MatRank: Text Re-ranking by Latent Preference Matrix

Jinwen Luo [*1], **Jiuding Yang** [*2], **Weidong Guo** [*1], **Chenglin Li** [2], **Di Niu** [2], **Yu Xu** [1]
[1]Platform and Content Group, Tencent
[2]University of Alberta
[1]{jamsluo,weidongguo,henrysxu}@tencent.com
[2]{jiuding,ch11,dniu}@ualberta.ca

Abstract

Text ranking plays a key role in providing content that best answers user queries. It is usually divided into two sub-tasks to perform efficient information retrieval given a query: text retrieval and text re-ranking. Recent research on pretrained language models (PLM) has demonstrated efficiency and gain on both sub-tasks. However, while existing methods have benefited from pre-trained language models and achieved high recall rates on passage retrieval, the ranking performance still demands further improvement. In this paper, we propose MatRank, which learns to re-rank the text retrieved for a given query by learning to predict the most relevant passage based on a latent preference matrix. Specifically, MatRank uses a PLM to generate an asymmetric latent matrix of relative preference scores between all pairs of retrieved passages. Then, the latent matrix is aggregated row-wise and column-wise to obtain global preferences and predictions of the most relevant passage in two of these directions, respectively. We conduct extensive experiments on MS MACRO, WikiAQ, and SemEval datasets. Experimental results show that MatRank has achieved new state-of-the-art results on these datasets, outperforming all prior methods on ranking performance metrics.

1 Introduction

The enormous growth of online content has posed significant challenges for information retrieval (IR) systems, e.g., Google, Baidu, Bing, and Quora. Ranking methods are crucial in IR systems in identifying the most relevant passage and document to answer a user query. Generally, text ranking is divided into two steps, i.e., text retrieval and text re-ranking. In the text retrieval step, a number of passages or documents of high relevance to a user query are retrieved to ensure the efficiency of the text ranking. In the re-ranking step, all the retrieved paragraphs are ranked in decreasing order of relevance to the given query.

Recent studies adopt pre-trained language models (PLMs) to boost the ranking performance of IR systems. Current state-of-the-art text retrieval techniques can provide decent performance in recalling the most relevant content of a query (Xiong et al., 2020; Qu et al., 2021; Karpukhin et al., 2020; Ren et al., 2021a; Gao and Callan, 2021, 2022). However, how to better present those retrieved texts in order of their relevance to the query is still challenging. Previous learning to rank methods can be roughly divided into three categories: 1) the point-wise methods (e.g., McRank (Li et al., 2007)), which score each candidate independently, 2) the pair-wise methods (e.g., RankNet (Burges et al., 2005), LambdaRank (Burges et al., 2006)), which employ a pair-wise loss between every pair of candidate paragraphs, and 3) list-wise methods (e.g., LambdaMart (Burges, 2010), ListRank-MF (Shi et al., 2010)), most of which try to rank all candidates at once. However, the sizes of the benchmarks are increasing due to the enormous growth of online texts. Such large sizes have limited the number of annotations per query (Nguyen et al., 2016). That is, only a few truth answers are available for each query, rather than a ground-truth rank list. Therefore, directly implementing the classical pair-wise or list-wise ranking methods on these benchmarks is infeasible due to the lack of ground-truth labels. Moreover, the increasing online text brings more homogeneous passages, making the re-ranking even more challenging.

To tackle the above challenges, recent research on text re-ranking (Ren et al., 2021b; Gao et al., 2021b; Ma et al., 2021; Xin et al., 2020; Li et al., 2020; Gao et al., 2020; Hofstätter et al., 2020) utilizes PLMs to better understand the content of text before evaluating its relevance to a query. In case of limited annotations, they try to convert the text re-ranking problem into a classification task, which

[*]These authors contributed equally to this work.

Findings of the Association for Computational Linguistics: EMNLP 2022, pages 2011–2023
December 7-11, 2022

learns to predict the best relevant candidate directly. The outputs for each candidate are their probability of becoming the best choice, which is used to represent their relevance score to the query. However, such classification tasks do not utilize preference comparisons between candidates (pair-wise comparison). Furthermore, the limitation on the sequence length of the PLMs makes the re-ranking of long documents even harder.

In this paper, we introduce the MatRank (Text Re-**rank**ing by Latent Preference **Mat**rix), an effective method for re-ranking the retrieved text by learning a latent preference matrix, which enables the preference assessment during the re-ranking process and makes full use of the annotation. Given a query, MatRank employs a PLM to encode every two candidates into latent vectors and an MLP selector to generate a latent preference Matrix, where each entry represents a preference score between two candidates. The matrix entries are then aggregated to generate the row-wise preference and the column-wise preference, which are used to predict the rank result. The entire model is trained end-to-end by simultaneously minimizing the list-wise classification losses on both the row-wise preference and the column-wise preference to identify their winners, which are used to match the ground truth. Specifically, we make the following contributions:

We proposed the MatRank method, which can implicitly assess the preference between every two candidates by generating a latent preference matrix even when the preference labels are unavailable. Through the aggregation of the estimated pair-wise preference scores in the matrix, the MatRank can also make use of the list-wise information to generate a better ranking result.

To handle the re-ranking for long documents, we design MatRank-Split, which splits the long documents into pieces and uses MatRank to generate a latent preference matrix of these text pieces. Each document will be represented by its text piece with the highest global preference. With such a design, our MatRank-Split can aggregate much more information from a document to assess the preference between every two documents more accurately.

To demonstrate the performance of the proposed method, we have tested MatRank on the MS MACRO (Nguyen et al., 2016) Passage, WikiQA (Yang et al., 2015), and three SemEval datasets (Nakov et al., 2015, 2016, 2017). The results indicate that the proposed method significantly outperforms a wide range of existing passage re-ranking methods. Furthermore, experimental results on the MS MACRO Document dataset demonstrate the superiority of the proposed MatRank-split model over state-of-the-art non-ensemble methods on long-document re-ranking tasks. We have also performed extensive ablation studies to demonstrate the effectiveness of the proposed model design of both the MatRank and the MatRank-Split models by exploring a variety of potential variants.

