Final assignment: Cross-encoder re-rankers

SIWEN TU and SHUPEI LI, Group 15, LIACS, Leiden University, the Netherlands

1 INTRODUCTION

1.1 Cross-Encoder

The two-stage retrieval is a common strategy when it comes to retrieving the relevant documents given a query. The first stage is retrieving from the whole document corpus using lexical matching methods such as the BM25 model. The second stage is re-ranking the retrieved documents with a neural model. Cross-encoder is a popular style of model in the second stage of retrieval, which means organizing task inputs (query and candidate texts) into an input template to feed into a transformer for inference, in contrast with the "bi-encoder" design where the queries and candidate texts are encoded independently [3]. Figure 1 illustrates the structure of "cross-encoder". In this task, we explore how to fine-tune and evaluate the cross-encoder re-rankers and how to ensemble different cross-encoder re-rankers using ensemble methods based on the MS Macro dataset.

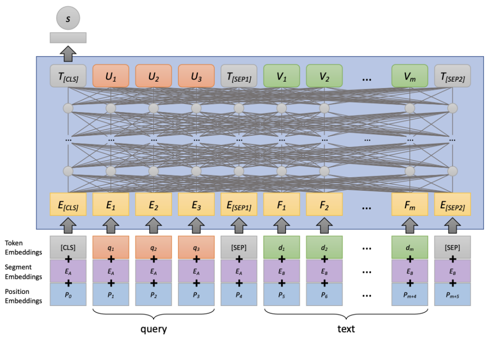

Fig. 1. Structure of monoBERT ranking model, a two-input cross-encoder re-ranker [3].

1.2 MS Macro Dataset

We conduct the experiments on the MS Macro dataset. To be more detailed, we fine-tune the cross-encoder models on the MS Macro dataset and evaluate the performance on the TREC 19 DL dataset which is based on the MS Macro dataset. MS Macro is a collection of datasets focused on deep learning in search provided by Microsoft. In this task, we use the passage retrieval dataset containing 8 million passages extracted from web documents retrieved by Bing [4]. In the fine-tuning phase, the training sample contains 5×10^6 query-passage pairs while the development sample contains 200 queries. However, the MS Macro dataset is sparse where usually only one passage is marked as relevant given a query. So in the evaluation phase, we change to the TREC 19 DL dataset, which is a more dense annotation dataset and suitable for evaluation. In this task, we use 200 queries for evaluation and only consider the queries which have at least one relevant passage.

IR'23, Master CS, Spring 2023, Leiden, the Netherlands

2 TASK 1: EVALUATING CROSS-ENCODERS

Table 1 shows the results obtained from the fine-tuning and evaluation for three specified cross-encoder models. As we can see from Table 1, the training steps of the three models differ significantly. The overall performance of MiniLM and TinyBERT surpasses that of distilroberta, which can be attributed to the significant difference in training steps and the fact that the former two models were pre-trained on the MS Macro dataset, unlike the latter. The performance of the MiniLM and TinyBERT is comparable despite the difference in training steps. The reason might be that the difference in architecture compensates for the difference in training steps and the evaluation dataset is suitable for both models. On the other hand, the values of Recall@100 for the three models are relatively close, in contrast to the other two metrics. This could be attributed to the fact that Recall@100 measures the proportion of relevant passages retrieved among all passages, regardless of the position of the relevant passages. Given that in the MS Macro dataset, the proportion of relevant passages among all passages is small for each query, it is reasonable to expect these three models to perform similarly.

Table 1. Effectiveness results of three fine-tuned cross-encoder models

Model	nDCG@10	Recall@100	MAP@1000	#Training Steps
MiniLM	0.674	0.495	0.433	44653 / 156250
TinyBERT	**0.692**	**0.508**	**0.456**	73999 / 156250
distilroberta	0.616	0.497	0.424	8999 / 156250

A higher-performing model is not necessarily more effective for all new, unseen queries. In this task, we train the models on MS Macro dataset and evaluate them on TREC 19 DL dataset. The queries and passages in these two datasets have many common characteristics and structures so the high-performing model tends to perform well during evaluation. However, for new and unseen queries that have different characteristics and distributions from the training dataset, the higher-performing model may not outperform the other models.

As for the limitations of these three models, we can conclude from the results that the training time of these models differs significantly. The training speed of the distilroberta model is much lower than the other two. So it is essential to take the training time and computational power into account. Besides, these three models are smaller versions of their respective base models. They sacrifice better performance in exchange for effectiveness so they are more suitable for simpler tasks with smaller datasets.

3 TASK 2: SELECT AND APPLY FIVE ENSEMBLE METHODS

For each ensemble method listed in Table 2, we search for the optimal hyperparameter setting before fusion and adopt the min-max normalization strategy. We can conclude from Table 2 that the Posfuse model performs best among the five models with an outstanding MAP@1000. The values of nDCG@10 and Recall@100 of the five models are similar to each other. The Posfuse model is a probability-based method that relies on the probability of a document appearing at a specific position. It is built upon a perfect probability model constructed using the same queries and results for both training and fusion [2]. During the optimization phase of this fusion task, we utilize queries and results from the MS Macro dataset, which is an integral part of the training dataset. This alignment could be the reason why Posfuse outperforms other methods.

In general, the five ensemble models outperform the best single model, TinyBERT, particularly in terms of nDCG@10 and MAP@1000. This outcome confirms that these ensemble methods

Table 2. Effectiveness results of five ensemble methods

Model	nDCG@10	Recall@100	MAP@1000
Mixed	0.707	0.513	0.465
Reciprocal Rank Fusion (RRF)	0.699	0.507	0.462
BayesFuse	0.697	0.507	0.451
PosFuse	**0.710**	**0.516**	**0.480**
Weighted BordaFuse	0.707	0.511	0.464

effectively enhance the retrieval performance of models through their combination. Every single model has its strengths and weaknesses and different models are good at capturing different aspects of the data. The ensemble methods could leverage the strength of each individual model to capture more features of the data. Additionally, given the fact that individual models may have high biases, ensemble methods could reduce the biases of the outcome by combining multiple models.

