Enhancing Document Ranking with Task-adaptive Training and Segmented Token Recovery Mechanism

Xingwu Sun[1†‡],**Yanling Cui**[2†],**Hongyin Tang**[1†],**Fuzheng Zhang**[1],**Beihong Jin**[2‡],**Shi Wang**[3‡]
[1]Meituan Inc.
[2]Institute of Software Chinese Academy of Sciences
[3]Institute of Computing Technology Chinese Academy of Sciences
[‡]The corresponding authors: sunxingwu01@gmail.com, beihong@iscas.ac.cn, wangshi@ict.ac.cn

Abstract

In this paper, we propose a new ranking model DR-BERT, which improves the Document Retrieval (DR) task by a task-adaptive training process and a Segmented Token Recovery Mechanism (STRM). In the task-adaptive training, we first pre-train DR-BERT to be domain-adaptive and then make the two-phase fine-tuning. In the first-phase fine-tuning, the model learns query-document matching patterns regarding different query types in a pointwise way. Next, in the second-phase fine-tuning, the model learns document-level ranking features and ranks documents with regard to a given query in a listwise manner. Such pointwise plus listwise fine-tuning enables the model to minimize errors in the document ranking by incorporating ranking-specific supervisions. Meanwhile, the model derived from pointwise fine-tuning is also used to reduce noise in the training data of the listwise fine-tuning. On the other hand, we present STRM which can compute OOV word representation and contextualization more precisely in BERT-based models. As an effective strategy in DR-BERT, STRM improves the matching perfromance of OOV words between a query and a document. Notably, our DR-BERT model keeps in the top three on the MS MARCO leaderboard since May 20, 2020.

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 (Nguyen et al., 2016), DR has achieved remarkable advancements.

Input Text **(a)**

Query: what does **bogue** mean?
Document: the definition of bogus is fake or untrue. a statement that is not true is an example of something that would be described as bogus.

The Input Text Tokenized by WordPiece In BERT

Query: what does bog ##ue mean ?
Document: the definition of bog ##us is fake or un ##tr ##ue . a statement that is not true is an example of something that would be described as bog ##us .

Input Text **(b)**

Query: define **insouciantly**?
Document: noun. ineptitude is defined as the quality of being bad at something, or an incompetent remark.

The Input Text Tokenized by WordPiece In BERT

Query: define ins ##ou ##cian ##tly ?
Document: noun . in ##ept ##itude is defined as the quality of being bad at something , or an incompetent remark .

Figure 1: Two cases, each listing an input query, a document candidate and the text tokenized by WordPiece in BERT. The OOV words, e.g., "insouciantly" and "bogue", are tokenized into 2 or more sub-tokens.

Lots of DR models have been studied over the last few years. Traditional machine learning based DR models, like LambdaRank (Burges et al., 2006), AdaRank (Xu and Li, 2007), etc., rely heavily on manual feature engineering which is time-consuming and unsustainable. Neural DR models, including DSSM (Huang et al., 2013), KNRM(Xiong et al., 2017), 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 (Cao et al., 2007; Qin et al., 2008).

Recently, the pre-trained language models have

[†]Equal Contribution.

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 3570–3579
November 7–11, 2021.

caused a stir in DR. Taking BERT (Devlin et al., 2018) as an example, it 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. No matter whether BERT is applied to DR in either a feature-based or a fine-tuning manner (Nogueira and Cho, 2019; Nogueira et al., 2019a), substantial improvements in DR have been attained.

The BERT-based models, despite their 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) The desired model could be more task-adaptive by considering text domains and query types. (3) Due to WordPiece (Devlin et al., 2018) segmentation method employed by BERT, the matching of OOV word might be miscalculated and further a document irrelevant to a query might be ranked high. Figure 1 gives two examples to illustrate the problem induced by WordPiece. In Figure 1(a), the query and the document are irrelevant, because the word "bogue" in the query and the word "bogus" in the document are unrelated. However, BERT fails to distinguish the two words as a result of being misled by the separated tokens generated by WordPiece, i.e., "bog". In Figure 1(b), the tokens generated by WordPiece, i.e., "ins" in the query and "in" in the document, lead to an undesirable matching and further a high query-document relevance score. (4) These models do not have the ability to deal with noise in the training data, which is common in NLP corpora.

