ColBERT-serve: Efficient Multi-Stage Memory-Mapped Scoring

Kaili Huang✉[1†], Thejas Venkatesh[2†], Uma Dingankar[3†‡], Antonio Mallia[4], Daniel Campos[5], Jian Jiao[1], Christopher Potts[6], Matei Zaharia[7], Kwabena Boahen[6], Omar Khattab[6], Saarthak Sarup[6], and Keshav Santhanam[6]

[1] Microsoft, Redmond, WA, USA
kaili.khuang@gmail.com, jian.jiao@microsoft.com
[2] Samaya AI, Mountain View, CA, USA
thejas@stanford.edu
[3] Foundry, Palo Alto, CA, USA
uma@mlfoundry.com
[4] Pinecone, New York, NY, USA
me@antoniomallia.it
[5] Snowflake, New York, NY, USA
daniel.campos@snowflake.com
[6] Stanford University, Stanford, CA, USA
{cgpotts, boahen, okhattab, ssarup, keshav2}@stanford.edu
[7] UC Berkeley, Berkeley, CA, USA
matei@berkeley.edu

Abstract. We study serving retrieval models, particularly late interaction retrievers like ColBERT, to many concurrent users at once and under a small budget, in which the index may not fit in memory. We present ColBERT-serve, a serving system that applies a memory-mapping strategy to the ColBERT index, reducing RAM usage by 90% and permitting its deployment on cheap servers, and incorporates a multi-stage architecture with hybrid scoring, reducing ColBERT's query latency and supporting many concurrent queries in parallel.

Keywords: Information Retrieval · ColBERT · Efficiency

1 Introduction

Multi-vector late-interaction retrievers like ColBERT [10] and ColPali [5] have demonstrated state-of-the-art quality and superior generalization [31] while maintaining low latency, but despite major progress in compressing their embeddings [28,8], hosting a ColBERT index of Wikipedia (20M passages) via PLAID [27] demands nearly 100GB of RAM. This poses a challenge for serving such models on cheap servers with little RAM, especially if we need to serve many concurrent users with low latency. Unfortunately, cost, latency, and quality tradeoffs

[†]K. Huang, T. Venkatesh, and U. Dingankar contributed equally to this work.
[‡]Work by U. Dingankar was done while at Stanford.

in such a high-concurrency, low-memory regime are rarely considered jointly in the existing neural IR literature.

We tackle this with the following contributions. First, we present a methodology and benchmark for evaluating the concurrent serving of neural IR models under different traffic workloads and memory budgets. Second, we introduce **ColBERT-serve**,[1] which (1) incorporates a new memory-mapping architecture, permitting the bulk of ColBERTv2's index to reside *on disk*, (2) **minimizes access to this index** via a multi-stage retrieval process, (3) **handles concurrent requests** in parallel with low latency and scales gracefully under load by adapting PISA and ColBERTv2, and (4) **preserves the quality of full ColBERTv2 retrieval** through a hybrid scoring technique. Third, we conduct an empirical evaluation that demonstrates the first ColBERT serving system that **can serve up to 4 queries per second on a server with as little as a few GBs of RAM** (90% reduction in RAM usage for loading the model compared to the full ColBERTv2) for massive collections while preserving quality.

2 Related Work

Memory-mapping is a technique for accessing data from disk while only materializing accessed portions in memory *on demand*. It is used in approximate nearest neighbor search [9,4], database systems [11], and concurrently with our work also in neural IR [30]. Memory-mapped indexes pose the challenge of minimizing the latency overhead incurred by page misses. Whereas [30] built a prefetcher to reduce the impact of SSD latencies, we seek to reduce the number of accesses to disk directly via multi-stage retrieval. Much existing work has studied improving the latency or memory footprint of ColBERT-like models [28,27,29,17,25,15,6] or the quality-cost tradeoff [10,21,19,3] of using ColBERT to re-rank results produced by simpler systems like BM25 [26] or LADR [12]. This general strategy is generally known to lead to improved latency but comes at the cost of a reduction in MRR and recall. We build a concurrent serving system for ColBERTv2 that permits the index to mostly reside on disk without sacrificing retrieval quality or latency under high traffic. We achieve this by leveraging a combination of memory-mapping and a multi-stage retrieval approach that utilizes scores from both the candidate generation and the re-ranking steps. This hybrid scoring method leverages the strengths of both stages, resulting in performance that can surpass fully in-memory ColBERTv2 retrieval.

