IDST at TREC 2019 Deep Learning Track: Deep Cascade Ranking with Generation-based Document Expansion and Pre-trained Language Modeling

Ming Yan, Chenliang Li, Chen Wu, Bin Bi, Wei Wang, Jiangnan Xia, Luo Si
Alibaba Group, Hangzhou, China
{ym119608,lcl193798,wuchen.wc,b.bi,hebian.ww,jiangnan.xjn,luo.si}@alibaba-inc.com

ABSTRACT

This paper describes our participation in the passage and document ranking tasks of TREC 2019 Deep Learning Track. We propose a two-stage cascade ranking pipeline by taking the advantages of sequence-to-sequence generation and pre-trained language modeling. Firstly, we use a simple and effective index-based method to retrieve a collection of candidate passages. To overcome the vocabulary mismatch problem, we propose a query generation method for document expansion based on the pointer-generator model, where each passage is expanded with a set of generated queries for higher recall in the retrieval of candidate passages. Then we pre-train a BERT language model with a new sentence prediction objective, and adopt a pointwise ranking strategy for re-ranking the remained candidate passages. Our cascade ranking method achieves the best results among all participants on both the passage ranking and document ranking tasks, according to the official evaluation metric NDCG@10.

KEYWORDS

cascade ranking, pre-trained language model, document expansion, sequence-to-sequence generation

ACM Reference Format:
Ming Yan, Chenliang Li, Chen Wu, Bin Bi, Wei Wang, Jiangnan Xia, Luo Si. 2020. IDST at TREC 2019 Deep Learning Track: Deep Cascade Ranking with Generation-based Document Expansion and Pre-trained Language Modeling . In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The Deep Learning Track is a new track first run in TREC 2019, which aims at studying information retrieval in a large training data regime. It consists of two tasks: passage ranking and document ranking. Both tasks use a large human-generated set of training labels, from the MS-MARCO [1] dataset. The passage ranking task focuses on ranking passages, where it contains 1,010,916 queries on a collection of 8,841,823 passages. The document ranking task is based on source documents, which contains passages in the passage ranking task. The full corpus is 3,213,835 documents and the training set has 367,013 queries. For both tasks, there are two subtasks related to this: full ranking and top-k re-ranking. In the full ranking subtask, the aim is to rank passages/documents directly from the full document collection provided, while in the re-ranking subtask we re-rank given an initial ranking set of top-k passages/documents. For the official evaluation, it will be using depth pooling and construct separate pools for the passage ranking and document ranking tasks. Passages/documents in these pools will then be labelled by NIST assessors using multi-graded judgments, allowing to measure the NDCG metric.

Our approach is mainly based on the BERT language model [2], which is a state-of-the-art model in various natural language understanding tasks. Different from many other ranking methods which directly finetune the original BERT model in downstream task, we modify the next sentence prediction task in BERT to a 3-class sentence classification as in StructBERT [9] and pre-train a new BERT language model from scratch. Based on the new pre-trained BERT model, we further finetune it with pointwise ranking strategy on the labeled query-passage data for re-ranking. Moreover, for full ranking subtask, we use a simple and effective BM25 method [6] to retrieve a collection of candidate passages, by leveraging the document expansion technique. Prior to indexing, we propose a query generation method for document expansion based on the pointer-generator model, where each document is expanded with a set of generated queries for improved retrieval. For both the passage and document ranking tasks, we train with the passage-level labeled data. For evaluating on document ranking, we split a document into overlapping passages. The BERT ranker predicts the relevance of each passage independently, and the document score is calculated as the maximum score of all the passages within the document. We tested and submitted runs for both the passage ranking and document ranking tasks, by combining the advantages of pre-trained language model and document expansion technique. The results shows that our method outperforms all the submission runs of other participants on both tasks, in terms of the official evaluation metric NDCG@10, which validates the effectiveness of our cascade ranking framework.

The remainder of the paper is organized as follows. Section 2 outlines our approach. The experiment results and analysis are given in Section 3. Finally, Section 4 concludes the work.

