University of Amsterdam at TREC 2020: Deep Learning Track

David Rau Nikolaos Kondylidis* Jaap Kamps

University of Amsterdam

Abstract

This paper documents the University of Amsterdam's participation in the TREC 2020 Deep Learning Track. Rather than motivated by engineering the best scoring system, our work is motivated by our interest in analysis informing our understanding of the opportunities and challenges of transformers for text ranking. Specifically, we focus on the passage retrieval task where we try to answer three of sets of questions.

First, transformers use different tokenization than traditional IR approaches such as stemming and lemmatizing, leading to different document representations. What is the effect of modern preprocessing techniques on traditional retrieval algorithms? Our main observation is that the limited vocabulary of the BERT tokenizer is affecting many long-tail tokens, which leads to large gains in efficiency at the cost of a small decrease in effectiveness.

Second, the effectiveness of transformers is a result of the self-supervised pre-training task promoting general language understanding, ignorant of the specific demands of ranking tasks. Can we make further correlate queries and relevant passages in the pre-training task? Our main observation is that there is a whole continuum between the original self-supervised training task of BERT and the final interaction ranker, and isolating ranking-aware pre-training tasks may leads to gains in efficiency (as these pretrained models can be reused for many tasks) as well as to gains in effectiveness (in particular when limited data on the target task is available).

Third, transformers combine large sequence length with many layers, with unclear what this deep semantics adds in the context of ranking. How complex do the models need to be in order to perform well on this task? Our main observation is that the deep layers of BERT lead to some, but relatively modest, gains in performance, but that the exact role of the presumed superior language understanding for search is far from clear.

1 Introduction

This paper documents our participation in the TREC 2020 Deep Learning Track. The Deep Learning Track started at TREC 2019 [Craswell et al., 2020b], and is in it's second year. The TREC Deep Learning Track is providing very large training data for its task, which is a necessity for training deep neural networks. The track consists of four tasks in total; two tasks for two collections. Full-Ranking and Re-Ranking tasks applied in collections of documents and passages. The Re-Ranking task focuses on ranking the top 1,000 candidates from the collection for each query. We focused on the collection of passages and performed experiments for both tasks. TREC provides training, evaluation and test queries, the collection of passages and training triplets that consist of a query, a couple of candidates and an indicator that defines the most relevant passage among the two candidates.

*Currently at the VU Amsterdam.

Table 1: Examples of the different tokenization techniques applied on the "orchestrating" term

Tokenization Technique	Resulting text
None	"orchestrating"
BERT Tokenizer	"orchestra", "##ting"
Porter Stemmer	"orchestr"
Porter Stemmer + BERT Tokenizer	"or", "##ches" "##tr"

This paper is structured in the following way. Our first experiment described in Section 2 studies the effects of BERT tokenizer and Porter stemmer as a preprocessing step before applying BM25 for the Full-Ranking task. The following two experiments in Sections 3 and 4 are using the Re-Ranking task. Specifically, Section 3 describes a proposed additional preprocessing step applied on BERT, which aims on further correlating tokens among queries and relevant passages. Finally, Section 4 evaluates the performance of BERT used as a re-ranker, when only some of its layers are being used and fine-tuned.

2 Impact of New Preprocessing Techniques

In this section we detail our tokenization experiments, motivated by the observation that modern transformers use different tokenization than traditional IR approaches such as stemming and lemmatizing, leading to different document representations. What is the effect of modern preprocessing techniques on traditional retrieval algorithms?

2.1 Tokenization in BERT

All retrieval systems are functioning under a multi-stage ranking setting. The first steps are using traditional IR techniques taking advantage of their efficiency and ability to be applied on large collections. The final stages are using modern ML techniques on a small subset of candidates. This is the case due to their superior text comprehension abilities which comes with a high computational cost which prohibits their application on large scale. The two approaches utilize documents differently and we wanted to study how the modern BERT tokenizer [Devlin et al., 2018] alone is affecting the BM25 term based similarity scores. Therefore we preprocessed text with BERT tokenizer and then run Full-Ranking experiments on the passage retrieval task. Additionally, we study the effect of the NLTK Porter stemmer [Porter, 1980] on both applications.

Pretrained BERT comes with a predefined vocabulary of size 30,522 which is also enforced on the pre-trained BERT tokenizer that was applied. The BERT tokenizer applies WordPiece tokenization [Devlin et al., 2018]. An example of word piece tokenization is that "orchestrating" will produce two tokens, namely "orchestra" and "##ting". The double # character indicates that this token is part of the previous word. This ability of BERT tokenizer allows it to be very descriptive taking into account its very small vocabulary size. Nevertheless, the vocabulary size of BERT tokenizer is still limiting the tokenized text, compared to traditional term based retrieval algorithms were there is no vocabulary size limitation. The words that cannot be represented by the BERT tokenizer are resulting to UNK tokens, which are being removed from the resulting text. An example of the different tokenization techniques applied is illustrated in Table 1.

