A Systematic Investigation of Distilling Large Language Models into Cross-Encoders for Passage Re-ranking

Ferdinand Schlatt[1], Maik Fröbe[1], Harrisen Scells[2], Shengyao Zhuang[3,4], Bevan Koopman[3], Guido Zuccon[4], Benno Stein[5], Martin Potthast[6,7,8], Matthias Hagen[1]

[1]Friedrich-Schiller-Universität Jena, [2]Leipzig University, [3]CSIRO, [4]University of Queensland, [5]Bauhaus-Universität Weimar, [6]University of Kassel, [7]hessian.AI, [8]ScadDS.AI

Correspondence: ferdinand.schlatt@uni-jena.de

Abstract

Cross-encoders distilled from large language models (LLMs) are often more effective re-rankers than cross-encoders fine-tuned on manually labeled data. However, the distilled models usually do not reach their teacher LLM's effectiveness. To investigate whether best practices for fine-tuning cross-encoders on manually labeled data (e.g., hard-negative sampling, deep sampling, and listwise loss functions) can help to improve LLM ranker distillation, we construct and release a new distillation dataset: Rank-DistiLLM. In our experiments, cross-encoders trained on Rank-DistiLLM reach the effectiveness of LLMs while being orders of magnitude more efficient. Our code and data is available at `https://github.com/webis-de/msmarco-llm-distillation`.

1 Introduction

Cross-encoders (Akkalyoncu Yilmaz et al., 2019; Nogueira and Cho, 2020; Xiong et al., 2021) using pre-trained transformer-based models are among the most effective passage re-rankers (Hofstätter et al., 2021; Rosa et al., 2022). However, they require large amounts of labeled data for fine-tuning. In contrast, large language models (LLMs) require no further fine-tuning to excel in re-ranking tasks (Sun et al., 2023; Pradeep et al., 2023a,b) and are often more effective than cross-encoders. The main drawback of LLMs is their computational cost. They are expensive to run and need several seconds to re-rank 100 passages for a single query. While this cost makes them impractical for production search engines, LLMs can be used to create training data for fine-tuning cross-encoders.

Initial work (Tamber et al., 2023; Baldelli et al., 2024) showed that cross-encoders distilled from LLMs are more effective re-rankers than cross-encoders fine-tuned on manually labeled data. However, the distilled cross-encoders did not reach the effectiveness of the LLMs. One reason could be that the distilled models were created without considering the best practices for fine-tuning re-rankers on manually labeled data. For example, no "hard-negative" sampling was used, which requires an effective first-stage retrieval model to sample data (Gao et al., 2021; Pradeep et al., 2022), at most 30 passages per query were provided, which is not deep enough to achieve optimal effectiveness (Zhuang et al., 2022), and no listwise losses were used, which are usually more effective than pair- and pointwise losses (Gao et al., 2021).

In this paper, we systematically investigate the distillation of cross-encoders from LLMs. Using our newly constructed Rank-DistiLLM dataset, we analyze the effect of the first-stage retrieval model, the ranking depth, and the amount of training data on the distilled cross-encoder's effectiveness. Additionally, we propose a novel listwise loss function for distillation from ranking data.

In an evaluation on the TREC 2019 and 2020 Deep Learning tracks (Craswell et al., 2019, 2020) and in the TIREx framework (Fröbe et al., 2023), we find that our listwise loss function yields no benefit over a pairwise loss function. However, distilling cross-encoders using our new Rank-DistiLLM dataset, which follows best practices like hard-negative sampling and deeper rankings, helps to close the effectiveness gap to LLMs: our distilled cross-encoders achieve similar effectiveness as state-of-the-art ranking LLMs while being orders of magnitude more efficient.

2 Related Work

MS MARCO Fine-Tuning MS MARCO is the most commonly used dataset for fine-tuning cross-encoders, as it contains over 500k query–passage pairs (Nguyen et al., 2016). However, most queries only have a single passage labeled as relevant. This label sparsity has two implications: (1) the options for suitable loss functions are limited and (2) "non-relevant" passages have to be sampled heuristically.

Regarding the first implication, listwise losses produce the most effective models (Gao et al., 2021; Pradeep et al., 2022; Zhuang et al., 2022). They use a single relevant passage and a set of k heuristically-sampled "non-relevant" passages to compute the loss. Generally, a higher k produces more effective models—with $k = 36$ being the highest reported value (Zhuang et al., 2022). We rely on recent work on memory-efficient fused-attention kernels (Dao et al., 2022; Lefaudeux et al., 2022; Dao, 2023) to circumvent the memory constraints of the Transformer's self-attention mechanism and fine-tune models on up to $k = 100$ passages.

Regarding the second implication, "hard negative" sampling, i.e., using an effective first-stage retrieval model to sample "non-relevant" samples, has produced the most effective models (Gao et al., 2021; Pradeep et al., 2022; Zhuang et al., 2022). For instance, models fine-tuned on negatives sampled from ColBERTv2 (Santhanam et al., 2022) are more effective than those fine-tuned on negatives sampled from BM25 (Robertson et al., 1994). However, Arabzadeh et al. (2022) found that MS MARCO contains passages that are more relevant than the labeled passage for a substantial number of queries, leading to noisy training data.

