Reranking Passages with Coarse-to-Fine Neural Retriever using List-Context Information

Hongyin Zhu
hongyin_zhu@163.com

Abstract

Passage reranking is a crucial task in many applications, particularly when dealing with large-scale documents. Traditional neural architectures are limited in retrieving the best passage for a question because they usually match the question to each passage separately, seldom considering contextual information in other passages that can provide comparison and reference information. This paper presents a list-context attention mechanism to augment the passage representation by incorporating the list-context information from other candidates. The proposed coarse-to-fine (C2F) neural retriever addresses the out-of-memory limitation of the passage attention mechanism by dividing the list-context modeling process into two sub-processes, allowing for efficient encoding of context information from a large number of candidate answers. This method can be generally used to encode context information from any number of candidate answers in one pass. Different from most multi-stage information retrieval architectures, this model integrates the coarse and fine rankers into the joint optimization process, allowing for feedback between the two layers to update the model simultaneously. Experiments demonstrate the effectiveness of the proposed approach.

Table 1: An example for passage reranking

Question: What causes heart disease?
Passages:
A. **Cardiovascular disease** (also called **heart disease**) is a class of diseases that involve the heart or blood vessels (arteries , capillaries , and veins).
B. Cardiovascular disease refers to any disease that affects the cardiovascular system , principally cardiac disease, vascular diseases of the brain and kidney , and peripheral arterial disease.
C. The causes of **cardiovascular disease** are diverse but atherosclerosis and/or hypertension are the most common.
D. Additionally, with aging come a number of physiological and morphological changes that alter cardiovascular function and lead to subsequently increased risk of cardiovascular disease, even in healthy asymptomatic individuals.
...
Answer: C

Introduction

Passage reranking (Zhu et al. 2021) is a subtask of question answering and machine reading comprehension that involves retrieving one or several passages (text options) that can best answer a given question. Each passage contains one or several sentences, as shown in Table 1. The most common approach is to model the question-answer (QA) pair (Severyn and Moschitti 2016), then compute various similarity measures between obtained representations. Finally, we can choose the high-score candidates as the answer.

Prior works focus on enriching the features for sentence representation or enhancing the interaction between a sentence pair, rarely considering the impact of other candidates, so the relationship between candidates is not fully exploited. Humans consider not only whether each independent passage matches the question but also whether there is a better choice. This implies that enhancing the representation of each passage by interacting with the context information from other candidates can improve the result's confidence. Context-independent representations potentially limit sentence semantics when other sentences also provide useful context information for this question. For instance, passage A in Table 1 explains that "Cardiovascular disease" refers to heart disease. Although passage C does not mention heart disease, we can still get relevant information from candidate A.

Modeling list-context is a nontrivial task. The first challenge is to emphasize the comparative and reference information. Previous studies have tackled this challenge by utilizing hierarchical GRU-RNNs to consider context information among sentences (Tan et al. 2018). However, they did not explicitly enhance sentence representation by leveraging other candidates. Another approach is multi-mention learning, which models multiple mentions in a document to answer questions (Swayamdipta, Parikh, and Kwiatkowski 2017). While these methods have made significant contributions, they do not explicitly model the context information in the passage list. To address this limitation, we propose a list-context attention mechanism, composed of static attention and adaptive attention. This mechanism injects list-context information into the passage, allowing each candidate to consider the whole list semantics by attending to all the candidates. Additionally, adaptive attention enables each passage to adaptively interact with other candidates by considering their correlation information.

The second challenge is the large number of candidate passages. It's difficult to analyze thousands of passages simultaneously without running into technical issues. Previous research typically broke down a long list of passages into smaller parts and then created a context-independent representation of each sentence pair. This approach often relies on a multi-stage retrieval architecture (Mackenzie et al. 2018), where the candidate documents are repeatedly narrowed down and reordered. Our paper addresses this issue by applying a two-stage retrieval approach to the neural model. Unlike previous methods, our model streamlines the multi-stage process into a two-level (coarse-to-fine) model. First, it selects a good passage set roughly, and then it fine-tunes the selection by ignoring irrelevant instances that are far from the classification hyperplane. By training two layers of model parameters jointly, our approach enables them to collaborate and interact more effectively.

