TABLE: A Task-Adaptive BERT-based ListwisE Ranking Model for Document Retrieval

Xingwu Sun[1*],Hongyin Tang[2,3],Fuzheng Zhang[1],Yanling Cui[2,3],Beihong Jin[2,3*],Zhongyuan Wang[1]
[1]Meituan Dianping
[2]State Key Laboratory of Computer Sciences, Institute of Software, Chinese Academy of Sciences
[3]University of Chinese Academy of Sciences, Beijing, China
[*]The corresponding authors: sunxingwu01@gmail.com, beihong@iscas.ac.cn

ABSTRACT

Document retrieval (DR) is a crucial task in NLP. Recently, the pre-trained BERT-like language models have achieved remarkable success, obtaining a state-of-the-art result in DR. In this paper, we come up with a new BERT-based ranking model for DR task, named TABLE. In the pre-training stage of TABLE, we present a domain-adaptive strategy. More essentially, in the fine-tuning stage, we develop a two-phase task-adaptive process, i.e., type-adaptive pointwise fine-tuning and listwise fine-tuning. In the type-adaptive pointwise fine-tuning phase, the model can learn different matching patterns regarding different query types. In the listwise fine-tuning phase, the model matches documents with regard to a given query in a listwise fashion. This task-adaptive process makes the model more robust. In addition, a simple but effective exact matching feature is introduced in fine-tuning, which can effectively compute matching of out-of-vocabulary (OOV) words between a query and a document. As far as we know, we are the first who propose a listwise ranking model with BERT. This work can explore rich matching features between queries and documents. Therefore it substantially improves model performance in DR. Notably, our TABLE model shows excellent performance on the MS MARCO leaderboard.

KEYWORDS

Neural Information Retrieval, Document Retrieval, BERT

ACM Reference Format:
Xingwu Sun, Hongyin Tang, Fuzheng Zhang,Yanling Cui, Beihong Jin, Zhongyuan Wang. 2020. TABLE: A Task-Adaptive BERT-based ListwisE Ranking Model for Document Retrieval. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20), Oct. 19–23, 2020, Virtual Event, Ireland.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3340531.3412071

1 INTRODUCTION

Document Retrieval (DR) requires the machine to retrieve and rank documents according to their relevance with the query which needs strong text understanding ability. As one basic and crucial task in NLP, it can aid several real applications, such as question answering systems and Web-based search engines, e.g., Google, Yahoo, Bing, etc. With the development of deep learning and the increasing emergence of large-scale datasets, e.g., MS MARCO [11], DR has achieved remarkable advancements.

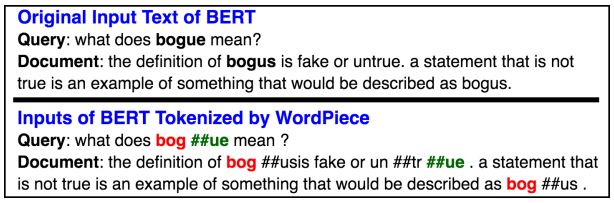

Figure 1: A case in baseline BERT model. The token "bogue" in the query is unrelated with "bogus" and "untrue" in the document. But WordPiece tokenization generates exact matching words, i.e., "bog" and "##ue", which induces a high relevance score.

Lots of DR models based on machine learning have been studied over the last few years. Traditional machine learning based DR models, like LambdaRank[2], AdaRank[20], etc., rely heavily on manual feature engineering which is time-consuming and unsustainable. Neural DR models, including DSSM[8], KNRM[19], etc., learn the query and document representation and ranking features in a continuous vector space, which obviate the need of manual feature design. Formally, ranking models can be divided into three categories: pointwise, pairwise and listwise. It has been reported that listwise models perform comparatively better in ranking tasks[3, 14].

Recently, the pre-trained language models have caused a stir in DR. These models are pre-trained on unlabeled large-scale corpora and then applied to DR in either a feature-based or a fine-tuning manner[12, 13]. Both offer substantial improvements. Currently, BERT[5] is undoubtedly one of the most successful pre-trained language models, which employs deep transformers to enhance language understanding from large-scale texts, obtaining state-of-the-art results in a wide variety of NLP tasks, including DR.

