LiT and Lean: Distilling Listwise Rerankers into Encoder-Decoder Models

Manveer Singh Tamber, Ronak Pradeep, and Jimmy Lin

David R. Cheriton School of Computer Science,
University of Waterloo, Waterloo, Ontario, Canada

{mtamber,rpradeep,jimmylin}@uwaterloo.ca

Abstract. Large-scale LLMs have driven listwise reranking research, achieving impressive state-of-the-art results. However, their massive parameter counts and limited context sizes limit efficient reranking. To address this, we present LiT5, a family of efficient listwise rerankers based on the T5 model. Our approach demonstrates competitive reranking effectiveness compared to listwise LLM rerankers, with far fewer parameters, greater computational efficiency, and the ability to rerank more passages in a single shot. Our models consistently deliver strong effectiveness with as few as 220M parameters, offering a scalable solution for listwise reranking. Code and scripts for reproducibility are available at https://github.com/castorini/rank_llm.

Keywords: Listwise Reranking · Efficient Reranking · Encoder-Decoder

1 Introduction

Listwise reranking using LLMs has seen success in recent work and has attained state-of-the-art results for reranking [18,14,15,11]. These approaches leverage the extensive capabilities of LLMs to take a query and a list of passages and rank the passages in terms of relevance to the query, considering all of the passages together. However, these listwise rerankers rely on large LLMs with billions of parameters and limited context-window sizes. This reliance on large-scale models introduces challenges in terms of computational demands.

This paper introduces LiT5, a family of models that leverages the Fusion-in-Decoder (FiD) architecture [7] to build efficient listwise rerankers based on T5 [17]. To our knowledge, we are the first to show that it is possible to distill large LLM listwise rerankers into much smaller models while maintaining competitive effectiveness. Our approach, based on encoder–decoder models, is both effective and computationally efficient, allowing for the first time, the listwise reranking of 100 passages in a single shot, surpassing the 20-passage limit from previous work due to context-window limitations.

2 Background and Related Work

The Fusion-in-Decoder (FiD) [7] model, based on T5, has proven highly effective in knowledge-intensive tasks like open-domain question answering [8]. FiD mod-

ifies the T5 encoder–decoder architecture to take multiple passages, encoding each one individually, then performs attention over the concatenated encoded passages in the decoder, with both encoder and decoder computations scaling linearly with the number of passages [9]. While the encoder processes each passage separately, the decoder synthesizes information across passages to produce an answer. In contrast, decoder-only LLMs scale quadratically in computation when handling multiple passages in a prompt, unless additional methods are used to manage this complexity [19].

RankGPT [18] demonstrated that $GPT_{3.5}$ and GPT_4 are strong listwise rerankers, with $RankGPT_4$ achieving state-of-the-art results. The work also showed that RankGPT's reranking effectiveness can be distilled into smaller cross-encoders, though with room for improvement. Later work distilled the effectiveness of RankGPT into smaller LLMs [14,15]. This distillation used instruction fine-tuning to train student models to replicate the teacher's rankings. In our LiT5 method, we distill ranking reorderings from a RankZephyr teacher model into smaller encoder–decoder models for efficient listwise reranking.

3 Methods

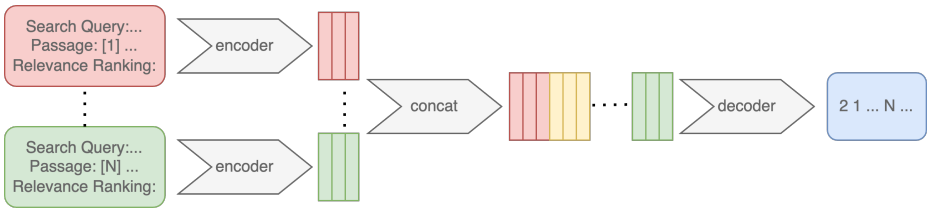

Fig. 1. LiT5 architecture. Each query–passage pair is encoded separately. Then, the decoder reads over the concatenated representations to generate a ranking such as: "2 1 . . . N . . . ".

We build on previous work [14,15] that uses RankGPT as a teacher model to distill ranking orderings into listwise student reranking models. One such model is RankZephyr [15], which bridges the effectiveness gap with GPT_4 and in some cases even surpasses the proprietary teacher. In LiT5, we use RankZephyr as a teacher model for distillation, avoiding the API costs of GPT_4 and leveraging a more transparent, open-source model.

