LEARNING-TO-RANK WITH BERT IN TF-RANKING

A PREPRINT

Shuguang Han, Xuanhui Wang, Michael Bendersky and Marc Najork

TF-Ranking Team, Google Research, Mountain View, CA

{hanshuguang,xuanhui,bemike,najork}@google.com

June 8, 2020

ABSTRACT

This paper describes a machine learning algorithm for document (re)ranking, in which queries and documents are firstly encoded using BERT [1], and on top of that a learning-to-rank (LTR) model constructed with TF-Ranking (TFR) [2] is applied to further optimize the ranking performance. This approach is proved to be effective in a public MS MARCO benchmark [3]. Our first two submissions achieve the best performance for the *passage re-ranking* task [4], and the second best performance for the *passage full-ranking* task as of April 10, 2020 [5]. To leverage the lately development of pre-trained language models, we recently integrate RoBERTa [6] and ELECTRA [7]. Our latest submissions improve our previously state-of-the-art re-ranking performance by 4.3% [8], and achieve the third best performance for the full-ranking task [9] as of June 8, 2020. Both of them demonstrate the effectiveness of combining ranking losses with BERT representations for document ranking.

1 Introduction

Recently, neural network models built on top of pretrained language models such as BERT [1] have achieved state-of-the-art performance on various machine learning tasks including question answering [10], key-phrase extraction [11], as well as document and passage ranking [12, 13]. In this paper, we are focusing on passage ranking, and particularly the MS MARCO passage full ranking and re-ranking tasks [3].

A common way to incorporate BERT for ranking tasks is to construct a finetuning classification model with the goal of determining whether or not a document is relevant to a query [13]. The resulting predictions are then used for ranking documents. We argue that such an approach is less suited for a ranking task, compared to a pairwise or listwise learning-to-rank (LTR) algorithm, which learns to distinguish relevance for document pairs or to optimize the document list as a whole, respectively [14].

To this end, we propose **TFR-BERT**, a generic document ranking framework that builds a LTR model through finetuning BERT representations of query-document pairs within TF-Ranking[1]. We apply this approach on the MS MACRO benchmark, and our submissions achieve the best leaderboard performance for the passage re-ranking task [8], and the third best performance for the passage full ranking task [9]. This demonstrates the effectiveness of combining ranking losses with BERT representations for passage ranking.

2 TFR-BERT

Our TFR-BERT model can be illustrated by Figure 1. Documents (passages) for a given query will be firstly flattened to query-document (query-passage) pairs, and then passed through BERT layers[2]. Specifically, query and each document (passage) are treated as two sentences, and are further concatenated to the following format:

[CLS] query text [SEP] passage text [SEP]

[1] TF-Ranking official page: https://github.com/tensorflow/ranking

[2] We use BERT checkpoints downloaded from the official BERT page: https://github.com/google-research/bert.