2 OUR APPROACH

This section presents our two-stage cascade ranking pipeline. An off-the-shelf BM25 retriever is first used to efficiently retrieve a

[1] http://www.msmarco.org/

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

collection of candidate passages/documents. To overcome the vocabulary mismatch problem, prior to indexing, we generate a set of queries for each passage/document to conduct document expansion. In the second stage, we further leverage the state-of-the-art BERT language model to re-rank the candidate passages/documents. Besides, The BERT model is pre-trained with a new sentence prediction objective for better modeling the sentence structure information. Since the passage ranking and document ranking tasks share the same collection of text data with passage-level training labels, we address both tasks in a passage level.

2.1 Document Expansion-based Retriever

For the retriever stage, we first expand each passage with a set of generated queries using sequence-to-sequence generation method. Then we build index on the collection of expanded passages, and use a simple and effective BM25 method to retrieve the top-k candidate passages.

As for document expansion, for each passage $\mathbf{p} = \{x_1, \cdots, x_N\} \in \mathcal{P}$, we aim to predict a set of queries $Q^{gen} = \{\mathbf{q}_1^{gen}, \cdots, \mathbf{q}_L^{gen}\}$ for which that passage will be relevant, where $\mathbf{q}^{gen} = \{q_1, \cdots, q_M\}$ and M is the total number of generated query words. We first extract a total collection of query-relevant passage pairs from the labeled training corpus, and use them to train an encoder-decoder network for query generation from the passage. The proposed model is based on the pointer-generator model [7], which is widely used in abstractive text summarization. The tokens of the input passage $\mathbf{p} = \{x_1, \cdots, x_N\}$ are fed into the passage encoder, which maps the text into a sequence of encoder hidden states $\{h_1, \cdots, h_N\}$. At decoding time, the decoder will sequentially generate query words by attending on the passage hidden states. At each decoding step, the attention distributions e_{ti} of encoder states and the context vector c_t are given as:

$$e_{ti} = v^T tanh(W_h h_i + W_s s_t) \tag{1}$$

$$c_t = \sum_{i=1}^{N} \alpha_{ti}^e h_i, \quad \alpha_t^e = softmax(e_t) \tag{2}$$

where v, W_h, W_s are trainable parameters, s_t is the decoder hidden state at time step t.

Traditional attention mechanism just calculates the attention distribution of the encoder hidden states but ignores the decoder hidden states. To better distinguish among generated words in the query, we propose a novel Attention Over Attention (AOA) mechanism to consider both the attention distributions of encoder hidden states and the previous decoder hidden states. Specifically, at each decoding time step t, we further calculate a new decoder hidden state $\bar{s}_t$ by attending on the previous decoder hidden states as:

$$d_{ti} = u^T tanh(W_d s_i + W_c c_t) \tag{3}$$

$$\bar{s}_t = \sum_{i=1}^{t} \alpha_{ti}^d s_i, \quad \alpha_t^d = softmax(d_t) \tag{4}$$

where u, W_d, W_c are trainable parameters.

Then, the new decoder state $\bar{s}_t$ is concatenated with the context vector c_t , and further fed through two linear layers to produce the vocabulary distribution $P_{vocab}(w)$ over all words in the vocabulary:

$$P_{vocab}(w) = softmax(f([\bar{s}_t, c_t])) \tag{5}$$

where $f(\cdot)$ is two linear layers.

Following [7], we also use a soft switch to choose between generating a word from the vocabulary or copying a word from the input sequence, and calculate the final probability distribution over the extended vocabulary as:

$$P(w) = p_{gen} P_{vocab}(w) + (1 - p_{gen}) \sum_{i:w_i=w} \alpha_{ti}^e \tag{6}$$

where p_{gen} is a soft switch used in [7].