We try to quantify these effects by reporting the number of (uniquely) affected tokenized words in Table 2. This clearly demonstrates that the limited BERT vocabulary has clear impact. Recall,

Table 2: Number of terms identified as unknown by BERT tokenizer, also after Porter stemming

Tokenizer	Stemmer	Total Unknown Terms	Unique Unknown Terms
BERT	-	15,401,102	4,172,669
BERT	Porter	74,231,878	3,980,886

Table 3: Preprocessing effects using BM25 on various tokenizations (scores in percentages)

Tokenizer	Stemmer	MAP	Rec@100	NDCG@10	P@10
Anserini	-	27.75	51.22	50.53	37.96
Anserini	Porter	28.19	55.92	48.06	35.19
BERT†	-	26.51	53.61	46.58	32.96
BERT	Porter	25.38	50.03	45.98	32.59

† *Official submission as* bm25_bert_token.

in standard IR approaches essentially all tokens (except for stop-words) are indexed. These absolute numbers of unknown tokens are massive. To put them into perspective, the untokenized corpus has 6,620,939 terms in our index, making also the fraction very large. The impact of this requires further research, the overwhelming majority of them occurs only once, and inspection of a sample reveals noisy character strings of unclear benefit to the performance.

2.2 Results

The BM25 Full-Ranking results are presented in Table 3. Although all scores are reasonably close, applying ther BERT tokenizer decreases the performance of BM25 across all metrics. In addition, Porter stemming leads to a performance increase under normal conditions, but if we first apply Porter stemmer and then the BERT tokenizer, there is a further performance drop. Above, in Table 2 we quantified the effects of the BERT tokenizer. Hence, the very large number of unknown words might be the main cause of the lower BM25 performance when BERT tokenizer is applied. Furthermore, by applying Porter stemmer before tokenization, the number of total unknown word instances increases by 400%, even though the number of unique unknown words slightly decreases. This suggests that by applying Porter stemmer initially, we affect less words in total but these words are more frequent, which can also explain the further performance drop of this technique.

Our general conclusion is that the limited vocabulary of the BERT tokenizer is affecting many long-tail tokens, which leads to large gains in efficiency at the cost of a small decrease in effectiveness.

3 Relevance Embedding Pre-training Task

In this section we detail our ranking-aware pre-training task experiments, motivated by the observation that modern transformers are pre-trained using a self-supervised pre-training task promoting general language understanding, ignorant of the specific demands of ranking tasks. Can we make further correlate queries and relevant passages in the pre-training task?

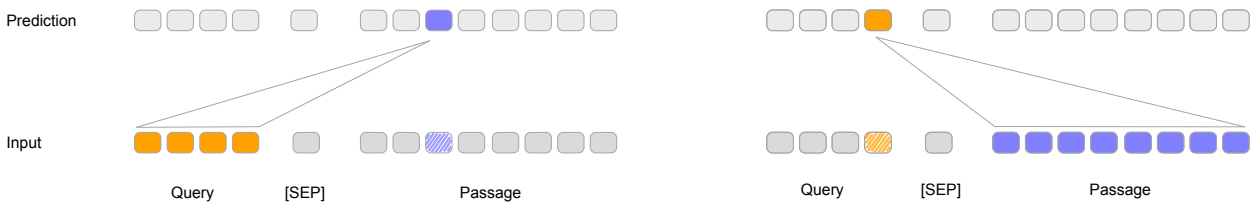

Figure 1: RelEmb pre-training task. Predicting randomly tokens in the passage from the query (left) or tokens in query from the passage (right).

3.1 Ranking-aware Pre-training

The success of the BERT model in various tasks can, next to its self-attention mechanism, be attributed to massive pre-training on large amounts of unlabeled text. Generally, pre-training allows to first build up a general representation that can later be helpful in the specialization of a down-stream task; particularly, in cases where the data for the down-stream task is very scarse and would not be sufficient to train a model on.

For the pre-training of BERT two special tasks have been developed: The Mask-LM (MLM) task where the model predicts random masked words in a sentence and the Next Sentence Prediction (NSP) task that requires the model to predict whether two sentences appear next to each other. While those pre-training tasks equip the model with a general language understanding that has been shown to significantly improve ranking [Nogueira and Cho, 2019b], they are not tailored upon ranking specific demands. Specifically, capturing the *semantic relevance between query and passage*, which plays an essential role for QA-focused datasets such as MS MARCO [Qiao et al., 2019], is not incorporated into the current pre-training of interaction-based rankers and is only indirectly learned during down-stream training. As pointed out by Yan et al. [2020], we also feel the urge for the design of appropriate pre-training tasks for ranking with BERT. To this end we propose a new relevance embedding (RelEmb) pre-training task for interaction-based rankers to better model the interaction between query and passage.