3 ColBERT-serve

Memory-Mapped Storage To deploy ColBERTv2 on memory-constrained machines, we introduce memory-mapping into the ColBERT implementation, specifically for the tensors encoding the compressed ColBERTv2 embeddings.

[1] https://github.com/stanford-futuredata/colbert-serve

This bypasses loading the index upfront and instead enables the operating system to manage limited memory resources, by bringing only accessed data into memory at the page granularity and evicting pages when RAM is insufficient. This reduces the RAM requirements by over 90%.

Concurrent Requests We build a server-client architecture for deployment as well as experimentation for ColBERTv2. To this end, we improve ColBERTv2's multithreading compatibility by releasing Python's Global Interpreter Lock while invoking all underlying functionality of ColBERTv2 implemented as C++ extensions.[2] Without this, multithreading for ColBERTv2 was prohibitively expensive as each query would block when extensions are invoked, so concurrency was only possible by launching multiple processes, which—without memory mapping—would scale memory consumption linearly with the number of processes. With support for memory mapping, we tune the number of threads used to serve each ColBERTv2 request and find that though multithreading improves performance under low load, single-threaded performance dominates under higher loads; hence, we use only a single thread for all our experiments. In addition, we adapt the PISA [22] engine for this setting to support our server-client architecture with the multi-stage retrieval discussed next. We leave the comparison with more recent dynamic pruning strategies specifically designed for learned sparse retrieval models [24,20] as future work.

Multi-Stage Retrieval Memory-mapping introduces a key challenge: due to the latency incurred by page misses, searching over MS MARCO with a memory-mapped index is approximately 2× slower than an in-memory index. We tackle this via a multi-stage ranking architecture, in which SPLADEv2 [7], a learned sparse model [18,32,33,2], serves as the first-stage retriever to minimize the number of documents we need to access from the ColBERT index. As a baseline,[3] we use the standard ColBERTv2 with PLAID [27] with a machine capable of fitting the entire index in memory. Then, we implement and study four different systems: (1) **MMAP ColBERTv2**, in which we apply memory-mapping to the end-to-end process of PLAID; (2) **SPLADEv2** w/ PISA, in which SPLADEv2 expands queries and the PISA engine performs efficient retrieval [22]; (3) **MMAP Rerank**, in which SPLADEv2 generates top-200 candidates per query and MMAP ColBERTv2 re-ranks them; and **MMAP Hybrid**, in which SPLADEv2's top-200 results are re-ranked via a linear interpolation between SPLADEv2 and MMAP ColBERTv2. For a given query Q and document D, the hybrid score is given by:

$$S_{\text{hybrid}}(D,Q) = \alpha N(S_{\text{SPLADE}}(D,Q)) + (1-\alpha)N(S_{\text{ColBERT}}(D,Q))$$

where $S(*,*)$ is the score function, $N(*)$ is the normalization function, and α is a coefficient between 0 and 1. SPLADEv2 and ColBERTv2 produce scores of drastically different distributions, a likely source of quality for hybrid scoring.

[2] This optimization was implemented in May 2024. Since then, Python 3.13 has introduced experimental support for a GIL-free mode.

[3] We build on code from https://github.com/stanford-futuredata/ColBERT, https://github.com/naver/splade, and https://github.com/pisa-engine/pisa.

To combine these scores, we explored (1) linearly mapping each to the range of $[0, 1]$, (2) min-max norm, and (3) z-norm. Among these, z-norm yielded the best results, so we select that as the normalization function, defined as: $N(x) = \frac{x-\bar{x}}{S}$ where $\bar{x}$ denotes the mean of samples and S denotes the standard deviation.

4 Evaluation

We now test the impact of multi-stage retrieval on quality, of memory-mapping on RAM usage, and of both together on latency under varying traffic.