Following the FiD architecture as shown in Figure 1, the model encoder encodes each passage individually alongside the query. For each query–passage pair, the input prompt begins with `Search Query:`, followed by the query, then `Passage:` with a unique numerical identifier (e.g., [1], [2]), and finally the passage text. The prompt ends with `Relevance Ranking:` to prompt the model to generate a ranking. The decoder then reads over the concatenated encoded token representations of all passages to produce an ordering of passage identifiers based on relevance, from most to least relevant, for example, "3 1 2 . . . ".

By fine-tuning LiT5 to produce the same orderings from RankZephyr for up to 100 passages, we train LiT5 to reorder up to 100 passages in a single shot. This eliminates the need for the sliding window strategy used in RankGPT, RankVicuna, and RankZephyr, which were limited to smaller context windows of 20 passages due to context-window constraints.

Model Training To train LiT5, we randomly sampled 20K queries from the MS MARCO v1 passage ranking dataset. For each query, we retrieved 100 passages from *both* the MS MARCO v1 and v2 passage corpora. For the MS MARCO v1 corpus, we used OpenAI's ADA_2 model due to its strong retrieval effectiveness. For MS MARCO v2, we used BM25 with RM3 for simplicity, avoiding the computational overhead of encoding the larger corpus. The top 100 retrieved passages were reranked using RankZephyr's sliding window approach, which includes three passes to optimize ranking effectiveness. LiT5 was trained to produce the same orderings as its RankZephyr teacher. LiT5 models were trained with variable window sizes and shuffled input orderings, similar to RankZephyr, to handle reranking a variable number of passages and passages provided in a random order. Of the 20K queries, 1.25K were set aside as a dev set, and models were evaluated after each epoch, to select the model with the lowest dev set loss.

When ranking passages, we observed that ordering the most relevant passages at the top is more critical than ordering less relevant ones further down. To capture this intuition, we applied a weighted cross-entropy loss function with exponential decay (0.95 weight per subsequent token), ensuring a higher focus on top-ranking passages while effectively handling up to 100 passages.

We adopted hyperparameters similar to FiD work [8], as these proved effective, using a batch size of 64, a 10% dropout, and the AdamW optimizer with a 5e-5 learning rate and 100-step linear warmup. For MS MARCO passage reranking, we limited the combined length of the query and passage to 150 tokens using the T5 SentencePiece tokenizer. For BEIR dataset reranking, the token limit was increased to 512 to accommodate longer queries and passages. Training the $LiT5_{base}$, $LiT5_{large}$, and $LiT5_{XL}$ models on 8 TPU-v4 cores took approximately 53, 53, and 80 hours, respectively. All model evaluations were performed on an NVIDIA RTX A6000 GPU, post-conversion to a PyTorch checkpoint.

Model Initialization To avoid data contamination, we initialized our models with the T5 1.1 LM-Adapted weights [17], rather than the FLAN-T5 weights [1], since the FLAN mixture includes both the MS MARCO QA task, as previously noted [14] and some BEIR datasets. We initialized the LiT5 models in three sizes (base, large, and XL) to explore model effectiveness across different scales, resulting in the $LiT5_{base}$, $LiT5_{large}$, and $LiT5_{XL}$ variants.

4 Results

For evaluation, we focused on MS MARCO passage reranking. We evaluate using the TREC Deep Learning Tracks from 2019 to 2022 [5,2,3,4] (referred to as

