *Proc. ACM Int. Conf. on Research and Development in Information Retrieval (SIGIR)*, 2021, pp. 1723–1727. doi:10.1145/3404835.3463030.
- [3] L. Gao, Z. Dai, J. Callan, COIL: Revisit exact lexical match in information retrieval with contextualized inverted list, in: *Proc. Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, 2021, pp. 3030–3042. doi:10.18653/v1/2021.naacl-main.241.
- [4] J. Lin, X. Ma, A few brief notes on DeepImpact, COIL, and a conceptual framework for information retrieval techniques, *arXiv:2106.14807* (2021). doi:10.48550/arXiv.2106.14807.
- [5] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, W. Yih, Dense passage retrieval for open-domain question answering, in: *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 6769–6781. doi:10.18653/v1/2020.emnlp-main.550.
- [6] Y. Qu, Y. Ding, J. Liu, K. Liu, R. Ren, W. X. Zhao, D. Dong, H. Wu, H. Wang, RocketQA: An optimized training approach to dense passage retrieval for open-domain question answering, in: *Proc. Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, 2021, pp. 5835–5847. doi:10.18653/v1/2021.naacl-main.466.
- [7] K. M. Risvik, T. Chilimbi, H. Tan, K. Kalyanaraman, C. Anderson, Maguro, a system for indexing and searching over very large text collections, in: *Proc. Conf. on Web Search and Data Mining (WSDM)*, 2013, pp. 727–736. doi:10.1145/2433396.2433486.
- [8] N. Tonello, C. Macdonald, I. Ounis, Efficient query processing for scalable web search, *Foundations & Trends in Information Retrieval* 12 (2018) 319–500. doi:10.1561/15000000057.
- [9] H. R. Turtle, J. Flood, Query evaluation: Strategies and optimizations, *Information Processing & Management* 31 (1995) 831–850. doi:10.1016/0306-4573(95)00020-H.
- [10] A. Z. Broder, D. Carmel, M. Herscovici, A. Soffer, J. Zien, Efficient query evaluation using a two-level retrieval process, in: *Proc. ACM Int. Conf. on Information and Knowledge Management (CIKM)*, 2003, pp. 426–434. doi:10.1145/956863.956944.
- [11] T. Strohman, W. B. Croft, Efficient document retrieval in main memory, in: *Proc. ACM Int. Conf. on Research and Development in Information Retrieval (SIGIR)*, 2007, pp. 175–182. doi:10.1145/1277741.1277774.
- [12] S. Ding, T. Suel, Faster top- k document retrieval using block-max indexes, in: *Proc. ACM Int. Conf. on Research and Development in Information Retrieval (SIGIR)*, 2011, pp. 993–1002. doi:10.1145/2009916.2010048.
- [13] C. M. Daoud, E. S. de Moura, A. L. Carvalho, A. S. da Silva, D. Fernandes, C. Rossi, Fast top- k preserving query processing using two-tier indexes, *Information Processing & Management* 52 (2016) 855–872. doi:10.1016/j.ipm.2016.03.005.

- [14] C. M. Daoud, E. S. de Moura, D. Fernandes, A. S. da Silva, C. Rossi, A. Carvalho, Waves: A fast multi-tier top- k query processing algorithm, *Information Retrieval* 20 (2017) 292–316. doi:10.1007/s10791-017-9298-6.
- [15] A. Kane, F. W. Tompa, Split-lists and initial thresholds for WAND-based search, in: *Proc. ACM Int. Conf. on Research and Development in Information Retrieval (SIGIR)*, 2018, pp. 877–880. doi:10.1145/3209978.3210066.
- [16] J. Mackenzie, M. Petri, A. Moffat, Anytime ranking on document-ordered indexes, *ACM Trans. on Information Systems* 40 (2022) 13:1–13:32. doi:10.1145/3467890.
- [17] A. Mallia, M. Siedlaczek, J. Mackenzie, T. Suel, PISA: Performant indexes and search for academia, in: *Proc. OSIRRC at SIGIR 2019*, 2019, pp. 50–56. URL: <http://ceur-ws.org/Vol-2409/docker08.pdf>.
- [18] J. Mackenzie, A. Trotman, J. Lin, Wacky weights in learned sparse representations and the revenge of score-at-a-time query evaluation, *arXiv:2110.11540* (2021). URL: <https://arxiv.org/abs/2110.11540>.