In order to solve the above-mentioned problems of BERT-based ranking models, we propose a new DR model named DR-BERT. In the DR-BERT model, we make the following improvements on the basis of BERT: (1) As depicted in Figure 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 a query type-adaptive fine-tuning strategy to adapt the model to this DR task. (3) We add a Segmented Token Recovery Mechanism (STRM) into DR-BERT, which can effectively improve the matching accuracy of OOV words. (4) We employ the model derived from pointwise fine-tuning to reduce noise in the training data of the listwise fine-tuning. We conduct extensive experiments on the MS MARCO dataset, 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 DR-BERT model showed excellent performance.

Our contributions can be summarized as follows.

- We propose a new task-adaptive BERT-based model for DR task. By a domain-adaptive pre-training and a two-phase (i.e., pointwise plus listwise) fine-tuning, the model turns to highly adaptive to the DR task. As a result, it substantially improves the model performance.
- We are the first to propose the pointwise plus listwise fine-tuning, which enables the model to not only learn document-level ranking features after grasping query-document matching features in the pointwise phase but also measure and optimize the document ranking errors.
- We are the first to find the OOV mismatching problem and give an effective mechanism called STRM, which can compute OOV word representation and contextualization more precisely. STRM effectively improves the matching performance of OOV words and it can be applied to most BERT-based models.
- Our DR-BERT model outperformed many strong baselines and keeps in the top three on the MS MARCO leaderboard with an MRR@10 of 0.419 since May 20, 2020.

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 (Shashua and Levin, 2002; Friedman, 2000) transform ranking tasks to classi-

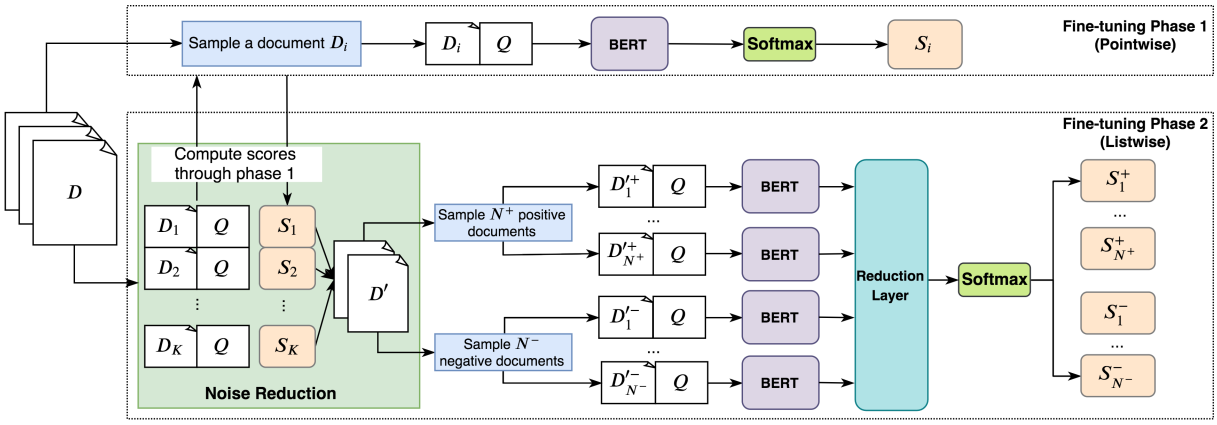

Figure 2: Architecture of the two-phase fine-tuning of the DR-BERT model, which comprises a type-adaptive pointwise fine-tuning, a listwise fine-tuning and a noise reduction method. Q represents the query. D_*^+ and D_*^- represent positive and negative documents, respectively. S_* represent the query-document relevance scores. Note that both the two fine-tuning phases use the query type feature, which is omitted in this figure for brevity.

fication 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 make comparisons between documents which is the core of ranking.