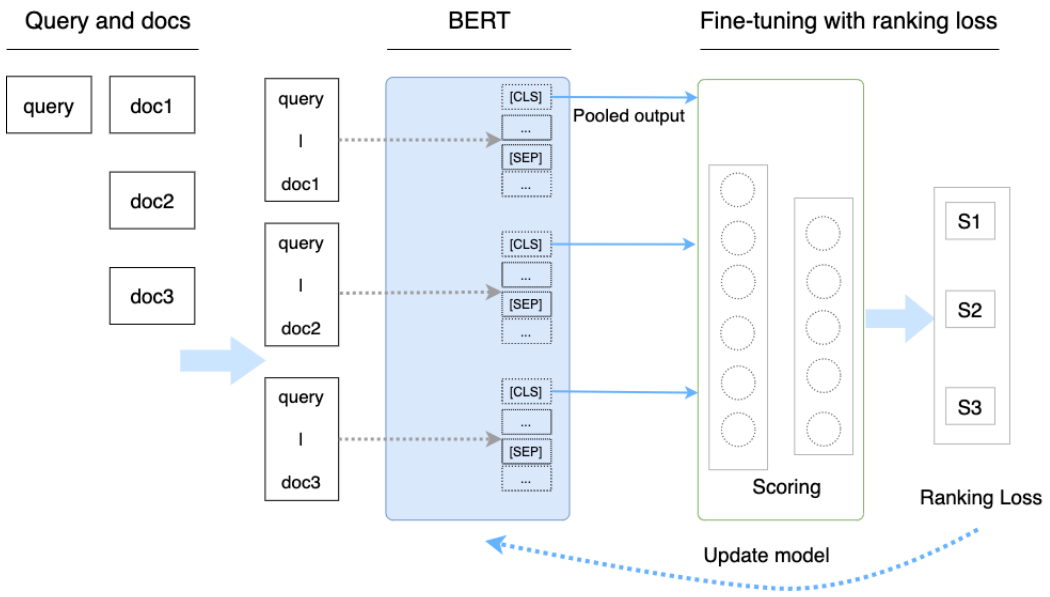

Figure 1: An illustration of the TFR-BERT framework, in which a Learning-to-Rank model is constructed on top of the BERT representations of query-document pairs.

Here, [CLS] indicates the start of a sequence and [SEP] denotes a separator between the first and second sentences. We also truncate the passage text if the whole sequence exceeds the maximum length of 512 tokens.

After that, the pooled BERT outputs (i.e., the hidden units of the [CLS] token) are fed into a ranking model built from TF-Ranking [2]. TF-Ranking provides a variety of pointwise, pairwise and listwise losses, which enable us to compare different LTR approaches in our TFR-BERT model.

3 MS MARCO Experiment

To understand the performance of TFR-BERT, we conduct a set of experiments using the publicly available MS MARCO dataset. The dataset contains 1 million real Bing queries (each query is a question), and 8.8 million candidate documents (each document is a passage). For each query, it also provides zero or more respective relevant passages marked by human annotators. In this work, we study both the passage *full ranking* and *re-ranking* tasks.

Passage Re-ranking Task. For each query, we are given the top 1000 candidate passages retrieved by BM25. The goal is to re-rank passages by their relevance to the query, i.e. the likelihood to be an answering passage for the question.

Passage Full Ranking Task. While the re-ranking performance is bounded by the recall of top 1000 passages from BM25, in this full ranking task, we are asked to rank relevant documents for each query from the whole collection of 8.8 million passages.

Ranking Dataset. To create the training set, we employ the data from *triples.train.full.tsv*. In this file, each data record is a triple containing the content of a query, a relevant passage and an irrelevant passage (query and the relevant passage can repeat multiple times in the dataset). For each query, there are roughly 1000 passages, and (in most cases) only one of them is relevant[3].

To better support the pairwise and listwise ranking models, we further group triples by query. Therefore, we obtain a list of up to 1000 passages for each query. With regards to the computer memory limit, we further break this passage list into roughly 90 lists, each taking one relevant passage and 11 irrelevant passages; thereby, creating a set of passage lists with size up to 12. Note that the above process is only used when building the training data. We leave the *dev* and *eval* datasets intact – 1000 passages per each query are present for these datasets.

Training. Our models are trained on TPU V3. We set the list size to be 12, as described above. The batch size is set to 32. As a result, a number of 32 * 12 = 384 query-document pairs are used in each training step. We checkpoint each

[3]More details about this dataset can be found in https://github.com/nyu-dl/dl4marco-bert.

model at the 50K steps. Our ensemble approach, which will be introduced in Section 4.2, aggregates over multiple models, each following the above training process.

4 Experimental Results

In this section, we report the results obtained by TFR-BERT, in which we take into account all of the pointwise, pairwise and listwise ranking approaches. The ranking models are constructed using the open-source TF-Ranking code. For more details about their implementation, the readers may refer to Pasumarthi et al. [2] and Bruch et al. [15].

4.1 Our Submissions

We made five submissions to the MS MARCO leaderboard (https://microsoft.github.io/msmarco/), as listed below. Submission #1, #2 and #4 focused on the passage re-ranking task (Section 4.2), whereas the other two submissions addressed the passage full ranking task (Section 4.3).

For pre-trained language models, we used the *BERT-Large, Uncased* checkpoint [1] for submissions #1 to #3. Later on, we switched to the *BERT-Large, Uncased (Whole Word Masking)* checkpoint for submissions #4 and #5 because of its better performance. For RoBERTa, we adopted the *roberta.large* checkpoint. And for ELECTRA, we utilized the *ELECTRA-Large* checkpoint.