2 Related Work

In this section, we introduce the related works in the following aspects.

2.1 Traditional Text Ranking

Text Ranking is a central problem for an Information retrieval system. Traditional approaches rank text mostly based on vector-based method (Baeza-Yates et al., 1999; Deerwester et al., 1990) and probabilistic-based method (Maron and Kuhns, 1960; Robertson, 1997). Later, the learning-to-rank method is developed by implementing supervised machine learning in ranking problems using manually-engineered features (Cao et al., 2007; Li, 2011; Liu, 2011). Those methods are usually grouped into three types: 1) Point-wise approaches which score each candidate independently from other text (Crammer et al., 2001; Li et al., 2007); 2) pair-wise approaches which compare every two candidates to ensure the correctness of the output rank list (Herbrich et al., 1999; Burges et al., 2005; Gao et al., 2014); 3) List-wise approaches which consider the entire candidate list when ranking (Xia et al., 2008; Valizadegan et al., 2009; Burges, 2010; Shi et al., 2010). The latter two kinds of approaches are believed to be closer to the "concept" of ranking (Liu, 2011), which inspired us to design our proposed method.

Then, with the development of neural networks, neural text ranking, which relies on soft matching between text representations, is introduced. Unlike traditional text ranking, neural text ranking is developed to utilize (deep) neural networks to abstract the relevance between query and text without manually-engineered features. Researches extract text features from pre-trained word embeddings such as word2vec (Mikolov et al., 2013) and GLoVe (Pennington et al., 2014) to compute similarities between query and text as relevance

score (Ganguly et al., 2015; Guo et al., 2016; Tymoshenko et al., 2016; Xiong et al., 2017).

2.2 PLM Based Ranking

The great success achieved by pre-trained language models (PLMs), e.g., BERT (Devlin et al., 2019), further draws the attention of researchers on building Transformers-based ranking model (Karpukhin et al., 2020; Zhu and Klabjan, 2020; Xin et al., 2020; Gao et al., 2020; Hofstätter et al., 2020; Gao and Callan, 2021, 2022; Ma et al., 2021; Zhuang and Zuccon, 2021). To balance the trade-off between efficiency and effectiveness, recent researches conventionally separate the text ranking procedure into text retrieval and text re-ranking, using different models. Text retrieval method mostly utilize dual-encoder architectures (Qu et al., 2021; Gao et al., 2021a; Ren et al., 2021a), which encode query and text separately for efficiency. The re-ranking step usually employs a cross-encoder model to extract a union representation for a query-candidate pair, which enables the interactions between the query and the candidate (Li et al., 2020; Gao et al., 2021b; Ren et al., 2021b). However, while those state-of-art retrieval methods can ensure relatively high recall rates, the rank performance of re-ranking tasks still needs further improvement.

Recent research has made multiple attempts on optimizing re-ranking learning (Ren et al., 2021b; Gao et al., 2021b; Ma et al., 2021; Li et al., 2020; Xin et al., 2020; Zhuang and Zuccon, 2021). However, those PLM-based methods mostly focus on effective representation learning, ignoring a human-like comparison between texts when ranking. Han et al. (Han et al., 2020) trained an ensemble of point-wise, pair-wise, and list-wise learning-to-rank models with BERT (Devlin et al., 2019). However, the authors combine those three methods simply by summing up the corresponding losses. In contrast, our proposed method effectively combines pair-wise preference and list-wise classification. MatRank learns a latent preference matrix and simultaneously minimizes the list-wise classification losses on both the row-wise and column-wise preferences aggregated from the matrix.

3 Methodology

In this section, we introduce the proposed MatRank, which learns to rank based on the content comparison between different candidates, in detail. We will mainly discuss the design parts of MatRank based on the MS MACRO datasets.

Given a query q and the corresponding retrieved candidate list $T = \{t_i\}_{1 \le i \le M_T}$, the goal of text re-ranking is to generate a rank list $R = \{r_i\}_{1 \le i \le M_R}$ from T, where M_T is the number of candidates and $M_R (\le M_T)$ represents the length of the generated rank list. Here, we assume $M_R = M_T = M$.

3.1 Passage Re-ranking

The overall architecture of MatRank is shown in the left part of Figure 1. Given a query q, and its retrieved passage set $T = \{t_i\}_{1 \le i \le M}$, where M is the number of candidate passages. Starting from the bottom left of Figure 1, we first employ a pre-trained language model (PLM) as a cross-encoder to encode each query-passage pair into a latent vector, e.g., BERT (Devlin et al., 2019), ERNIE (Sun et al., 2020). Those models have superior abilities in extracting semantic information from given texts, which our method utilizes to predict pair-wise preference.

Formally, we have

$$h_i = \underset{\text{[CLS]}}{\text{encode}}(q, t_i), \tag{1}$$

where h_i denotes the output embedding of the [CLS] token of each given query-passage pair. q is the query, and t_i is the i^{th} passage in the passage set T of query q. [CLS] is commonly used in PLM models to represent the overall semantic meaning of the input sequence. By feeding both the query and the passage into PLM, PLM can establish the connection between them and thus can better understand the passage.