	Model Params	Source Prev.	MSv1 DL19	MSv1 DL20	MSv2 DL21	MSv2 DL22
(1a) BM25	-	None	0.506†	0.480†	0.446†	0.269†
(2a) SPLADE++ED	110M	None	0.731	0.720†	0.684	0.570†
(3a) MonoT5 [16]	220M	BM25	0.715	0.670	-	-
(3b) MonoT5 [16]	3B	BM25	0.718	0.689	-	-
(4a) RankT5 [16]	3B	BM25	0.712	0.695	-	-
(5a) RankVicuna	7B	BM25	0.668	0.655	0.624†	0.430†
(5b) RankVicuna	7B	SPLADE++ED	0.746	0.747	0.701	0.582†
(6a) RankZephyr	7B	BM25	0.742	0.709	0.703	0.515
(6b) RankZephyr	7B	SPLADE++ED	0.782	**0.816**	0.760	0.669
(7a) $\text{RankGPT}_{3.5}$ [15]	?	BM25	0.686	0.620†	0.605†	0.418†
(8a) RankGPT_{4} [15]	?	BM25	0.750	0.704	0.707	0.508
(8b) RankGPT_{4} [15]	?	SPLADE++ED	0.746	0.708†	**0.772**	**0.718**
(9a) $\text{LiT5}_{\text{base}}$	220M	BM25	0.717	0.667	0.645	0.484
(9b) $\text{LiT5}_{\text{base}}$	220M	SPLADE++ED	0.783	0.751	0.693	0.626†
(9c) $\text{LiT5}_{\text{large}}$	770M	BM25	0.733	0.698	0.679	0.512
(9d) $\text{LiT5}_{\text{large}}$	770M	SPLADE++ED	**0.800**	0.766	0.728	0.686
(9e) LiT5_{XL}	3B	BM25	0.730	0.737	0.703	0.512
(9f) LiT5_{XL}	3B	SPLADE++ED	0.785	0.804	0.747	0.696

Table 1. nDCG@10 on DL19–DL22. Each reranker reranks top-100 BM25 or SPLADE++ED passages. Best scores are in bold, and the best scores with BM25 are underlined. A † indicates a significant difference from LiT5_{XL} reranking BM25 or SPLADE++ED, based on a one-sided, one-sample t-test ($p < 0.05$, Holm-Bonferroni corrected).

DL19–DL22). In addition, we also test using the BEIR collection [20], which spans a variety of diverse text retrieval tasks and domains.

4.1 Effectiveness

LiT5 demonstrates effective listwise reranking capabilities across all model sizes, competing with and sometimes surpassing current state-of-the-art models. We examine nDCG@10 scores for reranking the top-100 documents returned by first-stage retrieval using either BM25 or SPLADE++ EnsembleDistil (SPLADE++ED) [6]. BM25 provides a common baseline, while SPLADE++ED serves as a stronger supervised first-stage method.

MS MARCO In Table 1, we examine the reranking of MS MARCO passages across DL19–DL22 test collections. We observe that LiT5's reranking effectiveness improves with model size. $\text{LiT5}_{\text{base}}$ has the lowest scores for DL19-DL22, while LiT5_{XL} generally scores highest, though $\text{LiT5}_{\text{large}}$ slightly edges out LiT5_{XL} on DL19.

Comparisons to supervised T5-based methods Both MonoT5 [12] and RankT5 [21] are pointwise rerankers trained on the MS MARCO v1 training set. $\text{LiT5}_{\text{large}}$ (770M parameters) and LiT5_{XL} (3B parameters) outperform both 3B parameter MonoT5 and RankT5 models across datasets, shown in rows 3 and 4 vs 9(c, e).

	BM25	LiT5			RankZephyr
Dataset	-	220M	770M	3B	7B
TREC-COVID	59.5	79.5	82.1	81.8	85.6
BioASQ	52.3	55.2	57.4	58.2	55.6
NFCorpus	32.2	34.2	34.9	36.1	32.2
NQ	30.6	52.9	56.1	57.7	56.9
HotpotQA	63.3	68.8	72.0	73.8	72.1
FiQA	23.6	36.5	40.0	41.7	38.7
Signal-1M	33.0	31.5	32.2	32.0	31.5
TREC-NEWS	39.5	48.0	49.9	49.4	52.2
Robust04	40.7	52.7	56.5	55.4	54.7
Arguana	39.7	29.7	35.2	39.2	42.7
Touche-2020	44.2	32.8	34.1	34.4	32.9
Quora	78.9	80.7	84.7	85.4	80.6
DBPedia	31.8	40.7	43.6	44.7	44.6
SCIDOCS	14.9	16.4	18.8	19.3	19.3
FEVER	65.1	77.6	78.1	81.6	77.1
Climate-FEVER	16.5	22.0	21.9	22.9	23.5
SciFact	67.9	72.4	74.1	74.9	76.0
Average	43.2	48.9	51.3	52.3	51.5

Table 2. Average nDCG@10 score for reranking the top 100 passages returned by BM25 on all BEIR datasets, with the exception of CQADupStack for simplicity. The listed numbers underneath the models (220M, 770M...) indicate the parameter counts for LiT5 and RankZephyr.