Methodology We use MS MARCO Passage Ranking development set that contains 7K queries and 8.8M passages [1] as an "in-domain" benchmark for ColBERTv2 and SPLADEv2 and report MRR@10, Recall@5 and Recall@50. To test out-of-domain (OOD) generalization, we use Wikipedia Open-QA NQ-dev with 8.7K queries and 21M passages [13,16] and LoTTE Search Lifestyle-dev with 417 queries and 269K passages [28]. These popular datasets differ dramatically in size, with Wikipedia stressing RAM usage and LoTTE Lifestyle always fitting in memory. Following [28], we report Success@5. We report the mean latency and tail (95th and the 99th percentiles) latency observed by the concurrent clients in our client-server architecture under varying degrees of server load. We measure latency over the first 1K queries from each dataset, a sufficient size to saturate the system under high load conditions. The number of queries per second (QPS) is sampled using a Poisson distribution.

Choice of Hardware Since the ColBERTv2 baseline loads the entire index in memory, its experiments require a machine with a high-capacity RAM. In contrast, ColBERT-serve can run on significantly smaller machines. To highlight this important benefit, we run experiments for our method on strictly smaller, less powerful, cheaper machines rather than using the same machines as the control experiments (namely, the full ColBERTv2 baseline). This demonstrates that the proposed method has comparable quality and latency, while running on significantly cheaper and resource-constrained machines. For MS MARCO and Wikipedia, we use an AWS `r6a.4xlarge` instance for the control experiment, and `m5ad.xlarge` and `r6id.xlarge` instances for SPLADEv2/MMAP experiments, respectively. LoTTE Lifestyle's index is small enough to fit in a memory-restricted machine, so we run all experiments on a `c5ad.xlarge` instance. The key machine specifications are provided in Table 1.

Table 1: AWS machine specifications

	Control	**MMAP MARCO**	**MMAP Wiki**	**LoTTE**
AWS machine	r6a.4xlarge	m5ad.xlarge	r6id.xlarge	c5ad.xlarge
Disk Size (GB)	950	150	237	150
CPU Count	16	4	4	4
Memory (GB)	128	**16 (-88%)**	**32 (-75%)**	8
Cost ($/month)	438	**95 (-78%)**	**139 (-68%)**	54

Table 2: Results on MS MARCO, Wikipedia (NQ-dev) and LoTTE (Lifestyle-dev) datasets. For SPLADEv2, we use the BT-SPLADE-L [14] checkpoint and a PISA index compressed with a `block_simdbp` encoding, following [23], and block of size 40 with a quantized scorer. For MS MARCO, we report development results on Dev, on which we tune α for all datasets, and report evaluation results on the held-out evaluation set used by the ColBERT authors [10,28].

Method	**MS MARCO Dev**			**MS MARCO 5K Test**		
	MRR@10	R@5	R@50	MRR@10	R@5	R@50
ColBERTv2	39.51	56.62	86.30	40.57	57.78	86.14
SPLADEv2	38.00	54.70	85.04	38.62	54.92	84.84
Rerank	39.50	56.65	86.64	40.55	57.78	86.36
Hybrid ($\alpha = 0.3$)	**40.22**	**57.38**	**86.98**	**41.11**	**58.23**	**86.91**

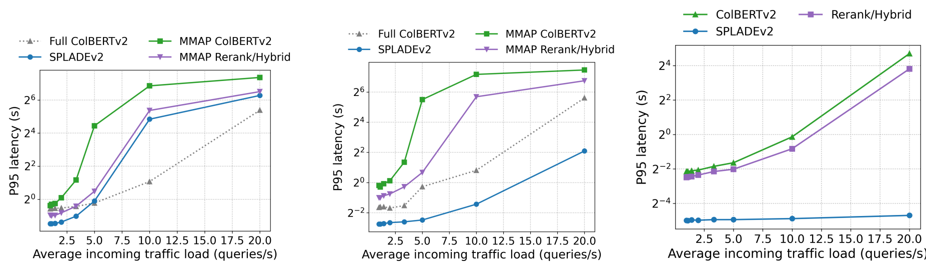
Method	**Wikipedia**		**LoTTE**	
	S@5	Δ	S@5	Δ
ColBERTv2	**67.51**		74.6	
SPLADEv2	59.60	-11.7%	70.7	-5.6%
Rerank	66.29	-1.8%	74.3	-0.4%
Hybrid ($\alpha = 0.3$)	65.78	-2.6%	74.8	+0.3%
Hybrid (optimal α)	66.34	-1.7%	**75.3**	**+0.9%**