We can utilize the logistic regression method for model ensembling. To elaborate further, we can construct a logistic regression model where the inputs consist of the results from different models, and the output represents the final ranking. Through the training process, we can identify the optimal weights that should be assigned to each model based on their strengths and weaknesses.

Furthermore, during the optimization of fuse methods, a grid search is performed over the search space of hyperparameters. However, it is possible to explore alternative fine-tuning methods, such as genetic algorithms, to discover better hyperparameters.

4 TASK 3: ANALYZING THE MOST EFFECTIVE ENSEMBLE METHOD

The PosFuse ensemble method in Task 2 outperforms all other methods. Therefore, we apply PosFuse to all combinations of two models out of three in this task and summarize experimental results in Table 3.

Table 3. Effectiveness results of all combinations

Combination	nDCG@10	Recall@100	MAP@1000
MiniLM + TinyBERT	**0.723**	**0.518**	**0.483**
MiniLM + distilroberta	0.691	0.509	0.463
TinyBERT + distilroberta	0.694	0.507	0.468

MiniLM + TinyBERT ensemble performs best on all metrics. MiniLM + distilroberta performs worst on nDCG@10 and MAP@1000, while TinyBERT + distilroberta performs worst on Recall@100. The result is consistent with our expectations. Because MiniLM and TinyBERT achieve significantly better results than distilroberta in Task 1, we think the performance of the ensemble including distilroberta will be affected by the inferiority of distilroberta. There is no big difference between the performance of MiniLM + distilroberta and that of TinyBERT + distilroberta. However, MiniLM + TinyBERT performs much better than other combinations, which may be caused by removing the distilroberta ranking file. It is worth mentioning that MiniLM + TinyBERT even achieves higher scores than the ensemble of three models in Task 2. This pattern indicates that more models do not mean better performance. A weak model may introduce more noise than information, which impairs the model's performance.

5 TASK 4: MODIFYING THE EVALUATION METRIC IN FINE-TUNING

We modify the CERerankingEvaluator class to change the evaluation metric to nDCG@10. We adopt the implementation in the sklearn package to calculate nDCG@10. Note that the modified class can be easily used by changing the corresponding API in the fine-tuning notebook. No other modification is needed. Our code has been included in the submission.

6 TASK 5: ENSEMBLING THROUGH SCORE INJECTION

In this task, we investigate the effect of the score injection. The score injection is a strategy described in paper [1] that aims at improving the performance of BERT-based re-rankers by injecting BM25 scores. However, we inject the average scores of MiniLM, TinyBERT, and distilroberta, instead of BM25 scores. Besides, we use the microsoft/MiniLM-L12-H384-uncased model as the re-ranker. Our modified version of the fine-tuning notebook first performs inference with three models and sets the average score as the first sentence of the document. Then the pairs of queries and injected documents are inputted in the re-ranker. Note that we convert the type of the average score into the integer following the suggestion in [1]. Due to the GPU quota of Colab, we fine-tune the microsoft/MiniLM-L12-H384-uncased model with and without injection for one hour and run the evaluation notebook. The evaluation results are reported in Table 4. We also include results in [1] for comparison.

Table 4. Effectiveness results of ensembling with and without score injection

Model	nDCG@10	Recall@100	MAP@1000
MiniLM-L12-H384-uncased without injection	0.668	**0.503**	0.450
MiniLM-L12-H384-uncased with injection	0.664	0.495	0.435
$\text{MiniLM}_{\text{CAT}}$[1]	0.704	-	0.452
$\text{MiniLM}_{\text{BM25CAT}}$[1]	**0.711**	-	**0.463**

MiniLM with BM25 injection has the best performance during evaluation. The performance of MiniLM with the injection of three BERT-based models' predictions does not achieve the anticipated effect. We think one reason is that the training process is not sufficient. Table 4 indicates that our version of MiniLM without injection is not at the expected level as that of $\text{MiniLM}_{\text{CAT}}$. Another reason may be the instability of the model performance. We notice that MiniLM with the average score injection is not very robust during the training. The improvement of the model's robustness is a possible future direction for developing score injection strategies.

REFERENCES

[1] Arian Askari, Amin Abolghasemi, Gabriella Pasi, Wessel Kraaij, and Suzan Verberne. 2023. Injecting the BM25 Score as Text Improves BERT-Based Re-rankers. (2023). arXiv:2301.09728 [cs.IR]

[2] David Lillis, Lusheng Zhang, Fergus Toolan, Rem W. Collier, David Leonard, and John Dunnion. 2010. Estimating probabilities for effective data fusion. In *Proceeding of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2010, Geneva, Switzerland, July 19-23, 2010.* ACM, 347–354.

[3] Jimmy Lin, Rodrigo Nogueira, and Andrew Yates. 2020. Pretrained Transformers for Text Ranking: BERT and Beyond. *arXiv preprint arXiv:2010.06467* (2020). https://arxiv.org/abs/2010.06467

[4] Tri Nguyen, Matthew Rosenberg, Xiaodong Song, Jianfeng Gao, Saurabh Tiwari, Rangan Majumder, and Li Deng. 2016. MS MARCO: A Human Generated MAchine Reading Comprehension Dataset. *CoRR* abs/1611.09268 (2016). http://arxiv.org/abs/1611.09268