We introduce a bucket policy learning (BPL) algorithm to model the two-level selection process end-to-end. The coarse selection sub-process uses a scoring function to rank the sentences in the bucket memory and dynamically selects the top-k scoring sentences for further processing. In addition to the coarse selection sub-process, our model also incorporates a fine ranker to further refine the representations. Our model performs passage reranking and parameter optimization simultaneously. We conduct experiments on the WIKIQA (Yang, Yih, and Meek 2015) and MARCO (Nguyen et al. 2016) datasets. The results show the effectiveness of our approach. The distinctive properties of this paper are as follows:

(i) This paper introduces the idea of enhancing passage representation by considering context information from other candidates.

(ii) This paper proposes a list-context attention mechanism, composed of static attention and adaptive attention, to model list-context information.

(iii) This paper introduces a C2FRetriever, which can select answers from coarse to fine level in one pass.

Related Work

Previous work employs deep learning models to enhance sentence representations and compute their similarity. (Rocktäschel et al. 2015) propose a textual entailment model that models word relations between sentences by using word-to-word attention on an LSTM-RNN. (Severyn and Moschitti 2016) proposes CNN$_R$ to consider overlapping words to encode relational information between question and answer. (Yin et al. 2016) proposes 3 attention methods on a CNN model (ABCNN) to encode mutual interactions between sentences. (Miller et al. 2016) proposes key-value memory networks (KV-MemeNN) to select answers by using key-value structured facts in the model memory. (Wang, Hamza, and Florian 2017) proposes a Bilateral Multi-View Matching (BiMPM) model, which utilizes an attention mechanism to model the mutual interaction of sentences at different scales. (Bachrach et al. 2017) apply two attention operations to capture more word-level contextual information, but their work still focuses on enhancing sentence-pair representations without considering list-level contextual information. A work close to ours is the hierarchical GRU-RNN (Tan et al. 2018), which is used to model word-level and sentence-level matching and provide a kind of contextual information. However, their approach does not explicitly enhance sentence representations by using contextual information from other candidates. Our approach incorporates list-context information to augment sentence representation. (Ran et al. 2019) propose a Paragraph Comparison Network (OCN) for Multiple Choice Reading Comprehension.

(Guo et al. 2016) propose the Deep Dependent Matching Model (DRMM), which introduces a histogram pooling technique to summarize the translation matrix. (Xiong et al. 2017) propose KNRM, which uses a kernel pooling layer to softly compute the frequency of word pairs at different similarity levels. MARCO's official baseline (Mitra, Diaz, and Craswell 2017) uses two separate DNNs to model query-document relevance using local and distributed representations, respectively. Conv-KNRM (Dai et al. 2018) enhances KNRM by utilizing CNN to compose n-gram embeddings from word embeddings and cross-matched n-grams of different lengths. SAN + BERT base (Liu, Duh, and Gao 2018) maintains a state and iteratively refines its predictions.

BERT (Devlin et al. 2018) provides a way to directly model sentence-pair interactions. BERT is a pretrained language model based on the bidirectional transformer (Vaswani et al. 2017) and has been shown to be effective at generating rich sentence-pair representations. Previous work mainly uses a multi-stage retrieval architecture (Mackenzie et al. 2018) in web search systems. A set of candidate documents is generated using a series of increasingly expensive machine learning techniques. (Chen et al. 2017) present a method for optimizing cascaded ranking models. Our C2FRetriever is trained in an end-to-end manner so that the parameters of the coarse ranker and fine ranker can be jointly optimized for better collaboration between the two levels. This approach can be used to simplify many cascaded pipelines (Zhu 2022a; Zhu 2023a; Zhu 2022b; Zhu 2023b).

Approach

Suppose we have a question Q with l tokens $\{w_1^q, w_2^q, ..., w_l^q\}$, and a candidate answer set O with n passages $\{O_1, O_2, ..., O_n\}$. n is the number of passages that can vary over a wide range (1$\sim$1000). Each passage O_i contains one or several sentences (passage) which consists of o_i tokens $\{w_1^o, w_2^o, ..., w_{o_i}\}$. The label $y_i \in \{0, 1\}$ with 1 denotes a positive answer and 0 otherwise. The goal of the model is to score each passage based on how well it answers the question, and then rank the passages based on the score.

Our main effort lies in designing a deep learning architecture that enhances representations by considering contextual information of other candidates. Its main building block has two layers, the coarse ranker and the fine ranker. In the following, we first describe the two layers and then describe the training algorithm.