In pairwise approaches (Burges et al., 2005, 2006; Wu et al., 2010), 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 might cause the training bias.

Further, the listwise approaches (Cao et al., 2007; Xia et al., 2008; Xu and Li, 2007; Taylor et al., 2008; Qin et al., 2008) train the models to learn the optimal ranking directly by taking the documents to be ranked as input. Cao et al. (2007) treats the sequence of documents ordered by the top-one probability distribution as the ranking list and optimizes the cross entropy. Xia et al. (2008) maximizes the likelihood of the golden ranking list. Qin et al. (2008) optimizes the similarity of the query and the documents. Xu and Li (2007); Taylor et al. (2008) directly optimize the metrics like NDCG (Järvelin and Kekäläinen, 2000) 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 (Guo et al., 2016) tempts to build neural ranking models which need no manual features and show impressive performance.

Recently, the pre-trained language models like BERT (Devlin et al., 2018) have achieved the state-of-the-art results on several NLP tasks. Leveraging their powerful language understanding abilities, several ranking models built on BERT are proposed to improve the effectiveness and efficiency of DR, For effectiveness of DR, Nogueira and Cho (2019) adopt a pointwise paradigm and Nogueira et al. (2019a) adopt a pairwise one. Further, Nogueira et al. (2019b) append predicted queries to the document and rank the documents with BERT as described in (Nogueira and Cho, 2019). 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. For efficiency of DR, Khattab and Zaharia (2020) present a novel ranking model which encodes the query and the document using BERT and employs an interaction mechanism to speed up the retrieval.

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 $\check{D}$, 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) (Dai and Callan, 2019) to get the top-K document candidates $D = \{D_1, D_2, \cdots, D_K\}$ for the query Q. Readers can refer to (Dai and Callan, 2019) for more details. In this work, we focus on the re-ranking stage.

In the following subsections, we describe our ranking model DR-BERT, whose training process comprises a domain-adaptive pre-training and a two-phase fine-tuning, i.e., type-adaptive pointwise fine-tuning and listwise fine-tuning, in which STRM is proposed to solve the OOV mismatching problem.

3.1 Task-adaptive Training

3.1.1 Domain-adaptive Pre-training

DR-BERT is based on BERT, which is pre-trained on the open corpora. Inspired by (Gururangan et al., 2020), we analyze the top frequent words in the corpora of BERT baseline and MS MARCO dataset, and find that the domains of the MARCO are different from those of the corpora in BERT baseline.

Specifically, we find that the top 10,000 frequent words are 44.3% different. Besides, Liu et al. (2019) proves that BERT is under-optimized. Therefore, it is necessary to adapt BERT to the task domains and continue pretraining. In detail, we employ MS MARCO as input and pre-train BERT for the DR task by maximizing the summation of the masked language model likelihood and the next sentence prediction likelihood. For more details, please refer to (Devlin et al., 2018).

3.1.2 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:

$$X_i = [\texttt{<CLS>}, T, \texttt{<SEP>}, Q, \texttt{<SEP>}, D_i], \quad (1)$$

where <SEP> is the separator and <CLS> indicates the position for query-document relation representation. Next, for the j-th token X_{ij} in sequence X_i, the embedding $\mathbf{E}_{ij}$ can be computed by:

$$\mathbf{E}_{ij} = \mathbf{E}_{tok_{ij}} + \mathbf{E}_{seg_{ij}} + \mathbf{E}_{pos_{ij}}, \quad (2)$$

where $\mathbf{E}_{tok_{ij}}$, $\mathbf{E}_{seg_{ij}}$ and $\mathbf{E}_{pos_{ij}}$ are the token embedding, segmentation embedding and position embedding of X_{ij}, respectively.

