E2Rank: Efficient and Effective Layer-wise Reranking

Cesare Campagnano[1,2], Antonio Mallia[2], Jack Pertschuk[2], and Fabrizio Silvestri[1]

[1] Sapienza University of Rome, Italy
{campagnano, fsilvestri}@diag.uniroma1.it
[2] Pinecone, US
{cesare, antonio, jack}@pinecone.io

Abstract. Reranking models are critical to enhancing the quality of retrieval systems by refining initial search results based on query relevance. Among these, cross-encoders demonstrate higher effectiveness because of their deep semantic understanding, achieved through transformer-based architectures. However, their high computational demands pose significant challenges for real-time applications and scalability.
This paper introduces E2Rank, a layer-wise reranking model that optimizes both efficiency and effectiveness by leveraging intermediate transformer outputs, progressively applying deeper model layers to a narrowed candidate set, to reduce computational costs with minimal impact on quality. Our training approach, which includes model merging and layerwise contrastive training, yields substantial gains in effectiveness. Extensive experiments conducted on standard benchmarks demonstrate that E2Rank achieves state-of-the-art performance, outperforming existing rerankers in both effectiveness and computational efficiency.

Keywords: Neural Retrieval · Reranking · Cross-Encoder

1 Introduction

Advancements in Retrieval-Augmented Generation (RAG) highlight the crucial role of retrieval in reducing hallucinations and grounding generative models in factual, relevant information [8, 15]. Effective retrieval reduces the risk of irrelevant outputs by providing only the most pertinent documents. In large-scale retrieval systems, however, directly reranking all candidates is impractical, so cascading architectures [1, 30] are used to filter out less relevant documents early, allowing complex models to focus on a smaller, high-quality subset.

Cross-encoders have emerged as powerful tools for reranking a subset of relevant candidates. A cross-encoder typically consists of a transformer-based architecture, e.g. BERT [10], where pairs of queries and candidate documents are jointly provided as input [33] producing relevance scores. This approach, while highly effective, comes with significant computational costs, especially as model sizes continue to increase. A key challenge in modern IR is balancing the effectiveness of neural models with their latency and computational demands.

Several techniques have been introduced to address this challenge, with notable improvements aimed at reducing latency without sacrificing performance. One such approach, CROSS-JEM [23], improves the efficiency of cross-encoders by jointly scoring multiple items in parallel, rather than processing each independently, as is typical in standard cross-encoder architectures. FIRST [24] optimizes reranking by reducing the number of output tokens required and employs strategies such as truncated ranked lists and top-down partitioning to further enhance efficiency. Additionally, some methods leverage adapters within transformer models [22] to minimize fine-tuning costs and boost efficiency. For real-time ranking, online production systems often employ sparse neural networks like MEB [4], which enable efficient ranking of thousands of items simultaneously. Late-interaction models, such as ColBERT [25] and TwinBERT [17], offer another solution by applying a late-interaction layer over query-item embeddings and pre-computed document embeddings, significantly reducing computational costs while preserving ranking quality. Furthermore, techniques such as knowledge distillation have been used to compress large language models, such as distilling RankGPT into cross-encoders for passage reranking [26], which dramatically lowers inference latency without compromising ranking effectiveness.

Very recently, layerwise reranking[3] has appeared as an implementation proposal rather than a fully researched methodology formally documented in the literature. This method makes use of only a subset of layers from the full transformer architecture to perform reranking, thus reducing computation costs.

We propose a novel approach that addresses both the cost and performance issues in current methods. This work provides the following contributions:

- We present an overview of the current reranking landscape and introduce a comprehensive benchmark that includes standard datasets and both open-source and proprietary models.
- We introduce a training strategy that combines model merging with layer-wise contrastive training, yielding substantial improvements.
- We present a low-parameter state-of-the-art reranker that outperforms its larger competitors across several in-domain and out-of-domain benchmarks.
- We propose a layerwise reranking strategy with adjustable efficiency-effectiveness trade-offs, allowing to operate across various efficiency requirements.[4]

2 E2Rank

Architecture. The proposed reranker employs a transformer-based cross-encoder architecture to evaluate the relevance between a query and a document by producing a similarity score. Specifically, the model takes both the query and the document as input, processes them jointly with the same transformer layers, and outputs a final score in the form of a logit through a projection layer. In our implementation, to maximize effectiveness, we use the DeBERTa v3 large

[3] `https://huggingface.co/BAAI/bge-reranker-v2-minicpm-layerwise`
[4] The code is available at: `https://github.com/caesar-one/e2rank`

model as the base encoder, selected for its disentangled attention mechanism and advanced pre-training. The disentangled attention allows the model to independently focus on content and positional information, which is crucial for capturing fine-grained relationships in query-document pairs. Additionally, DeBERTa v3 is trained with an ELECTRA-style objective, which enables the model to predict token replacements rather than just masked tokens. This approach yields richer representations, contributing to stronger retrieval performance and a more effective reranker overall.

