Naver Labs Europe @ TREC Deep Learning 2020

Thibault Formal[1,2], Benjamin Piwowarski[1], and Stéphane Clinchant[2]

[1] Sorbonne Université, LIP6, F-75005 Paris, France
benjamin.piwowarski@lip6.fr
[2] Naver Labs Europe, Meylan, France
{thibault.formal,stephane.clinchant}@naverlabs.com

Abstract. This paper describes our participation to the 2020 TREC Deep Learning challenge. While the track comprises 4 tasks in total (document and passage (re-)ranking), we only focused on the passage full ranking task, for which the goal is to retrieve and rank a set of 1000 passages directly from the collection of 8.8M entries. We submitted three runs, coming from a diverse set of experiments we conducted throughout the year regarding the use of BERT in ranking models. We explored simple dense embedding-based first stage retrieval, the impact of training transformer models from scratch with Masked Language Modeling (MLM) on the target collection, as well as diverse training settings and hyperparameter configurations.

1 Introduction

Over the last two years, the release of large pre-trained language models such as BERT [4] has led to tremendous progress in Information Retrieval. BERT-based models –following work from [10]– are now state-of-the-art for ad-hoc IR, and rank first on the MSMARCO passage and document ranking leaderboards[3]. Starting in 2019, the TREC Deep Learning track –building on the publicly released MSMARCO dataset [9]– has provided a testbed for IR researchers that aim to study new models (deep or not) in a large scale training regime. Evaluation methodology is similar to last year [2], thus we provide rankings for a test set of 200 queries, later annotated by NIST assessors.

Following the recent trend that transfers knowledge from pre-trained language models from the re-ranking setting to the full ranking one [3, 12, 17], part of our submission consists in moving from term-based retrieval (e.g. BM25) to dense embedding-based semantic retrieval. In the meantime, while techniques such as distillation lead to more efficient models that keep performance at the same level [5], we are interested in a slightly different –but related– question: can we train, *from scratch*, on the target collection, language models that are inherently smaller (in terms of number of parameters, training and inference

[3] https://microsoft.github.io/msmarco/

time) ?

Focusing on these two points, our submission is rooted in a set of experiments that studies various training settings, ranging from standard hyperparameters, losses, regularization techniques such as mixup [18], ensembling and so forth.

2 Methodology

From lexical to semantic first stage retrieval In a classic two-stage ranking system, term-based models like BM25 remain a go-to for the initial retrieval, as they provide baselines that are both efficient (through the inverted index) and actually hard to beat [16]. For this challenge, following recent works [1, 6, 14, 15, 17], we experimented with semantic first stage retrieval, based on a BERT Siamese architecture. Documents and queries are fed *independently* to BERT –allowing for offline document indexing–, and (fixed-length) document/query representations are obtained from a pooling of BERT embeddings: this is the so-called representation-based approach in IR. We found no clear difference between using `[CLS]` or taking the average of contextual embeddings (but we consider the latter as we got slightly better results). Relevance is obtained by computing similarity between representations (dot product in our case). For the similarity search, we used the FAISS library[4] [8], with the (exact, brute-force) `IndexFlatIP` index. We give the performance for this retrieval stage (best model) in Table 1.

	MRR@10 (dev)	R@1000 (dev)	NDCG@10 (DL 2019)	R@1000
`Siamese-BERT`	0.312	0.941	0.637	0.625

Table 1. Performance of first-stage semantic retrieval, on MSMARCO dev set (sparse labels) and TREC Deep Learning 2019 test set.

Learning Language Models from scratch We are also interested in the questions of models size as well as the role of pre-training in ranking models, particularly: (i) do models need to be *that* big (e.g. 12 layers, 110M parameters for BERT base) ?; (ii) is pre-training on such large collections (BooksCorpus + English Wikipedia) *necessary* to obtain reasonable performance ?; (iii) thus, can we get smaller and better models (for the task) by directly pre-training LMs on the target collection (more suited vocabulary, domain-specific pre-training) ?

