PASSAGE RE-RANKING WITH BERT

Rodrigo Nogueira
New York University
rodrigonogueira@nyu.edu

Kyunghyun Cho
New York University
Facebook AI Research
CIFAR Azrieli Global Scholar
kyunghyun.cho@nyu.edu

ABSTRACT

Recently, neural models pretrained on a language modeling task, such as ELMo (Peters et al., 2017), OpenAI GPT (Radford et al., 2018), and BERT (Devlin et al., 2018), have achieved impressive results on various natural language processing tasks such as question-answering and natural language inference. In this paper, we describe a simple re-implementation of BERT for query-based passage re-ranking. Our system is the state of the art on the TREC-CAR dataset and the top entry in the leaderboard of the MS MARCO passage retrieval task, outperforming the previous state of the art by 27% (relative) in MRR@10. The code to reproduce our results is available at https://github.com/nyu-dl/dl4marco-bert

1 INTRODUCTION

We have seen rapid progress in machine reading compression in recent years with the introduction of large-scale datasets, such as SQuAD (Rajpurkar et al., 2016), MS MARCO (Nguyen et al., 2016), SearchQA (Dunn et al., 2017), TriviaQA (Joshi et al., 2017), and QUASAR-T (Dhingra et al., 2017), and the broad adoption of neural models, such as BiDAF (Seo et al., 2016), DrQA (Chen et al., 2017), DocumentQA (Clark & Gardner, 2017), and QAnet (Yu et al., 2018).

The information retrieval (IR) community has also experienced a flourishing development of neural ranking models, such as DRMM (Guo et al., 2016), KNRM (Xiong et al., 2017), Co-PACRR (Hui et al., 2018), and DUET (Mitra et al., 2017). However, until recently, there were only a few large datasets for passage ranking, with the notable exception of the TREC-CAR (Dietz et al., 2017). This, at least in part, prevented the neural ranking models from being successful when compared to more classical IR techniques (Lin, 2019).

We argue that the same two ingredients that made possible much progress on the reading comprehension task are now available for passage ranking task. Namely, the MS MARCO passage ranking dataset, which contains one million queries from real users and their respective relevant passages annotated by humans, and BERT, a powerful general purpose natural language processing model.

In this paper, we describe in detail how we have re-purposed BERT as a passage re-ranker and achieved state-of-the-art results on the MS MARCO passage re-ranking task.

QA: ①、由标准机制(eg. BM25)检索的大量可能相关的 document.
②. 重排 document
③ topK 用于生成答案

2 PASSAGE RE-RANKING WITH BERT

Task A simple question-answering pipeline consists of three main stages. First, a large number (for example, a thousand) of possibly relevant documents to a given question are retrieved from a corpus by a standard mechanism, such as BM25. In the second stage, *passage re-ranking*, each of these documents is scored and re-ranked by a more computationally-intensive method. Finally, the top ten or fifty of these documents will be the source for the candidate answers by an answer generation module. In this paper, we describe how we implemented the second stage of this pipeline, passage re-ranking.

Method The job of the re-ranker is to estimate a score s_i of how relevant a candidate passage d_i is to a query q. We use BERT as our re-ranker. Using the same notation used by Devlin et al.

query : 64 个 token

query + 分隔符 + passage : 512 个 token

截断

(2018), we feed the query as sentence A and the passage text as sentence B. We truncate the query to have at most 64 tokens. We also truncate the passage text such that the concatenation of query, passage, and separator tokens have the maximum length of 512 tokens. We use a $\text{BERT}_{\text{LARGE}}$ model as a binary classification model, that is, we use the [CLS] vector as input to a single layer neural network to obtain the probability of the passage being relevant. We compute this probability for each passage independently and obtain the final list of passages by ranking them with respect to these probabilities.

passage的相关度分值.

单层 nn

bert

input

We start training from a pre-trained BERT model and fine-tune it to our re-ranking task using the cross-entropy loss:

候选文档j的相关分值

$$L = -\sum_{j \in J_{\text{pos}}} \log(s_j) - \sum_{j \in J_{\text{neg}}} \log(1 - s_j), \tag{1}$$

where J_{pos} is the set of indexes of the relevant passages and J_{neg} is the set of indexes of non-relevant passages in top-1,000 documents retrieved with BM25.