Distillation from LLMs To obtain less noisy data, Sun et al. (2023) proposed fine-tuning a cross-encoder on the rankings generated by an LLM applied in zero-shot manner. Models fine-tuned on their released dataset are more effective in low-data settings and out-of-domain re-ranking than those fine-tuned on MS MARCO. More recently, Baldelli et al. (2024) released a smaller dataset using a variety of first-stage retrieval models. Cross-encoders are even more effective when fine-tuned on this dataset, but an effectiveness gap between the cross-encoder and the LLMs remains. We investigate if this gap can be closed by applying the insights from fine-tuning on MS MARCO to LLM distillation.

3 Cross-Encoders

A cross-encoder processes the query and passage simultaneously using a pre-trained transformer-based encoder model. Given sequences of query tokens q and passage tokens p, the encoder's input sequence is [CLS] q [SEP] p [SEP], where [CLS] and [SEP] are special classification and separator tokens. The model outputs contextualized embedding vectors for every token. Using learnable weights $W \in \mathbb{R}^{d\times 1}$ and biases $b \in \mathbb{R}^1$, it then applies a linear layer to the [CLS] token's contextualized embedding $e_{[\text{CLS}]} \in \mathbb{R}^d$ to compute the relevance score $s_p = W \cdot e_{[\text{CLS}]} + b$.

3.1 Fine-Tuning on MS MARCO

Loss When fine-tuning cross-encoders on data sampled from MS MARCO, previous work obtains the most effective models by using listwise softmax cross entropy (Bruch et al., 2019a; Zhuang et al., 2022) or localized contrastive estimation loss (LCE) (Gao et al., 2021; Pradeep et al., 2022). Both are equivalent when only a single relevant passage is available. Given a set of passages $\mathcal{P}$ of which one is relevant $p_+ \in \mathcal{P}$, LCE is defined as:

$$\mathcal{L}_{\text{LCE}} = -\log \frac{\exp(s_{p_+})}{\sum_{p\in\mathcal{P}} \exp(s_p)}.$$

Data For highest effectiveness, $\mathcal{P}$ should be as large as possible, and the best available first-stage retrieval model should retrieve the other passages $\mathcal{P}_- = \mathcal{P} \setminus \{p_+\}$. Following Pradeep et al. (2022), we use ColBERTv2 (Santhanam et al., 2022) to retrieve the top 200 passages for all MS MARCO training queries. We then randomly sample up to 99 hard-negatives.

3.2 Fine-Tuning on LLM Distillation Data

Loss Instead of a set of passages $\mathcal{P}$, LLM distillation data consists of a ranked list of passages $\mathcal{R} = [p_1, p_2, \ldots, p_n]$ for a query q. Previous work (Sun et al., 2023; Baldelli et al., 2024) uses RankNet (Burges et al., 2005), a pairwise loss function, for distillation fine-tuning:

$$\mathcal{L}_{\text{RankNet}} = \sum_{i=1}^{n}\sum_{j=1}^{n} \mathbb{1}_{i<j} \log(1 + \exp(s_{p_i} - s_{p_j})),$$

where $\mathbb{1}$ is the indicator function.

For fine-tuning on MS MARCO, listwise loss functions are more effective than pairwise loss functions (Pradeep et al., 2022; Zhuang et al., 2022). To test if the same applies to LLM distillation, we propose a new loss function based on the Approx family of loss functions (Qin et al., 2010). Approx loss functions compute a smooth approximation of a passage's rank $\hat{\pi}(p)$ based on all passages' scores (see Appendix B for an in-depth description of $\hat{\pi}$). Since LLM distillation data does not contain explicit relevance judgments, we cannot apply previous Approx listwise losses directly. Our new loss, Approx Discounted Rank MSE (ADR-MSE), computes the mean squared error between a passage's actual and approximated rank. Inspired by nDCG, we also apply a logarithmic discount to give higher-ranked passages a higher weight. We define our loss function as:

$$\mathcal{L}_{\text{ADR-MSE}} = \sum_{i=1}^{n} \frac{1}{\log_2(i+1)} (i - \hat{\pi}(p_i))^2.$$

Data To our knowledge, only two datasets for distilling cross-encoders from LLMs have been released. Sun et al. (2023) released the first dataset (RankGPT) consisting of the top 20 passages retrieved by BM25 (Robertson et al., 1994) and re-ranked by RankGPT-3.5 for 100k queries from MS MARCO. Baldelli et al. (2024) released another dataset (TWOLAR) of the top 30 passages retrieved by three different retrieval models (BM25, DRAGON (Lin et al., 2023), and SPLADE (Formal et al., 2021)) and re-ranked by RankGPT-3.5 for a total of 20k-queries from MS MARCO. Cross-encoders fine-tuned on the TWOLAR dataset are more effective than when fine-tuned on the RankGPT dataset. Still, whether the improved first-stage retrieval models, deeper rankings, or both in combination lead to better effectiveness remains unclear.