Comparisons to listwise rerankers LiT5 outperforms both $\text{RankGPT}_{3.5}$ and RankVicuna, which is distilled from $\text{RankGPT}_{3.5}$. This advantage holds even for smaller LiT5 models, due to their effective distillation from RankZephyr, a model trained with RankGPT_4 as a teacher. LiT5 is also competitive with RankGPT_4 and RankZephyr. The scores are very close, with neither model scoring significantly higher than LiT5_{XL} ($p < 0.05$) in any collection.

LiT5_{XL} is able to score higher than RankZephyr and RankGPT_4 in many cases. With BM25 retrieval, LiT5_{XL} excels over RankGPT_4 in DL20 and DL22, shown in row 8(a), and over RankZephyr in DL20, shown in row 6(a). With SPLADE++ED first-stage retrieval, $\text{LiT5}_{\text{large}}$ and LiT5_{XL} have stronger reranking effectiveness than RankZephyr on DL19 and DL22, shown in rows 9(d,f) vs 6(b). All LiT5 models have stronger reranking effectiveness on DL19 and DL20 than RankGPT_4. Nonetheless, all LiT5 models have weaker effectiveness on DL21 and DL22 than RankGPT_4. Interestingly, the same is true for RankZephyr, which outperforms RankGPT_4 on both DL19 and DL20, but it underperforms RankGPT_4 on DL21 and DL22. This may be a result of RankZephyr being trained to rerank MS MARCO v1 passages only.

Considering SPLADE++ED retrieval, $\text{LiT5}_{\text{large}}$ achieves the highest ranking effectiveness score on DL19, RankZephyr has the strongest reranking effectiveness on DL20, and RankGPT_4 has the highest scores on DL21 and DL22. These best scores are bolded in the table. Considering BM25 first-stage retrieval, we underline the highest scores. LiT5_{XL} achieves the strongest reranking effectiveness on DL20, RankZephyr achieves the highest scores on DL22, and RankGPT_4 scores strongest on DL19 and DL21. Our methods along with RankZephyr and RankGPT_4 often trade places in attaining the highest reranking effectiveness

scores. We have shown that even though RankGPT$_4$ and RankZephyr are much larger than the LiT5 models in parameters, our results indicate that LiT5 can achieve reranking effectiveness that rivals that of RankGPT$_4$ and RankZephyr.

BEIR For BEIR test collections, we examine reranking the top 100 passages retrieved using BM25 retrieval in Table 2. The table shows the average nDCG@10 score for reranking on all BEIR datasets, with the exception of CQADupStack for simplicity. We observe that reranking with LiT5 improves nDCG@10 scores compared to BM25 scores, suggesting that the LiT5 models generalize well. The tables show that as our models are scaled up in parameters, the average scores improve. LiT5$_{\text{XL}}$ typically delivers the strongest results among all our models. We also evaluate RankZephyr on BEIR datasets ourselves using vLLM [10]. We see that LiT5$_{\text{XL}}$ attains a higher average score compared to RankZephyr. This is despite LiT5 being trained with RankZephyr as the teacher. We suspect that RankZephyr likely suffers in BEIR reranking due to its limited context window, forcing passages to be cut off.

4.2 Model Efficiency

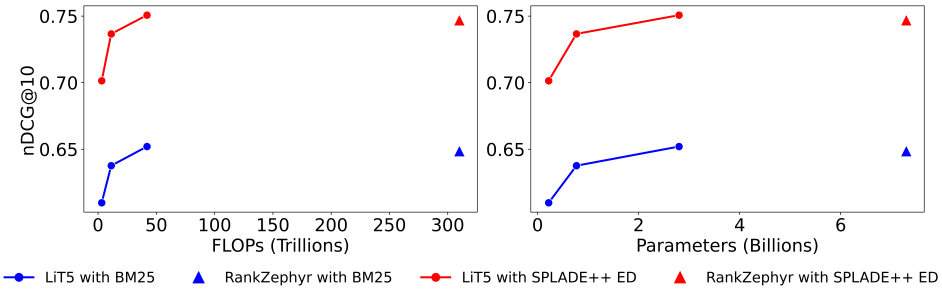

Fig. 2. Average nDCG@10 across DL collections weighted by the number of queries for the LiT5 model variants vs. RankZephyr, comparing reranking of the top 100 BM25 passages by average FLOPs per query (left) and model parameters (right).