3.2 Methodology

Given a query and a relevant document, we form the combined input in the standard form of "CLS + query + SEP + relevant_passage." Additionally, we add segmentation embeddings to allow the model to disambiguate query and the relevant passage. We truncate the query to a maximum 64 tokens and the total length of to input to 512. As the goal of this pre-training task is to model strong interactions between query and passage, we randomly mask tokens in the query and try to predict the words only from the passage and vice versa (as is shown in Figure 1). This way we explicitly learn a query and a document model given the other respectively. In 80% of the times mask tokens from the query, and 20% tokens from the document to account for the reason that predicting words in the document given the query is more challenging. We apply this additional pre-training step on the already pre-trained BERT, as a first preliminary experiment. We experimented with extracting relevant query/passage pairs from the ORCAS click data [Craswell et al., 2020a] and the provided MS MARCO training triples. After the pre-training we trained the BERT model following Nogueira and Cho [2019b].

Table 4: Passage re-ranking (top 1,000) performance for the relevance embedding (RelEmb) pre-training task evaluated on the TREC 2020 test topics

Model	Pre-training dataset	MAP	NDCG@10	MRR
BERT Vanilla	-	44.92	66.71	80.08
BERT RelEmb	MS Marco	**45.33**	**69.29**	80.93
BERT RelEmb†	Orcas	45.32	67.48	**83.36**

† *Official submission as relemb_mlm_0_2 (NDCG@10 0.6662, MRR 0.7677), scores differ as the model was not put into evaluation mode.*

3.3 Results

The results of the ranker pre-trained of our proposed RelEmb task alongside the vanilla BERT ranker can be found in Table 4. The RelEmb pre-training task increases the performance across all metrics compared to the vanilla pre-trained BERT model for both datasets. Particularly, NDCG@10 increases around 2.5 points for the BERT RelEmb trained on MS Marco, and MRR increases over 3 points when pre-trained on Orcas. These preliminary experiments show that designing new training tasks that go beyond general language understanding and target down-stream task specific characteristics holds potential and are worth exploring more in the future.

The proposed RelEmb pre-training task can especially be useful when the amount of query/passage pairs is insufficient to fine-tune a ranker on. In this case the model can be first pre-trained on available query/passage pairs (from another dataset) and then later be fine-tuned on the available ranking labels. Further, pre-training a ranker-like model on a large dataset has the advantage that its learned representations can be leveraged to many different tasks. Ranker that are trained directly on the down-stream task can not be reused have to be trained every single time from "scratch" requiring computational resources, time and data. In additional preliminary experiments we observed that after the RelEmb pre-training on Orcas the model is already closer to a BERT ranker and starts at a higher performance than the vanilla BERT. This experiment outlines the most naive way of modeling query-passage interaction and has to be seen as an initial effort in designing a ranking-aware pre-training task that needs refinement in future work.

Our general conclusion is that there is a whole continuum between the original self-supervised training task of BERT and the final interaction ranker, and isolating ranking-aware pre-training tasks may leads to gains in efficiency (as these pretrained models can be reused for many tasks) as well as to gains in effectiveness (in particular when limited data on the target task is available).

4 Impact of the Deep Layers of BERT

In this section, we detail our experiments with the depth of the transformer model, motivated by trying to understand the what this deep semantics adds in the context of ranking. How complex do the models need to be in order to perform well on this task?

4.1 Impact of Model Depth

Fine-tuning BERT on the binary classification task for identifying whether a passage is relevant or irrelevant proved to perform very well, as proposed by Nogueira and Cho [2019a]. The model's input is given in the form [CLS] + query + [SEP] + passage. The hidden activations of the CLS token are given to a binary classifier. The relevance probability was used as a relevance score. We

Table 5: Utilizing some of BERT's layers to be used as a passage re-ranker

Layers Finetuned	MAP	NDCG@10	MRR
BERT layers 1-12 (Vanilla)	44.92	66.71	80.08
BERT layers 1-9	41.50	64.49	78.06
BERT layers 1-6[†]	40.59	63.84	79.07
BERT layers 1-3	34.55	56.06	73.41
BERT layers 1-2	28.94	46.97	63.92

[†]*Official submission as bert_6 (NDCG@10 0.6149, MRR 0.7386), scores differ as the model was not put into evaluation mode.*

Table 6: Selection of topics with the most gain in performance comparing BERT 12 to BERT 2

Query	Gain (MAP)
average salary for dental hygienist in nebraska	+ 55.16
what is the un fao	+ 54.29
what is a statutory deed	+ 44.26
how many sons robert kraft has	+ 39.68
do google docs auto save	+ 39.03

extended this approach and question whether the full 12-layer BERT model has to be used for this task. The intuition is that this task might be simple enough, not requiring high level comparisons between the query and the passage that is provided by all 12 BERT layers of the pre-trained BERT. Instead, semantic matching of lower level might be enough. Our training scheme is the same with Nogueira and Cho [2019a] and in all cases the final binary classification linear layer is always randomly initialized.