Retrieval Quality Table 2 reports the quality of full ColBERTv2 scoring against more efficient approaches based on SPLADEv2, Rerank, and Hybrid scoring. We tune the parameter α for Hybrid on MS MARCO Dev and report the results of this setting (i.e., $\alpha = 0.3$) across all datasets. We can observe that Hybrid scoring is the most effective method on MS MARCO and that it outperforms the SPLADEv2 model and Rerank across every dataset. On Wikipedia, however, using a non-optimal α results in lower performance than Rerank. This suggests that tuning α on a dedicated set of queries can be important to OOD settings, though we leave this exploration for future work. Having confirmed the quality of the Rerank and especially Hybrid methods, we now proceed to evaluate the different efficiency dimensions.

Table 3: Retrieval quality of Hybrid with different α. When $\alpha = 0$, the method is equivalent to Rerank; when $\alpha = 1$, it's equivalent to SPLADEv2.

α	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Wiki S@5	66.29	**66.34**	66.21	65.78	65.33	64.76	63.80	63.01	62.12	61.04	59.60
LoTTE S@5	74.3	74.1	**75.3**	74.8	74.6	74.3	73.4	72.4	71.5	71.2	70.7
MARCO MRR@10	39.50	39.95	40.06	**40.22**	40.08	40.05	39.81	39.53	38.98	38.54	38.00

RAM Usage We measure memory usage for loading ColBERTv2 on MS MARCO and Wikipedia by recording the difference in RSS memory before and



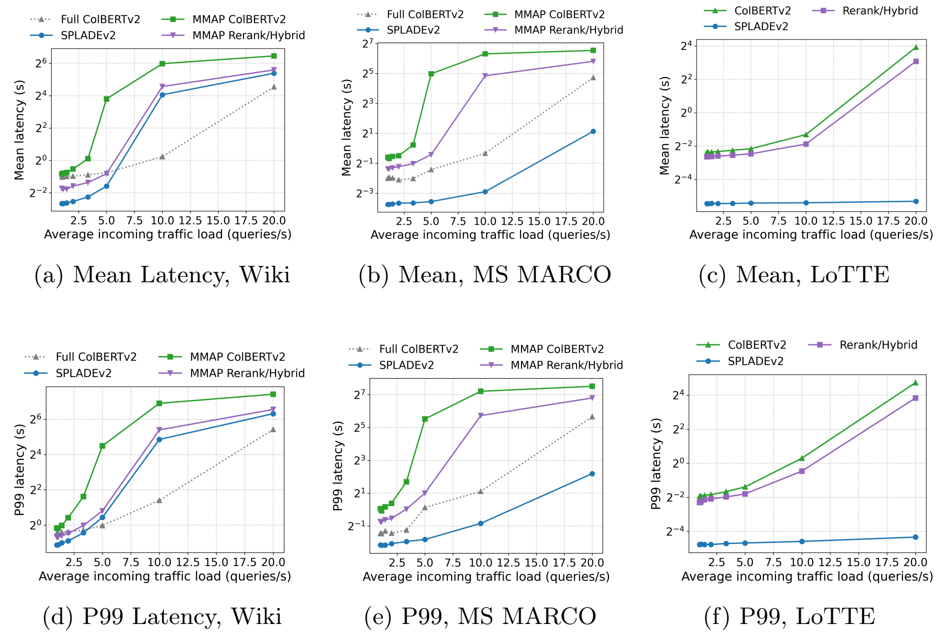
(a) Wikipedia (b) MS MARCO (c) LoTTE

Fig. 1: P95 latency for Wikipedia, MS MARCO, and LoTTE. Note that full ColBERTv2 on MS MARCO is evaluated on a higher-end and more expensive machine (refer to Table 1) with a different physical processor, so its latency is only for reference and is not directly comparable to the MMAP methods.