Then, we apply BERT with L layers of successive transformer blocks to obtain inter-representation of each token in X_i, i.e., the hidden state $\mathbf{H}_i^l$ in each layer:

$$\mathbf{H}_i^l = \{\mathbf{H}_{i1}^l, \mathbf{H}_{i2}^l, ..., \mathbf{H}_{i|X_i|}^l\} \quad (3)$$

$$\mathbf{H}_i^l = \texttt{Transformer}(\mathbf{H}_i^{l-1}), l = 1, 2, 3, ..., L, \quad (4)$$

in which $\mathbf{H}_i^0 = \mathbf{E}_i$ and $|X_i|$ indicates the sequence length of X_i. The hidden state corresponding to the <CLS> position in the last hidden layer can be used to calculate the query-document relevance score:

$$\mathbf{S}_i = \texttt{Softmax}(\mathbf{H}_{i1}^L) \quad (5)$$

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

Noise Reduction Method Large scale manually labeled training data usually suffers from noise because of data annotation limitations. For instance, the MS MARCO dataset suffers from the sparsity of annotations, which means the dataset contains much noise, i.e., positive samples which are labeled 0. For reducing noise in the training data of the second-phase listwise fine-tuning, we score the relevance between the query Q and document candidates in D using the model derived from the first-phase pointwise fine-tuning. Next, we remove the document candidates whose relevance is greater than a certain threshold, which are determined as unannotated positive samples. The left document

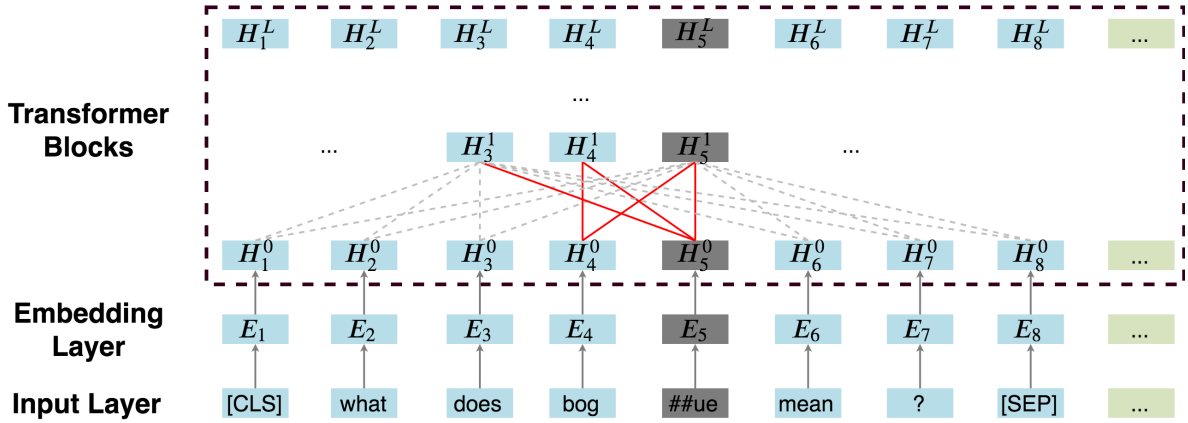

Figure 3: Working process of STRM. By making some restrictions on the attention calculation, we make the last sub-token representation represent the OOV word, which is then used for the computation of contextualized representation with other words. In this figure, we make the following restrictions: (1) The last sub-token of the OOV word is "##ue", which can be attended by all positions, i.e., ranging from 1 to 8. (2) The first sub-token of the OOV word is "bog", which can only be attended by positions ranging from 4 to 5. (3) The sub-tokens of the OOV word, i.e., "bog" and "##ue", can be attended by positions ranging from 4 to 5.

candidates form the set D', which is used in the second-phase listwise fine-tuning.

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.