The BERT model, despite its powerfulness, could be further improved for DR in the following aspects: (1) Current BERT-based ranking models are either pointwise or pairwise style. Their training targets are to optimize the relevance of query and document or the order of documents within pairs rather than minimizing errors in ranking of documents. (2) Besides that, the model could be more task-adaptive by considering text domains and query types. (3) The matching of OOV word could be miscalculated due to WordPiece[5] segmentation method. Fig. 1 gives an example that the query and the document are irrelevant because the word "bogue" in the query and the word "bogus" in the document are unrelated. However,

CIKM '20, October 19–23, 2020, Virtual Event, Ireland

ACM ISBN 978-1-4503-6859-9/20/10…$15.00
https://doi.org/10.1145/3340531.3412071

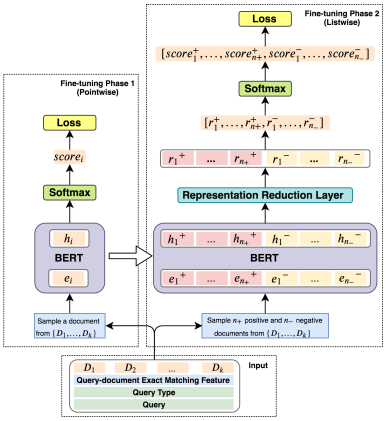

Figure 2: Architecture of the two-phase fine-tuning of the TABLE model. D_*^+ and D_*^- represent positive and negative documents, respectively.

BERT fails to distinguish the two words as a result of being misled by the separative terms generated by WordPiece, i.e., "bog".

In order to solve the above-mentioned problems of BERT-based ranking models, we propose a new DR model which we call Task-Adaptive and BERT-based ListwisE (TABLE) ranking model. In the TABLE model, we make the following improvements on the basis of BERT: (1) As depicted in Fig. 2, we construct a BERT-based listwise method to learn document-level comparison with regard to a given query in fine-tuning. (2) We present a domain-adaptive pre-training process and propose a query type-adaptive fine-tuning strategy to adapt the model to this DR task. (3) We add an exact matching feature in the input layer of BERT, which can effectively improve the matching accuracy of OOV words.

We conduct extensive experiments on MS MARCO, which is a large-scale benchmark from search engine Bing. In MS MARCO, all queries are sampled from real search queries and documents are real Web documents. With one million queries, it is one of the most comprehensive real-world datasets of its kind in both quantity and quality. Experimental results show that our TABLE model achieves excellent performance.

Our contributions can be summarized as follows.

- We present a new BERT-based model for DR task, in which we give a listwise fine-tuning strategy. As far as we know, we are the first who propose a listwise ranking model with BERT. This work can explore rich matching features between queries and documents. As a result, it substantially improves model performance in DR.
- We propose a domain-adaptive pre-training process and a query type-adaptive fine-tuning method to make the model more task-adaptive and robust in DR.
- We introduce an effective exact matching feature. It effectively improves the matching performance of OOV words.
- Our TABLE model outperforms many strong baselines and wins the second place on the MS MARCO leaderboard.

2 RELATED WORK

Learning to rank refers to adopting machine learning algorithms to train models for ranking tasks. These ranking models can be employed in a wide variety of applications. While applied to the DR tasks, ranking models output a ranked document list for each query based on relevance scores computed by the models. Depending on how the loss functions are defined, we can categorize learning to rank into three classes, namely pointwise, pairwise and listwise.

Pointwise approaches [6, 15] transform ranking tasks to classification or regression tasks by directly predicting the relevance score of a document with respect to a given query. Although easy to implement, pointwise approaches do not consider the relation with other documents which is the core of ranking.

In pairwise approaches [1, 2, 17], ranking is transformed into classification on the order of document pairs. In particular, each document is compared to the other one at a time according to their relevance with the query. The final ranked list is arranged by their relative positions. However, the assumption that these approaches require, i.e., the documents are generated in pairs, is too strong. Even worse, the uneven distribution of documents in different queries can cause training bias.

Further, the listwise approaches [3, 14, 16, 18, 20] train the models to learn the optimal ranking directly by taking the documents to be ranked as input. [3] treats the sequence ordered by the top-one probability distribution of documents as the ranking list and optimizes the cross entropy. [18] maximizes the likelihood of the golden ranking list. [14] optimizes the similarity of the query and the documents. [16, 20] directly optimizes the metrics like NDCG [9] which measure the quality of a ranking list.