Then, we compare every two query-passage pairs by feeding their latent embeddings, e.g., $(h_i, h_j)_{i \ne j}$, into an MLP module to predict the degree to which query q is more relevant to paragraph t_i than paragraph t_j. To be specific, we have

$$s_{i,j} = \text{MLP}(h_i \frown h_j), \tag{2}$$

where $h_i \frown h_j$ represents the concatenated embedding of h_i and h_j. Let

$$\mathcal{S} = [s_{i,j}]_{1 \le i,j \le M}, \tag{3}$$

be the latent preference matrix that stores all the comparison results where $s_{i,j} = 0$ if $i = j$. We further aggregate the asymmetric preference matrix row-wise and column-wise to generate the relative preference scores of all the retrieved paragraphs

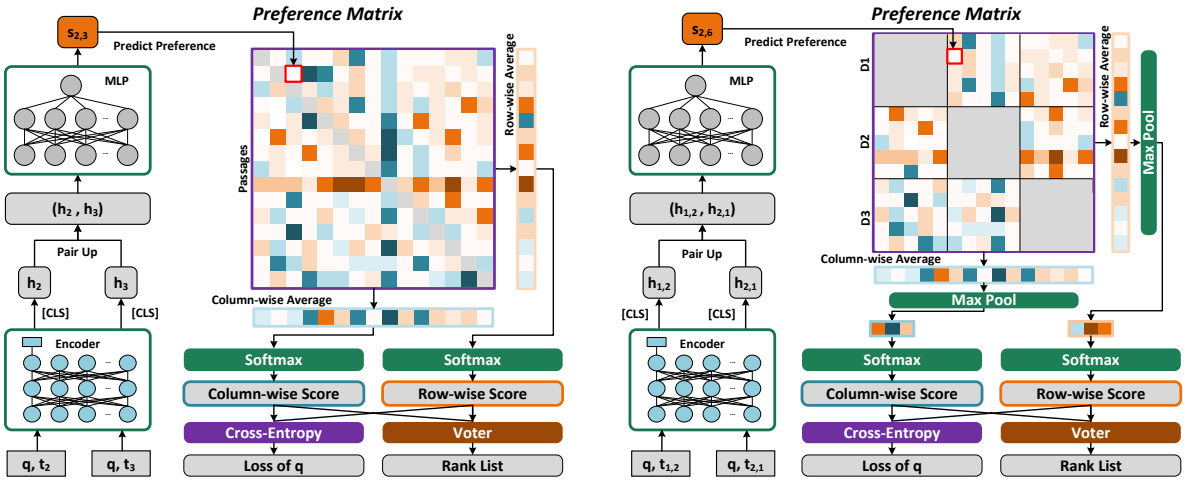

Figure 1: The architecture of MatRank (left) and MatRank-Split (right). The encoder is a pre-trained language model. For MatRank, We use the comparison between passage-2 and passage-3 as an example for preference prediction. For MatRank-Split, we use the comparison between the second text piece of document-1 ($t_{1,2}$) and the first text piece of document-2 ($t_{2,1}$) as an example for preference prediction.

both row-wise and column-wise. Finally, these two aggregated vectors are used to predict the truly clicked passage for query q, respectively. Here, we aggregate the preference matrix by taking the average of each row or column. Specifically,

$$\mathbf{r} = \frac{1}{M}\sum_{j=1}^{M}[s_{i,j}]_{1\le i,j\le M}, \tag{4}$$

$$\mathbf{c} = \frac{1}{M}\sum_{i=1}^{M}[s_{i,j}]_{1\le i,j\le M}, \tag{5}$$

where $\mathbf{r}$ denotes the raw-wise average of $\mathcal{S}$, and $\mathbf{c}$ denotes the column-wise average of $\mathcal{S}$. Then, we can calculate the row-wise average score β and the column-wise average score ω through the softmax operation:

$$\beta = \text{softmax}(\mathbf{r}), \tag{6}$$

$$\omega = \text{softmax}(-\mathbf{c}), \tag{7}$$

The average row score of a passage is obtained by comparing the passage to all other candidates, and the average column score is obtained by comparing each candidate to the passage. Therefore, these two scores calculated from the two directions can both represent the relative relevance of the passage compared to all the other candidates. Thus, we treat both the β and ω as the predicted label distribution of the truly clicked (preferred) paragraph.

Finally, we calculate the loss between the predictions and the ground truth with the Cross-entropy loss function. Let $b_u \in \beta$ and $w_v \in \omega$, the loss of the query q is formed as:

$$\ell = \sum_{u=1}^{M} y_u \log b_u + \sum_{v=1}^{M} y_v \log w_v, \tag{8}$$

where $y_x = 1$ if t_x is the ground truth, otherwise $y_x = 0$. The first term denotes the prediction loss of the row-wise average score, i.e., β, and the second term indicates the loss of the column-wise preference score, i.e., ω.

By optimizing the loss function defined in Equation (8), our model learns to compare query-passage pairs from two different directions, which makes full use of the retrieved candidate set. Thus, our model is more effective than previous work, which directly transfers the learn-to-rank task to a classification task. We will demonstrate the superiority of such a model design by comparing experimental results to previous state-of-the-art methods.

To get the rank list of query q, we sort β and ω in descending order. The resulting indices are two rank lists that indicate the orders of a paragraph row-wise and column-wise. We then use the voting method to fuse these two lists into one rank list as the final output.

3.2 Document Re-ranking

We further propose MatRank-Split to handle the document re-ranking. As shown in the right part of Figure 1, given a query q and its retrieved document set $T = \{t_i\}_{1\le i\le M}$ where M is the number of candidates, we first split every document t_i into

$t_i = \{\hat{t}_{i,k}\}_{1 \le k \le K}$ where $t_{i,k}$ is a text piece of the document t_i with a max length of L, and K is the max number of text pieces that each document can be split into. Similarly, we utilized Equation (1) and Equation 2 to build the latent preference matrix:

$$\hat{\mathcal{S}} = [\hat{s}_{g,f}]_{1 \le g,f \le MK}, \tag{9}$$

where $\hat{s}_{g,f} = 0$ if the g-th and f-th text pieces are from the same document. Note that each document is split into K text pieces, thus $\hat{\mathcal{S}} \in \mathbb{R}^{MK \times MK}$. To predict the most relevant document, we need to find the most relevant text pieces among all candidates. As shown in Figure 1, we calculate the row-wise average score $\hat{\beta}$ and the column-wise average score $\hat{\omega}$ by adding a maxpool function which extracts the max value of every K elements in a vector:

$$\hat{\beta} = \text{softmax}(\text{maxpool}(\hat{\mathbf{r}})), \tag{10}$$

$$\hat{\omega} = \text{softmax}(\text{maxpool}(-\hat{\mathbf{c}})). \tag{11}$$

where $\hat{\mathbf{r}}$ denotes the row-wise average of $\hat{\mathcal{S}}$, and $\hat{\mathbf{c}}$ denotes the column-wise average of $\hat{\mathcal{S}}$. Finally, following similar procedure of the MatRank model, the prediction loss can be formulated as:

$$\hat{\ell} = \sum_{u=1}^{M} \hat{y}_u \log \hat{b}_u + \sum_{v=1}^{M} \hat{y}_v \log \hat{w}_v, \tag{12}$$

where $\hat{y}_x$ is one if the x-th test piece belongs to a ground truth document, otherwise is zero. $\hat{b}_u \in \hat{\beta}$ and $\hat{w}_v \in \hat{\omega}$ denote the relative row-wise preference score of the u-th and column-wise preference score of the v-th documents, respectively.

