Cross-Encoder Re-Rankers and Query Expansion with an LLM

KACPER KADZIOLKA, Leiden University, Netherlands, *Student Number: s4115945*
GIULIA RIVETTI, Leiden University, Netherlands, *Student Number: s4026543*

1 INTRODUCTION

In the final assignment, we investigate the performance and effectiveness of cross-encoder re-rankers and various techniques of query expansion using LLMs. The objective is to initially fine-tune each cross-encoder model for one hour, and subsequently evaluate and understand the potential and limitations of each model. In the following section, we compare five ensemble methods applied to all three fine-tuned models, leading to the final benchmark of the most effective ensemble method applied to all combinations of two models. Finally, we study and explore query prompting techniques for improving information retrieval tasks, utilizing both standard and PRF techniques.

2 TASK 1: CROSS-ENCODERS EVALUATION

All three models described in Table (1) were fine-tuned for one hour using the MS Marco dataset. 'MiniLM' and 'TinyBERT' converged similarly, reaching around 41,000 to 44,000 training steps out of a total 156,250. In contrast, 'distilroberta-base,' being more complex, required about four times more time to tune and completed only 10,000 steps in the first hour. Models like BERT, TinyBERT, and MiniLM, which leverage deep learning and self-attention mechanisms inherent to Transformer architectures, show significant improvement over traditional retrieval approaches. Their published implementations, such as 'DistilRoBERTa' pre-trained on the OpenWebTextCorpus dataset [7], provide robust baselines for fine-tuning on specialized datasets.

Table 1. Performance metrics of three different models after fine-tuning on the MS Marco dataset for approximately one hour only.

	NDCG@10	Recall@100	MAP@1000	Training Steps
cross-encoder/ms-marco-MiniLM-L-2-v2	68.03	49.29	43.39	40860/156250
cross-encoder/ms-marco-TinyBERT-L-2-v2	**69.90**	**50.47**	**45.55**	**43999/156250**
distilroberta-base	60.55	48.42	40.36	9999/156250

We evaluated our models using the TREC Deep Learning Track (DL'19) benchmark with 43 queries. *MiniLM* and *TinyBERT* performed similarly, but MiniLM had slightly lower scores. *TinyBERT* achieved the best scores with NDCG@10 (69.90), Recall@100 (50.47), and MAP@1000 (45.55). The 'distilroberta-base' model lagged behind with significantly lower scores, completing only 9,999 training steps in the same time frame, suggesting a longer convergence time that results in noticeably suboptimal performance.

Designed as efficient yet powerful alternatives to larger models, *MiniLM* and *TinyBERT* offer a comprehensive understanding of data within a shorter training period. This advantage resulted in higher scores compared to the more complex *distilroberta* model, which requires more training for optimal performance. However, while *MiniLM* and *TinyBERT* performed better on 'TREC DL'19' queries, they may not generalize equally well to all unseen queries due to variations in unseen data.

We recognize the limitations of the presented models. The *distilroberta* model's longer training time makes it less suitable for rapid fine-tuning, which in our case was limited to one hour. The smaller models, though

Authors' addresses: Kacper Kadziolka, Leiden University, Leiden, Netherlands , *Student Number: s4115945*; Giulia Rivetti, Leiden University, Leiden, Netherlands , *Student Number: s4026543*.

quicker to train, are more prone to overfitting. Despite promising results, the models' ability to generalize remains uncertain, emphasizing the need for comprehensive evaluation on diverse datasets.

3 TASK 2: ENSEMBLE METHODS

For this task, we have chosen 5 ensemble methods and applied them to the three models considered in the Task (2). To experiment with different kinds of ensemble strategies, we have selected two score-based methods (*CombSUM* and *CombMED*), two voting-based methods (*BordaFuse* and *Condorcet*) and one rank-based method (*ISR*). Table 2 shows the effectiveness of these 5 ensemble methods over three metrics: NDCG@10, Recall@100 and MAP@1000. By looking at the table, it can be noticed that *Condorcet* outperforms the other methods for

Table 2. Performance metrics of five different ensemble methods applied to the three models considered in Task 1.