We create Rank-DistiLLM to systematically investigate the effect of the first-stage retrieval model and the rank depth on a cross-encoder's downstream effectiveness. We retrieve the top 100 passages using BM25 and ColBERTv2 for 10k randomly sampled queries from the MS MARCO training set. We then use RankZephyr (Pradeep et al., 2023b), an open-source alternative to RankGPT, to re-rank them. To evaluate the effect of ranking depth, we subsample additional datasets by removing all passages that were not within the top 10, 25, and 50 passages of the first-stage retrieval. We release Rank-DistiLLM to the community to facilitate further research.

4 Evaluation

Labeled Data vs LLM Distillation Table 1 lists nDCG@10 of monoELECTRA (a cross-encoder using ELECTRA (Clark et al., 2020) as the backbone encoder) fine-tuned using the data described in Section 3.2 on the TREC DL 2019 and 2020 tasks when re-ranking the top 100 passages retrieved by BM25 and ColBERTv2. We refer to Appendix C for details on fine-tuning settings. The effectiveness of RankGPT-4, RankZephyr, and monoELECTRA fine-tuned using MS MARCO labels are provided for comparison.

Table 1: Comparison of nDCG@10 on TREC DL 2019 and 2020 of monoELECTRA fine-tuned on various LLM distillation datasets (Single-Stage) or further fine-tuned from an already fine-tuned model (Two-Stage). The highest and second-highest scores per task are bold and underlined, respectively.

	BM25		**ColBERTv2**	
Model	DL 19	DL 20	DL 19	DL 20
First Stage	0.480	0.494	0.732	0.724
RankGPT-4	0.713	<u>0.713</u>	0.766	<u>0.793</u>
RankZephyr	<u>0.719</u>	**0.720**	0.749	**0.798**
monoELECTRA	0.687	0.698	0.739	0.760
Data	*monoELECTRA – Single-Stage Fine-Tuning*			
RankGPT	0.696	0.666	0.690	0.662
TWOLAR	0.693	0.669	0.754	0.730
RankZephyr BM25	0.644	0.622	0.674	0.654
RankZephyr CBv2	0.709	0.704	**0.774**	0.754
Data	*monoELECTRA – Two-Stage Fine-Tuning*			
RankGPT	0.664	0.634	0.477	0.472
TWOLAR	0.715	0.706	0.763	0.760
RankZephyr BM25	0.672	0.638	0.714	0.683
RankZephyr CBv2	**0.720**	0.711	<u>0.768</u>	0.770

Of all monoELECTRA cross-encoders, only the one fine-tuned using our new ColBERTv2-then-RankZephyr data is more effective than monoELECTRA fine-tuned using MS MARCO labels. The first-stage retrieval model substantially impacts the distilled cross-encoder's effectiveness, shown by the poor effectiveness of the model fine-tuned on BM25-then-RankZephyr data. In conclusion, sampling hard rankings is essential for effectively distilling cross-encoders from LLM rankings.

To further increase effectiveness, (Baldelli et al., 2024) suggest a two-stage approach by first fine-tuning on noisy data and continuing to fine-tune on LLM distillation data. In our experiments, two-stage fine-tuning also boosts the effectiveness of models fine-tuned on our newly proposed dataset. Fine-tuning on MS MARCO labels and then fine-tuning using our ColBERTv2-then-RankZephyr rankings produces the most effective model. It achieves slightly higher effectiveness than RankGPT-4 and RankZephyr on TREC DL 2019 and slightly lower effectiveness on TREC DL 2020. None of the differences are statistically significant (t-test, $p < 0.05$, Bonferroni corrected).

Listwise Fine-Tuning Using ADR-MSE as the loss function produces a 0.002 (single-stage) and

Table 2: Effectiveness in nDCG@10 of various re-ranking models micro-averaged across all queries from a collection from the TIREx framework (Fröbe et al., 2023). See Appendix D for details on the per-corpus tasks. Macro-averaged arithmetic and geometric means are computed across all corpora. Model sizes are given in the number of parameters. Both monoELECTRA models are fine-tuned on our new ColBERTv2-then-RankZephyr distillation data. All monoT5 models are taken from the TIREx experiments and were fine-tuned on MS MARCO labels. The highest and second-highest scores per corpus are bold and underlined, respectively.