Training Objective. We modify the above architecture to enable reranking scores to be computed at each individual layer of the transformer. This flexibility offers the advantage of leveraging intermediate representations that capture relevant features more efficiently. We propose a training objective that incorporates layer-wise supervision and distribution alignment between intermediate layers and the final layer of the reranker model.

Let q be a query, and $D = \{d_0^+, d_1^-, \ldots, d_N^-\}$ be a set of documents associated with q, where d_0^+ is the relevant (positive) document and $\{d_i^-\}$ are the N non-relevant (negative) documents. The model computes relevance scores for each document at each layer $l \in L$, producing logits $\hat{\mathbf{s}}_l \in \mathbb{R}^{N+1}$ for the documents in D. These logits are converted to probabilities via the softmax function $\hat{\mathbf{p}}_l = \text{softmax}(\hat{\mathbf{s}}_l)$. The training objective consists of two components:
First, we apply a cross-entropy loss at each layer l to encourage them to assign higher scores to relevant documents:

$$\mathcal{L}_{\text{layerwise}} = \frac{1}{|L|} \sum_{l=1}^{|L|} \text{CE}(\hat{p}_l, \hat{y}), \tag{1}$$

where $\hat{p}_l$ is the probability distribution at the l-th layer, and $\hat{y}$ is the target one, defined as $\hat{y} = [1, 0, \ldots, 0]$, indicating that the first document d_0^+ is relevant.

Then, to encourage output distributions of intermediate layers to align with the final layer's distribution, we use a Kullback-Leibler Divergence (KL) loss:

$$\mathcal{L}_{\text{divergence}} = \frac{1}{|L| - 1} \sum_{l=1}^{|L|-1} \text{KL}(\hat{p}_l, \hat{p}_{|L|}), \tag{2}$$

where $\hat{p}_l$ and $\hat{p}_{|L|}$ are the probability distributions for layer l (student) and the final layer (teacher), respectively. The total loss is the sum of the two losses.

We found that using multiple hard negatives – challenging negative samples that are somewhat similar to the anchor but are not true positives – substantially improves training, helping the model generalize more robustly. This well studied technique [13, 32, 34] enhances performance by enabling the model to better distinguish between similar but incorrect examples.

Model Merging. Model merging refers to combining the weights of multiple models trained independently, typically on different datasets or with different hyperparameter configurations, to improve robustness and accuracy. In our work, we merge two checkpoints – trained on MSMARCO and QA datasets – using

linear averaging of their weights, a method that enables us to benefit from the strengths of both models without increasing memory or inference costs, unlike traditional ensembles. This approach ensures that the merged model retains performance improvements while remaining efficient in terms of resource usage [31].

Multi-step reranking. In Algorithm 1, we propose a multi-step cascading reranking approach that improves model efficiency by gradually applying more transformer layers as the document set is refined. Initially, a limited number of layers generate approximate ranking scores for a broad set of candidates, acting as a coarse filter. In each subsequent step, as the candidate set narrows, the model progressively uses more layers to capture complex patterns, refining the ranking with each pass while minimizing computational costs, and ensuring that the whole model is used only on the most promising documents.

Figure 1 depicts this process, showing how the number of transformer layers and candidate documents change across steps. Starting with a high document count and few layers, the method results in a deeper model usage only on a small, more relevant subset of documents, balancing computational load with ranking quality. Unlike a pipeline composed of multiple models that increase in size, this method benefits from using the same model throughout the process, enabling the reuse of intermediate representations. This reduces redundant computations and ensures that each step builds upon the prior, leading to both efficiency gains.

