From doc2query to docTTTTTquery

Preprint · December 2019

2 authors, including:

Rodrigo Nogueira
University of Campinas
84 PUBLICATIONS **1,876** CITATIONS

From doc2query to docTTTTTquery

An MS MARCO passage retrieval task [1] μpublication

Rodrigo Nogueira[1] **and Jimmy Lin**[2]
[1] Epistemic AI
[2] David R. Cheriton School of Computer Science, University of Waterloo

Abstract: Nogueira et al. [7] used a simple sequence-to-sequence transformer [9] for document expansion. We replace the transformer with T5 [8] and observe large effectiveness gains.

Code and Data: `https://github.com/castorini/docTTTTTquery`

The idea behind doc2query [7], a form of document expansion, is to train a model, that when given an input document, generates questions that the document might answer. These predicted questions are then appended to the original documents, which are then indexed as before.

The setup in this work follows doc2query, but with T5 [8] as the expansion model. T5 is a sequence-to-sequence model that uses a similar pretraining objective as BERT [3] to pretrain its encoder-decoder architecture. In this model, all target tasks are cast as sequence-to-sequence tasks. In our case, we feed as input the passage and train the model to generate the question. We train the model with a constant learning rate of 10^{-4} for 4k iterations with batches of 256, which corresponds to 2 epochs with the MS MARCO training set. We use a maximum of 512 input tokens and 64 output tokens. In the MS MARCO dataset, none of the inputs or outputs have to be truncated when using these lengths. Similar to Nogueira et al. [7], we find that the top-k sampling decoder [4] produces more effective queries than beam search. We use $k = 10$. In all experiments, we use T5-base as we did not notice any improvement in retrieval effectiveness with the large model. We did not experiment with T5-3B and T5-11B due to their computational cost.

We use Google's TPU v3s to train and run inference. Training takes less than 1.5 hours on a single TPU. For inference, sampling 5 queries per document for 8.8M documents requires approximately 40 hours on a single TPU, costing $96 USD (40 hours $\times$ $2.40 USD/hour) using preemptible TPUs. Note that inference is trivially parallelizable and linear with respect to the number of samples.

All expanded documents are then indexed with the Anserini IR toolkit [10] (post-v0.6.0); the expanded queries are appended to the original documents, but not specially delimited. For evaluation, dev/test questions are issued against the index as "bag of words" queries, using the BM25 ranking function with Anserini's default parameters.

Table 1 shows results in terms of effectiveness on the dev and test sets as well as query latency. Latency for docTTTTTquery is the average time to retrieve 1000 documents per query on an Intel Xeon E5-2690v4 2.6GHz CPU "Broadwell" with 64GB of memory. The BM25 + BERT Large latency figures are copied from Nogueira et al. [7]. Since neural inference is applied prior to indexing, the increase in query latency is attributable solely to longer documents. Note that in a multi-stage reranking architecture, this represents the initial candidate generation stage; other techniques, such as mono/duoBERT [6], can be applied to further improve effectiveness. We have not done so (yet).

Table 1 also provides points of comparison: BM25 (Anserini) baseline; doc2query and BERT-based reranking (high score but very slow), both taken from Nogueira et al. [7]; Hofstätter et al. [5], which is, from what we can tell, the best non-ensemble, non-BERT method from the leaderboard with an associated paper; and DeepCT [2], a recently-introduced BERT-based document expansion method.

We also evaluate the queries produced by the models against the ground truth dev queries in terms of BLEU: docTTTTTquery scores 0.21 BLEU, which is much higher than doc2query, 0.088 BLEU. We attribute this large difference in output quality to pretraining and not to the size of the model

	MRR@10		R@1000	Latency
	Dev	Test	Dev	(ms/query)
BM25 (Anserini)	0.184	0.186	0.853	55
doc2query, top-k, 10 samples	0.218	0.215	0.891	61
docTTTTTquery, top-k, 5 samples	0.259	-	0.929	58
docTTTTTquery, top-k, 10 samples	0.265	-	0.939	61
docTTTTTquery, top-k, 20 samples	0.272	-	0.944	62
docTTTTTquery, top-k, 40 samples	0.277	0.272	0.947	64
docTTTTTquery, top-k, 80 samples	0.278	-	0.945	66
DeepCT [2]	0.243	0.239	0.913	55
Best non-ensemble, non-BERT [5]	0.290	0.277	-	-
BM25 + BERT Large [7]	0.375	0.368	0.853	3,500

Table 1: Main results on MS MARCO the passage retrieval task.

itself, as even the T5-small model, which has a similar number of parameters as the doc2query model, achieves 0.18 BLEU.

Interestingly, doc2query and docTTTTTquery produce similar proportions of copied (67%) and new words (33%) with respect to the original document. This analysis was performed for both models using 10 samples drawn from the top-k sampling decoder; stopwords are not considered when computing these statistics. As noted by Nogueira et al. [7], copying terms has the effect of term re-weighting while expanding with new terms mitigates the vocabulary mismatch problem, thus increasing recall.

Acknowledgments

We would like to thank Google Cloud for credits to support this research. Also, thanks to Wei (Victor) Yang and Kyunghyun Cho for helpful discussions.

History

[v1] December 2, 2019: First version.
[v2] December 7, 2019: This version. Added additional DeepCT numbers.

References

[1] P. Bajaj, D. Campos, N. Craswell, L. Deng, J. Gao, X. Liu, R. Majumder, A. McNamara, B. Mitra, T. Nguyen, et al. MS MARCO: A human generated MAchine Reading COmprehension dataset. *arXiv:1611.09268*, 2016.

[2] Z. Dai and J. Callan. Context-aware sentence/passage term importance estimation for first stage retrieval. *arXiv:1910.10687*, 2019.

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186, 2019.

[4] A. Fan, M. Lewis, and Y. Dauphin. Hierarchical neural story generation. *arXiv:1805.04833*, 2018.

[5] S. Hofstätter, N. Rekabsaz, C. Eickhoff, and A. Hanbury. On the effect of low-frequency terms on neural-IR models. *arXiv:1904.12683*, 2019.

[6] R. Nogueira, W. Yang, K. Cho, and J. Lin. Multi-stage document ranking with BERT. *arXiv:1910.14424*, 2019.

[7] R. Nogueira, W. Yang, J. Lin, and K. Cho. Document expansion by query prediction. *arXiv:1904.08375*, 2019.

[8] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv:1910.10683*, 2019.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[10] P. Yang, H. Fang, and J. Lin. Anserini: reproducible ranking baselines using Lucene. *Journal of Data and Information Quality*, 10(4):Article 16, 2018.