Document Re-ranking MS MARCO using BERT

Niels van Harten
s1012159

Jonan Richards
s1036117

Floris Rossel
s1010794

1 INTRODUCTION

The use of neural retrieval models has been a relatively new development in the field of information retrieval. However, as opposed to other fields in computing, neural models have not been able to significantly improve ranking in the first couple of years when these models were developed. This changed with the advent of language transformer models like BERT, where ranking performance finally improved over traditional non-neural methods, breaking the stagnating progress in information retrieval. Given that using BERT for information retrieval is computationally expensive, it is mostly used for re-ranking of non-neural retrieval models like BM25. Applied this way, inference cost is somewhat limited, and results are still impressive, outperforming BM25 significantly.

BERT models optimized for information retrieval often use a query-passage pair as input and output a score which can be used to rank passages for a given query. We would like to see how easy it is to use such a model to re-rank MS MARCO documents instead of passages having a maximum length of 512 tokens. Is the usage of BERT for document ranking a profitable alternative to simple non-neural approaches on MS MARCO? How should one split the documents to feed into the BERT model and how should one aggregate the scores produced by BERT? We will compare two methods to split documents into passages and investigate multiple aggregation methods combining the passage scores into a single document score for a given query.

2 RELATED WORK

Our work tries to solve a document ranking task within the field of information retrieval. Here, the task is to rank a collection of documents as being relevant to the user given a user-defined query. A strong baseline for this problem is BM25, a model that estimates the probability that a document-query pair is relevant based on the similarity between a query and document. We use IndriQueryLikelihood as baseline retrieval model for our approach which is another baseline often used for document retrieval. This retrieval model is based on the Indri Query Language and used as baseline for the MS MARCO document retrieval task.

An important characteristic of document retrieval is that the top-K documents are most important as users typically do not look further than the first 10 queries. Therefore, re-ranking of the top-K documents using a better but more expensive retrieval model is found to be a good approach to improve retrieval. We will use the neural language transformer BERT for re-ranking. BERT (Bidirectional Encoder Representations from Transformers, [3]) learns contextual word embeddings based on bidirectional training of a transformer on huge amounts of text.

Our work follows up on the work by Nogueira et al. [5] which uses BERT BASE and fine-tunes the model on MS MARCO for re-ranking query-passage results from a top-1000 ranking created with BM25. We will expand on this work by re-ranking top-100 document rankings from IndriQueryLikelihood using their fine-tuned model. Our task differs from theirs as we re-rank complete documents instead of passages. The major challenge of using complete documents is that not all documents can be used as input of the BERT model directly given that it has a input size limitation of 512 tokens. Therefore, our focus lies on how to split the documents and how to aggregate the partial BERT results for a document.

Because of the input-size limitation of BERT BASE, we have to split the documents that are too large into segments. Each segment will therefore get its own score. Aggregating these scores to get the final document score can be done in several ways, some of which have been explored by Dai et al., 2019 [2]. Our approach of splitting documents to feed into BERT followed by aggregation partial results is similar to theirs. However, we apply the task on the MS MARCO dataset instead of on Robust04 and ClueWeb09-B.

3 METHOD

In document ranking tasks, documents are often longer than the limited input size of BERT allows. An approach around this limitation is to split the documents into segments. These segments are fed into BERT separately, after which their scores are aggregated into an overall score for the entire document. After all (relevant) documents for a given query have been scored, they can be ranked based on this score.

In this paper, we will evaluate two methods of splitting documents. The first, most basic approach we have dubbed the 'length' method is to cut off segments at the maximum allowed BERT input length. However, we hypothesize that in some cases this may lead to a decrease in performance, due to potentially important words losing part of their context. In order to help remedy this, we will evaluate a second method dubbed the 'period' method where segments are cut off at the last period ('.') token before the length limit. Hopefully, this approach will leave most of the context of the words neighbouring the cutoff intact.