On the other hand, most of the above methods rely heavily on handcrafted features which require much expertise. As the rapid developing of deep learning techniques, lots of work [7] tempts to build neural ranking models which need no manual features and show impressive performance.

Recently, the pre-trained language models like BERT [5] have achieved the state-of-the-art results on several NLP tasks. Leveraging their powerful language understanding abilities, [12, 13] propose ranking models built on BERT which outperform other neural models by salient margins. [12] adopts a pointwise paradigm and [13] adopts a pairwise one. However, their training targets are minimizing errors in classification of query-document relevance or the order of document pairs rather than minimizing errors in ranking of documents, which restricts their performance.

3 OUR APPROACH

First of all, we formally describe the task definition as follows. Given a query Q and a very large set of documents D^*, our goal is to produce a ranked list of documents, which is as close as possible to the oracle ranking of documents according to their relevance levels, i.e., $y \in \{0, 1\}$, where 1 indicates positive sample and 0 means negative one.

Basically, the pipeline of DR contains two stages: a retrieval stage and a re-ranking stage. The retrieval stage is to get a smaller set of document candidates for a query, which is an effective way to balance the performance and costs. In this work, we apply BM25 (DeepCT-Index) [4] to get the top-k document candidates $D = \{D_1, D_2, \cdots, D_k\}$ for the query Q. Readers can refer to [4] for more details. Here, we focus on the re-ranking stage.

In the following subsections, we describe our ranking model TABLE. TABLE is pre-trained to be domain-adaptive, and then fine-tuned in a two-phase paradigm.

3.1 Domain-adaptive Pre-training

TABLE is based on BERT. As we know, BERT is usually pre-trained on the open corpora, which may be inconsistent with the domains of the target task. Therefore, it is necessary to adapt BERT to the task domains. Specifically, we employ MS MARCO as input and pre-train BERT for the DR task while keeping its loss function unchanged, i.e., the masked language model loss and the next sentence prediction loss. For more details, please refer to [5].

3.2 Two-phase Fine-tuning

Type-adaptive Pointwise Fine-tuning Phase Considering that the matching patterns between query and document are closely related to the query type, the query type should be involved in the fine-tuning. In MS MARCO, each query is manually labeled with its type, i.e., location, numeric, person, description, entity. Therefore, the first phase fine-tuning aims to learn different matching patterns regarding different query types by predicting the query-document relation in a pointwise fashion. Using the query-document pairs in MS MARCO as input, the BERT is fine-tuned to conduct the query-document matching task. Here, we model the query, the query type and the document using BERT to compute a deep inter-representation. Specifically, we first concatenate the query type T, the query Q and the i-th document D_i as one sequence:

$$X_i = [< CLS >, T, < SEP >, Q, < SEP > D_i] \tag{1}$$

where <SEP> is the separator and <CLS> indicates the position for query-document relation representation. Next, for the j-th token x_i^j in sequence X_i, the embedding e_i^j can be computed by:

$$e_i^j = {e_{tok}}_i^j + {e_{seg}}_i^j + {e_{pos}}_i^j \tag{2}$$

where ${e_{tok}}_i^j$, ${e_{seg}}_i^j$ and ${e_{pos}}_i^j$ are the token embedding, segmentation embedding and position embedding of x_i^j, respectively. Then, we apply BERT to obtain inter-representation of each token in X_i, i.e., the hidden state in the final hidden layer of BERT:

$$h_i = \text{BERT}(e_i) \tag{3}$$

in which the hidden state corresponding to the <CLS> position can be used to calculate the query-document relation score:

$$score_i = \text{Softmax}(h_i^0) \tag{4}$$

We use cross entropy as the loss function. After the first phase of fine-tuning, the model learns different matching patterns regarding different query types, and turns towards type adaptive.

Listwise Fine-tuning Phase The second phase fine-tuning is a listwise paradigm, which enables the model to learn the document-level ranking features in a listwise manner and to minimize errors in the ranking of documents during training.

In detail, for each query, we employ a document set composed of n_+ positive samples and n_- negative samples as input. This document set is randomly selected from D. To be aware, due to hardware limitation, we do not take all candidate documents in D as input, which is the same as common listwise models.