Model	**Parameters**	**Antique**	**Args.me**	**ClueWeb09**	**ClueWeb12**	**CORD-19**	**Cranfield**	**Disks4+5**	**GOV**	**GOV2**	**MEDLINE**	**MS MARCO**	**NFCorpus**	**Vaswani**	**WaPo**	**A. Mean**	**G. Mean**
First Stage	–	0.516	**0.404**	0.177	**0.364**	0.586	**0.012**	0.436	0.235	0.466	0.358	0.487	0.281	0.447	0.364	0.367	0.394
RankZephyr	7B	0.534	0.363	<u>0.213</u>	0.303	**0.767**	0.009	0.556	**0.294**	<u>0.560</u>	**0.457**	0.720	0.314	0.512	**0.508**	**0.437**	**0.478**
Fine-tuned on MS MARCO relevance labels																	
monoT5$_{BASE}$	220M	0.510	0.304	0.185	0.260	0.688	0.009	0.535	0.264	0.486	0.253	0.705	0.310	0.306	0.451	0.376	0.420
monoT5$_{LARGE}$	770M	0.532	0.337	0.181	0.266	0.636	0.010	<u>0.566</u>	0.265	0.512	0.313	0.717	0.311	0.414	0.492	0.397	0.438
monoT5$_{3B}$	3B	0.543	<u>0.391</u>	0.199	0.279	0.603	<u>0.011</u>	**0.569**	<u>0.289</u>	0.513	0.348	**0.736**	**0.324**	0.458	0.476	0.410	0.448
Distilled from ColBERTv2-then-RankZephyr data																	
monoELECTRA$_{BASE}$	110M	**0.593**	0.375	0.209	0.295	0.692	0.010	0.521	0.264	0.541	0.326	0.715	0.306	<u>0.522</u>	0.458	0.416	0.457
monoELECTRA$_{LARGE}$	330M	<u>0.575</u>	0.368	**0.221**	<u>0.313</u>	<u>0.716</u>	0.008	0.559	0.288	**0.572**	<u>0.376</u>	<u>0.730</u>	<u>0.316</u>	**0.526**	<u>0.504</u>	<u>0.434</u>	<u>0.475</u>

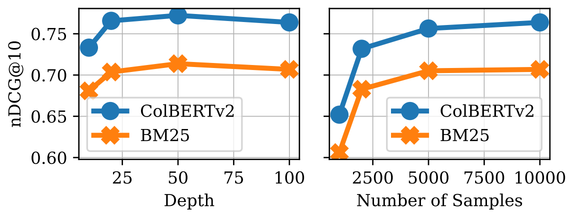

Figure 1: Effectiveness averaged across TREC Deep Learning 2019 and 2020 of monoELECTRA models fine-tuned on subsamples of the ColBERTv2-then-RankZephyr distillation dataset using different ranking depths and numbers of samples.

0.001 (two-stage) nDCG@10 less effective model compared to using RankNet averaged across TREC DL 2019 / 2020 and BM25 / ColBERTv2 for initial retrieval. Since the difference in effectiveness is marginal and monoELECTRA fine-tuned using RankNet already reaches the effectiveness of RankZephyr, we conclude that listwise loss functions are unnecessary for distillation from LLMs.

Data Ablation Since LLMs are costly, we investigate how much data is necessary to achieve the highest possible effectiveness. Figure 1 shows that effectiveness peaks at 50 samples per query and slightly decreases at 100 samples per query. When downsampling the number of training samples, we achieve the highest effectiveness using all 10k queries. Since we can reach the effectiveness of RankZephyr in two-stage fine-tuning, we assume 10k queries are sufficient. However, more data may improve effectiveness in single-stage fine-tuning.

Out-of-Domain Effectiveness Table 2 shows that monoELECTRA$_{BASE}$ is more effective than all previous cross-encoders in TIREx, improving over monoT5$_{3B}$ (Nogueira et al., 2020), the previously best cross-encoder, on average. Using a larger model further improves effectiveness. We match the current state-of-the-art RankZephyr's effectiveness using monoELECTRA$_{LARGE}$.

Efficiency Our monoELECTRA$_{LARGE}$ model uses approximately 89% and 95% fewer parameters compared to monoT5$_{3B}$ and RankZephyr, respectively. This reduces memory consumption and latency. Our model requires approximately 300 milliseconds to re-rank 100 passages for a single query. In contrast, monoT5$_{3B}$ requires approximately 3 seconds and RankZephyr about 25 seconds per query.

5 Conclusion

Using our new Rank-DistiLLM datset, we have systematically investigated several aspects of distilling cross-encoders from LLM rankings. Our findings indicate that rankings of the top-50 passages for 10,000 queries suffice to achieve competitive effectiveness compared to LLMs, but the passages need to be sampled using a very effective first-stage retrieval model. By first fine-tuning on MS MARCO labels and then further on Rank-DistiLLM, our best model is more effective than previous cross-encoders and matches the effectiveness of LLMs for in- and out-of-domain re-ranking while being orders of magnitude more efficient.

6 Limitations

In this short paper, we could only test a limited range of distillation settings. For instance, analyzing ranking depth or number of queries at finer granularity could provide further insights. Additionally, as we have only tested a single cross-encoder architecture and a single LLM, further experiments are needed to analyze whether and how our findings generalize to other architectures and LLMs.

7 Ethical Considerations

We are aware that fine-tuning and running large transformer-based language models requires considerable amounts of energy and may contribute to climate change. Especially creating our new Rank-DistiLLM dataset required substantial computational resources and energy. We have tried to minimize the environmental impact by fine-tuning models with (comparatively) few parameters. Nonetheless, we acknowledge that our research has an environmental impact.