More specifically, Submission #1 was a single run of TFR-BERT with softmax loss; Submission #2 was an ensemble of pointwise, pairwise and listwise TFR-BERT models; Submission #3 adopted the same ensemble technique as Submission #2, but re-ranked top 1000 passages from both BM25 and DeepCT [16], and further combined the two ranking lists; Submission #4 only adopted the listwise loss in TF-Ranking but used ensemble over BERT, RoBERTa and ELECTRA; Submission #5 applied the same ensemble technique as Submission #4, but combined both DeepCT [16] and BM25 results for re-ranking.

- **Submission #1** (re-ranking): TF-Ranking + BERT (Softmax Loss, List size 6, 200k steps) [17].
- **Submission #2** (re-ranking): TF-Ranking + BERT (Ensemble of pointwise, pairwise and listwise losses) [4].
- **Submission #3** (full ranking): DeepCT Retrieval + TF-Ranking BERT Ensemble [5].
- **Submission #4** (re-ranking): TF-Ranking Ensemble of BERT, RoBERTa and ELECTRA [8].
- **Submission #5** (full ranking): DeepCT + TF-Ranking Ensemble of BERT, RoBERTa and ELECTRA [9].

4.2 Re-ranking Experiments

Experimental results for re-ranking tasks are provided in Table 1. In addition to the official BM25 and Duet V2 baselines, we also include a baseline from Nogueira and Cho [13].

TFR-BERT Single Run. We experimented with three types of TFR-BERT models – pointwise model with sigmoid cross-entropy loss, pairwise model with pairwise logistic loss and listwise with softmax loss. We run each model 5 times, and the reported numbers are the average of 5 runs. For **Submission #1** [17], we choose the softmax loss run with the best MRR@10 performance on the Dev data set over the 5 runs.

According to Table 1, TFR-BERT models outperform the official baselines by a large margin. More importantly, they further improve upon the existing state-of-the-art approach [13] that uses the same training data and BERT checkpoint. This demonstrates the effectiveness of combining ranking losses with BERT representations for passage ranking.

The **Submission #1** achieved the second best performance for the passage re-ranking task at the time of its submission on March 19, 2020. Compared with the best method at that time [19], which used auxiliary information to enrich BERT, and introduced additional index information for ranking[4], TFR-BERT only adopted the original BERT checkpoint, and can be reproduced easily in TF-Ranking.

Ensemble of Multiple Losses. After a manual examination of model predictions, we discovered that, despite similar MRR performance, different TFR-BERT runs (even with the same type of loss) show non-trivial difference in predictions. Therefore, we further include an approach to **ensemble** models trained from different runs. It worked as follows:

1: Supposes we have n runs (models) to ensemble: R_1, R_2, ..., R_n.

[4]However, the author did not disclose further details about his approach.

Table 1: MRR@10 performance for passage **re-ranking**. Note that 1) only the models submitted to the leaderboard have MRR@10 for the Eval dataset, 2) for multiple BERT ensemble, we switched the checkpoint from *BERT-Large, Uncased* to *BERT-Large, Uncased (Whole Word Masking)*, which slightly improved MRR@10 from 0.3856 to 0.3898.

	Model	Dev (MRR@10)	Eval (MRR@10)
Baselines	BM25	0.1670	0.1649
	Duet V2 ([18])	0.2517	0.2527
	BERT + Small training ([13])	0.3653	0.3587
	Previous Leaderboard Best [19]	0.3730	0.3676
TFR-BERT Single Run	Sigmoid cross entropy loss (pointwise)	0.3716	-
	Pairwise logistic loss (pairwise)	0.3718	-
	Softmax loss (listwise)	0.3725	-
	Submission #1 [17]	*0.3782*	*0.3660*
Multiple Losses (Ensemble)	Sigmoid cross entropy loss (5 runs)	0.3839	-
	Pairwise logistic loss (5 runs)	0.3849	-
	Softmax loss (5 runs)	0.3856	-
	Submission #2 [4]	*0.3877*	*0.3747*
Multiple BERTs (Ensemble)	BERT (5 runs, listwise loss*)	0.3898	-
	RoBERTa (5 runs, listwise loss)	0.3958	-
	ELECTRA (5 runs, listwise loss)	0.3976	-
	Submission #4 [8]	**0.4046**	**0.3905**

2: For each run R_k and a query q_i, we rank the corresponding documents based on prediction scores, and then obtain the rank position $P_{k,i,j}$ for each document d_j.