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 document candidates 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 and negative documents can be computed by equation 4:

$$\mathbf{H}_i^+ = \texttt{BERT}(\mathbf{E}_i^+), \mathbf{H}_i^- = \texttt{BERT}(\mathbf{E}_i^-) \quad (6)$$

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

$$\mathbf{R}_i^+ = \mathbf{W}^T \mathbf{H}_{i1}^{L\,+} + \mathbf{b}, \mathbf{R}_i^- = \mathbf{W}^T \mathbf{H}_{i1}^{L\,-} + \mathbf{b}, \quad (7)$$

where $\mathbf{W}^T$ and $\mathbf{b}$ are trainable variables, $\mathbf{H}_{i1}^{L\,+}$ and $\mathbf{H}_{i1}^{L\,-}$ are the last hidden states corresponding to the <CLS> positions for the i-th positive and negative input respectively, $\mathbf{R}_i{}^+$ and $\mathbf{R}_i{}^-$ are unnormalized query-document relevance scores.

Then, we concatenate all the query-document relevance scores to make a document-level normalization:

$$\mathbf{S}_i^+ = \frac{exp(\mathbf{R}_i^+)}{\sum_{j=1}^{N^+} exp(\mathbf{R}_j{}^+) + \sum_{j=1}^{N^-} exp(\mathbf{R}_j{}^-)} \quad (8)$$

where $\mathbf{S}_i^+$ is the normalized query-document relevance score of the i-th positive document. Till now, we get a ranked list of documents according to their relevance scores with the query. Then, we use supervision information to guide the optimization of measuring document rankings. With only positive and negative labels, the optimizing of the ranking can be regarded as maximizing the normalized scores 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:

$$\mathcal{L} = \frac{\sum_{i=1}^{N^+} -log(\mathbf{S}_i^+)}{N^+} \quad (9)$$

3.2 Segmented Token Recovery Mechanism

In practice, since the word vocabulary is limited, the OOV words are segmented by WordPiece in BERT-based models. As a result, the matching of OOV words is not dealt with properly. As the example in Figure 1(a) 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", which we call OOV mismatching problem. In order to avoid such traps, we propose STRM.

Here, we give the working process of STRM. A direct solution of OOV mismatching problem is to merge all the sub-token representations in an OOV word and then conduct inter-representation with other words by taking the OOV as a whole. Following this direction, we make the following restrictions in the calculation of attention to make the last sub-token representation as the OOV word representation and compute inter-relations with other words as a whole.

- As the representation of the OOV word, the last sub-token in an OOV word can be attended by all positions. For instance, in Figure 3, the last sub-token of the OOV word is "##ue", it can be attended by all positions, i.e., ranging from 1 to 8.
- All the sub-tokens in an OOV word, except for the last one, cannot be attended by other positions outside the OOV word. That is because we only take the last token as the representation of the OOV. For instance, in Figure 3, the first sub-token of the OOV word is "bog", which cannot be attended by positions ranging from 1 to 3 or positions ranging from 6 to 8.
- All the sub-tokens in an OOV word can be attended by each other to make a better self-representation of the OOV word. For instance, in Figure 3, the sub-tokens of the OOV word, i.e., "bog" and "##ue", can be attended by positions in the range of 4 to 5.

Specifically, we introduce a masking matrix $\mathbf{M}$ in the calculation of the l-th layer's attention $\mathbf{A}^l$:

$$\mathbf{A}^l = \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}} + \mathbf{M}\right)\mathbf{V} \tag{10}$$

$$\mathbf{Q} = \mathbf{H}^{l-1}\mathbf{W}^Q, \mathbf{K} = \mathbf{H}^{l-1}\mathbf{W}^K, \mathbf{V} = \mathbf{H}^{l-1}\mathbf{W}^V \tag{11}$$

$$\mathbf{M}_{ab} = \begin{cases} 0, & \textit{allowed to attend} \\ -\infty, & \textit{not allowed to attend} \end{cases} \tag{12}$$

where the previous layer's output $\mathbf{H}^{l-1}$ is projected to a triple of query, key and value using different parameter matrices $\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V$. The masking matrix $\mathbf{M}$ determines whether different positions can be attended to each other. We use different masking matrices to control what positions can be attended to in the computation of the contextualized representations of OOV words, as illustrated in Figure 3.