Document re-ranking is much more challenging than passage re-ranking due to the large text length. Despite that they have richer information than passages, their relevance with a query may only depend on a small portion of their texts. Moreover, limited by hardware, the documents usually can not be directly fed into most of the existing PLMs for cross-encoding. How to effectively locate the text pieces relevant to a query becomes the key to a better prediction. By splitting documents into text pieces, our method can perform a piece-wise comparison between different documents. For each candidate, the most preferred text piece will be used to represent the document for rank list prediction. Our design can include more information from a long document when comparing, which leads to a more accurate assessment of preference.

4 Experiments

We develop and test our method on the MS MACRO Passage and the MS MACRO Document datasets, which are widely used benchmarks in many recent studies on passage ranking (Gao et al., 2021a; Qu et al., 2021; Gao and Callan, 2021; Li et al., 2020). To further justify the robustness of the MatRank, we also conduct experiments on other public datasets such as the WikiQA (Yang et al., 2015) and the SemEval datasets (Nakov et al., 2015, 2016, 2017). Detailed descriptions of those datasets are given in Appendix A.

We adopt the same settings and retrievers as the current state-of-the-art methods on each dataset for fair comparisons. We report the MRR (Mean Reciprocal Rank) and, if applicable, the MAP (Mean Average Precision) as the evaluation metrics.

We fine-tune our model on 8 NVIDIA Tesla A100 GPUs (with 40G RAM) under FP16. We employ the AdaFactor (Shazeer and Stern, 2018) optimizer with a learning rate of 1e-5 and a 10% warm-up ratio to train our models on all the datasets. The epochs are set to be 3 on the MS MARCO dataset and 5 on the WikiQA and the SemEval datasets. The positive to the hard negative ratio is set to be 1:96 on the MS MARCO Passage dataset, 1:16 on the MS MARCO Document dataset, and 1:4 on the WikiQA and the SemEval datasets. In addition, we set the max text piece of document re-ranking to 5 and the max length of each piece to 512. Our model contains 120 million parameters. The implementation details of the compared baselines are given in Appendix B.

4.1 Passage Re-ranking

Table 1 and Table 2 summarize the results of the passage re-ranking performance of MatRank and other baselines on the MS MACRO Passage, SemEVal, and WikiQA datasets. The results of the baselines are collected from their corresponding published papers. We further reproduce the LCE with the ANCE retriever on MS MACRO Passage since the original paper does not report it.

On the MS MACRO Passages dataset, we achieved the best MRR@10 score of 42.6, which outperforms the SOTA method, i.e., Ren et al. (2021b), by 0.7 points, following the same experimental settings. On the SemEval datasets, we outperform CETE by 0.4 to 1.9 on MAP and 1.2 to 2.3 on MRR. Furthermore, on the WikiQA dataset, we outperform CETE by 1.6 and 1.9 on MAP and

Model	Retriever	Passage MRR@10	Document MRR@100	Retriever	Passage MRR@10	Document MRR@100
BM25 (Robertson and Zaragoza, 2009)	BM25	18.7	23.0	-	-	-
ColBERT (Khattab and Zaharia, 2020)[1]	BM25	34.9	-	-	-	-
TFR-BERT (Han et al., 2020)[1]	BM25	40.5	-	RocketQAv2	41.1	-
RocketQAv2 (Ren et al., 2021b)[2]	BM25	40.1	-	RocketQAv2	41.9	-
LCE Gao et al. (2021b)[1]	HDCT	-	43.4	ANCE	-	44.7
DML Zhang and Yang (2021)[1]	HDCT	-	42.5	ANCE	-	44.2
MatRank[1]	BM25	40.2	-	RocketQAv2	**41.9**	-
MatRank[2]	BM25	**40.8**	-	RocketQAv2	**42.6**	-
MatRank-Split[1]	HDTC	-	**44.9**	ANCE	-	**45.2**
MatRank-Split[2]	HDTC	-	**45.4**	ANCE	-	**45.8**

Table 1: Experiment results on the MS MACRO Passage dataset. We use BM25@1000 and RocketQAv2@50 (Ren et al., 2021b) for passage retrieval, and HDTC@100 (Dai and Callan, 2020) and ANCE@100 (Xiong et al., 2020) for document retrieval. Methods with [1] use $\text{BERT}_{\text{base}}$ as text encoder. Similarly, [2] indicates the $\text{ERNIE}_{\text{base}}$ encoder.

Model	SemEval2015		SemEval2016		SemEval2017		WikiQA	
	MAP	MRR	MAP	MRR	MAP	MRR	MAP	MRR
KeLP (Filice et al., 2017)	-	-	79.2	88.8	88.4	92.8	-	-
MVFNN (Sha et al., 2018)	-	-	80.1	87.2	-	-	75.6	75.8
GSAMN (Lai et al., 2019)	-	-	-	-	-	-	85.7	87.2
CETE (Laskar et al., 2020)	93.3	95.6	85.1	90.0	90.9	94.4	84.7	86.0
MatRank	**93.7**	**96.8**	**87.0**	**92.3**	**91.6**	**96.1**	**86.3**	**87.9**

Table 2: Experiment results on the SemEval datasets. Following Laskar et al. (2020), we encode query-passage pairs using $\text{RoBERTa}_{\text{base}}$.

MRR, respectively.