3: For each query q_i, we re-compute a new score $s_{i,j}$ for document d_j based on the average reciprocal rank ($\frac{1}{n}\sum_k \frac{1}{P_{k,i,j}}$) of n runs.

4: Finally, we rank documents based on the new score $s_{i,j}$.

We firstly experimented with the ensemble of 5 different runs using the same loss function. According to Table 1, the ensemble approach improves the performance of a single run by 3.5% for all three loss functions. Through a further ensemble over all the three loss functions (total of 15 runs), we achieve the best overall MRR on the Dev data set. The 15-run ensemble is chosen as the **Submission #2** [4], which outperforms the previously best submission [19] by 4.0% on the development dataset, and 1.9% on the evaluation dataset.

Ensemble of Multiple BERTs. To incorporate the recent advancement of pre-trained BERT models, we further integrated RoBERTa [6] and ELECTRA [7] into TF-Ranking. The ensemble process for each BERT model worked the same as the above. From Table 1, we observed that ensemble with RoBERTa slightly outperformed BERT, and ensemble with ELECTRA slightly outperformed RoBERTa. Through a further ensemble over all of the three models (total of 15 runs, **Submission #4**), we achieve the best MRR@10 for the re-ranking task [8], outperforming the previously best performance (also from us [4]) by 4.4% on the dev dataset and by 4.3% on the evaluation dataset.

4.3 Full Ranking Experiments

In addition to the re-ranking task, we made another submission to the full ranking task, in which we re-ranked the top 1000 passages from both BM25 and DeepCT [16] using the TFR-BERT ensemble model, and further combined the two resulting ranking lists. It worked as follows.

1: Re-rank the top 1000 passages retrieved by BM25 using the TFR-BERT ensemble model.

2: Re-rank the top 1000 passages retrieved by DeepCT [16] using the TFR-BERT ensemble model.

3: Combine the re-ranking scores (we use reciprocal rank to be consistent with the ensemble model) from the above two lists. For passages occurring in both lists, we take the average; otherwise, we keep its original score.

4: Finally, we re-rank passages based on the new score.

The full-ranking results are reported in Table 2. Same as the re-ranking results, we include the official BM25 and Duet V2 baselines for reference. In addition, we introduce a baseline (W-index + BERT F-rerank) from Dai et al. [16], as it is the original entry that proposed the DeepCT retrieval approach.

According to Table 2, we discovered that DeepCT helps boost the re-ranking of BM25 results by a large margin, and a further combination of both BM25 and DeepCT re-ranked lists brings additional gains. With Submission #3, we achieved the second best overall performance on the leaderboard as of April 10, 2020. With the recent Submission #5, we further improved our previous performance, and obtained the third best performance on the leaderboard as of June 8, 2020 (with tens of new leaderboard submissions in between).

The above results, again, demonstrate the effectiveness and robustness of the TFR-BERT ensemble model – it works well on both re-ranking and full ranking tasks, and more importantly, it does not require auxiliary information other than the original BERT checkpoints, and can be easily reproduced with TF-Ranking.

Table 2: MRR@10 performance for the passage **full ranking** task. Note that only the models submitted to the leaderboard have MRR@10 for the Eval dataset.

	Model	Dev (MRR@10)	Eval (MRR@10)
Baselines	BM25	0.1670	0.1649
	Duet V2 ([18])	0.2517	0.2527
	W-index + BERT-F rerank ([16])	0.3935	0.3877
	Leaderboard Best [20] (as of April 10, 2020)	0.4012	0.3998
	Leaderboard Best [21] (as of June 8, 2020)	0.4200	**0.4190**
Multiple Losses (Ensemble)	Re-ranking over BM25	0.3877	0.3747
	Re-ranking over DeepCT	0.4012	-
	Submission #3: combining the above [5]	0.4049	0.3946
Multiple BERTs (Ensemble)	Re-ranking over BM25	0.4046	0.3905
	Re-ranking over DeepCT	0.4175	-
	Submission #5: combining the above [9]	**0.4213**	0.4073

5 Conclusion

In this paper, we propose the TFR-BERT framework for document and passage ranking. It combines state-of-the-art developments from both pretrained language models, such as BERT, and learning-to-rank approaches. Our experiments on the MS MARCO passage ranking task demonstrate its effectiveness.