4 Experiments

4.1 Dataset and Metric

We evaluate our model on the passage ranking dataset in MS MARCO, where MS MARCO provides multiple large-scale real-world datasets and the passage ranking dataset provides over 1M queries and a corpus of 8.8M paragraphs extracted from 3.6M Web documents. Our goal 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. This dataset has been split into training, development and evaluation sets. The train set contains about 0.5M queries, while the development and evaluation set each has about 6800 queries. We note that the dataset suffers from the sparsity of annotations, which means the dataset contains much noise, i.e., positive samples which are labeled 0.

Evaluation is performed by submitting the ranking results to the online leaderboard and the official metric is MRR@10.

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) + DR-BERT**	**0.420**	**0.419**
5	**4**-Domain-adaptive Pre-training	0.413	-
6	**4**-Pointwise Fine-tuning	0.413	-
7	**4**-Query Type Feature	0.415	-
8	**4**-Listwise Fine-tuning	0.390	-
9	**4**-STRM	0.405	-

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

Model	MRR@10		Rank
	Dev	Eval	
RocketQA + ERNIE	0.439	0.426	1
UED-Large Anonymous	0.436	0.424	2
BM25 + DR-BERT	**0.420**	**0.419**	**3**
Expando-mono-duo-T5	0.420	0.408	4
DeepCT + TFR-BERT Ensemble	0.421	0.407	5
BM25 + duoBERT (Pairwise)	0.390	0.379	23

Table 2: Top models on MS MARCO leaderboard.

4.2 Model Settings

In the retrieval stage, we first use BM25 (DeepCT Index) to get top-1000 candidates for re-ranking. DR-BERT is initialized with a publicly available uncased version of BERT large model and readers can refer to (Devlin et al., 2018) for more details. In the pre-training, we train the model continuously for 5 epochs. We use Adam(Kingma and Ba, 2014) optimizer with a learning rate of 1e-6 and warmup over the first 10% of total steps. The batch size is set to 128. In the two-phase fine-tuning, we first train the model for 1 epoch to be type-adaptive. Then, we conduct listwise fine-tuning for another 5 epochs and select the best model on the development dataset. We directly use the query type labels in this dataset. Since each query has about 1 positive candidate on average, we set N^+ to 1 and N^- to 5. We also use Adam optimizer. The batch size is 16 and the input sequence length is 180.

4.3 Performance Evaluation

We conduct the ablation study to evaluate the individual contribution of each component in DR-BERT. Table 1 lists the results of ablation study, along with the performance of several baselines in terms of MRR@10.

From Table 1, we have the following observations:

- Our DR-BERT model outperforms all the baselines. It outperforms BERT by a large margin, which indicates that the components of our DR-BERT model are effective in DR.
- The result of the domain-adaptive pre-training ablation shows that adapting the model to the target domains can improve its performance.
- The ablation of pointwise fine-tuning phase is conducted by only using listwise fine-tuning for additional epoches with the same training data. The experimental result indicates that learning matching and ranking features successively in the two-phase fine-tuning can aid the model.
- As for the query type feature,it induces about 0.5% improvement by MRR@10 over DR-BERT model without the query type feature.
- The result of the listwise fine-tuning ablation reveals its superiority compared to the pointwise fine-tuning.