Additionally, there are multiple ways to aggregate the segment scores into a document score. In this paper, we will address three: a 'average', a 'max' and a 'first' operation, corresponding to the following formulas respectively:

$$avg = \frac{\Sigma_{i \in n} \text{segment}_i}{n}$$

and

$$max = \max(\text{segment}_1, ...\text{segment}_n) \textbf{ and } first = \text{segment}_1$$

As aggregation methods we also considered normalized max and sum (not normalized). However, the results of those two approaches looked unconvincing.

The document ranking dataset we will be using is from the TREC 2019 Deep Learning track [1], which is based on the MS MARCO dataset [4]. Since our focus is on the application of BERT on document ranking, we will not be pre-ranking the entire document collection for every query. Rather, we will re-rank a provided top

100 documents for every query, which corresponds to the ‘document re-ranking’ task of the TREC 2019 Deep Learning track.

For every query, we will run the fine-tuned BERT model from [5] on the segments obtained from splitting the top 100 documents for that query. These segment scores will be aggregated into document scores, which will be converted into ranks. These ranks will be compared to the relevancy labels for that query based on several metrics. This process will be done for both document splitting methods and both aggregation methods in order to determine the optimal approach to document re-ranking using BERT.

4 EXPERIMENTAL SETUP

Since we have limited time and computational resources, we will limit our experiment to a subset of our dataset. We have sampled 100 queries from the TREC 2019 development (‘dev’) subset to form a validation set, and another 1000 queries to form a test set.

We will compare the document splitting methods and aggregation methods in two separate experiments. In the first experiment (table 1), we will attempt to determine which document splitting method is the best by running them on the validation set, and comparing them across both aggregation methods. The second experiment will be done on the test set, comparing the two aggregation methods only for the splitting method that came out as best in experiment 1 (table 2). Should the results of experiment 1 be ambiguous (e.g. the ‘period’ splitting method works best for the ‘average’ aggregation method and the ‘length’ splitting method for ‘max’), we will run experiment 2 for both splitting methods as well.

Because the weights of the BERT model fine-tuned on MS MARCO described in [5] is not accompanied with a model implementation, we have based our implementation on the one that can be found on https://github.com/Tomjg14/Master_Thesis_MSMARCO_Passage_Reranking_BERT. During preprocessing, this implementation removes all non-alphanumeric characters from its inputs. However, our implementation of the ‘period’ splitting method requires us to leave period characters in the documents. In order to evaluate whether or not removing periods has an effect on the performance of the fine-tuned BERT, we have also executed an initial experiment on the validation set where a comparison is made with the ‘length’ splitting method between removing periods or leaving them in (table 3).

The code of the implementation for this paper can be found on https://github.com/jonan-richards/BERT-MSMARCO-document-ranking

5 RESULTS

In table 1, every combination of document splitting method (‘length’ or ‘period’) and aggregation method (‘average’, ‘max’ or ‘first’) is compared on several metrics, evaluated on the validation dataset. The best score per aggregation method on every metric is bold and underlined. While the length splitting method seems to perform better than the period method on most metrics for the average aggregation, the reverse is true for the max and first aggregation. Therefore, it is not clear which splitting method is better, meaning that both will have to be taken into account in experiment 2 (table 2).