Apparently, our method achieves the best performance among all the baseline methods under the same settings, i.e., the same pretrained language model and the same candidate retriever. Unlike those baselines that compare scores to generate permutations, our method can better measure the semantic interaction between different passages to encode query-passage pairs under the supervision of all retrieved candidates by comparing the content representation of candidates.

Figure 2 shows a real case example of the content comparison, where passage-11 is the ground truth, on the MS MACRO Passage dataset. However, as shown in Figure 3, passage-1 is also a candidate of high relevance and can provide answers to the query "how long to get a bachelor". This kind of high-relevance negative sample is common in the MS MACRO dataset, which makes it harder for models to distinguish the best answer (passage-11) from those highly relevant candidates (e.g., passage-1). For example, RocketQAv2, a score-based ranking model, failed to predict the ground truth and rank the passage-1 at TOP-1. In contrast, our MatRank successfully ranks passage-11 as Top-1 by comparing the content of all candidates, and the reason for such a decision can be explicitly shown by the heat map in Figure 2. Circled by the red dashed lines, although the preference between passage-1 and passage-11 is very small, one can easily tell that passage-11 should be more relevant to the query since it is preferred by the model compared with all the other candidates. The comparison to the content of other candidates becomes the key to a more accurate permutation.

4.2 Document Re-ranking

Table 1 shows the result of the document re-ranking of MatRank-Split and the other two baselines. From the table, we can observe that the MatRank-Split achieved the best performance. With the HDTC retriever, MatRank-Split outperforms the LCE baseline by 1.5 points and DML by 2.4 on MRR@100 score. Meanwhile, the MatRank-Split with the ANCE retriever outperforms the LCE baseline by 0.5 and the DML by 1.0 points on MRR@100.

Restricted by the built-in text length limitation of BERT, DML and LCE can only extract information from a small portion of a long document. Instead, we split at most 5×512 terms of a document into 5 pieces for later content comparison. Such a length can cover most of the candidates in the dataset, whose average document length is 584. Figure 4 shows a real case from the MS MARCO Document dataset. According to the figure, the third text piece

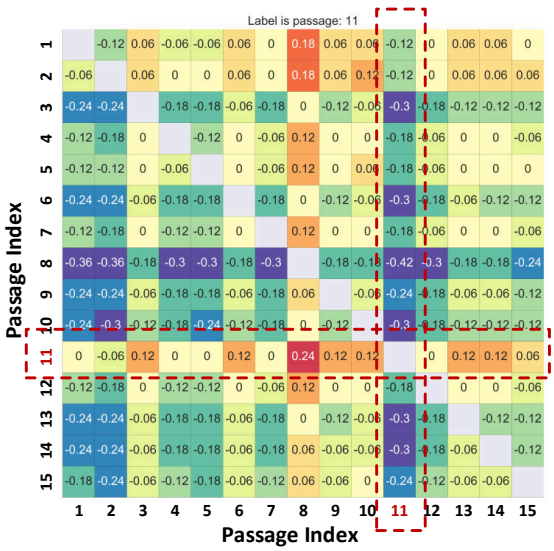

Figure 2: A real case from the MS MACRO Passage dataset, where passage-11 is the ground truth.

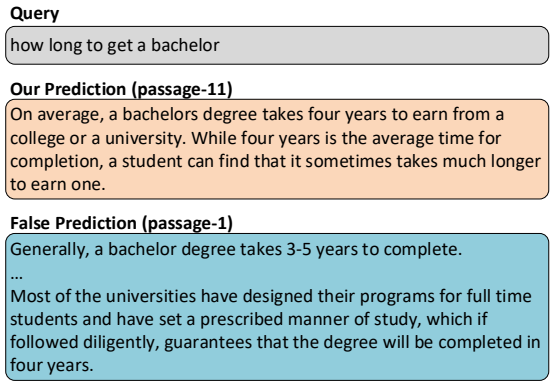

Figure 3: The query and two passages mentioned in Figure 2.

of document-3 is the most preferred, and the first text piece of document-2 is the second preferred. Based on the results, our model correctly predicts document-3 as the TOP-1, while other baselines can only extract limited information from documents and may falsely rank document-2 as the TOP-1. Moreover, as shown in Table 1, without the document splitting, our MatRank still outperforms the two baselines with the same retriever, which further demonstrates the effectiveness of our model design that enables content comparison.

4.3 Ablation Study

To further test the effectiveness of our designed content comparison method on passage and document re-ranking, we compare our model to a variety of

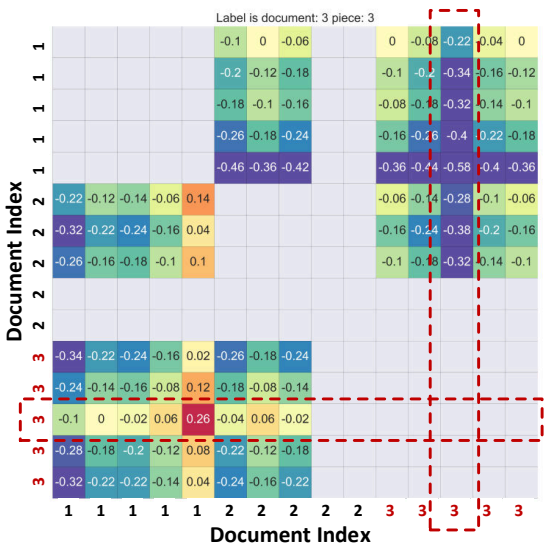

Figure 4: A real case from the MS MACRO Document dataset, where document-3 is the ground truth.

penitential variants of both MatRank and MatRank-Split. Tale 3 gives the performance comparison and Figure 5 provides an illustration of the variants of MatRank.

For the "Point-wise" model, we collect the latent vectors of all query-passage pairs of the q using $\text{BERT}_{\text{base}}$ and feed them into an MLP to obtain the score of every passage to generate the permutation. For the "List-wise" model, we concatenate the latent vectors obtained from the cross-encoder and employ an MLP model to generate the passage score in a multi-class classification setting. This design aims to build the interaction directly between passages by feeding all information at once. However, as shown in Table 3, the method does not have a visible performance impact on the result, demonstrating the limitation of pure MLP models in handling interaction between candidates. To help build interactions between candidates, we further modified "List-wise" to the "+BigBird" model and the "+Attention" model.