after loading. For the memory-mapped approaches, only the model checkpoint and index metadata are loaded into memory, resulting in a substantial reduction of RAM usage, by 90% for MS MARCO (from 23.4 GB to 2.3 GB) and 92% for Wikipedia (from 98.3 GB to 8.2 GB). Our approach allows us to host the indexes on machines with significantly lower RAM capacities, and reduces machine cost by 78% for MS MARCO and 68% for Wikipedia, as shown in Table 1.

Latency on Varying Traffic Figure 1a compares the P95 latency across methods on Wikipedia. The optimized PISA implementation of SPLADEv2, using the efficiency-optimized BT-SPLADE-L model checkpoint [14], has the lowest latency, although this comes at the steep reduction in quality, especially out of domain, presented earlier. Next, although the Rerank/Hybrid methods incur higher latency than SPLADE, they are markedly faster than the memory-mapped ColBERTv2 method. The Rerank/Hybrid methods maintain low latency with QPS up to $1/0.2 = 5$ queries per second. When QPS exceeds this, the system is saturated, leading to a sharper increase in latency due to queuing time. Note that as shown in Table 1, full ColBERTv2 experiments were conducted on a more expensive machine that fits the index in RAM, for reference. Despite this, the Rerank/Hybrid methods still achieve lower latency than full ColBERTv2 on QPS $< 1/0.3 = 3.3$, highlighting the value of multi-stage retrieval.

Figure 1b shows similar trends on MS MARCO, where our Rerank/Hybrid systems greatly reduce the latency of memory-mapping ColBERTv2 across every traffic load. Note that full ColBERTv2 is evaluated on a machine with a different physical processor, so its latency is only for reference and is not directly comparable to the memory-mapped methods. Lastly, Figure 1c reports very similar patterns for for LoTTE. Note that we do not apply memory mapping for LoTTE, whose ColBERTv2 index fits easily in the RAM of our smallest machines. We also report mean latency and P99 latency as additional metrics in Figure 2, with similar trends as P95 latency.



(a) Mean Latency, Wiki (b) Mean, MS MARCO (c) Mean, LoTTE

(d) P99 Latency, Wiki (e) P99, MS MARCO (f) P99, LoTTE

Fig. 2: Mean Latency and P99 Latency on Three Datasets.

5 Conclusion

We presented a highly practical serving system for ColBERT models that combines memory-mapping, hybrid scoring, and support for concurrent requests. We introduced an evaluation methodology for assessing the neural IR tradeoffs in the concurrent, memory-constrained regime and demonstrated for the first time to our knowledge that a ColBERT serving system can serve several queries per second over large datasets *on a server with as little as a few GBs of RAM.* While we expect that serving multi-vector models will continue to become faster and cheaper in other ways, this work presents that a simple yet effective strategy to balance a large number of deployment tradeoffs.

Acknowledgments. This work was partially supported by a Stanford HAI Hoffman-Yee Research Grant and by IBM as a founding member of the Stanford Institute for Human-Centered Artificial Intelligence (HAI), Oracle, Virtusa, and Cigna Healthcare. This research was supported in part by affiliate members and other supporters of the Stanford DAWN project—Facebook, Google, and VMware.