	NDCG@10	Recall@100	MAP@1000
CombSUM	0.6840	0.5135	0.4542
BordaFuse	0.6966	0.5138	0.4567
ISR	0.6757	0.5057	0.4469
CombMED	0.6963	0.5046	0.4571
Condorcet	**0.6978**	**0.5144**	**0.4618**

all the three metrics, whereas *ISR* obtained the lowest score for both NDCG@10 and MAP@1000; the lowest value for the Recall@100 was obtained when employing the *CombMED* method. Even though we identified these differences in the performance of the ensemble methods, we want to remark the fact that these differences are not significant.

The superiority of *Condorcet* over the other methods, can be traced back to its voting mechanism: this method uses pairwise comparisons of ranking positions to determine the overall ranking [1], making it possible to capture and aggregate the preferences of multiple search engines more accurately, since in this way the relative performance of each search engine can be easily captured. Moreover, even if *BordaFuse* is a voting-based method too, it is not a majoritarian algorithm, as opposed to *Condorcet* [6], and its lower performance, as also testified in [6] may be due to its reliance solely on ranking positions, which may fail to fully reflect the comparative strengths of models.

In our opinion, ensemble methods have several advantages they make them useful: these methods combine the strengths of individual models by leveraging their diverse perspectives on the data. By combining the models' predictions, ensemble methods can improve overall accuracy by capturing a more comprehensive understanding of the data. Moreover, combining diverse models reduces the risk of biases or inaccuracies from individual models, improving both reliability and performance of the model [2].

We believe that a meta-learning approach can be utilized, where the agent would use the outputs of multiple ensemble methods as its meta-features. It could combine their outputs, which depends on various based-model predictions, in the most optimal manner. The meta-learner would optimally combine the scores and ranks from each ensemble method, producing weights that will later maximize the evaluation metrics.

4 TASK 3: MOST EFFECTIVE ENSEMBLE METHOD

In the previous chapter (3), we discussed the performance of various ensemble methods applied to the predictions of the three base models. Table (2) aids us in selecting the most optimal one, the *Condorcet* method. In this task, we will apply this specific ensemble method to all combinations of two from the initial three ranking run files.

Studying Table (3) clearly shows that the combination of *MiniLM* and *TinyBERT* outperforms the other candidates. This aligns with Table (1), where both models were the most optimal, making their combination logically the best. In two out of three metrics, the combination of *TinyBERT* and *distilroBERTa* is slightly worse than

Table 3. Performance metrics of *Condorcet* ensemble method applied to all combinations of two from the three ranking files.

	NDCG@10	Recall@100	MAP@1000
(*MiniLM*, *TinyBERT*)	**0.6782**	**0.4817**	**0.4251**
(*MiniLM*, *distilroBERTa*)	0.6418	0.4535	0.3841
(*TinyBERT*, *distilroBERTa*)	0.6540	0.4238	0.3818

the other combination with the second model. This might be due to the combination of two BERT architectures, which could lack diversity and lead to similar ranking outputs.

The performance of the Condorcet ensemble method varies slightly across all metrics. However, we do not find these deviations substantial. These differences are further minimized when comparing both combinations containing the *distilroBERTa* model, indicating that the *MiniLM* and *TinyBERT* architectures have similar characteristics. Because of this similarity, we would have expected the combination of *MiniLM* and *TinyBERT* to perform worse, since Condorcet often benefits from models which are diverse with respect to one another: indeed, more different models are likely to capture different aspects of the data or learn different patterns, possibly leading to a more robust ensemble method. However, in our case, the combination of two "similar" models, lead to achieving higher performance.

5 TASK 4

In this task, we executed query expansion by leveraging the generative abilities of the *Zephyr-7b-beta* LLM. Specifically, we used the Chain-of-Thought prompt (CoT) to expand the queries, and then fed the output of the LLM to the Tiny-BERT re-encoder. Table (4) presents the results obtained by evaluating the model across three different metrics (NDCG@10, Recall@100, and MAP@1000) for two cases: with the original queries and with the expanded queries.

Table 4. Performance metrics obtained with and without query expansion using the *Zephyr-7b-beta* LLM.