Additionally, while our work derives from publicly available and widely used datasets and models, we are aware that these may contain biases. We release our data, code, and models to the public but caution that they may contain biases that could be harmful if used in production systems.

References

Zeynep Akkalyoncu Yilmaz, Wei Yang, Haotian Zhang, and Jimmy Lin. 2019. Cross-Domain Modeling of Sentence-Level Evidence for Document Retrieval. In *Proceedings of EMNLP-IJCNLP 2019*, pages 3490–3496.

Negar Arabzadeh, Alexandra Vtyurina, Xinyi Yan, and Charles L. A. Clarke. 2022. Shallow Pooling for Sparse Labels. *Information Retrieval Journal*, 25:365–385.

Davide Baldelli, Junfeng Jiang, Akiko Aizawa, and Paolo Torroni. 2024. TWOLAR: A TWO-Step LLM-Augmented Distillation Method for Passage Reranking. In *Proceedings of ECIR 2024*, pages 470–485.

Alexander Bondarenko, Maik Fröbe, Johannes Kiesel, Shahbaz Syed, Timon Gurcke, Meriem Beloucif, Alexander Panchenko, Chris Biemann, Benno Stein, Henning Wachsmuth, Martin Potthast, and Matthias Hagen. 2022. Overview of Touché 2022: Argument Retrieval. In *Proceedings of CLEF 2022*, pages 311–336.

Alexander Bondarenko, Lukas Gienapp, Maik Fröbe, Meriem Beloucif, Yamen Ajjour, Alexander Panchenko, Chris Biemann, Benno Stein, Henning Wachsmuth, Martin Potthast, and Matthias Hagen. 2021. Overview of Touché 2021: Argument Retrieval. In *Proceedings of CLEF 2021*, pages 450–467.

Vera Boteva, Demian Gholipour, Artem Sokolov, and Stefan Riezler. 2016. A Full-Text Learning to Rank Dataset for Medical Information Retrieval. In *Proceedings of ECIR 2016*, pages 716–722.

Sebastian Bruch, Xuanhui Wang, Michael Bendersky, and Marc Najork. 2019a. An Analysis of the Softmax Cross Entropy Loss for Learning-to-Rank with Binary Relevance. In *Proceedings of SIGIR 2019*, pages 75–78.

Sebastian Bruch, Masrour Zoghi, Mike Bendersky, and Marc Najork. 2019b. Revisiting Approximate Metric Optimization in the Age of Deep Neural Networks. In *Proceedings of SIGIR 2019*, pages 1241–1244.

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to Rank Using Gradient Descent. In *Proceedings of ICML 2005*, pages 89–96.

Stefan Büttcher, Charles L. A. Clarke, and Ian Soboroff. 2006. The TREC 2006 Terabyte Track. In *Proceedings of TREC 2006*.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. In *Proceedings of ICLR 2020*.

Charles L. A. Clarke, Nick Craswell, and Ian Soboroff. 2004. Overview of the TREC 2004 Terabyte Track. In *Proceedings of TREC 2004*.

Charles L. A. Clarke, Nick Craswell, and Ian Soboroff. 2009. Overview of the TREC 2009 Web Track. In *Proceedings of TREC 2009*.

Charles L. A. Clarke, Nick Craswell, Ian Soboroff, and Gordon V. Cormack. 2010. Overview of the TREC 2010 Web Track. In *Proceedings of TREC 2010*.

Charles L. A. Clarke, Nick Craswell, Ian Soboroff, and Ellen M. Voorhees. 2011. Overview of the TREC 2011 Web Track. In *Proceedings of TREC 2011*.

Charles L. A. Clarke, Nick Craswell, and Ellen M. Voorhees. 2012. Overview of the TREC 2012 Web Track. In *Proceedings of TREC 2012*.

Charles L. A. Clarke, Falk Scholer, and Ian Soboroff. 2005. The TREC 2005 Terabyte Track. In *Proceedings of TREC 2005*.

Cyril W. Cleverdon. 1991. The Significance of the Cranfield Tests on Index Languages. In *Proceedings of SIGIR 1991*, pages 3–12.

Kevyn Collins-Thompson, Paul N. Bennett, Fernando Diaz, Charlie Clarke, and Ellen M. Voorhees. 2013. TREC 2013 Web Track Overview. In *Proceedings of TREC 2013*.

Kevyn Collins-Thompson, Craig Macdonald, Paul N. Bennett, Fernando Diaz, and Ellen M. Voorhees. 2014. TREC 2014 Web Track Overview. In *Proceedings of TREC 2014*.

Nick Craswell and David Hawking. 2002. Overview of the TREC-2002 Web Track. In *Proceedings of TREC 2002*.

Nick Craswell and David Hawking. 2004. Overview of the TREC 2004 Web Track. In *Proceedings of TREC 2004*.