During training, we minimize a maximum-likelihood loss, which is most widely used in generation tasks. We define q_t^* as the target word for the decoding time step t and the overall loss is:

$$\mathcal{L} = -\frac{1}{T} \sum_{t=0}^{T} logP(q_t^* | q_1^*, ..., q_{t-1}^*, \mathbf{p}) \tag{7}$$

After the query generation model is trained, for each passage in $\mathcal{P}$, we generate the top-L queries $Q^{gen} = \{\mathbf{q}_1^{gen}, \cdots, \mathbf{q}_L^{gen}\}$ using beam search [10], which are used for passage expansion.

Next, we append the generated top-L queries Q^{gen} to each original passage, respectively. The expanded passages are then indexed and we adopt the simple and effective term-based BM25 method [6] for retrieval by considering the term weight. For the BM25 search engine, we use the off-the-shelf Anserini open-source IR toolkit [11].

2.2 BERT-based Re-ranker

With the top-k candidate passages from retriever stage, we further take the BERT language model to re-rank the candidate passages for final ranking. The pre-trained BERT model is expected to capture certain semantic relevance between query and passage. Inspired from StructBERT [9], we first modify the next sentence prediction task of original BERT into a more difficult previous sentence and next sentence prediction task, and pre-train a modified BERT language model from scratch using the new sentence prediction task. Then, we finetune the pre-trained BERT model on labeled query-passage data with a point-wise ranking strategy.

BERT [2] is a self-supervised approach for pre-training a deep transformer encoder [8], before fine-tuning it for a particular downstream task. Given a single text sentence or a pair of text sentences, BERT packs them in one token sequence and learns a contextualized vector representation for each token. The input representations are fed into a stack of multi-layer bidirectional Transformer blocks, which uses self-attention to compute the text representations by considering the whole input sequence.

There are two training objectives in the original BERT model – masked language modeling (MLM) and next sentence prediction (NSP), where one is at token level and the other at sentence level. One of the key points of BERT lies in how to design more appropriate pre-training objectives with the unlabeled text. Recently, some variants [4, 12] of BERT language model found that the NSP task may be harmful to learn a effective language model compared with MLM task. They remove the NSP task when pre-training the language model. Some others found it beneficial to modify the NSP task and design a more difficult NSP task for pre-training the language model [3, 9]. In our search scenario which involves modeling the

relevance between the query and passage, we think it is beneficial to keep the NSP task and design a more difficult one to capture certain sentence structure information. Therefore, we follow the line of StructBERT and adopt the more difficult previous sentence and next sentence prediction task. Specifically, given a segment $\mathbf{S}_1$ from the unlabeled corpus as segment 1, $\frac{1}{3}$ of the time we choose the actual next text span $\mathbf{S}_2$ as segment 2, $\frac{1}{3}$ of the time we switch the positions of $\mathbf{S}_1$ and $\mathbf{S}_2$ as input, and the final $\frac{1}{3}$ of the time we randomly sample a segment $\mathbf{S}_{rand}$ from the other documents as segment 2. The new task is to distinguish among the three situations with a three-class classification problem. The MLM task is kept the same as in the original BERT. We jointly train the token-level MLM task and new NSP task on the unlabeled data, which is the same data as in the original BERT pre-training.

After the new BERT language model is pre-trained from scratch, we follow the method in [5] and treat the re-ranking task as a binary classification problem and use the [CLS] vector in the final layer of new BERT to compute a score s_i for each passage. The final list of passages are ranked by the score s_i. In both the BERT pre-training and fine-tuning, we use the 12-layer BERT base architecture ($L = 12, H = 768, A = 12$), the max sequence length is set at 384.

We start training from the pre-trained BERT base model, and fine-tune it with the pointwise objective as:

$$\mathcal{L} = -\sum_{i=1}^{N} y_i \cdot log(s_i) + (1 - y_i) \cdot log(1 - s_i) \tag{8}$$

where $s_i = sigmoid(\mathbf{w} \cdot \mathbf{h}_{CLS}^L)$, $y_i \in \{0, 1\}$ is the ground-truth label of the query-passage pair, $\mathbf{w}$ is a trainable parameter and $\mathbf{h}_{CLS}$ is the hidden state of [CLS] token in the final layer of BERT.