3 Experiments

We train and evaluate our models on two passage-ranking datasets, MS MARCO and TREC-CAR.

3.1 MS MARCO

The training set contains approximately 400M tuples of a query, relevant and non-relevant passages. The development set contains approximately 6,900 queries, each paired with the top 1,000 passages retrieved with BM25 from the MS MARCO corpus. On average, each query has one relevant passage. However, some have no relevant passage because the corpus was initially constructed by retrieving the top-10 passages from the Bing search engine and then annotated. Hence, some of the relevant passages might not be retrieved by BM25.

这6900个query中，有的query可能没有passage是相关的.

An evaluation set with approximately 6,800 queries and their top 1,000 retrieved passages without relevance annotations is also provided.

Training We fine-tune the model using TPUs[1] with a batch size of 32 (32 sequences * 512 tokens = 16,384 tokens/batch) for 400k iterations, which takes approximately 70 hours. This corresponds to training on 12.8M (400k * 32) query-passage pairs or *less than 2% of the full training set*. We could not see any improvement in the dev set when training for another 10 days, which equivalent to seeing 50M pairs in total. (使用全部的 train pair 和使用 2% 没看到区别，在 dev set 上)

We use ADAM (Kingma & Ba, 2014) with the initial learning rate set to 3×10^{-6}, $\beta_1 = 0.9$, $\beta_2 = 0.999$, L2 weight decay of 0.01, learning rate warmup over the first 10,000 steps, and linear decay of the learning rate. We use a dropout probability of 0.1 on all layers.

3.2 TREC-CAR

Introduced by Dietz et al. (2017), in this dataset, the input query is the concatenation of a Wikipedia article title with the title of one of its section. The relevant passages are the paragraphs within that section. The corpus consists of all of the English Wikipedia paragraphs, except the abstracts. The released dataset has five predefined folds, and we use the first four as a training set (approximately 3M queries), and the remaining as a validation set (approximately 700k queries). The test set is the same one used to evaluate the submissions to TREC-CAR 2017 (approx. 1,800 queries).

Although TREC-CAR 2017 organizers provide manual annotations for the test set, only the top five passages retrieved by the systems submitted to the competition have manual annotations. This means that true relevant passages are not annotated if they rank low. Hence, we evaluate using the automatic annotations, which provide relevance scores for all possible query-passage pairs.

Training We follow the same procedure described for the MS MARCO dataset to fine-tune our models on TREC-CAR. However, there is an important difference. The official pre-trained BERT

[1] https://cloud.google.com/tpu/

Method	MS MARCO MRR@10 Dev	MS MARCO MRR@10 Eval	TREC-CAR MAP Test
BM25 (Lucene, no tuning)	16.7	16.5	12.3
BM25 (Anserini, tuned)	-	-	15.3
Co-PACRR⋆ (MacAvaney et al., 2017)	-	-	14.8
KNRM (Xiong et al., 2017)	21.8	19.8	-
Conv-KNRM (Dai et al., 2018)	29.0	27.1	-
IRNet†	27.8	28.1	-
BERT Base	34.7	-	31.0
BERT Large	36.5	35.8	33.5

Table 1: Main Result on the passage re-ranking datasets. ⋆ Best Entry in the TREC-CAR 2017. † Previous SOTA in the MS MARCO leaderboard as of 01/04/2019; unpublished work.

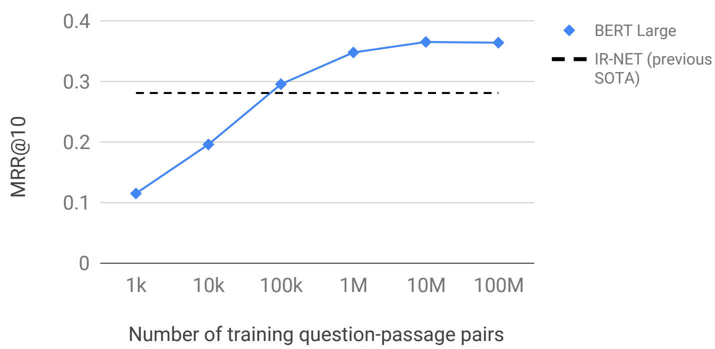

Figure 1: Number of MS MARCO examples seen during training vs. MRR@10 performance.

models[2] were pre-trained on the full Wikipedia, and therefore they have seen, although in an unsupervised way, Wikipedia documents that are used in the test set of TREC-CAR. Thus, to avoid this leak of test data into training, we pre-trained the BERT re-ranker only on the half of Wikipedia used by TREC-CAR's training set.