Algorithm 1: Layer-wise Reranking Algorithm

```
Function LayerwiseRerank(ids, hidden_states, ranking_layers)
    ranking_layer, k ← Pop(ranking_layers)
    for l ← 1 to TotalLayers do
        hidden_states ← ApplyLayer(l, hidden_states)
        if l = ranking_layer then
            logits ← Classify(hidden_states)
            doc_ids, hidden_states ← TopK(k, logits, ids, hidden_states)
            ranking_layer, k ← Pop(ranking_layers)
    return ids
```

3 Experiments

Dataset and query logs. Our evaluation uses test collections from the TREC Deep Learning Tracks (2019 and 2020) [6, 7] based on the MS MARCO v1 passage corpus [20]. For the BEIR benchmark [27], we focused on 12 datasets, excluding the four that are not publicly available to ensure reproducibility. We also excluded MS MARCO, as it is evaluated separately, and ArguAna, since its task of finding counter-arguments to the query contrasts with the purpose of a reranker. In all experiments, we rerank 200 candidates retrieved by SPLADE-v3 [14] leveraging BMP [19] as a retrieval algorithm. We also tested candidates

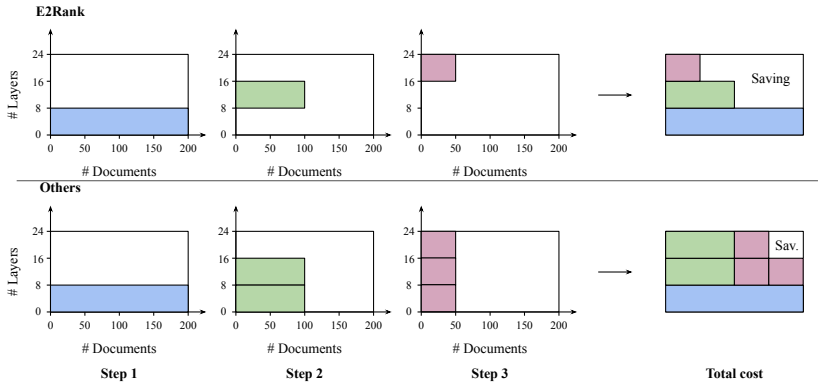

Fig. 1. A comparison of cascading reranking highlighting computation reuse in E2Rank.

from diverse retrievers, dense, sparse, and multi-vector models [2, 18, 25, 29], yielding similar results, but omitted them from the main experiments for brevity.

Competing Methods. We employ a diverse set of reranker models with distinct architectures and goals. The Naver model, based on DeBERTa-v3 large [11] by Naver Labs Europe[5] [9], and the BGE family from BAAI [3, 16] include BGE_{Layer} (MiniCPM-based with layer selection) and BGE_{M3} (M3-based lightweight model). We also test T5-based models MonoT5_{Base} and MonoT5_{3B}[21] fine-tuned on MS MARCO. Other models include JinaAI's JinaAI[12], Mixedbread[6] from Mixedbread, Cohere's Cohere_{En} and Cohere_{Multi}[5], Voyage_{v2} by VoyageAI [28], and Google's VertexAI[7]. Our models, E2Rank_{Marco}, E2Rank_{QA}, and E2Rank, are fine-tuned on MS MARCO, QA datasets, and their merged version. All models are evaluated on an NVIDIA A100 GPU with a standardized batch size of 8 and context length of 512. To ensure fairness, we avoid engineering optimizations like flash attention, CUDA kernels, and model compilation.

Accuracy. Our first experiment aims to show the effectiveness improvements of E2Rank. Table 1 shows the results for several reranking models on the TREC datasets. A weighted average of the NDCG@10 for the two query logs is computed as well as a mean response time (MRT) expressed in milliseconds per document. E2Rank is the fastest model, due to its low number of parameters. MonoT5_{Base} is the only model with fewer parameters, but it is slightly slower due to the more complex encoder-decoder architecture. Some of the proprietary models are tested though their official remote API and we were not able to collect latencies in a comparable way. In terms of retrieval quality, the weighted average NDCG@10 measured for the two TREC tracks shows that E2Rankis superior than all the competitors, although with a small margin.

[5] https://huggingface.co/naver/trecdl22-crossencoder-debertav3
[6] https://huggingface.co/mixedbread-ai/mxbai-rerank-large-v1
[7] https://cloud.google.com/generative-ai-app-builder/docs/ranking

Table 1. Effectiveness and mean response time (MRT, in ms/doc), on MSMARCO using the TREC 2019 and TREC 2020 queries. Avg refers to the weighted average.