Nick Craswell, David Hawking, Ross Wilkinson, and Mingfang Wu. 2003. Overview of the TREC 2003 Web Track. In *Proceedings of TREC 2003*, pages 78–92.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2020. Overview of the TREC 2020 Deep Learning Track. In *Proceedings of TREC 2020*.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Jimmy Lin. 2021. Overview of the TREC 2021 Deep Learning Track. In *Proceedings of TREC 2021*.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, Jimmy Lin, Ellen M. Voorhees, and Ian Soboroff. 2022. Overview of the TREC 2022 Deep Learning Track. In *Proceedings of TREC 2022*.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. 2019. Overview of the TREC 2019 Deep Learning Track. In *Proceedings of TREC 2019*.

Tri Dao. 2023. FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning. arXiv:2307.08691.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. In *Proceedings of NeurIPS 2022*, pages 16344–16359.

William Falcon and The PyTorch Lightning team. 2023. PyTorch Lightning.

Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking. In *Proceedings of SIGIR 2021*, pages 2288–2292.

Maik Fröbe, Jan Heinrich Reimer, Sean MacAvaney, Niklas Deckers, Simon Reich, Janek Bevendorff, Benno Stein, Matthias Hagen, and Martin Potthast. 2023. The Information Retrieval Experiment Platform. In *Proceedings of SIGIR 2023*, pages 2826–2836.

Luyu Gao, Zhuyun Dai, and Jamie Callan. 2021. Rethink Training of BERT Rerankers in Multi-stage Retrieval Pipeline. In *Proceedings of ECIR 2021*, pages 280–286.

Ralf Gommers, Pauli Virtanen, Matt Haberland, Evgeni Burovski, Warren Weckesser, Tyler Reddy, Travis E. Oliphant, David Cournapeau, Andrew Nelson, alexbrc, Pamphile Roy, Pearu Peterson, Ilhan Polat, Josh Wilson, endolith, Nikolay Mayorov, Stefan van der Walt, Matthew Brett, Denis Laxalde, Eric Larson, Atsushi Sakai, Jarrod Millman, Lars, peterbell10, C. J. Carey, Paul van Mulbregt, eric-jones, Kai, Nicholas McKibben, and Lucas Colley. 2024. Scipy/scipy: SciPy 1.14.0rc1. Zenodo.

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. 2020. Array Programming with NumPy. *Nature*, 585(7825):357–362.

Helia Hashemi, Mohammad Aliannejadi, Hamed Zamani, and W. Bruce Croft. 2020. ANTIQUE: A Non-factoid Question Answering Benchmark. In *Proceedings of ECIR 2020*, pages 166–173.

William R. Hersh, Ravi Teja Bhupatiraju, L. Ross, Aaron M. Cohen, Dale Kraemer, and Phoebe Johnson. 2004. TREC 2004 Genomics Track Overview. In *Proceedings of TREC 2004*.

William R. Hersh, Aaron M. Cohen, Jianji Yang, Ravi Teja Bhupatiraju, Phoebe M. Roberts, and Marti A. Hearst. 2005. TREC 2005 Genomics Track Overview. In *Proceedings of TREC 2005*.

Sebastian Hofstätter, Sophia Althammer, Michael Schröder, Mete Sertkan, and Allan Hanbury. 2021. Improving Efficient Neural Ranking Models with Cross-Architecture Knowledge Distillation. arXiv:2010.02666.

John D. Hunter. 2007. Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering*, 9(03):90–95.

Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, Carol Willing, and Jupyter Development Team. 2016. Jupyter Notebooks – A Publishing Format for Reproducible Computational Workflows. In *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87–90. IOS Press.

Benjamin Lefaudeux, Francisco Massa, Diana Liskovich, Wenhan Xiong, Vittorio Caggiano, Sean Naren, Min Xu, Jieru Hu, Marta Tintore, Susan Zhang, Patrick Labatut, and Daniel Haziza. 2022. xFormers: A modular and hackable Transformer modelling library. `https://github.com/facebookresearch/xformers`.

Sheng-Chieh Lin, Akari Asai, Minghan Li, Barlas Oguz, Jimmy Lin, Yashar Mehdad, Wen-tau Yih, and Xilun Chen. 2023. How to Train Your Dragon: Diverse Augmentation Towards Generalizable Dense Retrieval. In *Findings of EMNLP 2023*, pages 6385–6400.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. In *Proceedings of ICLR 2019*.

Sean MacAvaney, Craig Macdonald, and Iadh Ounis. 2022. Streamlining Evaluation with `ir-measures`. In *Proceedings of ECIR 2022*, pages 305–310.

Sean MacAvaney, Andrew Yates, Sergey Feldman, Doug Downey, Arman Cohan, and Nazli Goharian. 2021. Simplified Data Wrangling with `ir_datasets`. In *Proceedings of SIGIR 2021*, pages 2429–2436.

Craig Macdonald, Nicola Tonellotto, Sean MacAvaney, and Iadh Ounis. 2021. PyTerrier: Declarative Experimentation in Python from BM25 to Dense Retrieval. In *Proceedings of CIKM 2021*, pages 4526–4533.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. In *Proceedings of COCO@NeurIPS 2016*.

Rodrigo Nogueira and Kyunghyun Cho. 2020. Passage Re-ranking with BERT. arXiv:1901.04085[v5].

Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document Ranking with a Pretrained Sequence-to-Sequence Model. In *Findings of EMNLP 2020*, pages 708–718.

The pandas development team. 2024. Pandas-dev/pandas: Pandas. Zenodo.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Proceedings of NeurIPS 2019*, pages 8024–8035.

Ronak Pradeep, Yuqi Liu, Xinyu Zhang, Yilin Li, Andrew Yates, and Jimmy Lin. 2022. Squeezing Water from a Stone: A Bag of Tricks for Further Improving Cross-Encoder Effectiveness for Reranking. In *Proceedings of ECIR 2022*, pages 655–670.

Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023a. RankVicuna: Zero-Shot Listwise Document Reranking with Open-Source Large Language Models. arXiv:2309.15088.

Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023b. RankZephyr: Effective and Robust Zero-Shot Listwise Reranking is a Breeze! arXiv:2312.02724.

Tao Qin, Tie-Yan Liu, and Hang Li. 2010. A general approximation framework for direct optimization of information retrieval measures. *Information Retrieval*, 13:375–397.

Kirk Roberts, Dina Demner-Fushman, Ellen M. Voorhees, William R. Hersh, Steven Bedrick, and Alexander J. Lazar. 2018. Overview of the TREC 2018 Precision Medicine Track. In *Proceedings of TREC 2018*.

Kirk Roberts, Dina Demner-Fushman, Ellen M. Voorhees, William R. Hersh, Steven Bedrick, Alexander J. Lazar, and Shubham Pant. 2017. Overview of the TREC 2017 Precision Medicine Track. In *Proceedings of TREC 2017*.

Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. 1994. Okapi at TREC-3. In *Proceedings of TREC 1994*, pages 109–126.

Guilherme Rosa, Luiz Bonifacio, Vitor Jeronymo, Hugo Abonizio, Marzieh Fadaee, Roberto Lotufo, and Rodrigo Nogueira. 2022. In Defense of Cross-Encoders for Zero-Shot Retrieval. arXiv:2212.06121.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2022. ColBERTv2: Effective and Efficient Retrieval via Lightweight Late Interaction. arXiv:2112.01488.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. 2023. Is ChatGPT Good at Search? Investigating Large Language Models as Re-Ranking Agents. In *Proceedings of EMNLP 2023*, pages 14918–14937.

Manveer Singh Tamber, Ronak Pradeep, and Jimmy Lin. 2023. Scaling Down, LiTting Up: Efficient Zero-Shot Listwise Reranking with Seq2seq Encoder-Decoder Models. arXiv:2312.16098.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C. J. Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, and Paul van Mulbregt. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17(3):261–272.

Ellen Voorhees, Tasmeer Alam, Steven Bedrick, Dina Demner-Fushman, William R. Hersh, Kyle Lo, Kirk

Roberts, Ian Soboroff, and Lucy Lu Wang. 2020. TREC-COVID: Constructing a Pandemic Information Retrieval Test Collection. arXiv:2005.04474.

Ellen M. Voorhees. 2004. Overview of the TREC 2004 Robust Track. In *Procedings of TREC 2004*.

Ellen M. Voorhees and Donna Harman. 1998. Overview of the Seventh Text REtrieval Conference (TREC-7). In *Proceedings of TREC 1998*.

Ellen M. Voorhees and Donna Harman. 1999. Overview of the Eigth Text REtrieval Conference (TREC-8). In *Proceedings of TREC 1999*.

Lucy Lu Wang, Kyle Lo, Yoganand Chandrasekhar, Russell Reas, Jiangjiang Yang, Doug Burdick, Darrin Eide, Kathryn Funk, Yannis Katsis, Rodney Kinney, Yunyao Li, Ziyang Liu, William Merrill, Paul Mooney, Dewey Murdick, Devvret Rishi, Jerry Sheehan, Zhihong Shen, Brandon Stilson, Alex Wade, Kuansan Wang, Nancy Xin Ru Wang, Chris Wilhelm, Boya Xie, Douglas Raymond, Daniel S. Weld, Oren Etzioni, and Sebastian Kohlmeier. 2020. CORD-19: The COVID-19 Open Research Dataset. arXiv:2004.10706.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. HuggingFace's Transformers: State-of-the-art Natural Language Processing. arXiv:1910.03771[v5].

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. 2021. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. In *Proceedings of ICLR 2021*.

Honglei Zhuang, Zhen Qin, Rolf Jagerman, Kai Hui, Ji Ma, Jing Lu, Jianmo Ni, Xuanhui Wang, and Michael Bendersky. 2022. RankT5: Fine-Tuning T5 for Text Ranking with Ranking Losses. arXiv:2210.10634.