For the fine-tuning data, we generate our query-passage pairs by retrieving the top ten passages from the entire TREC-CAR corpus using BM25.[3] This means that we end up with 30M example pairs (3M queries * 10 passages/query) to train our model. We train it for 400k iterations, or 12.8M examples (400k iterations * 32 pairs/batch), which corresponds to only 40% of the training set. Similarly to MS MARCO experiments, we did not see any gain on the dev set by training the models longer.

3.3 Results

We show the main result in Table 1. Despite training on a fraction of the data available, the proposed BERT-based models surpass the previous state-of-the-art models by a large margin on both of the tasks.

Training size vs performance: We found that the pretrained models used in this work require few training examples from the end task to achieve a good performance 1. For example, a $\text{BERT}_{\text{LARGE}}$ trained on 100k question-passage pairs (less than 0.3% of the MS MARCO training data) is already 1.4 MRR@10 points better than the previous state-of-the-art, IR-NET.

[2] https://github.com/google-research/bert

[3] We use the Anserini toolkit (Yang et al., 2018) to index and retrieve the passages.

4 CONCLUSION

We have described a simple adaptation of BERT as a passage re-ranker that has become the state of the art on two different tasks, which are TREC-CAR and MS MARCO. We have made the code to reproduce our MS MARCO entry publicly available.

REFERENCES

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*, 2017.

Christopher Clark and Matt Gardner. Simple and effective multi-paragraph reading comprehension. *arXiv preprint arXiv:1710.10723*, 2017.

Zhuyun Dai, Chenyan Xiong, Jamie Callan, and Zhiyuan Liu. Convolutional neural networks for soft-matching n-grams in ad-hoc search. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pp. 126–134. ACM, 2018.

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.

Bhuwan Dhingra, Kathryn Mazaitis, and William W Cohen. Quasar: Datasets for question answering by search and reading. *arXiv preprint arXiv:1707.03904*, 2017.

Laura Dietz, Manisha Verma, Filip Radlinski, and Nick Craswell. Trec complex answer retrieval overview. TREC, 2017.

Matthew Dunn, Levent Sagun, Mike Higgins, V Ugur Guney, Volkan Cirik, and Kyunghyun Cho. Searchqa: A new q&a dataset augmented with context from a search engine. *arXiv preprint arXiv:1704.05179*, 2017.

Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pp. 55–64. ACM, 2016.

Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. Co-pacrr: A context-aware neural ir model for ad-hoc retrieval. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pp. 279–287. ACM, 2018.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*, 2017.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Jimmy Lin. The neural hype and comparisons against weak baseline. 2019.

Sean MacAvaney, Andrew Yates, and Kai Hui. Contextualized pacrr for complex answer retrieval. 2017.

Bhaskar Mitra, Fernando Diaz, and Nick Craswell. Learning to match using local and distributed representations of text for web search. In *Proceedings of the 26th International Conference on World Wide Web*, pp. 1291–1299. International World Wide Web Conferences Steering Committee, 2017.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*, 2016.

Matthew E Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. Semi-supervised sequence tagging with bidirectional language models. *arXiv preprint arXiv:1705.00108*, 2017.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/language-unsupervised/language_ understanding_paper. pdf*, 2018.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*, 2016.

Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. End-to-end neural ad-hoc ranking with kernel pooling. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 55–64. ACM, 2017.

Peilin Yang, Hui Fang, and Jimmy Lin. Anserini: Reproducible ranking baselines using lucene. *Journal of Data and Information Quality (JDIQ)*, 10(4):16, 2018.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*, 2018.