SIB Text Mining at TREC 2020 Deep Learning Track

Julien Knafou[1,2,3], Matthew Jeffryes[1,2], Sohrab Ferdowsi[1], and Patrick Ruch[1,2]

[1]HES-SO, University of Applied Sciences and Arts of Western Switzerland
[2]SIB, Swiss Institute of Bioinformatics, Geneva, Switzerland
[3]University of Geneva, Switzerland

February 15, 2021

Abstract

This second campaign of the TREC Deep Learning Track was an opportunity for us to experiment with deep neural language models reranking techniques in a realistic use case. This year's tasks were the same as the previous edition: (1) building a reranking system and (2) building an end-to-end retrieval system. Both tasks could be completed on both a document and a passage collection. In this paper, we describe how we coupled Anserini's information retrieval toolkit with a BERT-based classifier to build a state-of-the-art end-to-end retrieval system. Our only submission which is based on a RoBERTa$_{large}$ pretrained model achieves for (1) a ncdg@10 of .6558 and .6295 for passages and documents respectively and for (2) a ndcg@10 of .6614 and .6404 for passages and documents respectively.

1 Introduction

Last year's TREC Deep Learning task was the first edition of the track where both traditional information retrieval tools and newer techniques involving machine learning performances were compared in a real-world scenario. This was the most prominent effort to benchmark information retrieval methods in very large scales. In particular, the aim was to see whether large data could make certain methods, perhaps those relying on deep learning, excel other methods used for other data regimes.

This year, the second edition is essentially the same task on the same collections (MS MARCO) [Bajaj et al., 2018]. It brought us a validation set which is last year's test set and the opportunity to look back at what has been done over the last year in and out of the competition.

For this campaign, we tried new information retrieval systems along with BERT-based models [Devlin et al., 2018]. This paper describes our methodology for our submission that achieves a ndcg of 0.7186 and 0.6389 for passage and document reranking, respectively.

2 Related work

A seminal effort in language modeling is BERT, showing significant success in a wide range of NLP tasks. This is based on bi-directional transformers, and is pretrained with the tasks of masked language modeling, as well as next sentence prediction. Ever since BERT was introduced, several works tried to augment its performance. A successful work in this direction is RoBERTa [Liu et al., 2019], using larger and more diverse corpora for pretraining, as well as removing the next sentence prediction task, but using a novel masked language modeling. While it needs larger compute power, it noticeably improves the performance of BERT across different tasks.

The information retrieval community has also been using BERT for the learning-to-rank task. In particular, on the MS MARCO database, the work of [Nogueira and Cho, 2019] uses BERT for passage reranking by concatenating the query and the passage, similar to the next sentence prediction setup, but by obtaining the probability of relevance of a passage to a query. Another recent effort is [Han et al., 2020] that uses more specialized learning-to-rank losses to fine-tune BERT. While this work also uses RoBERTa, when we started training our model, this was absent within the information retrieval communities. Our motivation was to see whether similar performance improvements for NLP tasks using RoBERTa can be observed also for reranking.

3 Tasks

The TREC 2020 Deep Learning Track is divided in two tasks; a document and a passage reranking tasks. Both of them are divided in two sub-tasks. The first one is the *full ranking task* where, for a given query, a ranked list of candidates needs to be retrieved in a corpus. For this, one needs to develop an end-to-end solution. The maximum returned list length is either 1000 passages or a 100 documents. The second sub-task is the *top-k reranking task* where a list of candidates is provided for each query by the track organizer and only a reranking is required. In this sub-task too, the number of candidates differ depending on the task; for the passages, a list of 1000 candidates is provided, while for documents, a list of 100 candidates is provided.

Corpus	# of items
Passages	8,841,823
Documents	3,213,835

Table 1: Number of documents and passages in the both corpus

	Split	# of Queries
Passages	Train	808,731
	Dev	101,093
	Validation	101,092
	Test (2020)	200
Documents	Train	367,013
	Dev	5,193
	Validation	200
	Test (2020)	200

Table 2: Number of passages' and documents' queries

4 Data

We can divide the provided data into four categories; first, the corpus where a huge collection of either passages or documents are listed, second, a list of queries mapped to either passages or documents in the collections, third, a dataset that not only map query to passages, but also tells if a passage is irrelevant to a given query, finally, the ORCAS click data that has been released at the end of the campaign (which we did not have the opportunity to exploit, due to the timing of its release).