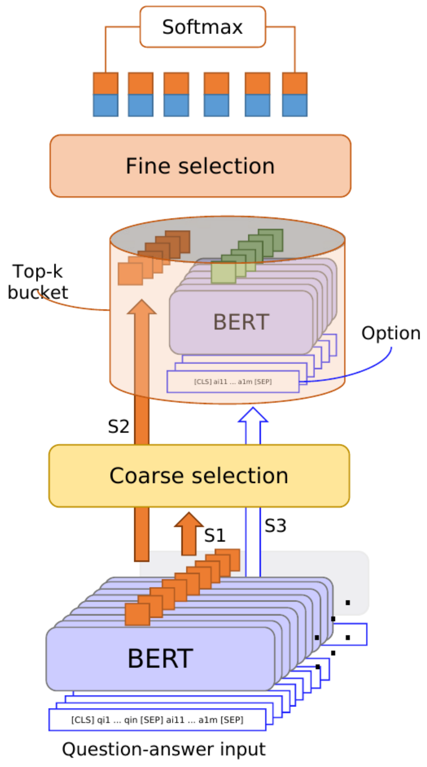


Figure 1: Schematic diagram of the model structure

Coarse Ranker

The architecture of the coarse ranker is shown in the lower part of Figure 1. This layer aims to filter some answers that are irrelevant to the given question to generate an intermediate set for performing fine selection. We use a BERT model (Devlin et al. 2018) to generate representations of QA pairs and a single-layer neural network to compute matching scores.

Passage O_i is concatenated with question Q to form a complete sequence, denoted as $\langle$[CLS]; Q; [SEP]; O_i; [SEP]$\rangle$.

$$O_i^{(c)} = \mathcal{F}_{bert}(\langle \text{[CLS]}; Q; \text{[SEP]}; O_i; \text{[SEP]} \rangle) \tag{1}$$

where $O_i^{(c)} \in \mathbb{R}^d$ is the QA representation. $\mathcal{F}_{bert}(\cdot)$ denotes the network defined in (Devlin et al. 2018). The representations are then projected to matching scores using a single-layer neural network.

$$p_i^{(c)} = \sigma(W^c O_i^{(c)} + b^c) \tag{2}$$

where $W^c \in \mathbb{R}^{d\times 1}$ and b^c is a scalar. $\sigma(\cdot)$ is the sigmoid activation function. This layer takes all question-answer pairs as input $[(Q, O_1), (Q, O_2), ..., (Q, O_n)]$, and then the model assigns a score to each candidate.

Here we use different paths to represent different functions. The coarse ranker creates a buffer as a bucket which is used to maintain the top-k passages ever seen before. According to p_i, passages are dynamically ranked through path S1 (ranking path). This path contains the functions shown below.

$$\mathbf{h}_t = \mathcal{K}(p_i, \mathbf{h}_{t-1}) \tag{3}$$

where $\mathbf{h}_0$ is the initial state of the empty bucket. $\mathbf{h}_t$ is the bucket state after t step update and contains the selected passages. $\mathcal{K}(\cdot)$ denotes the ranking function.

Prior early-stage models typically process all the candidates and then retain the top-k candidates which are also written to disk. Different from them, we incrementally maintain the bucket memory which only retains the top-k scoring QA pairs. k is a hyper-parameter. Then, path S2 (the remaining path of the QA pair) is used to connect the corresponding QA representation to the fine ranker. The path contains the function of the equation (4).

$$\mathbf{O}_{\mathbf{h}_t}^{(f)} = \mathcal{C}(\mathbf{O}_{:t}^{(c)}, \mathbf{h}_t) \tag{4}$$

where $\mathbf{O}_{:t}^{(c)}$ represents the $[O_1^{(c)}, O_2^{(c)}, ..., O_t^{(c)}]$. $\mathbf{O}_{\mathbf{h}_t}^{(f)}$ is the matrix for top-k QA pairs. $^{(f)}$ denotes the fine ranker. $\mathcal{C}(\cdot)$ is the selection function.