Table 1: Splitting methods ‘length’ and ‘period’ compared

	Average		Max		First	
Metric	Length	Period	Length	Period	Length	Period
precision@1	**0.170**	0.150	0.240	**0.270**	**0.270**	**0.270**
precision@5	**0.110**	**0.110**	0.112	**0.116**	**0.116**	0.122
precision@10	0.063	**0.064**	0.068	**0.070**	**0.074**	**0.074**
precision@25	**0.030**	**0.030**	**0.031**	**0.031**	**0.031**	**0.031**
recall@1	**0.170**	0.150	0.240	**0.270**	**0.270**	**0.270**
recall@5	**0.550**	**0.550**	0.560	**0.580**	0.580	**0.610**
recall@10	0.630	**0.640**	0.680	**0.700**	**0.740**	**0.740**
recall@25	**0.740**	**0.740**	**0.780**	**0.780**	**0.770**	**0.770**
F-score@1	**0.170**	0.150	0.240	**0.270**	**0.270**	**0.270**
F-score@5	**0.183**	**0.183**	0.187	**0.193**	0.193	**0.203**
F-score@10	0.115	**0.116**	0.124	**0.127**	**0.135**	**0.135**
F-score@25	**0.057**	**0.057**	**0.060**	**0.060**	**0.059**	**0.059**
NDCG@1	**0.170**	0.150	0.240	**0.270**	**0.270**	**0.270**
NDCG@5	**0.370**	0.359	0.411	**0.437**	0.437	**0.448**
NDCG@10	**0.395**	0.387	0.450	**0.477**	0.489	**0.490**
NDCG@25	**0.422**	0.411	0.474	**0.497**	**0.497**	**0.497**
MAP	**0.329**	0.315	0.384	**0.413**	0.414	**0.415**
MRR	**0.329**	0.315	0.384	**0.413**	0.414	**0.415**

In table 2, our method is compared to the baseline method (IndriQueryLikelihood) on several metrics. Our method has been evaluated on the test dataset for every combination of document splitting method (‘length’ or ‘period’) and aggregation method (‘average’, ‘max’ or ‘first’). The best score on every metric is bold and underlined. The ‘first’ aggregation method performs best on most evaluation metrics followed closely by ‘max’ aggregation. The ‘average’ aggregation does not perform best for any of the metrics indicating it to be the worst of the three methods. For ‘first’, the length splitting method seems to marginally outperform period splitting while for ‘max’ it seems that period splitting method is marginally better than the length splitting method. Our approach consistently outperforms the baseline by a large margin for all the tried methods.

In table 3, the ‘length’ splitting method is compared on several metrics both with (A) or without (B) period characters. The comparison is done across all aggregation methods (‘average’, ‘max’ or ‘first’) and evaluated on the validation dataset. From the scores between A and B being only marginally different for the aggregation methods, it is clear that not removing period characters does not have a significant positive or negative effect on performance.

Table 2: Our different splitting and aggregation methods compared to the baseline method

	Our method						Baseline
	Average		Max		First		
Metric	Length	Period	Length	Period	Length	Period	IQL
precision@1	0.194	0.199	0.232	0.239	**0.244**	0.243	0.123
precision@5	0.086	0.085	**0.104**	**0.104**	0.103	0.103	0.068
precision@10	0.057	0.056	0.062	**0.063**	0.062	0.062	0.044
precision@25	0.028	0.028	0.028	0.028	**0.029**	**0.029**	0.024
recall@1	0.194	0.199	0.232	0.239	**0.244**	0.243	0.123
recall@5	0.432	0.427	**0.520**	0.518	0.514	0.516	0.340
recall@10	0.569	0.560	0.625	**0.629**	0.615	0.616	0.443
recall@25	0.693	0.691	0.711	0.709	**0.715**	0.713	0.592
F-score@1	0.194	0.199	0.232	0.239	**0.244**	0.243	0.123
F-score@5	0.144	0.142	**0.173**	**0.173**	0.171	0.172	0.113
F-score@10	0.103	0.102	**0.114**	**0.114**	0.112	0.112	0.081
F-score@25	0.053	0.053	**0.055**	**0.055**	**0.055**	**0.055**	0.046
NDCG@1	0.194	0.199	0.232	0.239	**0.244**	0.243	0.123
NDCG@5	0.316	0.316	0.383	0.383	**0.388**	**0.388**	0.235
NDCG@10	0.360	0.359	0.417	0.419	**0.420**	0.420	0.268
NDCG@25	0.391	0.392	0.438	0.439	**0.445**	0.444	0.305
MAP	0.306	0.308	0.358	0.360	**0.367**	0.366	0.227
MRR	0.306	0.308	0.358	0.360	**0.367**	0.366	0.227

Table 3: Results for the 'length' splitting method, with periods included (A) or removed (B)