References

1. Bajaj, P., Campos, D., Craswell, N., Deng, L., Gao, J., Liu, X., Majumder, R., McNamara, A., Mitra, B., Nguyen, T., et al.: Ms marco: A human generated machine reading comprehension dataset. arXiv preprint arXiv:1611.09268 (2016)

2. Basnet, S., Gou, J., Mallia, A., Suel, T.: Deeperimpact: Optimizing sparse learned index structures. arXiv preprint arXiv:2405.17093 (2024)
3. Bergum, J.K.: Improving zero-shot ranking with vespa hybrid search - part two (2023), https://blog.vespa.ai/improving-zero-shot-ranking-with-vespa-part-two/
4. Bernhardsson, E.: Spotify/annoy: Approximate nearest neighbors in c++/python optimized for memory usage and loading/saving to disk, https://github.com/spotify/annoy
5. Faysse, M., Sibille, H., Wu, T., Omrani, B., Viaud, G., Hudelot, C., Colombo, P.: Colpali: Efficient document retrieval with vision language models (2024), https://arxiv.org/abs/2407.01449
6. Formal, T., Clinchant, S., Déjean, H., Lassance, C.: Splate: Sparse late interaction retrieval. In: Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval. p. 2635–2640. SIGIR '24, Association for Computing Machinery, New York, NY, USA (2024). https://doi.org/10.1145/3626772.3657968, https://doi.org/10.1145/3626772.3657968
7. Formal, T., Lassance, C., Piwowarski, B., Clinchant, S.: SPLADE v2: Sparse lexical and expansion model for information retrieval. CoRR **abs/2109.10086** (2021), https://arxiv.org/abs/2109.10086
8. Hofstätter, S., Khattab, O., Althammer, S., Sertkan, M., Hanbury, A.: Introducing neural bag of whole-words with colberter: Contextualized late interactions using enhanced reduction. In: Proceedings of the 31st ACM International Conference on Information & Knowledge Management. p. 737–747. CIKM '22, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3511808.3557367, https://doi.org/10.1145/3511808.3557367
9. Johnson, J., Douze, M., Jégou, H.: Billion-scale similarity search with gpus. IEEE Transactions on Big Data **7**(3), 535–547 (2019)
10. Khattab, O., Zaharia, M.: Colbert: Efficient and effective passage search via contextualized late interaction over bert. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. p. 39–48. SIGIR '20, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3397271.3401075, https://doi.org/10.1145/3397271.3401075
11. Kotek, J.: Jankotek/mapdb: Mapdb provides concurrent maps, sets and queues backed by disk storage or off-heap-memory. it is a fast and easy to use embedded java database engine., https://github.com/jankotek/mapdb/
12. Kulkarni, H., MacAvaney, S., Goharian, N., Frieder, O.: Lexically-accelerated dense retrieval. In: Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval. p. 152–162. SIGIR '23, Association for Computing Machinery, New York, NY, USA (2023). https://doi.org/10.1145/3539618.3591715, https://doi.org/10.1145/3539618.3591715
13. Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Devlin, J., Lee, K., Toutanova, K., Jones, L., Kelcey, M., Chang, M.W., Dai, A.M., Uszkoreit, J., Le, Q., Petrov, S.: Natural Questions: A Benchmark for Question Answering Research. Transactions of the Association for Computational Linguistics **7**, 453–466 (08 2019). https://doi.org/10.1162/tacl_a_00276, https://doi.org/10.1162/tacl_a_00276
14. Lassance, C., Clinchant, S.: An efficiency study for splade models. In: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval. p. 2220–2226. SIGIR '22, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3477495.3531833, https://doi.org/10.1145/3477495.3531833