Furthermore, path S3 is the remaining path of paragraphs, aiming at extracting top-k candidate paragraphs for generating list context information. This path can be described as the equation (5).

$$\mathbf{P}_{\mathbf{h}_t}^{(f)} = \mathcal{C}(\mathbf{P}_{:t}^{(c)}, \mathbf{h}_t) \tag{5}$$

where $\mathbf{P}_{\mathbf{h}_t}^{(f)}$ is the matrix for top-k passages. $P_i^{(c)}$ is the representation for i-th passage, which can be calculated by equation (6).

$$P_i^{(c)} = \mathcal{F}_{bert}(\langle \text{[CLS]}; O_i; \text{[SEP]} \rangle) \tag{6}$$

Finally, the selected passages and representations in the bucket are sent to the fine ranker.

Fine Ranker

The inputs to the fine ranker are vectors of top-k QA pairs $\mathbf{O}_{\mathbf{h}_t}^{(f)}$ and top-k passages $\mathbf{P}_{\mathbf{h}_t}^{(f)}$. Figure 2 shows the architecture of the fine ranker, which consists of a list-context attention mechanism combining static and adaptive attention.

Static attention. To capture and compose the context semantics from the answer list, our model uses an attention mechanism (Parikh et al. 2016; Yang et al. 2016) which achieves the best performance in different alternatives to obtain the list-context representation. Our model extracts informative passages and aggregates their representations to form the list-context representation V_l. This method first measures the weight of each passage in the list context.

$$u_i = C_l^T P_i^{(c)} \tag{7}$$

where $C_l \in \mathbb{R}^d$ is the context vector that can be jointly trained. Then, this model computes the normalized weight of each passage through a softmax function and aggregates these representations.

$$\alpha_i = \frac{\exp(u_i - \max(\mathbf{u}))}{\sum_j \exp(u_j - \max(\mathbf{u}))} \tag{8}$$

where $\max(\mathbf{u})$ gets the max value of $[u_0, u_1, ..., u_k]$ where k is the candidate number. This operation encodes the list context into a vector which summarizes the main semantics of this list. This allows the model to softly consider all candidates in the entire list.

$$V_l = \sum_i \alpha_i \cdot P_i^{(c)} \tag{9}$$

where V_l is the list-context representation.

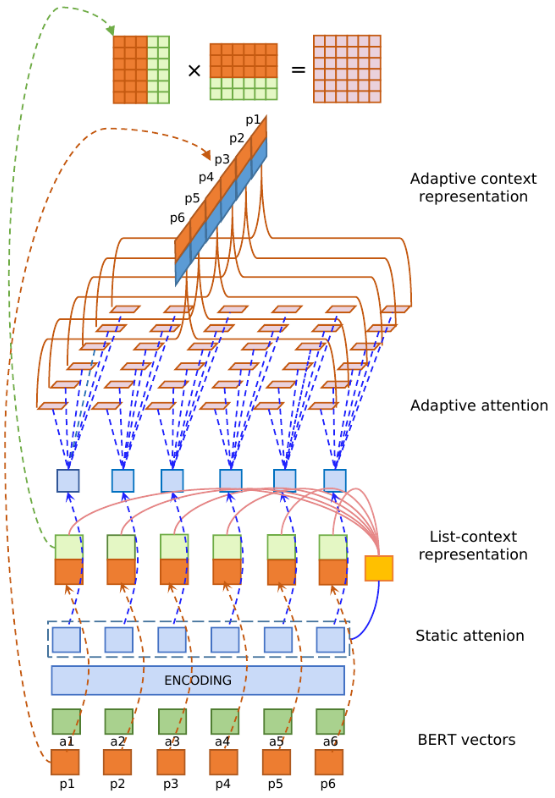


Figure 2: Schematic diagram of the fine ranker

Adaptive attention. While the static attention supplements the list-context information for each passage, V_l is the same for all candidates and does not consider the question. Ideally, we would like to overcome the above two problems, by considering list-context information and adaptively incorporating the context information for each passage based on the question. Our model resolves this problem by using adaptive attention which injects the correlation information of passages into passage representation directly, as shown in Figure 2. This method first computes the correlation weight between these passages by considering the semantic similarity of the QA pair and the list-context information.

$$\begin{aligned} w_{ij} &= [{O_i^{(c)}}^T, V_l^T] \begin{bmatrix} O_i^{(c)} \\ V_l \end{bmatrix} \\ &= {O_i^{(c)}}^T O_j^{(c)} + ||V_l||_2 \end{aligned} \tag{10}$$

where $|| \cdot ||_2$ is the L2-norm. Then, this method obtains normalized correlation weights through a softmax function.

$$\beta_{ij} = \frac{\exp(w_{ij})}{\sum_j \exp(w_{ij})} \tag{11}$$

The adaptive context representation is obtained by calculating the weighted sum of the passage representations.

$$Z_i = \sum_j \beta_{ij} \cdot O_j^{(c)} \tag{12}$$

where Z_i is the adaptive context of i-th passage. By using this attention scheme, each passage has a set of adaptive correlation weights with other candidates. This correlation information can softly aggregate the passage representations flexibly. This interaction operation encodes the correlation between candidates while also considering the list-context information and question.