	Model	TREC 2019	TREC 2020	Avg	Model Size	MRT
		NDCG@10			# params	ms/doc
OSS	Naver	77.5	75.5	76.4	304M	8.2
OSS	BGE$_{Layer}$	77.0	76.1	76.5	2.4B	10.5
OSS	BGE$_{M3}$	76.0	75.1	75.5	560M	7.6
OSS	MonoT5$_{Base}$	71.2	68.0	69.4	220M	4.1
OSS	MonoT5$_{3B}$	74.2	75.7	75.0	3B	29.8
Proprietary	Mixedbread	75.4	73.1	74.1	440M	8.1
Proprietary	JinaAI	75.1	76.2	75.7	560M	–
Proprietary	Cohere$_{En}$	73.5	71.2	72.2	–	–
Proprietary	Cohere$_{Multi}$	76.5	73.7	74.9	–	–
Proprietary	Cohere$_{3.5}$	75.2	72.7	73.8	–	–
Proprietary	Voyage$_{v2}$	76.1	76.5	76.3	–	–
Proprietary	VertexAI	65.1	64.7	64.9	–	–
Ours	E2Rank$_{Marco}$	76.7	75.9	76.3	304M	8.2
Ours	E2Rank$_{QA}$	72.8	70.5	71.5	304M	8.2
Ours	E2Rank	76.5	76.7	**76.6**	304M	8.2

Table 2 shows the results for several reranking models on the BEIR datasets. E2Rank is the best performing model, reaching the highest average NDCG@10 across all datasets. E2Rank outperforms the other models by achieving the best performance on 4 out of the 12 datasets, the highest number in our evaluation.

Table 2. Effectiveness as NDCG@10 on 12 datasets of BEIR and their average.

Model	c-fever	dbpedia	fever	fiqa	hotpotqa	nfcorpus	nq	quora	scidocs	scifact	touche	t-covid	Avg
Naver	26.6	48.9	87.2	50.1	75.2	37.7	66.0	84.1	19.7	77.1	33.3	**89.2**	57.9
BGE$_{M3}$	35.6	47.8	89.8	42.6	82.4	33.5	69.1	**89.1**	17.0	73.4	34.6	82.7	58.1
JinaAI	31.6	49.4	**92.1**	44.8	79.6	37.1	67.1	87.7	19.5	76.8	33.3	76.6	58.0
Cohere$_{En}$	29.3	45.6	88.0	45.6	76.4	37.2	62.2	83.7	19.2	76.7	27.9	86.6	56.5
Cohere$_{Multi}$	24.6	45.0	87.3	40.8	73.7	33.7	62.6	76.5	17.1	74.1	**37.6**	80.6	54.5
Cohere$_{3.5}$	34.3	49.3	90.7	47.4	79.7	35.1	69.9	87.2	17.8	77.3	33.5	84.8	58.9
Voyage$_{v2}$	24.2	48.8	87.6	**53.2**	80.8	37.7	69.9	85.9	18.7	76.3	25.1	84.7	57.7
VertexAI	23.5	36.6	55.9	36.8	69.5	31.4	52.9	66.8	15.8	72.7	27.3	66.3	46.3
E2Rank$_{Marco}$	29.9	50.2	87.5	50.7	75.9	**39.1**	66.3	83.1	19.7	76.7	34.7	87.6	58.5
E2Rank$_{QA}$	**40.8**	47.7	90.5	45.5	**83.4**	34.7	**72.2**	79.9	19.3	77.1	26.6	78.3	58.0
E2Rank	34.3	**52.1**	89.2	51.1	80.5	38.8	69.9	83.7	**20.1**	**78.3**	33.8	87.7	**59.9**

Efficiency. In this experiment, we evaluate the efficiency-effectiveness trade-offs for the only two models that allow it, BGE$_{Layer}$ and E2Rank, as shown in Figure 2. For BGE$_{Layer}$, the model is configured with different layer cutoffs (16, 24, 32, and 40 layers) to explore varying levels of computational cost. As expected, using fewer layers results in lower latency, but with a trade-off in effectiveness, indicated by a decrease in NDCG@10 scores. The performance improves as more layers are included, but at the expense of increased latency. In contrast, E2Rank

is evaluated with three configurations. First, we test it with all 24 layers, then with only the first 8 layers, and finally with a multi-step reranking strategy. In the multi-step approach, all documents are initially reranked using just 8 layers, then the top candidates are progressively reranked using 16 and finally 24 layers. This strategy enables E2Rank to achieve a balanced reduction in latency while maintaining quality by concentrating computational resources on fewer, more relevant documents. The results place E2Rank on the Pareto frontier, indicating that it achieves better efficiency-effectiveness trade-offs compared to BGE_{Layer}.