This issue has already been investigated in [11], with Target Corpus Pre-training, but our goal differs a bit, as we also aim to learn smaller models, while having control on parameters such as the vocabulary, which could be critical, for instance on very specific corpora (e.g. medical or legal).

[4] https://github.com/facebookresearch/faiss

In order to do so, we investigate the performance of lighter transformer models, directly pre-trained with MLM on the MSMARCO passage collection. These checkpoints are then later used as a starting point for fine-tuning ranking models. We were able to train such models with the same level of performance, thus reducing inference time at no cost. We tested several configurations (in terms of architecture, training setting etc.), and we give some representative results in Table 2 (for a RoBERTa-type pre-training). While the study of the trade-off between model size and performance is interesting, we chose to use the RoBERTa-medium checkpoint for the challenge, as we are more interested in performance than efficiency.

	MRR@10 (dev)	architecture	# params
`BERT-base` [10]	0.347	$L = 12$, $H = 768$	110M
`BERT-base (our training)`	0.365	$L = 12$, $H = 768$	110M
`RoBERTa-medium`	0.360	$L = 6$, $H = 1024$	98M
`RoBERTa-small`	0.358	$L = 4$, $H = 1024$	56M
`RoBERTa-mini`	0.269	$L = 2$, $H = 512$	17M

Table 2. Performance of RoBERTa re-rankers, pre-trained with MLM on MSMARCO collection and later fine-tuned on BM25-based pairs. On this example, the number of heads is kept fixed across experiments ($A = 12$).

Training details We consider the MSMARCO passage ranking dataset, which contains approximately 8.8M passages. For training the LMs, we use `fairseq` [13] and adopt a fairly standard procedure borrowed from machine translation: we removed non-printing characters, normalized unicode punctuation, and used moses tokenizer. Then, a vocabulary of 20k subwords is learned with subword-nmt. For fine-tuning models –either from the standard checkpoint or our own pre-training–, we generally follow the same strategy. We experimented tuning the learning rate, the training loss (pointwise cross-entropy and various pairwise losses), some regularization strategies like mixup [18], the number of iterations and so forth. For training first stage ranking models, we acknowledge the discrepancy between training with pairs provided by MSMARCO (hence built from BM25) and the objective of increasing recall w.r.t. to say BM25, following observations already made in [15,17]. To reduce such gap, we use *in-batch negative* training pairs, where for a given query q, positive documents from other queries in the batch are used as negatives. Similarly, while training re-rankers, we want the distribution of items seen by the models during training and inference to be the same. Thus, when they are expected to re-rank documents from a dense first stage retrieval, we train the models with pairs built from this model (and not the ones provided by MSMARCO). We train and evaluate each model using 4 TESLA V100 GPUs (with 32G memory), and we generally set the batch size so that we fit memory.

Runs description and results Our submission follows a logic where runs are (supposedly) incrementally richer.

1. `NLE_pr1`: first stage Siamese-BERT (1) + ensemble of 8 BERT re-rankers (averaging scores) (2)
2. `NLE_pr2`: first stage ensemble of 2 Siamese-BERT (averaging scores), first is (1), second comes from our own RoBERTa checkpoint trained from scratch with MLM on MSMARCO, and later fine-tuned (RoBERTa-medium in Table 2). Re-ranking with an ensemble (averaging scores) of 8 BERT re-rankers (2) + 4 ELECTRA + 3 RoBERTa (from own RoBERTa-medium checkpoint)
3. `NLE_pr3`: Last run is an ensemble of `NLE_pr2` and a pipeline of BM25 + 6 BERT re-rankers, using reciprocal rank following [7]

In table 3, we provide results from our runs on the 2019 and 2020 tracks.

model	NDCG@10 (2019)	NDCG@10 (2020)
`BEST`	0.765	0.803
`NLE_pr1`	0.739	0.733
`NLE_pr2`	0.749	0.734
`NLE_pr3`	0.753	0.746

Table 3. Performance of submitted runs on TREC Deep Learning 2019 and 2020 test sets.