2.3 Re-ranker Ensemble

Since the test set is rather small which contains no more than 200 examples, we train multiple BERT re-rankers for ensemble learning. For each BERT re-ranker, we output a probability score s_i for each query-candidate passage pair. The probability score is accumulated with different BERT re-rankers, and the sum of the probability scores is used to calculate the final rankings. For the ensemble method, we just use a simple ensemble of the same model training with 4~8 different random seeds.

2.4 Document Ranking

Applying BERT to long documents causes increasing memory usage and run time due to the complexity in interacting every pair of tokens. Therefore, we adopt a simple passage-level approach for document retrieval. Since the document ranking and passage ranking tasks share the same text corpus, we directly use the model trained on the passage ranking task for document retrieval. When evaluating and predicting, we split the document into overlapping passages with the same maximum length of 384 and doc stride of 192. The BERT re-ranker predicts the relevance of each passage with respect to the query independently, and the document score is calculated as the maximum relevance of all the passages within the document. Besides, the title information is important in document retrieval. Therefore, during the retriever and re-ranker stages, we both add the title to the beginning of every splitted passage to provide context.

3 EXPERIMENT

This section we presents the results of our runs in both the passage ranking and document ranking tasks. In total, there are 37 passage task runs and 38 document task runs from all the 15 teams. In each task, there are two subtasks: full ranking and re-ranking. The main official metric in both tasks is NDCG@10, since it makes use of the 4-level judgments and focuses on the first results that users will see. The details of task construction, evaluation methods and result analysis can be found in the overview paper of the TREC 2019 Deep Learning Track [1]. From the statistics, we can see that in most of our runs, our results can obtain the best performance among all the runs, especially for passage ranking task.

3.1 Passage Ranking Task Performance

Table 1 presents the results of our submitted runs and several other top competitive runs in passage ranking task. The last four runs listed are top performing runs of other groups and the two runs without run tags are single model ones which are self-evaluated with official tools without submitting. We can see that:

- Our submitted runs can obtain superior performance compared with other competitive runs, which shows the effectiveness of our deep cascade ranking framework by leveraging BERT pre-trained language model.
- Our results of full ranking style are much higher than the re-ranking style on all metrics, which demonstrates the effectiveness of our document expansion-based retriever by leveraging sequence-to-sequence modeling.
- Model ensembling can help improve the ranking performance, but not that significant.

3.2 Document Ranking Task Performance

Table 2 presents the results of our submitted runs and several other top competitive runs in document ranking task. To test the effectiveness of different components in our method, we submit the first three full ranking runs with different settings, i.e. 1) the method without adding generation-based document expansion, 2) the method with generation-based document expansion, and 3) the method with generation-based document expansion, but only recall half amount of passages in the first document expansion-based retriever stage. For comparison, the last four runs listed are top performing runs of other groups and the two runs without run tags are single model ones which are self-evaluated with official tools without submitting. We can see that:

- By directly adopting a passage-level approach for document retrieval, our method can still obtain superior performance compared with other competitive runs, which also shows the effectiveness of our method to deal with the ranking of long documents.
- Our full ranking method can largely improve the results of metrics AP and Recall@100 than the re-ranking method, but can bring little improvement in terms of metric NDCG@10. It may be due to that NDCG@10 focuses on the top-ranked results, but the revised retriever (passage-level) in full ranking method of document ranking task mainly helps to improve the recall of “wider" document candidates. Besides, the generation-based

Table 1: Overall ranking performance of submitted runs of our group and other top competitive runs in passage ranking task.