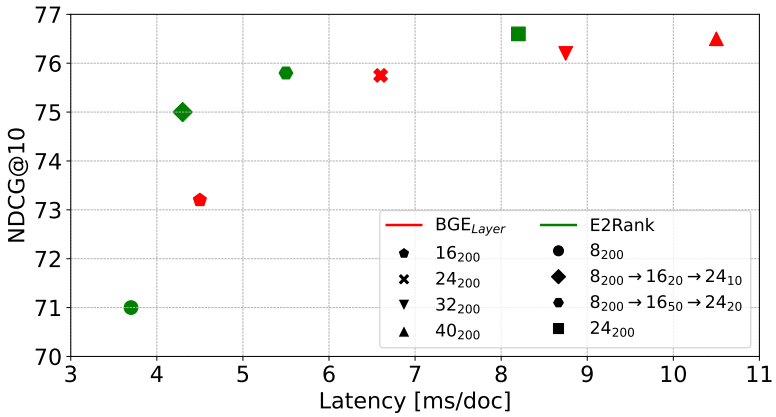

Fig. 2. Effectiveness-efficiency trade-offs for BGE_{Layer} and E2Rank on the weighted average between TREC 2019 and TREC 2020.

4 Conclusion

We introduced E2Rank, a novel layer-wise reranking model that balances efficiency and effectiveness. Our experiments show that E2Rank achieves state-of-the-art performance across a range of datasets while lowering computational costs compared to existing models. Our layer-wise reranking strategy enables progressive result refine offering adjustable efficiency-effectiveness trade-offs.

Acknowledgments. We would like to express our gratitude to Ram Sriharsha for his invaluable support throughout this work and Torsten Suel for the insightful discussions and constructive feedback, which significantly enriched the quality of this research. This work was partially supported by projects FAIR (PE0000013) and SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union – NextGenerationEU. Supported also by the project NEREO (Neural Reasoning over Open Data) project funded by the Italian Ministry of Education and Research (PRIN) Grant no. 2022AEFHA and project SEED funded by Sapienza University of Rome.

Bibliography

[1] Asadi, N., Lin, J.: Effectiveness/efficiency tradeoffs for candidate generation in multi-stage retrieval architectures. In: Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval. pp. 997–1000 (2013)

[2] Basnet, S., Gou, J., Mallia, A., Suel, T.: Deeperimpact: Optimizing sparse learned index structures. arXiv preprint arXiv:2405.17093 (2024)

[3] Chen, J., Xiao, S., Zhang, P., Luo, K., Lian, D., Liu, Z.: Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation (2024)

[4] Chen, J., Li, J., Majumder, R.: Make every feature binary: A 135b parameter sparse neural network for massively improved search relevance. https://www.microsoft.com/enus/research/blog/make-every-feature-binary-a-135b-parameter-sparse-neural-network-formassively-improved-search-relevance (2021), accessed: 2024-09-19

[5] Cohere: Introducing rerank 3: A new foundation model for efficient enterprise search & retrieval. https://cohere.com/blog/rerank-3 (2024), accessed: 2024-09-19

[6] Craswell, N., Mitra, B., Yilmaz, E., Campos, D.: Overview of the trec 2020 deep learning track. arXiv:2102.07662 (2021)

[7] Craswell, N., Mitra, B., Yilmaz, E., Campos, D., Voorhees, E.M.: Overview of the trec 2019 deep learning track. arXiv:2003.07820 (2020)

[8] Cuconasu, F., Trappolini, G., Siciliano, F., Filice, S., Campagnano, C., Maarek, Y., Tonellotto, N., Silvestri, F.: The power of noise: Redefining retrieval for rag systems. In: Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval. p. 719–729. SIGIR '24, Association for Computing Machinery, New York, NY, USA (2024). https://doi.org/10.1145/3626772.3657834, https://doi.org/10.1145/3626772.3657834

[9] Déjean, H., Clinchant, S., Formal, T.: A thorough comparison of cross-encoders and llms for reranking splade. arXiv preprint arXiv:2403.10407 (2024)

[10] Devlin, J.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)

[11] He, P., Gao, J., Chen, W.: Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. arXiv preprint arXiv:2111.09543 (2021)

[12] JinaAI: Jina reranker v2 for agentic rag: Ultra-fast, multilingual, function-calling & code search. https://jina.ai/news/jina-reranker-v2-for-agentic-rag-ultra-fast-multilingual-function-calling-and-code-search/ (2024), accessed: 2024-09-19