Then, the ranking score of the options in the bucket can be calculated as below.

$$p^{(f)} = softmax(W^{(f)}[\mathbf{P}_{\mathbf{h}_t}^{(f)}; \mathbf{Z}]) \tag{13}$$

where $W^{(f)} \in \mathbb{R}^{1\times\tau}$ and $b^{(f)}$ are linear composition matrix and bias. τ is vector dimension. $[;]$ denotes the concatenation operation. $p^{(f)}$ is the score vector.

Training Algorithm

Prior multi-stage retrieval methods typically cascade different machine learning techniques. Different from the cascaded ranking architecture, we introduce the BPL algorithm to train a two-level network for the two-stage retrieval problem. This algorithm performs ranking operations during the training process and can optimize the two-stage retrieval processes jointly.

As shown in Algorithm 1, in line 1, this model first initializes the bucket memory. Then, during model training, this model adds the ground truth answer to the memory, in line 2. In lines 3-12, this model maintains the bucket memory by coarsely selecting the top-k candidates. In lines 4-5, this model calculates the matching score based on the BERT representation. If the current candidate is a positive answer, its index j, matching score $p^{(c)}$ and representation p_{Bert} will be added to the bucket, in line 7. If the current candidate does not match this question, this model will maintain the top-k candidates based on their scores. In line 13, this model merges the bucket memory. In line 14, this model incorporates the context information of other sentences in the fine ranker to augment the passage representation. Finally, this model calculates the loss values of coarse and fine rankers. Then, we can update the model parameters by back-propagation using an optimization algorithm.

After getting the top-k sentences, this model will carry out the fine selection process as described in the Fine Ranker

Algorithm 1 The BPL algorithm

```
Input: QA pairs $pair$, Label $l$, Bucket size $n$
Output: The loss values of sub-layers
1: Initialize lists $p_i$,$p_v$,$p_{bert}$,$n_i$,$n_v$,$n_{bert}$
2: Get the index $p_{in}$ of positive passages
3: for $j \leftarrow 0, pair.length - 1$ do
4:     $O_j^{(c)} = \mathcal{F}_{bert}(pair[j])$
5:     $p_j^{(c)} = \sigma(Linear(O_j^{(c)}))$
6:     if $j$ in $p_{in}$ then
7:         Update ($p_i$,$p_v$,$p_{bert}$) using ($j$,$p^{(c)}$,$O_j^{(c)}$) and maintain the score ($p^{(c)}$) order
8:     else
9:         Update ($n_i$,$n_v$,$n_{bert}$) using ($j$,$p^{(c)}$,$O_j^{(c)}$) and maintain the score ($p^{(c)}$) order
10:        Maintain the size of the above three ordered lists smaller than ($n$-$p_i.length$)
11:    end if
12: end for
13: Merge two groups of ordered list ($p_i$,$p_v$,$p_{bert}$) and ($n_i$,$n_v$,$n_{bert}$) into bucket memory ($c_i$,$c_v$,$c_{bert}$)
14: Generate the representations $p^{(f)}$ of fine ranker from passage list
15: Calculate the loss value of two selection processes $loss_c$($p^{(c)}$,$l[c_i]$), $loss_f$($p^{(f)}$,$l[c_i]$)
```

subsection. This model is trained using the log loss of two-level selection as shown below.

$$\mathcal{L}^{(c)} = -\sum_j [y_j \cdot \log p_j^{(c)} + (1 - y_j) \cdot \log(1 - p_j^{(c)})] \tag{14}$$

$$\mathcal{L}^{(f)} = -\sum_j [y_j \cdot \log p_j^{(f)} + (1 - y_j) \cdot \log(1 - p_j^{(f)})] \tag{15}$$

where y_j is the label of the j-th passage. p_{c_j} and p_{f_j} are the predicted score of j-th passage in the coarse and fine rankers respectively. Then, these two layers are jointly trained to find a balance between passage selection and joint parameter optimization, as shown below.

$$\min_\theta \mathcal{L} = \sum_i^{|\mathbb{D}|} (\mathcal{L}_i^{(c)} + \lambda \mathcal{L}_i^{(f)})$$

where λ[1] is a hyper-parameter to weigh the influence of the fine ranker.

