ICTNET at TREC 2019 Complex Answer Retrieval Track

Hongfei Ren[1,2], Ruibin Xiong[1,2], Yutao Zeng[1,2], Jiangui Chen[1,2], Yinqiong Cai[1,2], Haoquan Jiang[1,2]
[1]University of Chinese Academy of Sciences, Beijing, China
[2]CAS Key Lab of Network Data Science and Technology, Institute of Computing Technology
{renhongfei18s, xiongruibin18s, zengyutao18s}@ict.ac.cn
{chenjiangui18z, caiyinqiong18s, jianghaoquan18g}@ict.ac.cn

Abstract

We participate in the Complex Answer Retrieval(CAR) track at TREC 2019. We applied several useful models in this work. In the rough ranking, we applied doc2query model to predict queries and retrieve using BM25. In the re-ranking, we submitted 5 different runs which use 5 different models, include BM25, DRMMTKS, Bert-DRMMTKS, Bert-ConvKNRM and Bert-ConvKNRM-50, to try to get the best result.

1 Introduction

The goal of the TREC 2019 Complex Answer Retrieval is decribed as following: given a outline of a page (may be the title or hierarchical section headings), then we retrieve a ranking paragraphs for each section so that we can give a smooth passage. The complex query have many aspects, so we should give all related paragraph. For exam-ple, when we give a query 'Effects of Water Pol-lution', we should give the paragraph about fertil-izers, ocean acidification, and aquatic debris and the effects. In terms of dataset, it is based on the assumption that each Wikipedia page represents a complex topic, with further details under each sections. Over time, many different types of retrieval models have been proposed and tested. For rough sort, We have read many retrieval model, such as BM25-like method and the recently proposed language modeling approach. But the "vocabulary mismatch" problem has not been solved very well. So we used doc2query model. For precise sort,we hope to find the best model to solve this problem, so we used 4 model which are bert+convknrm, bert+drmmtks, bm25, drmmtks and gave 5 runs.

2 Model Description

2.1 Overview

In regard to rough sort, we used the doc2query model to predict the 5 most likely queries. we concate the predicted queries and the paragraph so that we can raise the recall. In the respect of precise sort, there are several different attempts. We give 5 runs which used different models. We want to select a best model from them. The first run used only bert, the second run used drmmts, the third run used bert+drmmtks, the forth run used bert+convknrm, the fifth run is a contradistinction to forth one. The last run set the return number of paragraphs of a section to 50.

2.2 Rough Ranking

For each document, our task is to predict several queries and then concate them. We use a Transformer model to produce the queries. For the optimizer, we choose Adam. The input is documents and target queries. The document and target query are tokenized with tokenizer and to avoid excessive memory usage, we truncate each document to 400 tokens and queries to 100 tokens. Our implementation use the OpenNMT framework. After the training, we use model to predict 5 most rele-vant query and concate them together. At the last, we use BM25 to retrieval a ranked list document for queries.

2.3 Re-ranking

After rough ranking, we obtains a list documents that may be relevant to the given query. We applied three main strategies to re-rank the documents.

2.3.1 DRMMTKS

DRMM was proposed by (Guo et al., 2016). It employs a joint deep architecture at the query term level for relevance matching. Its contains three main parts, which are matching histogram mapping, a feed forward matching network, and a term gating network. By utilizing those networks, DRMM can effectively deal with the exact match-ing signals, query term importance, and diverse matching requirements.

DRMMTKS(Yang et al., 2018) is an variant ver-sion of DRMM. In DRMMTKS, the matching his-togram layer is replaced by a sorted top-k pooling layer. Hence each query term is able to interacted with all the document terms to produce the query term-level interaction vector. Then, the model picks out the top-k signals by a sorted top-k pooling layer.

2.3.2 BERT

Because the effectiveness of BERT (Devlin et al., 2018), we use it to re-rank the documents. We first regards query and document as sentence A and sentence B, respectively. And then the concatentated representation is fed into BERT model. A learning-to-rank layer is added on the the representation of '[CLS]' in last layer to generate the finally matching score.

For every query and document pair in the dataset, we can gain a score that stands for the matching information between them. The document which has highest score is selected as the best relevant document to the query.

2.3.3 Conv-KNRM

Convolutional Kernel-based Neural Ranking Model (Conv-KNRM) was proposed by (Dai et al., 2018). Conv-KNRM outperformed prior neural IR methods and feature-based methods because of its precise design. Conv-KNRM first embed words into continuous vectors. Then a convolutional neural network is applied to compose embeddings of adjacent word. to n-gram embeddings. Soft-matching n-gram is subsequently utilized by the kernel pooling and learning-to-rank layers to genearte the final score.

Conv-KNRM is an end-to-end model and can be optimized from user feedback.