As described in the last subsection, we first apply BERT to compute the deep relation representation between the query and each document. Here, the i-th positive document can be computed by:

$${h_i}^+ = \text{BERT}({e_i}^+) \tag{5}$$

And so does the negative one, as shown in Eq. 6:

$${h_i}^- = \text{BERT}({e_i}^-) \tag{6}$$

Next, we use a representation reduction layer to convert each query-document relation representation to a one-dimension vector, i.e., query-document relation score. The representation reduction layer is a single-layer perceptron:

$$r_i^+ = W h_i^{0+} + b; r_i^- = W h_i^{0-} + b \tag{7}$$

where W and b are trainable parameters, h_i^{0+} and h_i^{0-} are the hidden states corresponding to the <CLS> positions for the i-th positive and negetive input respectively, r_i^+ and r_i^- are unnormalized query-document relation scores.

Then, we normalize the query-document relation scores by a softmax function:

$$score_i^+ = \frac{e^{r_i^+}}{\sum_{j=1}^{n_+} e^{r_j^+} + \sum_{j=1}^{n_-} e^{r_j^-}} \tag{8}$$

where $score_i^+$ is the normalized query-document relation score of the i-th positive document after document-level comparison.

Till now, we can get a ranked list of documents according to their relation score with the query. Then, we use supervision information to guide the optimization of measuring document rankings. With only positive and negative label, the optimizing of the ranking can be regarded as maximizing the normalized score of the positive samples. To this end, we use the averaged negative log likelihood of all positive sample scores to calculate the loss during training:

$$loss = \frac{\sum_{i=1}^{n_+} -log(score_i^+)}{n_+} \tag{9}$$

3.3 Exact Matching of OOV Words

The exact matching word refers to a word which occurs both in the query and in the document. Exact matching is one of the most important requirements in DR and machine reading comprehension. However, in practice, since the word vocabulary is limited, the OOV words are either deleted directly or segmented using WordPiece. As a result, the exact matching of OOV words is not dealt with properly. As the example Fig. 1 shows, the query and the document are irrelevant because the word "bogue" in the query and the word "bogus" in the document are unrelated. But BERT fails in this case as a result of matching words generated by WordPiece tokenization, i.e., "bog".

To avoid such traps, we propose the exact matching strategy in TABLE. Specifically, at the fine-tuning stage, we bind an exact matching feature for each word. The value of the feature is determined by whether the word appears in both the query and the document before WordPiece segmentation. Then, the value is passed on to the tokens after WordPiece segmentation. Further, we use this exact matching feature as an extra input in the fine-tuning stage, and then we can encode the exact matching information before the calculation of the inter-representation. Now, the embedding of the j-th token in X_i will be calculated as follows.

$$e_i^j = {e_{tok}}_i^j + {e_{seg}}_i^j + {e_{pos}}_i^j + \alpha {e_{em}}_i^j \quad (10)$$

where em denotes exact matching, ${e_{em}}_i^j$ is the exact matching feature embedding of x_i^j, α is a hyperparameter to balance the importance of this feature, specified at the beginning of fine-tuning.

4 EXPERIMENTS

4.1 Dataset and Metric

We evaluate our model on the MS MARCO dataset, which is a large-scale benchmark from search engine Bing. With over 1M queries, it is one of the most comprehensive real-world dataset in both quantity and quality. We focus on the document retrieval task with a corpus of 8.8M paragraphs extracted from 3.6M Web documents. The goal of the system is to retrieve and rank paragraphs that can answer the query. We refer to these basic units of ranking as "documents" to maintain terminological consistency throughout this paper. Evaluation is performed by submitting the ranking results to the online leaderboard and the official metric is MRR@10.

4.2 Model Settings

We first use BM25 (DeepCT Index) to get top-1000 candidates for re-ranking. Our model is initialized with a publicly available uncased version of BERT large model and readers can refer to [5] for more details. In the pre-training, we train the model continuously for 5 epochs. We use Adam[10] optimizer with a learning rate of 1e-6 and warmup over the first 10% of total steps. The batch size is set to 32. In the two-phase fine-tuning stage, we first train the model for 5 epochs to be type-adaptive and then conduct listwise fine-tuning for another 3 epochs. Since each query has about 1 positive candidate in average, we set $n_+ = 1$ and $n_- = 5$. As for the exact matching strategy, we set $\alpha = 0.01$. We also use Adam optimizer with a learning rate of 5e-5 and warmup over the first 10% of total steps. The batch size is 4 and the input sequence length is 180.