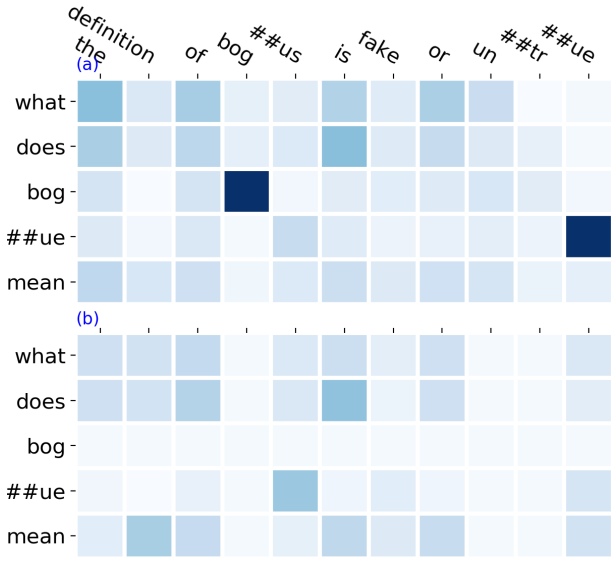

Figure 4: Attention weights between query (row) and document (column) tokens. (a), (b) indicate the BERT baseline, BERT + STRM, respectively. The darker the color is, the greater the attention weight is.

- The ablation of STRM proves the importance of solving the OOV mismatching problem.

Further, we submitted DR-BERT to the MS MARCO leaderboard. By this leaderboard, we can compare our DR-BERT with other models carrying out the same task. Table 2 gives the top models on the MS MARCO leaderboard. Except for DR-BERT, the other five models are as follows.

- RocketQA (Ding et al., 2020) plus ERNIE (Sun et al., 2019) is the best model based on ERNIE , which is well pretrained using more complex tasks.
- UED Anonymous is a competitive model that has not been published.
- Expando-mono-duo-T5 is performed based on T5 (Raffel et al., 2019) model and their previous published model duoBERT (Nogueira et al., 2019a), which is a pairwise document ranking model.
- DeepCT plus TFR-BERT Ensemble (Han et al., 2020) is a well performed DR model, which is a generic document ranking framework that builds a learning to rank model through fine-tuning BERT representations of query-document pairs.

As shown in Table 2, our DR-BERT model shows excellent performance and it behaves much better than many competitive models.

Model	Score	Ranking
BERT	0.864	1
BERT + STRM	0.253	139

(a)

Model	Score	Ranking
BERT	0.806	1
BERT + STRM	0.348	71

(b)

Table 3: Relevance scores and document rankings. (a) and (b) correspond to the query-document case in Figure 1(a) and Figure 1(b), respectively.

4.4 OOV Mismatching Analyses

In this subsection, we analyze the effects of STRM in our DR-BERT model. Firstly, as shown in Table 1, the ablation test of STRM shows that it plays an important role in the DR-BERT.

Secondly, we conduct the case study as follows. (1) For the case in Figure 1(a), we observe the relevance scores calculated by the BERT baseline (which does not have STRM) and the baseline plus STRM. As shown in Table 3(a), the BERT baseline outputs a high relevance score and the document which is irrelevant to the query is ranked the first. After adding STRM, the model behaves better than the baseline and the ranking, i.e., 137, implies that the document is almost impossible to be the one that matches with the query. (2) For the case in Figure 1(b), Table 3(b) lists the relevance scores and document rankings under different models. We also find that the BERT baseline ranks this document unrelated with the query high, but adding STRM changes the situation, where the document ranking is adjusted to 71, thus preventing from irrelevant matching. Further, to clarify how STRM affects the model, we take the case in Figure 1(a) as an example again and observe the effects of the attention weights between the query and the document. As shown in Figure 4, Figure 4(a) shows the scores calculated by the BERT baseline model, where the relevance scores between the segmented OOV words, i.e., “bog", “##ue", are high. As a result, it induces OOV mismatching problem. Figure 4(b) shows that when we add STRM, the OOV mismatching problem is solved in the same way as we expected.

The above analyses indicate STRM can effectively solve the OOV mismatching problem.