4.2 Results

The results of our experiments are displayed in Table 5. Top line (1-12) represents the vanilla approach were all 12 layers were utilized, as proposed by Nogueira and Cho [2019a]. We can see that when using only the first 2 or three layers of BERT the performance drops significantly (comparing 1-2 & 1-3 to 1-12). On the other hand, if we use the first six or nine layers, the performance does not deteriorate that much. Specifically, with respect to MRR the performance difference is almost insignificant. Regarding NDCG@10 and MAP@1000 the performance drops by around 3 absolute percentage points. Taking into account that the models 1-9 and 1-6 use roughly 75% and 50% of the parameters used by the vanilla 1-12 model, one can argue that this is a very beneficial trade-off, since these models can be trained and applied much faster. Whether this trade-off is beneficial depends on the task where the models will be applied.

In Table 6 we show a selection of the topics that gained the most in performance comparing BERT 12 to BERT 2. All of these requests are more complex (so no simple navigational search) and have compound multi-term concepts, and all of them have specific relations between multiple concepts as signaled by the prepositions and verbs. Arguably, here the language understanding encoded in BERT can be potentially helpful, and this may explain the gains in performance. However, when inspecting topics that don't gain from BERT's depth, we see very similar query characteristics, defying the naive explanation that these gains are due to the required language

understanding of these particular topics. Further research is needed to better understand the impact of transformer depth on search.

Our general conclusion is that the deep layers of BERT lead to some, but relatively modest, gains in performance, but that the exact role of the presumed superior language understanding for search is far from clear.

5 Conclusions

This paper documented our participation in the TREC 2020 Deep Learning Track. We conducted three sets of relatively independent experiments, analyzing the impact of the tokenization of modern transformers, experimenting with ranking-aware pre-training tasks, and analyzing the impact of the deep layers of BERT on ranking. Our main conclusions were the following. First, the limited vocabulary of the BERT tokenizer is affecting many long-tail tokens, which leads to large gains in efficiency at the cost of a small decrease in effectiveness. Second, there is a whole continuum between the original self-supervised training task of BERT and the final interaction ranker, and isolating ranking-aware pre-training tasks may leads to gains in efficiency (as these pretrained models can be reused for many tasks) as well as to gains in effectiveness (in particular when limited data on the target task is available). Third, the deep layers of BERT lead to some, but relatively modest, gains in performance, but that the exact role of the presumed superior language understanding for search is far from clear.

Acknowledgments

We thank the track organizers for their amazing service and effort in making realistic benchmarks for neural ranking available. This research is funded in part by the Netherlands Organization for Scientific Research (NWO STW # 17752; NWO CI # CISC.CC.016), Facebook Research (Computationally Efficient NLP grant), and the Innovation Exchange Amsterdam (POC grant). Views expressed in this paper are not necessarily shared or endorsed by those funding the research.

References

N. Craswell, D. Campos, B. Mitra, E. Yilmaz, and B. Billerbeck. Orcas: 18 million clicked query-document pairs for analyzing search, 2020a.

N. Craswell, B. Mitra, E. Yilmaz, D. Campos, and E. M. Voorhees. Overview of the TREC 2019 deep learning track. In *TREC 2019: Proceedings of the Twenty-Eighth Text REtrieval Conference*. NIST Special Publication 1250, 2020b.

J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL http://arxiv.org/abs/1810.04805.

R. Nogueira and K. Cho. Passage re-ranking with BERT. *CoRR*, abs/1901.04085, 2019a. URL http://arxiv.org/abs/1901.04085.

R. Nogueira and K. Cho. Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*, 2019b.

M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980. doi: 10.1108/eb046814. URL https://doi.org/10.1108/eb046814.

Y. Qiao, C. Xiong, Z. Liu, and Z. Liu. Understanding the behaviors of BERT in ranking. *CoRR*, abs/1904.07531, 2019. URL http://arxiv.org/abs/1904.07531.

M. Yan, C. Li, C. Wu, B. Bi, W. Wang, J. Xia, and L. Si. IDST at TREC 2019 deep learning track: Deep cascade ranking with generation-based document expansion and pre-trained language modeling. In *TREC 2019: Proceedings of the Twenty-Eighth Text REtrieval Conference*. NIST Special Publication 1250, 2020.