6 Acknowledgement

We would like to thank Zhuyun Dai from Carnegie Mellon University for kindly sharing her DeepCT retrieval results. We would also like to thank Sebastian N. Bruch from Google Research for creating the MS MARCO datasets for our experiments. This work would not be possible without the support provided by the TF-Ranking team.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[2] Rama Kumar Pasumarthi, Sebastian Bruch, Xuanhui Wang, Cheng Li, Michael Bendersky, Marc Najork, Jan Pfeifer, Nadav Golbandi, Rohan Anil, and Stephan Wolf. Tf-ranking: Scalable tensorflow library for learning-to-rank. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2970–2978, 2019.

[3] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*, 2016.

[4] Shuguang Han, Xuanhui Wang, Michael Bendersky, and Marc Najork. Tf-ranking + bert (ensemble of pointwise, pairwise and listwise losses). https://microsoft.github.io/msmarco/, 2020. Online; accessed 30 March 2020. See the entry starts with TF-Ranking + BERT.

[5] Shuguang Han, Zhuyun Dai, Xuanhui Wang, Michael Bendersky, and Marc Najork. Deepct retrieval + tf-ranking bert ensemble. https://microsoft.github.io/msmarco/, 2020. Online; accessed 10 April 2020. See the entry starts with DeepCT Retrieval + TF-Ranking.

[6] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[7] Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*, 2020.

[8] Shuguang Han, Xuanhui Wang, Michael Bendersky, and Marc Najork. Tf-ranking ensemble of bert, roberta and electra. https://microsoft.github.io/msmarco/, 2020. Online; accessed 2 June 2020. See the entry starts with TF-Ranking + BERT.

[9] Shuguang Han, Xuanhui Wang, Michael Bendersky, and Marc Najork. Deepct + tf-ranking ensemble of bert, roberta and electra. https://microsoft.github.io/msmarco/, 2020. Online; accessed 2 June 2020. See the entry starts with TF-Ranking + BERT.

[10] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

[11] Lee Xiong, Chuan Hu, Chenyan Xiong, Daniel Campos, and Arnold Overwijk. Open domain web keyphrase extraction beyond language modeling. *arXiv preprint arXiv:1911.02671*, 2019.

[12] Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. Multi-stage document ranking with BERT, 2019.

[13] Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*, 2019.

[14] Tie-Yan Liu et al. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, 3(3):225–331, 2009.

[15] Sebastian Bruch, Masrour Zoghi, Mike Bendersky, and Marc Najork. Revisiting approximate metric optimization in the age of deep neural networks. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '19)*, pages 1241–1244, 2019.

[16] Zhuyun Dai and Jamie Callan. Context-aware sentence/passage term importance estimation for first stage retrieval. *arXiv preprint arXiv:1910.10687*, 2019.

[17] Shuguang Han, Xuanhui Wang, Michael Bendersky, and Marc Najork. Tf-ranking + bert (softmax loss, list size 6, 200k steps). https://microsoft.github.io/msmarco/, 2020. Online; accessed 30 March 2020. See the entry starts with TF-Ranking + BERT.

[18] Bhaskar Mitra and Nick Craswell. An updated duet model for passage re-ranking. *arXiv preprint arXiv:1903.07666*, 2019.

[19] Ming Yan. Enriched bert base + aoa index. https://microsoft.github.io/msmarco/, 2019. Online; accessed 19 March 2020.

[20] Xinwu Sun. Table model. https://microsoft.github.io/msmarco/, 2020. Online; accessed 16 April 2020.

[21] Xinwu Sun. Dr-bert. https://microsoft.github.io/msmarco/, 2020. Online; accessed 8 June 2020.