Based on the "List-wise" model, we test the "+BigBird" by leveraging the ability of BigBird (Zaheer et al., 2020) to deal with long text and extract the context signal of all candidate passages at once. Specifically, we concatenate the query with all its candidates, feed the newly formed long text into a BigBird model, and directly use the output [SEP] embeddings of all passages to calculate their scores. However, the performance of such a design is even worse than the "Point-wise" model. One po-

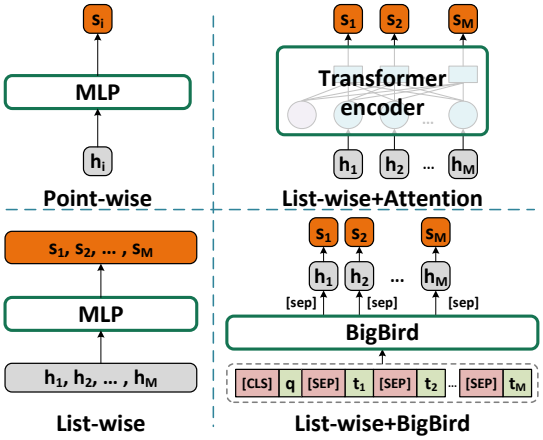

Figure 5: An illustration of other alternative comparison methods.

tential reason for this deduction is that, compared with MatRank, the sparse-attention mechanism in the Transformer layer of BigBird does not fit the idea of comparison. The term-level interactions between queries and passages may be too far to handle.

Model	Passage MRR@10	Document MRR@100
Point-wise	41.9	43.2
List-wise	41.9	-
+BigBird	41.2	-
+Attention	42.3	-
$\text{MatRank}_{\text{row}}$	42.4	-
MatRank	**42.6**	44.0
BigBird	-	44.1
MatRank+BigBird	-	44.3
MatRank-Split	-	**45.4**

Table 3: Ablation study on MS MARCO datasets. The passage retriever is RocketQAv2, and the document retriever is HDTC.

For the "+Attention" method, we employ a Transformer encoder layer to build the attention among all query-passage pairs of q and set the output dimension to 1 to score their relevance to the query. The relevance of a passage is scored by aggregating the content information of all candidates. Benefits from the self-attention interaction mechanism between passages, the "+Attention" method outperforms the "Point-wise" model by 0.4 on MRR@10. However, MatRank still performs better than "+Attention".

From the above observations, we can conclude that proper content comparison between candidates is the key to increasing the re-ranking performance since the ranking process is naturally done by comparing. Unlike the other approaches, our MatRank can better measure the content preference between candidates by calculating the preference matrix, thus achieving a more accurate prediction of the permutation.

As mentioned in Section 3, we also test the learning performance from only the row-wise average score with "$\text{MatRank}_{\text{row}}$" (the column-wise only MatRank has the same performance). The decreased performance shown in Table 3 clearly proves that learning from both the row-wise and the column-wise score is more effective.

Table 3 also give the performance comparison on MS MACRO Document dataset. As shown in the table, with our proposed split method, MatRank-Split achieves better performance than the point-wise model by 2.2 and the MatRank model by 1.4, which proves the effectiveness of our MatRank-Split design. We also report the performance of two potential variants: "BigBird", and "MatRank+BigBird". The candidates of all ablation models are prepared by the HDTC method for consistency.

To start, we first test the performance of BigBird with a max sequence of 1024. As shown in Table 3, since the BigBird can handle longer sequences than the "Point-wise" model, it outperforms "Point-wise" by 0.9. Moreover, we replace the encoder of the MatRank module with BigBird to further test the effectiveness of our proposed content-aware comparison method. From the table, we can observe that "MatRank+BigBird" achieves higher performance than its non-comparison version (i.e., "BigBird") by learning from the latent preference matrix built by MatRank.

From the above observations, we can conclude that both the proper comparison and the capability of handling long sequences are essential for increasing the document re-ranking performance. Our MatRank-Split method can effectively combine those two advantages by constructing a preference matrix on split documents, thus achieving the best performance among all compared methods.

5 Conclusion

In this paper, we introduce MatRank for passage re-ranking. MatRank adopts a pre-trained language model to generate an asymmetric latent matrix of relative preference scores, which is aggregated row-wise and column-wise to predict the most relevant passage in both directions, respectively. The model is trained in an end-to-end manner with classifi-

cation loss. By minimizing the list-wise classification loss generated from the pair-wise comparison scores, MatRank makes full use of all the retrieved passages, even with only a few annotations. Furthermore, we propose MatRank-Split for long document re-ranking. In MatRank-Split, the documents are split into text pieces to fit the comparison method designed in MatRank. Experimental results on several benchmark datasets demonstrate the effectiveness of both the MatRank and MatRank-Split models.

Limitations

Compared with other methods, MatRank and MatRank-Split require more GPU memories for training. Thus we use a smaller batch size to train the models. Furthermore, since our computing resources are limited, we have not implemented helpful learning strategies, such as adversarial training. Also, we didn't fine-tune the hyper-parameters of our model. Instead, we followed the settings of those parameters with existing works. To overcome the short-comes addressed above, we will further test our method with available learning strategies and perform a grid search to find the best parameter settings in the future.

Acknowledgements

We would like to thank the anonymous reviewers for their constructive comments.

References

Ricardo Baeza-Yates, Berthier Ribeiro-Neto, et al. 1999. *Modern information retrieval*, volume 463. ACM press New York.

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*, pages 89–96.

Christopher Burges, Robert Ragno, and Quoc Le. 2006. Learning to rank with nonsmooth cost functions. *Advances in neural information processing systems*, 19:193–200.

Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11(23-581):81.

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.

Koby Crammer, Yoram Singer, et al. 2001. Pranking with ranking. In *Nips*, volume 1, pages 641–647.