4.3 Model Comparison

ID	Model	MRR@10	
		Dev	Eval
1	BM25 (Microsoft Baseline)	0.167	0.165
2	BM25 (DeepCT Index)	0.243	-
3	BM25 (DeepCT Index) + BERT	0.367	-
4	**BM25 (DeepCT Index) + TABLE**	**0.412**	**0.401**
5	**4** - Type-adaptive Pointwise Fine-tuning	0.403	-
6	**4** - Exact Matching of OOV Words	0.399	-
7	**4** - Listwise Fine-tuning	0.384	-
8	**4** - Domain-adaptive Pre-training	0.406	-

Table 1: Results of different models on MS MARCO dataset.

Model	**MRR@10**		**Rank**
	Dev	Eval	
An Anonymous Model	**0.414**	**0.405**	1
BM25 + TABLE	0.412	0.401	2
BM25 + TF-Ranking BERT Ensemble	0.394	0.388	3
AOA Index + BERT + CAS	0.408	0.393	4
BM25 + duoBERT (Pairwise)	0.390	0.379	11

Table 2: Top models on the MS MARCO leaderboard. Our TABLE model wins the second place.

As shown in Table 1, our TABLE model outperforms all the baselines. It outperforms BERT by a large margin, which indicates that the components of our TABLE model are effective in DR. The ablation of type-adaptive fine-tuning phase shows that this strategy can aid DR greatly. Further ablation of exact matching feature proves its effectiveness. Moreover, the ablation of the listwise fine-tuning phase reveals its superiority compared to BERT. Table 2 shows that our TABLE model is near the top of the MS MARCO leaderboard and it behaves much better than the pairwise model duoBERT[13] and many other strong models.

5 CONCLUSION

We propose a model named TABLE for DR task, in which we adopt a domain-adaptive pre-training process and present a two-phase fine-tuning strategy. Besides, a very useful exact matching feature is introduced to improve the matching performance of OOV words. Experimental results show our TABLE model outperforms many strong baselines and is near the top of the MS MARCO leaderboard.

REFERENCES

[1] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to Rank Using Gradient Descent. In *Proceedings of the 22nd International Conference on Machine Learning*. 89–96.

[2] Christopher J. C. Burges, Robert Ragno, and Quoc Viet Le. 2006. Learning to Rank with Nonsmooth Cost Functions. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*. 193–200.

[3] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*. 129–136.

[4] Zhuyun Dai and Jamie Callan. 2019. Context-aware sentence/passage term importance estimation for first stage retrieval. *arXiv preprint arXiv:1910.10687* (2019).

[5] 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).

[6] Jerome Friedman. 2000. Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics* 29 (11 2000).

[7] 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*. 55–64.

[8] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. 2333–2338.

[9] Kalervo Järvelin and Jaana Kekäläinen. 2000. IR Evaluation Methods for Retrieving Highly Relevant Documents. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 41–48.

[10] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[11] 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. (2016).

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

[13] Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. 2019. Multi-stage document ranking with BERT. *arXiv preprint arXiv:1910.14424* (2019).

[14] Tao Qin, Xu-Dong Zhang, Ming-Feng Tsai, De-Sheng Wang, Tie-Yan Liu, and Hang Li. 2008. Query-level loss functions for information retrieval. *Information Processing & Management* 44, 2 (2008), 838–855.

[15] Amnon Shashua and Anat Levin. 2002. Ranking with Large Margin Principle: Two Approaches. In *Proceedings of the 15th International Conference on Neural Information Processing Systems*. 961–968.

[16] Mike Taylor, John Guiver, Stephen Robertson, and Tom Minka. 2008. SoftRank: Optimising Non-Smooth Rank Metrics. In *WSDM 2008*.

[17] Qiang Wu, Chris J.C. Burges, Krysta M. Svore, and Jianfeng Gao. 2010. Adapting Bboosting for Information Retrieval Measures. *Information Retrieval* 13 (2010), 254–270.

[18] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. 2008. Listwise Approach to Learning to Rank: Theory and Algorithm. In *Proceedings of the 25th International Conference on Machine Learning*. 1192–1199.

[19] Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017. 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*. 55–64.

[20] Jun Xu and Hang Li. 2007. AdaRank: A Boosting Algorithm for Information Retrieval. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 391–398.