We used data from tables 1 and 2 only for the information retrieval part, the first step of our end-to-end solution. Table 1 shows the number of passages and documents in the corpus for a size of about 2.9GB and 22GB, respectively. Those corpora are the collections from which we had to retrieve candidates in the first sub-tasks.

Table 2 shows the number of queries for both documents and passages per split. The document queries map to a single document. Most passage queries map to a single passage, but in the development set, 12% map to more than one passage. The test (2020) split was released later during the campaign. We currently have only the queries and top-k candidates for this one, the mapping will come later.

Table 3 shows the number of both positive and negative passages linked to a set of queries. In order to get comparable results with [Han et al., 2020] and [Nogueira and Cho, 2019], and also because the dataset size was already big enough, we only used the small version of it.

In table 4, we see the number of query-candidate pairs for passages and documents per split. We only used the validation set in order to assess our models and the test (2020) set for the submission of our run.

Dataset	# query-positive & negative passage pairs
Train small	39,780,811
Train large	397,756,691

Table 3: Number of positive and a negative passages linked to a set of queries

	Split	# query-candidate pairs
Passages	Train	478,002,393
	Dev	6,668,967
	Validation	189,877
	Test (2020)	190,699
Documents	Train	36,701,116
	Dev	519,300
	Validation	20,000
	Test (2020)	20,000

Table 4: Number of query and candidate pairs

5 Methodology

Our end-to-end method is based on a two-fold strategy. First, we use information retrieval methods in order to narrow down a given query to a limited amount of candidates. Then, we rerank the query-candidate pairs using a BERT-based classifier. The following subsection will describe in detail both steps.

5.1 Information retrieval

Initial document retrieval used conventional information retrieval methods to produce a smaller set of candidates to which more computationally intensive methods could be applied. We used the Anserini toolkit to insert the document and passage datasets into Lucene indexes, which were then queried to produce the candidate sets [Yang et al., 2018]. For the document retrieval task, we used BM25 with $k = 3.44$ and $b = 0.87$. On the development set, this gave recall at 1000 of 0.933 and on the validation set, this gave recall at 1000 of 0.681. For the passage retrieval task we used BM25 with $k = 0.9$ and $b = 0.4$. On the development set, this gave recall at 1000 of 0.857 and on the validation set, this gave recall at 1000 of 0.738.

5.2 Reranking

As discussed in the data section, for the reranking part, we used the small version of query-positive and negative passages pairs training dataset. Our reranking model pipeline is essentially based on [Nogueira and Cho, 2019] where we use a classifier with a cross-entropy loss that tries to predict if a given passage is

Model	Batch size	Learning rate	Total step	FP Precision
BERT$_{base}$	128	3e-5	50000	32
RoBERTa$_{large}$	40	2e-5	300000	Mixed (16 & 32)

Table 5: Fine-tuning setting for both models

relevant to a query. In order to do that, we concatenate each query-passage pair with a separator token in-between and assigned a positive or a negative label to it whether the passage is relevant to a query or not. We focused our fine-tuning on two BERT-based pretrained models [Devlin et al., 2018] namely BERT$_{base}$ and RoBERTa$_{large}$ [Liu et al., 2019]. We fixed the maximum amount of tokens at 64 and 512 for the query length and the sequence concatenation length, respectively. We tried to maintained a certain balance of positive and negative passages for each batch. We used an Adam optimizer [Kingma and Ba, 2017] for both models.