15. Lee, J., Dai, Z., Duddu, S.M.K., Lei, T., Naim, I., Chang, M.W., Zhao, V.: Rethinking the role of token retrieval in multi-vector retrieval. Advances in Neural Information Processing Systems **36** (2024)
16. Lee, K., Chang, M.W., Toutanova, K.: Latent retrieval for weakly supervised open domain question answering. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics (2019)
17. Li, M., Lin, S.C., Oguz, B., Ghoshal, A., Lin, J., Mehdad, Y., Yih, W.t., Chen, X.: Citadel: Conditional token interaction via dynamic lexical routing for efficient and effective multi-vector retrieval. In: Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 11891–11907 (2023)
18. Lin, J., Ma, X.: A few brief notes on deepimpact, coil, and a conceptual framework for information retrieval techniques. arXiv preprint arXiv:2106.14807 (2021)
19. MacAvaney, S., Tonellotto, N.: A reproducibility study of plaid. In: Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval. p. 1411–1419. SIGIR '24, Association for Computing Machinery, New York, NY, USA (2024). https://doi.org/10.1145/3626772.3657856, https://doi.org/10.1145/3626772.3657856
20. Mackenzie, J., Mallia, A., Moffat, A., Petri, M.: Accelerating learned sparse indexes via term impact decomposition. In: Findings of the Association for Computational Linguistics: EMNLP 2022. pp. 2830–2842 (2022)
21. Mallia, A., Khattab, O., Suel, T., Tonellotto, N.: Learning passage impacts for inverted indexes. In: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 1723–1727 (2021)
22. Mallia, A., Siedlaczek, M., Mackenzie, J., Suel, T.: PISA: performant indexes and search for academia. In: Proceedings of the Open-Source IR Replicability Challenge co-located with 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, OSIRRC@SIGIR 2019, Paris, France, July 25, 2019. pp. 50–56 (2019), http://ceur-ws.org/Vol-2409/docker08.pdf
23. Mallia, A., Siedlaczek, M., Suel, T.: 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. pp. 353–368. Springer (2019)
24. Mallia, A., Suel, T., Tonellotto, N.: Faster learned sparse retrieval with block-max pruning. In: Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 2411–2415 (2024)
25. Nardini, F.M., Rulli, C., Venturini, R.: Efficient multi-vector dense retrieval with bit vectors. In: European Conference on Information Retrieval. pp. 3–17. Springer (2024)
26. Robertson, S., Walker, S., Jones, S., Hancock-Beaulieu, M.M., Gatford, M.: Okapi at trec-3. In: Overview of the Third Text REtrieval Conference (TREC-3). pp. 109–126. Gaithersburg, MD: NIST (January 1995), https://www.microsoft.com/en-us/research/publication/okapi-at-trec-3/
27. Santhanam, K., Khattab, O., Potts, C., Zaharia, M.: Plaid: an efficient engine for late interaction retrieval. In: Proceedings of the 31st ACM International Conference on Information & Knowledge Management. pp. 1747–1756 (2022)
28. Santhanam, K., Khattab, O., Saad-Falcon, J., Potts, C., Zaharia, M.: ColBERTv2: Effective and efficient retrieval via lightweight late interaction. In: Carpuat, M., de Marneffe, M.C., Meza Ruiz, I.V. (eds.) Proceedings of the 2022 Conference of the

North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 3715–3734. Association for Computational Linguistics, Seattle, United States (Jul 2022). https://doi.org/10.18653/v1/2022.naacl-main.272, https://aclanthology.org/2022.naacl-main.272
29. Santhanam, K., Saad-Falcon, J., Franz, M., Khattab, O., Sil, A., Florian, R., Sultan, M.A., Roukos, S., Zaharia, M., Potts, C.: Moving beyond downstream task accuracy for information retrieval benchmarking. In: Rogers, A., Boyd-Graber, J., Okazaki, N. (eds.) Findings of the Association for Computational Linguistics: ACL 2023. pp. 11613–11628. Association for Computational Linguistics, Toronto, Canada (Jul 2023). https://doi.org/10.18653/v1/2023.findings-acl.738, https://aclanthology.org/2023.findings-acl.738
30. Shrestha, S., Reddy, N., Li, Z.: Espn: Memory-efficient multi-vector information retrieval. In: Proceedings of the 2024 ACM SIGPLAN International Symposium on Memory Management. p. 95–107. ISMM 2024, Association for Computing Machinery, New York, NY, USA (2024). https://doi.org/10.1145/3652024.3665515, https://doi.org/10.1145/3652024.3665515
31. Thakur, N., Reimers, N., Rücklé, A., Srivastava, A., Gurevych, I.: Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models. arXiv preprint arXiv:2104.08663 (2021)
32. Yu, P., Mallia, A., Petri, M.: Improved learned sparse retrieval with corpus-specific vocabularies. In: European Conference on Information Retrieval. pp. 181–194. Springer (2024)
33. Zhuang, S., Zuccon, G.: Fast passage re-ranking with contextualized exact term matching and efficient passage expansion. arXiv preprint arXiv:2108.08513 (2021)