Zhuyun Dai and Jamie Callan. 2020. Context-aware document term weighting for ad-hoc search. In *Proceedings of The Web Conference 2020*, pages 1897–1907.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Simone Filice, Giovanni Da San Martino, and Alessandro Moschitti. 2017. KeLP at SemEval-2017 task 3: Learning pairwise patterns in community question answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 326–333, Vancouver, Canada. Association for Computational Linguistics.

Debasis Ganguly, Dwaipayan Roy, Mandar Mitra, and Gareth JF Jones. 2015. Word embedding based generalized language model for information retrieval. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 795–798.

Jianfeng Gao, Patrick Pantel, Michael Gamon, Xiaodong He, and Li Deng. 2014. Modeling interestingness with deep neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2–13, Doha, Qatar. Association for Computational Linguistics.

Luyu Gao and Jamie Callan. 2021. Condenser: a pre-training architecture for dense retrieval. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 981–993, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Luyu Gao and Jamie Callan. 2022. Unsupervised corpus aware language model pre-training for dense passage retrieval. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2843–2853, Dublin, Ireland. Association for Computational Linguistics.

Luyu Gao, Zhuyun Dai, and Jamie Callan. 2020. Understanding bert rankers under distillation. *arXiv preprint arXiv:2007.11088*.

Luyu Gao, Zhuyun Dai, and Jamie Callan. 2021a. COIL: Revisit exact lexical match in information retrieval with contextualized inverted list. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3030–3042, Online. Association for Computational Linguistics.

Luyu Gao, Zhuyun Dai, and Jamie Callan. 2021b. Rethink training of bert rerankers in multi-stage retrieval pipeline. *arXiv preprint arXiv:2101.08751*.

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.

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

Ralf Herbrich, Thore Graepel, and Klaus Obermayer. 1999. *Support vector learning for ordinal regression*. IET.

Sebastian Hofstätter, Sophia Althammer, Michael Schröder, Mete Sertkan, and Allan Hanbury. 2020. Improving efficient neural ranking models with cross-architecture knowledge distillation. *arXiv preprint arXiv:2010.02666*.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 39–48.

Tuan Lai, Quan Hung Tran, Trung Bui, and Daisuke Kihara. 2019. A gated self-attention memory network for answer selection. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5953–5959, Hong Kong, China. Association for Computational Linguistics.

Md Tahmid Rahman Laskar, Jimmy Xiangji Huang, and Enamul Hoque. 2020. Contextualized embeddings based transformer encoder for sentence similarity modeling in answer selection task. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 5505–5514, Marseille, France. European Language Resources Association.

Canjia Li, Andrew Yates, Sean MacAvaney, Ben He, and Yingfei Sun. 2020. Parade: Passage representation aggregation for document reranking. *arXiv preprint arXiv:2008.09093*.

Hang Li. 2011. A short introduction to learning to rank. *IEICE TRANSACTIONS on Information and Systems*, 94(10):1854–1862.

Ping Li, Qiang Wu, and Christopher Burges. 2007. Mcrank: Learning to rank using multiple classification and gradient boosting. *Advances in neural information processing systems*, 20:897–904.

Tie-Yan Liu. 2011. *Learning to rank for information retrieval*. Springer Science & Business Media.

Xiaofei Ma, Cicero Nogueira dos Santos, and Andrew O. Arnold. 2021. Contrastive fine-tuning improves robustness for neural rankers. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 570–582, Online. Association for Computational Linguistics.

Melvin Earl Maron and John Larry Kuhns. 1960. On relevance, probabilistic indexing and information retrieval. *Journal of the ACM (JACM)*, 7(3):216–244.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Preslav Nakov, Doris Hoogeveen, Lluís Màrquez, Alessandro Moschitti, Hamdy Mubarak, Timothy Baldwin, and Karin Verspoor. 2017. SemEval-2017 task 3: Community question answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 27–48, Vancouver, Canada. Association for Computational Linguistics.

Preslav Nakov, Lluís Màrquez, Walid Magdy, Alessandro Moschitti, Jim Glass, and Bilal Randeree. 2015. SemEval-2015 task 3: Answer selection in community question answering. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 269–281, Denver, Colorado. Association for Computational Linguistics.

Preslav Nakov, Lluís Màrquez, Alessandro Moschitti, Walid Magdy, Hamdy Mubarak, Abed Alhakim Freihat, Jim Glass, and Bilal Randeree. 2016. SemEval-2016 task 3: Community question answering. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 525–545, San Diego, California. Association for Computational Linguistics.

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. In *CoCo@ NIPS*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2021. RocketQA: An optimized training approach to dense passage retrieval for open-domain question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5835–5847, Online. Association for Computational Linguistics.

Ruiyang Ren, Shangwen Lv, Yingqi Qu, Jing Liu, Wayne Xin Zhao, QiaoQiao She, Hua Wu, Haifeng Wang, and Ji-Rong Wen. 2021a. PAIR: Leveraging passage-centric similarity relation for improving dense passage retrieval. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2173–2183, Online. Association for Computational Linguistics.

Ruiyang Ren, Yingqi Qu, Jing Liu, Wayne Xin Zhao, QiaoQiao She, Hua Wu, Haifeng Wang, and Ji-Rong Wen. 2021b. RocketQAv2: A joint training method for dense passage retrieval and passage re-ranking. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2825–2835, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Stephen Robertson and Hugo Zaragoza. 2009. *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc.

Stephen E Robertson. 1997. Overview of the okapi projects. *Journal of documentation*.

Lei Sha, Xiaodong Zhang, Feng Qian, Baobao Chang, and Zhifang Sui. 2018. A multi-view fusion neural network for answer selection. In *Thirty-second AAAI conference on artificial intelligence*.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pages 4596–4604. PMLR.

Yue Shi, Martha Larson, and Alan Hanjalic. 2010. List-wise learning to rank with matrix factorization for collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 269–272.

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2020. Ernie 2.0: A continual pre-training framework for language understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8968–8975.