LiT5 is designed with fewer parameters and higher computational efficiency than RankVicuna and RankZephyr. By encoding passages separately and decoding over their concatenated representations, LiT5 achieves linear computation scaling with the number of passages, unlike the quadratic scaling discussed in Section 2. Our model variants, initialized with T5 models of different sizes, balance computational cost with reranking effectiveness.

RankZephyr and RankGPT also make use of a sliding window approach for reranking, handling only 20 passages at a time. This means that these models would have to repeatedly consider some passages in the sliding window pass

to rerank 100 passages. It is also worth mentioning that Pradeep et al., have shown that performing up to three sliding window passes generally results in slightly better final effectiveness for reranking 100 passages [14]. However, this is at the cost of even more computation, making these methods even more costly in comparison to LiT5.

In Figure 2, we present average nDCG@10 scores across DL19-22 for LiT5 and RankZephyr as a function of average FLOPs and model parameters. Both models benefit from KV caching [13], significantly reducing FLOPs. The triangular points depict RankZephyr's efficiency/effectiveness, showing that LiT5 models have fewer parameters and require fewer FLOPs than RankZephyr. Although RankZephyr has 2.6× the parameters of LiT5_{XL}, it needs 7.4× the FLOPs for reranking. This demonstrates LiT5 's superior parameter efficiency, with LiT5_{XL} also achieving a higher weighted average nDCG@10 score, indicating a much better efficiency-effectiveness trade-off than RankZephyr.

5 Conclusion and Future Work

We introduce LiT5, a family of efficient listwise rerankers, demonstrating that strong reranking effectiveness from models like RankZephyr can be distilled into smaller, computationally efficient encoder–decoder models. LiT5 enables the reranking of up to 100 passages in a single shot, surpassing prior context-window limitations. Our largest LiT5 model, LiT5_{XL}, in some cases, scores higher than the current state-of-the-art listwise rerankers RankGPT_4 and RankZephyr, despite having much fewer parameters, greater computational efficiency, and building on the outdated T5 model.

LiT5 also shows stronger effectiveness than supervised methods trained with human-annotated relevance labels using the same T5 models. We test models ranging from 220M to 3B parameters, showing that a small 220M parameter model can *still* excel at listwise reranking, all while generalizing well to out-of-domain reranking tasks and running efficiently.

The findings from this work have some interesting applications for future work. We show that we can successfully distill reranking effectiveness from RankZephyr to much smaller encoder-decoder models with the added benefits of greater computational efficiency and being able to rerank 100 passages at once instead of only 20. As listwise rerankers advance, these effective teachers can be distilled into compact LiT5 models, offering scalable, efficient alternatives capable of reranking more passages at once in a listwise manner.

Acknowledgements

This research was supported in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada. We thank Google Cloud and the TPU Research Cloud Program for credits to support some of our experimental runs.