Same to Section 2.3.2, Conv-KNRM generates a list of ranking scores. And the scores are sorted by descent order to pick out the most relevant document.

3 Experiment

We conducted a series of experiments on TREC-Deep Complex Answer Retrieval datasets.

Data. The query is the concatenation of a Wikipedia article title with the title of one of its sections. The ground-truth documents are the paragraphs in that section. We removed repeated queries. For the dataset, we use "benchmarkY1-train.v2.0" to expand queries and retrieval.

Evaluate Metrics. We adopt two evaluation measures, i.e. MRR and NDCG, to evluate the performance of our proposed system.

Experimental Details. We first expand the documents using the proposed Doc2query method. Then we index the expanded documents. At the last, we use BM25 to retrieve top 1000 relevant passages for each query in train dataset. At training time, we construct the training set in the following way. The passages from qrels file are positive samples and negative samples are sampled from results of BM25. Concretely, each query has 100 negative samples, which are consisted of top 50 relevant passages of BM25 and 50 passages randomly sampled from the remaining 950 rele-vant passages. Because each query has 1.04 pos-itive samples on average, we will duplicate each positive sample twice for making full use of neg-ative samples. All our experiments are carried out using MatchZoo-py (Fan et al., 2017; Guo et al., 2019)[1]. With MatchZoo-py, we can easily use pairwise method for model training. A training pair is consisted of a query, a positive sample and four negative samples. For each training epoch, we will set to resample four negative samples for each positive sample. At prediction time, for im-proving performance, we only use top 50 or 100 relevant passages of BM25 to rerank.

We conducted five experiments on required dataset, named BM25, DRMMTKS, Bert-DRMMTKS, Bert-ConvKNRM and Bert-ConvKNRM-50 respectively.

BM25 directly uses the top 1000 relevant passages retrieved for each query in test dataset. The parameter of BM25 is set to b1=0.8, k=0.6.

DRMMTKS uses Glove (Pennington et al., 2014) as word embedding, and set to trainable during training time. We set top-k is 10, layers of MLP is 1, and hidden size is 6. The model is trained using Adadelta optimizer on a typical GPU, and batch size is 32. Initial learning rate is set to 1e-3.

[1]https://github.com/NTMC-Community/MatchZoo-py

The lr will decay with factor 0.9 after every three epoches.

Bert-DRMMTKS uses the sequence output of last layer in BERT as input. BERT will be fine-tuned with DRMMTKS end-to-end. The setting of DRMMTKS part is same as DRMMTKS run.

Bert-ConvKNRM uses the sequence output of last layer in BERT as input. BERT will be fine-tuned with Conv-KNRM end-to-end. For Conv-KNRM, we set max n-gram is 3, the number of Gaussian kernels is 11 and use 128 filters in each convolution layer. The model is trained using Adam optimizer on a typical GPU, and batch size is 64. Initial learning rate is set to 1e-3. The lr will decay with factor 0.9 after every three epoches.

Bert-ConvKNRM-50 uses the same model as Bert-ConvKNRM, but only use top 50 relevant passages of BM25 at prediction time.

4 Conclusion

In this work, we presented a query expansion approach, the expansion approach is better than baseline. Then we tried five approaches to improve Re-rank results.We did five comparison experiments and five results were submitted. The next step is to make further improvements based on the results returned.

References

[1] Zhuyun Dai, Chenyan Xiong, Jamie Callan, and Zhiyuan Liu. 2018. Convolutional neural networks for soft-matching n-grams in ad-hoc search. In Pro-ceedings of the eleventh ACM international confer-ence on web search and data mining, pages 126– 134. ACM.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understand-ing. arXiv preprint arXiv:1810.04805.

[3] Yixing Fan, Liang Pang, JianPeng Hou, Jiafeng Guo, Yanyan Lan, and Xueqi Cheng. 2017. Matchzoo: A toolkit for deep text matching.

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

[5] Jiafeng Guo, Yixing Fan, Xiang Ji, and Xueqi Cheng. 2019. Matchzoo: A learning, practicing, and devel-oping system for neural text matching. In Proceed-ings of the 42Nd International ACM SIGIR Confer-ence on Research and Development in Information Retrieval, SIGIR'19, pages 1297–1300, New York, NY, USA. ACM.

[6] Jeffrey Pennington, Richard Socher,and Christopher Manning. 2014. Glove: Global vectors for word representation. In Proceedings of the 2014 confer-ence on empirical methods in natural language pro-cessing (EMNLP), pages 1532–1543.

[7] Zhou Yang, Qingfeng Lan, Jiafeng Guo, Yixing Fan, Xiaofei Zhu, Yanyan Lan, Yue Wang, and Xueqi Cheng. 2018. A deep top-k relevance matching model for ad-hoc retrieval. In China Conference on Information Retrieval, pages 16–27. Springer.