In addition, from Table 2, we can see that the accuracy of BM25 + DR-BERT on the evaluation set is higher than Expando-mono-duo-T5 while their accuracy on the development set is the same. It illustrates that DR-BERT is less likely to overfit on the training data compared to other methods. The reason behind is that STRM works well by merging representations of several sub-tokens as shown in Figure 3, which is like the “dropout mechanism” functionally. Besides, lots of noise in the MS MARCO dataset is eliminated by our noise reduction method, thus avoiding overfitting.

4.5 Hyperparameter Sensitivity

In this subsection, we analyze the effects of the key hyperparameter in DR-BERT, i.e., the number of negative samples in the listwise fine-tuning N^-.

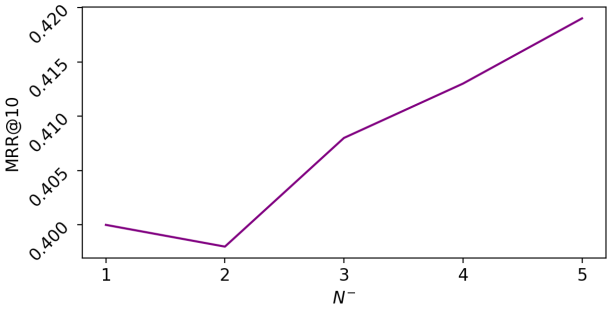

Figure 5: Model performance under different numbers of negative samples.

For the ranking task of MS MARCO, each query has about 1 positive document and hundreds of negative documents on average after first-stage document retrieval and noise reduction. Figure 5 shows the performance under different numbers of negative samples and illustrates that involving more negative samples in the listwise fine-tuning can improve the effect. Specifically, if N^- is set to 1, it becomes a pairwise method. Note that because of hardware limitation, the maximum of N^- is set to 5 in our model.

5 Conclusion

We propose a model named DR-BERT for DR task, in which we adopt a domain-adaptive pre-training and present a two-phase fine-tuning strategy, i.e., type-adaptive pointwise fine-tuning and listwise fine-tuning. Besides, we present a very useful segmented token recovery mechanism to improve the matching performance of OOV words, which can be also applicable to other BERT-based models. Experimental results show our model outperforms many strong baselines and keeps in the top three on the MS MARCO leaderboard since May 20, 2020.

6 Acknowledgments

This work is supported by Beijing NOVA Program (Cross-discipline, Z191100001119014), the National Key Research and Development Program of China (2017YFB1002300,2017YFC1700300), National Natural Science Foundation of China (61702234).

References

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*, page 89–96.

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*, page 193–200.

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*, pages 129–136.

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

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*.

Yingqi Qu Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2020. Rocketqa: An optimized training approach to dense passage retrieval for open-domain question answering.

Jerome Friedman. 2000. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29.

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*, page 55–64.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks.

Shuguang Han, Xuanhui Wang, Mike Bendersky, and Marc Najork. 2020. Learning-to-rank with bert in tf-ranking. *arXiv preprint arXiv:2004.08476*.

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*, pages 2333–2338.

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*, page 41–48.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. *arXiv preprint arXiv:2004.12832*.

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

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.

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.

Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*.

Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. 2019a. Multi-stage document ranking with bert. *arXiv preprint arXiv:1910.14424*.

Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. 2019b. Document expansion by query prediction.

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):838–855.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

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*, page 961–968.

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2019. Ernie 2.0: A continual pre-training framework for language understanding.

Mike Taylor, John Guiver, Stephen Robertson, and Tom Minka. 2008. Softrank: Optimising non-smooth rank metrics. In *WSDM 2008*.

Qiang Wu, Chris J.C. Burges, Krysta M. Svore, and Jianfeng Gao. 2010. Adapting bboosting for information retrieval measures. *Information Retrieval*, 13:254–270.

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*, page 1192–1199.

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*, pages 55–64.

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*, page 391–398.