Run Tag	Group	Run Description	Subtask	NDCG@10	AP	R@1000
idst_bert_p1	IDST	Ensemble 6 models	Full Ranking	**0.764**	0.503	0.835
idst_bert_p2	IDST	Ensemble 4 models	Full Ranking	0.763	0.504	**0.844**
idst_bert_p3	IDST	Ensemble 8 models	Full Ranking	0.759	**0.505**	**0.844**
idst_bert_pr1	IDST	Ensemble 6 models	Re-Ranking	0.738	0.457	0.694
idst_bert_pr2	IDST	Ensemble 8 models	Re-Ranking	0.738	0.457	0.694
-	IDST	Single model	Full Ranking	0.750	0.501	0.838
-	IDST	Single model	Re-Ranking	0.725	0.453	0.690
p_exp_rm3_bert	h2oloo	-	Full Ranking	0.742	**0.505**	0.806
test1	Brown	-	Re-Ranking	0.731	0.457	0.694
TUA1-1	TUA1	-	Re-Ranking	0.731	0.457	0.694
TUW19-p3-f	TU-Vienna	-	Full Ranking	0.688	0.420	0.739

Table 2: Overall ranking performance of submitted runs of our group and other top competitive runs in document ranking task.

Run Tag	Group	Run Description	Subtask	NDCG@10	AP	R@100
idst_bert_v1	IDST	No document expansion (8 ensemble)	Full Ranking	0.718	0.383	0.419
idst_bert_v2	IDST	With document expansion (8 ensemble)	Full Ranking	0.718	0.385	0.430
idst_bert_v3	IDST	Only recall half amount (8 ensemble)	Full Ranking	**0.726**	0.368	0.422
idst_bert_r1	IDST	Ensemble 8 models	Re-Ranking	0.719	0.291	0.387
idst_bert_r2	IDST	Ensemble 4 models	Re-Ranking	0.714	0.291	0.387
-	IDST	Single model (w/ document expansion)	Full Ranking	0.706	0.378	0.423
-	IDST	Single model (w/ document expansion)	Re-Ranking	0.704	0.284	0.382
bm25exp_marcomb	h2oloo	-	Full Ranking	0.646	**0.424**	**0.467**
TUW19-d3-re	TU-Vienna	-	Re-Ranking	0.644	0.271	0.387
ucas_runid1	UCAS	-	Re-Ranking	0.644	0.264	0.387
ms_ensemble	Microsoft	-	Full Ranking	0.578	0.237	0.368

document expansion makes not that much difference compared with passage task (idst_bert_v1 v.s. idst_bert_v2).

- Recalling less passages/documents in retriever stage gives the best result in terms of NDCG@10, but decreases the AP and Recall@100 metrics slightly. In this setting, more possible positive candidates are discarded with less passages/documents remained in retriever stage.

4 CONCLUSION

In this paper, we propose an effective cascade ranking framework for ad-hoc passage and document retrieval. Firstly, we propose to leverage a sequence-to-sequence generation method to conduct document expansion, which helps to retain a higher recall of the candidate passages from the whole passage collection. Then, we design a new pre-trained BERT language model for re-ranking, by enriched with more fine-grained sentence structure information. The experiment results show that our method can obtain superior performance compared with other competitive submission runs on both the passage ranking and document ranking tasks.

REFERENCES

[1] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2019. Overview of the TREC 2019 deep learning track. In *TREC (to appear)*.

[2] 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).

[3] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A lite BERT for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942* (2019).

[4] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).

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

[6] Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends® in Information Retrieval* 3, 4 (2009), 333–389.

[7] Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368* (2017).

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[9] Wei Wang, Bin Bi, Ming Yan, Chen Wu, Zuyi Bao, Liwei Peng, and Luo Si. 2019. StructBERT: Incorporating Language Structures into Pre-training for Deep Language Understanding. *arXiv preprint arXiv:1908.04577* (2019).

[10] Sam Wiseman and Alexander M Rush. 2016. Sequence-to-sequence learning as beam-search optimization. *arXiv preprint arXiv:1606.02960* (2016).

[11] Peilin Yang, Hui Fang, and Jimmy Lin. 2017. Anserini: Enabling the use of Lucene for information retrieval research. In *Proceedings of the 40th International ACM SIGIR Conference*. ACM, 1253–1256.

[12] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. *arXiv preprint arXiv:1906.08237* (2019).