A Software

This work made use of the following software packages: HuggingFace Transformers (Wolf et al., 2020), ir_datasets (MacAvaney et al., 2021), ir_measures (MacAvaney et al., 2022), Jupyter (Kluyver et al., 2016), Lightning (Falcon and The PyTorch Lightning team, 2023), matplotlib (Hunter, 2007), NumPy (Harris et al., 2020), pandas (pandas development team, 2024), PyTerrier (Macdonald et al., 2021), PyTorch (Paszke et al., 2019), and SciPy (Virtanen et al., 2020; Gommers et al., 2024),

B Approximate Ranking Function

Given a set of scores s_p for passages $p \in \mathcal{P}$, we can compute a smooth approximation of a passage's rank $\hat{\pi}(p)$ by (Qin et al., 2010):

$$\hat{\pi}(p) = 1 + \sum_{p_j \in \mathcal{R} \setminus p} \frac{\exp(-\alpha \cdot s_p)}{1 + \exp(-\alpha \cdot s_{p_j})}.$$

The parameter α controls the smoothness of the approximation. As α becomes greater, the approximation more closely resembles the actual rank. See Bruch et al. (2019b) for an in-depth analysis of the effect of alpha on Approx loss functions.

C Fine-Tuning Settings

We mostly follow Pradeep et al. (2022) for fine-tuning cross-encoders. We use HuggingFace (Wolf et al., 2020) ELECTRA$_{\text{BASE}}$ or ELECTRA$_{\text{LARGE}}$ (Clark et al., 2020) checkpoints as starting points.[1][2] For fine-tuning using MS MARCO (Nguyen et al., 2016) labels, we randomly sample 7 hard-negatives from the top 200 passages retrieved by ColBERTv2 (Santhanam et al., 2022) for every training query and fine-tune for 20k steps using LCE loss (Gao et al., 2021). For fine-tuning on LLM distillation data, we use the TREC Deep Learning 2021 and 2022 tracks (Craswell et al., 2021, 2022) as validation sets and train until nDCG@10 does not improve for 100 steps using either RankNet (Burges et al., 2005) or our novel ADR-MSE loss (using $\alpha = 1$). All models are fine-tuned using a batch size of 32 and the AdamW (Loshchilov and Hutter, 2019) optimizer with a 10^{-5} learning rate. We truncate queries longer than 32 tokens and passages longer than 256 tokens. All models are trained on a single NVIDIA A100 40GB GPU and implemented using PyTorch (Paszke et al., 2019) and Lightning (Falcon and The PyTorch Lightning team, 2023).

D TIREx Dataset

Table 3 provides an overview of the 31 retrieval tasks over 14 corpora contained in TIREx (Fröbe et al., 2023) used for evaluation. Citations for each corpus (except for Vaswani and WaPo, which do not have specific references) are provided below:

- Antique — QA Benchmark (Hashemi et al., 2020)
- Args.me — Touché (Bondarenko et al., 2021, 2022)
- ClueWeb09 — TREC Web Tracks (Clarke et al., 2009, 2010, 2011, 2012)

[1] `google/electra-base-discriminator`
[2] `google/electra-large-discriminator`

Table 3: Overview of the retrieval tasks in TIREx (Fröbe et al., 2023) used for evaluation. The number of tasks per corpus, number of queries, average judgments per query, and average document length are provided.

Corpus	**Tasks**		**Queries**		**Docs.**
	Details	#	Judg.	#	Length
Antique	QA Benchmark	1	32.9	200	49.9
Args.me	Touché 2020–2021	2	60.7	99	435.5
ClueWeb09	Web Tracks 2009–2012	4	421.8	200	1132.6
ClueWeb12	Web Tracks, Touché	4	163.8	200	5641.7
CORD-19	TREC-COVID	1	1386.4	50	3647.7
Cranfield	Fully Judged Corpus	1	8.2	225	234.8
Disks4+5	TREC-7/8, Robust04	3	1367.4	350	749.3
GOV	Web tracks 2002–2004	3	603.9	325	2700.5
GOV2	TREC TB 2004–2006	3	902.3	150	2410.3
MEDLINE	Genomics, PM	4	518.3	180	309.1
MS MARCO	DL 2019–2020	2	212.8	97	77.1
NFCorpus	Medical LTR Benchmark	1	48.7	325	364.6
Vaswani	Scientific Abstracts	1	22.4	93	51.3
WaPo	Core 2018	1	524.7	50	713.0

- ClueWeb12 — TREC Web Tracks, Touché (Collins-Thompson et al., 2013, 2014; Bondarenko et al., 2021, 2022)
- CORD-19 — TREC-COVID (Voorhees et al., 2020; Wang et al., 2020)
- Cranfield — Fully Judged Corpus (Cleverdon, 1991)
- Disks4+5 — TREC-7/8, Robust04 (Voorhees and Harman, 1998, 1999; Voorhees, 2004)
- GOV — TREC Web Tracks (Craswell and Hawking, 2002; Craswell et al., 2003; Craswell and Hawking, 2004)
- GOV2 — TREC TB (Clarke et al., 2004, 2005; Büttcher et al., 2006)
- MEDLINE — TERC Genomics, TREC Precision Medicine (Hersh et al., 2004, 2005; Roberts et al., 2017, 2018)
- MS MARCO — TREC Deep Learning (Craswell et al., 2019, 2020)
- NFCorpus — Medical LTR Benchmark (Boteva et al., 2016)
- Vaswani — Scientific Abstracts
- WaPo — Core ’18