	NDCG@10	Recall@100	MAP@1000
TinyBERT without query expansion	69.90	50.47	45.55
TinyBERT with query expansion	46.89	40.73	32.94
MiniLM without query expansion	68.03	49.29	43.39
MiniLM with query expansion	36.94	34.51	26.27
distilroBERTa without query expansion	60.55	48.42	40.36
distilroBERTa with query expansion	14.98	17.31	14.09

Our findings indicate that query expansion using the *Zephyr-7b-beta* LLM resulted in decreased performance across all models and metrics compared to the original queries. This is indeed a contrary conclusion to the one reported in the referenced paper [4], where the authors claim that query expansion via LLMs, particularly with Chain-of-Though (CoT) prompts, leads to improved retrieval performance.

The authors of the paper [4] are conscious of limitations in their work, noting that their experiments were conducted only on a sparse retrieval model (BM25). They assume that dense retrieval systems (dual-encoders) are less prone to the vocabulary gap and, therefore, less likely to benefit from query expansion. This is indeed the case in our work, where we explore such dense encoders, which resulted in suboptimal performance compared to the original queries.

6 TASK 5

For the last task, we have performed query expansion with the same LLM used before, the *Zephyr-7b-beta* LLM. This time, however, we have employed the PRF (Pseudo Relevance Feedback) technique. In particular, we have considered the top-3 documents retrieved from the original query, under the assumption that they were all relevant, and then we have performed relevance feedback. The results of this procedure are shown in table 5, where we have reported the scores for three different metrics (NDCG@10, Recall@100 and MAP@1000) with three different cross-encoders, taking into account two cases: one without query expansion, and the other with PRF query expansion. By looking at the table we can see that also in this case the results after query expansion are

Table 5. Performance metrics obtained with and without PRF with *Zephyr-7b-beta* LLM. For both cases three cross-encoders were considered: Mini-LM, Tiny-BERT and distilroBERTa.

	NDCG@10	Recall@100	MAP@1000
Mini-LM without PRF	68.03	49.29	43.39
Mini-LM with PRF	46.30	35.49	28.20
Tiny-BERT without PRF	69.90	50.47	45.55
Tiny-BERT with PRF	57.28	39.45	33.67
distilroBERTa without PRF	60.55	48.42	40.36
distilroBERTa with PRF	8.31	11.86	8.84

lower with respect to the original case. At first, we didn't expect this outcome, since we read that this technique can improve the performance of retrieval systems[3]. However, as it also happened for the previous task, this kind of query expansion is not always effective, especially for dense retrieval tasks. In general, it may be beneficial to use PRF for query expansion, but it also has some downsides that can limit its performance, such as the introduction of irrelevant or redundant terms to the query, and the risk of amplifying the bias or error of the initial ranking algorithm [5]; this aspects may indeed contribute to lower the performance of the retrieval.

7 CONCLUSION

With this assignment, we have gained a deeper knowledge on cross-encoder re-rankers, firstly by evaluating and fine-tuning their performance; after that, we have also experimented with some ensemble methods to combine the ranking run files of the different cross-encoder re-rankers. Finally, we have performed some experiments on query expansion, to analyze the effect of this technique on the performance of the retrieval procedure.

REFERENCES

[1] Elias Bassani and Luca Romelli. 2022. ranx.fuse: A Python Library for Metasearch. *Proceedings of the 31st ACM International Conference on Information and Knowledge Management* (2022), 4808–4812 pages.

[2] Encord. [n. d.]. What is Ensemble Learning? https://encord.com/blog/what-is-ensemble-learning/#:~:text=By%20leveraging%20the%20strengths%20of,applications%20like%20healthcare%20or%20finance.

[3] The Stanford Natural Language Processing Group. [n. d.]. Pseudo relevance feedback. https://nlp.stanford.edu/IR-book/html/htmledition/pseudo-relevance-feedback-1.html

[4] Rolf Jagerman, Honglei Zhuang, Zhen Qin, Xuanhui Wang, and Michael Bendersky. 2023. Query Expansion by Prompting Large Language Models. arXiv:2305.03653 [cs.IR]

[5] Linkedin. [n. d.]. What are the benefits and drawbacks of using pseudo relevance feedback in web search? https://www.linkedin.com/advice/0/what-benefits-drawbacks-using-pseudo-relevance

[6] Mark Montague and Javed A. Aslam. 2001. Condorcet Fusion for Improved Retrieval. *Proceedings of the 2001 ACM CIKM International Conference on Information and Knowledge Management* (2001), 427–433 pages.

[7] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *ArXiv* abs/1910.01108 (2019), 5 pages.