[13] Karpukhin, V., Oğuz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., Yih, W.t.: Dense passage retrieval for open-domain question answering. arXiv preprint arXiv:2004.04906 (2020)
[14] Lassance, C., Déjean, H., Formal, T., Clinchant, S.: Splade-v3: New baselines for splade. arXiv preprint arXiv:2403.06789 (2024)
[15] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.t., Rocktäschel, T., et al.: Retrieval-augmented generation for knowledge-intensive nlp tasks. Advances in Neural Information Processing Systems **33**, 9459–9474 (2020)
[16] Li, C., Liu, Z., Xiao, S., Shao, Y.: Making large language models a better foundation for dense retrieval (2023)
[17] Lu, W., Jiao, J., Zhang, R.: Twinbert: Distilling knowledge to twin-structured compressed bert models for large-scale retrieval. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management. pp. 2645–2652 (2020)
[18] 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)
[19] 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)
[20] Nguyen, T., Rosenberg, M., Song, X., Gao, J., Tiwary, S., Majumder, R., Deng, L.: Ms marco: A human-generated machine reading comprehension dataset (2016)
[21] Nogueira, R., Jiang, Z., Pradeep, R., Lin, J.: Document ranking with a pretrained sequence-to-sequence model. In: Findings of the Association for Computational Linguistics: EMNLP 2020 (2020)
[22] Pal, V., Lassance, C., Déjean, H., Clinchant, S.: Parameter-efficient sparse retrievers and rerankers using adapters. In: European Conference on Information Retrieval. pp. 16–31. Springer (2023)
[23] Paliwal, B., Saini, D., Dhawan, M., Asokan, S., Natarajan, N., Aggarwal, S., Malhotra, P., Jiao, J., Varma, M.: Cross-jem: Accurate and efficient cross-encoders for short-text ranking tasks. arXiv preprint arXiv:2409.09795 (2024)
[24] Reddy, R.G., Doo, J., Xu, Y., Sultan, M.A., Swain, D., Sil, A., Ji, H.: First: Faster improved listwise reranking with single token decoding. arXiv preprint arXiv:2406.15657 (2024)
[25] Santhanam, K., Khattab, O., Saad-Falcon, J., Potts, C., Zaharia, M.: Colbertv2: Effective and efficient retrieval via lightweight late interaction. In: Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 3715–3734 (2022)
[26] Schlatt, F., Fröbe, M., Scells, H., Zhuang, S., Koopman, B., Zuccon, G., Stein, B., Potthast, M., Hagen, M.: A systematic investigation of distill-

ing large language models into cross-encoders for passage re-ranking. arXiv preprint arXiv:2405.07920 (2024)
[27] Thakur, N., Reimers, N., Rücklé, A., Srivastava, A., Gurevych, I.: BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In: Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2) (2021)
[28] VoyageAI: rerank-2 and rerank-2-lite: the next generation of Voyage multilingual rerankers. https://blog.voyageai.com/2024/09/30/rerank-2/ (2024), accessed: 2024-10-16
[29] Wang, L., Yang, N., Huang, X., Yang, L., Majumder, R., Wei, F.: Multilingual e5 text embeddings: A technical report. arXiv preprint arXiv:2402.05672 (2024)
[30] Wang, Q., Dimopoulos, C., Suel, T.: Fast first-phase candidate generation for cascading rankers. In: Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval. pp. 295–304 (2016)
[31] Wortsman, M., Ilharco, G., Gadre, S.Y., Roelofs, R., Gontijo-Lopes, R., Morcos, A.S., Namkoong, H., Farhadi, A., Carmon, Y., Kornblith, S., et al.: Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In: International conference on machine learning. pp. 23965–23998 (2022)
[32] Xiong, L., Xiong, C., Li, Y., Tang, K.F., Liu, J., Bennett, P., Ahmed, J., Overwijk, A.: Approximate nearest neighbor negative contrastive learning for dense text retrieval. arXiv preprint arXiv:2007.00808 (2020)
[33] Yates, A., Nogueira, R., Lin, J.: Pretrained transformers for text ranking: Bert and beyond. In: Proceedings of the 14th ACM International Conference on web search and data mining. pp. 1154–1156 (2021)
[34] Zhan, J., Mao, J., Liu, Y., Guo, J., Zhang, M., Ma, S.: Optimizing dense retrieval model training with hard negatives. In: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 1503–1512 (2021)