Kateryna Tymoshenko, Daniele Bonadiman, and Alessandro Moschitti. 2016. Convolutional neural networks vs. convolution kernels: Feature engineering for answer sentence reranking. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1268–1278, San Diego, California. Association for Computational Linguistics.

Hamed Valizadegan, Rong Jin, Ruofei Zhang, and Jianchang Mao. 2009. Learning to rank by optimizing ndcg measure. In *NIPS*, volume 22, pages 1883–1891.

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*, pages 1192–1199.

Ji Xin, Rodrigo Nogueira, Yaoliang Yu, and Jimmy Lin. 2020. Early exiting bert for efficient document ranking. In *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing*, pages 83–88.

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.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate nearest neighbor negative contrastive learning for dense text retrieval. *arXiv preprint arXiv:2007.00808*.

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 on Research and Development in Information Retrieval*, pages 1253–1256.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. WikiQA: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018, Lisbon, Portugal. Association for Computational Linguistics.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. In *NeurIPS*.

Xuanyu Zhang and Qing Yang. 2021. Dml: Dynamic multi-granularity learning for bert-based document reranking. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 3642–3646.

Xiaofeng Zhu and Diego Klabjan. 2020. Listwise learning to rank by exploring unique ratings. In *Proceedings of the 13th international conference on web search and data mining*, pages 798–806.

Shengyao Zhuang and Guido Zuccon. 2021. Fast passage re-ranking with contextualized exact term matching and efficient passage expansion. *arXiv preprint arXiv:2108.08513*.

A Dataset

MS MACRO Passage. MS MACRO Passage dataset is one of the largest relevance dataset consisting of over 500k queries with 8.8 million passages. The dataset is built with an MRR metric, where each query has at least one high-relevant passage labeled as the ground truth. Different from most previous relevance dataset, their queries are collected from real-user, which can best represent our daily needs of information.

MS MACRO Document. MS MACRO Document dataset is a recently released relevance dataset which is consist of over 367k queries and 3.2 million documents. Similar with the MS MACRO Passage dataset, it designed for document retrieval and re-ranking.

WikiQA. WikiQA (Yang et al., 2015) is an open domain QA dataset created from Wikipedia. The questions can be viewed as user queries, and the model will generate a permutation from the answers.

SemEvalCQA. SemEvalCQA (Nakov et al., 2015, 2016, 2017) is a series of datasets collected from Qatar Living Forums[1]. Each candidate is labeled with either "Good", "Potentially Useful" or "BAD". Here, we follow CETE (Laskar et al., 2020) to set "Good" as ground truth and the other two kinds of labels as negative. We test our models on the sub-task A of the dataset.

B Baselines

MB25. BM25 (Robertson and Zaragoza, 2009) is a popular sparse retrieval method which is widely used by search engines. The relevance score between a query and a candidate is calculated based on the bag-of-words representations. We use the baseline provided by Anserini (Yang et al., 2017) to run BM25.

ColBERT. The contextualized late interaction over BERT (ColBERT) (Khattab and Zaharia, 2020) uses a late interaction consists of 1) dual-encoder that independently encodes the query and the document using BERT, 2) max-sim operator that interacts every query embedding with all document embeddings, and 3) summed scalar outputs across query terms.

TFR-BERT. The TFR-BERT (Han et al., 2020) fine-tunes BERT in cross-encoder setting to extract the representation of each query-document pair, and trains an ensemble of a point-wise, a pair-wise and a list-wise prediction models to optimize the performance.

RocketQAv2. RocketQAv2 (Ren et al., 2021b) jointly trains a retrieval model and a re-ranking model for more effective training. To alleviate the false negative problem, they perform data augmentation on the negative samples, which further improves the model performance.

HDCT+LCE. Gao et al. (Gao et al., 2021b) designs a cross-encoder that evaluates the candidate relevance using a localized contrastive estimation (LCE) loss, which focus on learning on hard negatives rather than randomly sampled noisy negatives. They use Context-aware Hierarchical Document Term weighting framework(HDCT) (Dai and Callan, 2020) as their retriever, which projects BERT representation into weight of terms, and provides a widely-used document retrieval baseline.

DML. The Dynamic Multi-Granularity Learning(DML) (Zhang and Yang, 2021) integrate Gaussian function into the loss calculation, which pays more attention to hard negatives while avoids the influence of confusing samples. They utilize the Approximate nearest neighbor negative contrastive learning (ANCE) (Xiong et al., 2020) as the document retriever, which extracts global negatives rather than random or in-batch local negatives. We reproduce the retrieval candidates by the MaxP method for document re-ranking, where the document is split to at most four 512-token passages.

KeLP. The Kernel-based Learning Platform(KeLP) (Filice et al., 2017) uses kernel-based classifiers to sort the instances and produce the final ranking. Especially, it employs a SVM learning algorithm which operates on a combination of multiple kernels to extract information.

MVFNN. The Multi-View Fusion Neural Network(MVFNN) uses an attention-based Recurrent Neural Network model to integrate the four views of a QA pair. These four views are named as inquiry type view, inquiry main verb view, inquiry semantic view, and co-attention view.

CETE. The Contextualized Embeddings based

[1]https://www.qatarliving.com/forum

Dataset	#Questions			#Candidate Answers		
	Train	Dev	Test	Train	Dev	Test
MS MACRO P.	502,939	6,980	6,837	8,841,823	-	-
MS MACRO D.	367,013	5,193	5,793	3,213,835	-	-
WikiQA	2,118	296	633	20,360	2,733	6165
SemEval-2015	2,600	300	329	16,541	1,645	1,976
SemEval-2016	4,879	244	327	36,198	2,440	3,270
SemEval-2017	4,879	244	293	36,198	2,440	2,930

Table 4: The statistics of used datasets. For MS MACRO datasets, the candidates are shared for training, developing and testing. P. represents passage dataset, D. represents document dataset.

Transformer Encoder(CETE) (Laskar et al., 2020) uses a cross-encoder to measure the similarity of sentence pairs. CETE also compares the performance of cross/dual encoders and different PLMs.