	Average		Max		First	
Metric	A	B	A	B	A	B
precision@1	0.180	0.170	0.240	0.240	0.260	0.270
precision@5	0.106	0.110	0.110	0.112	0.122	0.116
precision@10	0.106	0.063	0.069	0.068	0.073	0.074
precision@25	0.030	0.030	0.031	0.031	0.031	0.031
recall@1	0.180	0.170	0.240	0.240	0.260	0.270
recall@5	0.530	0.550	0.550	0.560	0.610	0.580
recall@10	0.620	0.630	0.690	0.680	0.730	0.740
recall@25	0.740	0.740	0.770	0.780	0.770	0.770
F-score@1	0.180	0.170	0.240	0.240	0.260	0.270
F-score@5	0.177	0.183	0.183	0.187	0.203	0.193
F-score@10	0.113	0.115	0.125	0.124	0.133	0.135
F-score@25	0.057	0.057	0.059	0.060	0.059	0.059
NDCG@1	0.180	0.170	0.240	0.240	0.260	0.270
NDCG@5	0.368	0.370	0.411	0.411	0.447	0.437
NDCG@10	0.397	0.395	0.458	0.450	0.486	0.489
NDCG@25	0.426	0.422	0.478	0.474	0.496	0.497
MAP	0.335	0.329	0.391	0.384	0.413	0.414
MRR	0.335	0.329	0.391	0.384	0.413	0.414

6 DISCUSSION

We demonstrated that using the BERT BASE transformer network for re-ranking documents is very effective, even when these documents needed to be split into chunks due to the input size limitation of this model. Using the first and maximum scores of all document segments significantly improved ranking performance compared to using the average over all the segments. For the maximum aggregation method, this makes a lot of sense if we think about the goal of these queries, which is finding relevant pieces of information in large amounts of information. Many documents contain text about much more than most queries refer to. Using an average score drags the document score down when the document contains additional text that does not closely match the query semantics. This should not happen, because the relevant text in the document is equally important, regardless of other text. It is somewhat more surprising that the 'first' method results are very comparable to the 'max' results. This suggests that the first passage of a document often suffices when determining the relevance of the entire document for MS MARCO.

We also showed that, in the 'max' aggregation method, splitting the input text on periods slightly improved the ranking performance compared to splitting based on maximum input size, where sentences are often split in two between input chunks. However, in the 'average' aggregation method, this performance difference seemed to be reversed in most metrics. Because the score differences are so small, we can not definitively say that there is an effect. We hypothesise that there is a slight negative impact on performance when sentences are cut in half, since sentence context is removed for both halves of the split. This lost context might be important for matching query semantics to document semantics. Another way to solve this problem that we could have followed is to use overlapping passages, as in the work of Dai et al., 2019 [2]. This method might give better ranking results than splitting on periods, because context that depends on multiple sequential sentences is better preserved near the split locations in the text.

One other deviation from the training procedure of the model is the way that we feed the pre-trained BERT BASE model the data.

The model input is slightly different from the data it was trained on, since we do not remove the periods in the input text. We concluded that this was not a problem, given the negligible score differences in our test which compared leaving the periods in and removing them. Another difference is that the model was trained with the task of passage ranking, which is not exactly the same task as document ranking. Training on document ranking might improve results in our pipeline.

Finally, it is important to note that we used small subsets of the TREC 2019 dataset for testing because of time constraints. Running the whole test set should give more consistent and reliable results, and can therefore give more confidence in the decision for which methods are superior.

REFERENCES

[1] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. 2020. Overview of the TREC 2019 deep learning track. https://doi.org/10.48550/ARXIV.2003.07820

[2] Zhuyun Dai and Jamie Callan. 2019. Deeper Text Understanding for IR with Contextual Neural Language Modeling. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Paris, France) *(SIGIR'19)*. Association for Computing Machinery, New York, NY, USA, 985–988. https://doi.org/10.1145/3331184.3331303

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

[4] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. In *CoCo@NIPS*. http://ceur-ws.org/Vol-1773/CoCoNIPS_2016_paper9.pdf

[5] Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage Re-ranking with BERT. *arXiv preprint arXiv:1901.04085* (2019).