Table 5 shows the main hyperparameters we set for this experiment. Due to the limited amount of resources during the first period of this experiment, we had to make decisions about the size of our training strategy. As this kind of model training is computationally intense, we first tried to train BERT$_{base}$ on only 640k of query-passage pairs. Once we were sure that the setup was working, we decided to train the RoBERTa$_{large}$ model on 12M pairs. In order to increase the batch size and decrease training time, we used mixed precision floating points (FP), this had an impact on the learning rate we decided to use as using a bigger rate could result in an unstable optimization.

As another strategy to reduce the memory requirements for fine-tuning the BERT models, we freezed the weights of the network and used the latent codes as fixed inputs. Instead of the usual one fully connected layer, this input was fed to a fully convolutional network and was trained for reranking using the binary cross-entropy loss. We noticed a very small drop in performance and did not report it in our submission. However, this can be an interesting future direction to economise on resource consumption by decoupling the BERT weights from those of the task-specific network.

5.2.1 Passage reranking

Once the model is trained, each query-passage pair (candidate) is evaluated and each passage is assigned a score according to its probability of being a relevant passage with respect to a query. We then return a list of query-passage pairs ranked according to their scores.

5.2.2 Document reranking

For the document reranking, we first divide each document to a list of passages. We then evaluate each passage the same way we did with the passage reranking. The document score will be equal to the biggest passage score of a given query-document pair.

Model name		RR	ndcg@10	MAP	R@100
Passages	BERT$_{base}$	**0.9360**	0.6677	0.4312	0.4852
	RoBERTa$_{large}$	0.9291	**0.7107**	**0.4644**	**0.5148**
Documents	BERT$_{base}$	0.9070	0.6456	0.2846	**0.3871**
	RoBERTa$_{large}$	**0.9244**	**0.6566**	**0.2863**	**0.3871**

Table 6: Results on the validation set using provided IR; 1000 candidates per query and 100 candidates for passage and document respectively

Run type		RR	ndcg@100	ndcg@10	MAP	R@100
Passages	reranking	0.8635	0.6587	0.7169	0.4823	0.6986
	end-to-end	**0.8769**	**0.6816**	**0.7192**	**0.4990**	**0.7438**
Documents	reranking	0.9185	0.6060	0.6295	0.4199	0.5985
	end-to-end	**0.9365**	**0.6390**	**0.6404**	**0.4423**	**0.6410**

Table 7: Official results

6 Results and discussion

Table 6 shows our results for both BERT$_{base}$ and RoBERTa$_{large}$ models on the validation split using top-k candidates of the second sub-task. Looking at the ndcg@10, which was the main metric of last year's leaderboard, we can see that for both tasks, RoBERTa$_{large}$ obtains the best results by about 5 points in the passage task. Looking at the document results, it is for the moment unclear if the small difference between the models is due to the model or the way we handled the transition from passage to document (see previous section).

Table 7 shows the official results for each sub-task. Our end-to-end solution outperformed the reranking results which means that our retrieval solution was better than the baseline provided.

7 Conclusion

Looking at the results, we're unable to draw any concrete conclusion, other than that a BERT-based reranker is an easy to implement tool that improves top-k metrics when compared to traditional information retrieval systems.

References

[Bajaj et al., 2018] Bajaj, P., Campos, D., Craswell, N., Deng, L., Gao, J., Liu, X., Majumder, R., McNamara, A., Mitra, B., Nguyen, T., Rosenberg, M., Song, X., Stoica, A., Tiwary, S., and Wang, T. (2018). Ms marco: A human generated machine reading comprehension dataset.

[Devlin et al., 2018] Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2018). BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

[Han et al., 2020] Han, S., Wang, X., Bendersky, M., and Najork, M. (2020). Learning-to-rank with bert in tf-ranking.

[Kingma and Ba, 2017] Kingma, D. P. and Ba, J. (2017). Adam: A method for stochastic optimization.

[Liu et al., 2019] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

[Nogueira and Cho, 2019] Nogueira, R. and Cho, K. (2019). Passage re-ranking with BERT. *CoRR*, abs/1901.04085.

[Yang et al., 2018] Yang, P., Fang, H., and Lin, J. (2018). Anserini: Reproducible ranking baselines using lucene. *J. Data and Information Quality*, 10(4).