References

1. Chung, H.W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, Y., Wang, X., Dehghani, M., Brahma, S., Webson, A., Gu, S.S., Dai, Z., Suzgun, M., Chen, X., Chowdhery, A., Castro-Ros, A., Pellat, M., Robinson, K., Valter, D., Narang, S., Mishra, G., Yu, A., Zhao, V., Huang, Y., Dai, A., Yu, H., Petrov, S., Chi, E.H., Dean, J., Devlin, J., Roberts, A., Zhou, D., Le, Q.V., Wei, J.: Scaling Instruction-Finetuned Language Models. arXiv:2210.11416 (2022)
2. Craswell, N., Mitra, B., Yilmaz, E., Campos, D.: Overview of the TREC 2020 deep learning track. In: Proceedings of the Twenty-Ninth Text REtrieval Conference Proceedings (TREC 2020). Gaithersburg, Maryland (2020)
3. Craswell, N., Mitra, B., Yilmaz, E., Campos, D., Lin, J.: Overview of the TREC 2021 deep learning track. In: Proceedings of the Thirtieth Text REtrieval Conference (TREC 2021) (May 2021)
4. Craswell, N., Mitra, B., Yilmaz, E., Campos, D., Lin, J., Voorhees, E.M., Soboroff, I.: Overview of the TREC 2022 deep learning track. In: Proceedings of the Thirty-First Text REtrieval Conference (TREC 2021). Gaithersburg, Maryland (2022)
5. Craswell, N., Mitra, B., Yilmaz, E., Campos, D., Voorhees, E.M.: Overview of the TREC 2019 deep learning track. In: Proceedings of the Twenty-Eighth Text REtrieval Conference Proceedings (TREC 2019). Gaithersburg, Maryland (2019)
6. Formal, T., Lassance, C., Piwowarski, B., Clinchant, S.: 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. p. 2353–2359. SIGIR '22, Association for Computing Machinery, New York, NY, USA (2022)
7. Izacard, G., Grave, E.: Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering. In: EACL 2021-16th Conference of the European Chapter of the Association for Computational Linguistics. pp. 874–880. Association for Computational Linguistics (2021)
8. Izacard, G., Lewis, P., Lomeli, M., Hosseini, L., Petroni, F., Schick, T., Dwivedi-Yu, J., Joulin, A., Riedel, S., Grave, E.: Few-shot Learning with Retrieval Augmented Language Models. arXiv:2208.03299 (2022)
9. de Jong, M., Zemlyanskiy, Y., Ainslie, J., FitzGerald, N., Sanghai, S., Sha, F., Cohen, W.: FiDO: Fusion-in-Decoder optimized for stronger performance and faster inference. arXiv:2212.08153 (2022)
10. Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C.H., Gonzalez, J.E., Zhang, H., Stoica, I.: Efficient memory management for large language model serving with pagedattention. In: Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles (2023)
11. Ma, X., Zhang, X., Pradeep, R., Lin, J.: Zero-Shot Listwise Document Reranking with a Large Language Model. arXiv:2305.02156 (2023)
12. Nogueira, R., Jiang, Z., Pradeep, R., Lin, J.: Document Ranking with a Pretrained Sequence-to-Sequence Model. In: Cohn, T., He, Y., Liu, Y. (eds.) Findings of the Association for Computational Linguistics: EMNLP 2020. pp. 708–718. Online (Nov 2020)
13. Pope, R., Douglas, S., Chowdhery, A., Devlin, J., Bradbury, J., Heek, J., Xiao, K., Agrawal, S., Dean, J.: Efficiently scaling transformer inference. Proceedings of Machine Learning and Systems **5** (2023)
14. Pradeep, R., Sharifymoghaddam, S., Lin, J.: RankVicuna: Zero-Shot Listwise Document Reranking with Open-Source Large Language Models. arXiv:2309.15088 (2023)

15. Pradeep, R., Sharifymoghaddam, S., Lin, J.: RankZephyr: Effective and Robust Zero-Shot Listwise Reranking is a Breeze! arXiv:2312.02724 (2023)
16. Qin, Z., Jagerman, R., Hui, K., Zhuang, H., Wu, J., Shen, J., Liu, T., Liu, J., Metzler, D., Wang, X., et al.: Large Language Models are Effective Text Rankers with Pairwise Ranking Prompting. arXiv:2306.17563 (2023)
17. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J.: Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. The Journal of Machine Learning Research **21**(1), 5485–5551 (2020)
18. Sun, W., Yan, L., Ma, X., Wang, S., Ren, P., Chen, Z., Yin, D., Ren, Z.: Is ChatGPT Good at Search? Investigating Large Language Models as Re-Ranking Agents. In: Bouamor, H., Pino, J., Bali, K. (eds.) Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. pp. 14918–14937. Singapore (Dec 2023)
19. Tay, Y., Dehghani, M., Bahri, D., Metzler, D.: Efficient Transformers: A Survey. ACM Comput. Surv. **55**(6) (dec 2022)
20. Thakur, N., Reimers, N., Rücklé, A., Srivastava, A., Gurevych, I.: BEIR: A Heterogenous Benchmark for Zero-shot Evaluation of Information Retrieval Models. arXiv:2104.08663 (2021)
21. Zhuang, H., Qin, Z., Jagerman, R., Hui, K., Ma, J., Lu, J., Ni, J., Wang, X., Bendersky, M.: RankT5: Fine-Tuning T5 for Text Ranking with Ranking Losses. In: Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 2308–2313 (2023)