3 Conclusion

We briefly introduced our methodology and the resulting evaluations of our submissions for the 2020 TREC Deep Learning passage ranking task. The challenge was a way for us to summarize and organize past experiments, and confront some novels ideas to the IR community. We plan to dig more into those, by exploring how to extend and/or improve the standard Siamese architecture for first stage retrieval, as well as complementing our study of LM pre-training for IR (by e.g. varying vocabulary size or architecture configurations).

References

1. Chang, W.C., Yu, F.X., Chang, Y.W., Yang, Y., Kumar, S.: Pre-training tasks for embedding-based large-scale retrieval (2020)
2. Craswell, N., Mitra, B., Yilmaz, E., Campos, D., Voorhees, E.M.: Overview of the trec 2019 deep learning track (2020)
3. Dai, Z., Callan, J.: Context-aware sentence/passage term importance estimation for first stage retrieval (2019)

4. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. CoRR **abs/1810.04805** (2018), http://arxiv.org/abs/1810.04805
5. Gao, L., Dai, Z., Callan, J.: Understanding bert rankers under distillation. Proceedings of the 2020 ACM SIGIR on International Conference on Theory of Information Retrieval (Sep 2020). https://doi.org/10.1145/3409256.3409838, http://dx.doi.org/10.1145/3409256.3409838
6. Gao, L., Dai, Z., Chen, T., Fan, Z., Durme, B.V., Callan, J.: Complementing lexical retrieval with semantic residual embedding (2020)
7. Han, S., Wang, X., Bendersky, M., Najork, M.: Learning-to-rank with bert in tf-ranking (2020)
8. Johnson, J., Douze, M., Jégou, H.: Billion-scale similarity search with gpus (2017)
9. Nguyen, T., Rosenberg, M., Song, X., Gao, J., Tiwary, S., Majumder, R., Deng, L.: MS MARCO: A human generated machine reading comprehension dataset. CoRR **abs/1611.09268** (2016), http://arxiv.org/abs/1611.09268
10. Nogueira, R., Cho, K.: Passage re-ranking with BERT. CoRR **abs/1901.04085** (2019), http://arxiv.org/abs/1901.04085
11. Nogueira, R., Yang, W., Cho, K., Lin, J.: Multi-stage document ranking with bert (2019)
12. Nogueira, R., Yang, W., Lin, J., Cho, K.: Document expansion by query prediction. arXiv preprint arXiv:1904.08375 (2019)
13. Ott, M., Edunov, S., Baevski, A., Fan, A., Gross, S., Ng, N., Grangier, D., Auli, M.: fairseq: A fast, extensible toolkit for sequence modeling. In: Proceedings of NAACL-HLT 2019: Demonstrations (2019)
14. Reimers, N., Gurevych, I.: Sentence-bert: Sentence embeddings using siamese bert-networks (2019)
15. Xiong, L., Xiong, C., Li, Y., Tang, K.F., Liu, J., Bennett, P., Ahmed, J., Overwijk, A.: Approximate nearest neighbor negative contrastive learning for dense text retrieval (2020)
16. Yang, W., Lu, K., Yang, P., Lin, J.: Critically examining the "neural hype". Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (Jul 2019). https://doi.org/10.1145/3331184.3331340, http://dx.doi.org/10.1145/3331184.3331340
17. Zhan, J., Mao, J., Liu, Y., Zhang, M., Ma, S.: Repbert: Contextualized text embeddings for first-stage retrieval (2020)
18. Zhang, H., Cissé, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: Beyond empirical risk minimization. CoRR **abs/1710.09412** (2017), http://arxiv.org/abs/1710.09412