The asymptotic complexity is described as follows. Assuming that all hidden dimensions are ρ, the complexity of matrix ($\rho \times \rho$)-vector ($\rho \times 1$) multiplication is $O(\rho^2)$. BERT takes $O(C_{bert})$. For the coarse ranker, calculating the BERT representation of n QA pairs and passages takes $O(nC_{bert})$. To maintain the bucket we need a top-k sort operation which takes $O(nk)$ at the worst case.

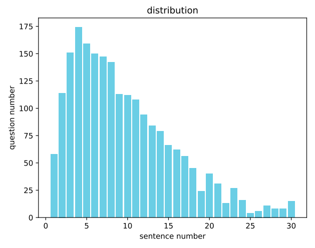
[1]We set λ=1.0.

For the fine ranker, the list context information requires $O(\rho^2)$. The adaptive context information requires $O(k\rho^2)$. Therefore, the total complexity is $O(nC_{bert} + k\rho^2)$. For BERT, it mainly includes matrix-vector multiplication, so the optimized calculation requires $O(ml\rho^2)$, where m is the number of matrix-vector multiplications, and l is the sequence length. The computational complexity of this model is still close to that of BERT.

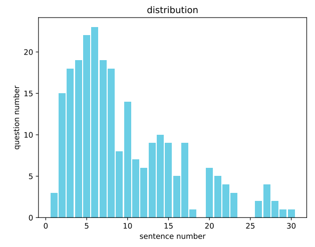
Experiments

Datasets

WIKIQA This is a standard open-domain QA dataset. The questions are sampled from the Bing query logs, and the candidate sentences are extracted from paragraphs of the associated Wiki pages. This dataset includes 3,047 questions and 29,258 sentences (Yang, Yih, and Meek 2015), where 1,473 candidate passages are labeled as answer sentences. Each passage contains only one sentence. We use the standard data splits in experiments. Figure 3 visualizes the data distribution of this dataset. The x- and y- axes denote the number of candidate sentences and the number of questions respectively. The candidate number of each question ranges from 1 to 30 and the average candidate number is 9.6. The average length of question and answer are 6.5 and 25.1 words respectively.



(a) Training set

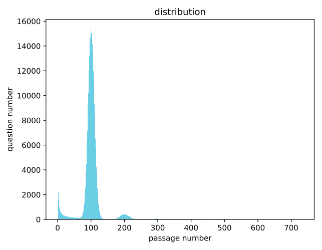


(b) Test set

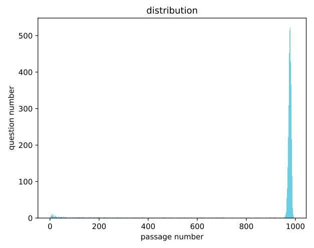
Figure 3: Passage distribution of the WIKIQA dataset

MARCO 2.0 This is a large-scale machine reading comprehension dataset (Nguyen et al. 2016) sampled from Bing's search query logs. We choose the dataset for the passage re-ranking task. Figure 4 visualizes the data distribution of this dataset. The x- and y- axes denote the number of candidate passages and the number of questions respectively. The training set contains 398,792 questions. The number of passages in each question ranges from 2 to 732.

The question length ranges from 1 to 38 words. The passage length ranges from 1 to 362 words. Each question average has 100.7 passage candidates. On average, each question has one relevant passage. The development set and test set contain 6,980 and 6,837 questions respectively. Each question has 1000 passages candidates retrieved with BM25 from the MS MARCO corpus. In the test set, the question length ranges from 2 to 30 words. The passage length ranges from 1 to 287 words.



(a) Training set



(b) Test set

Figure 4: Passage distribution of the MARCO dataset

Hyper-parameters

This paper uses the uncased BERT-base model to generate sentence pair representations. We use the output of "[CLS]" on the last layer of BERT as a representation of the QA pair. The encoding layer uses a single-layer neural network with the hyperbolic tangent activation function to generate the 200-D vector representations. We use 40 tokens and 200 tokens as the maximum question length and the maximum answer length respectively. We set the bucket size to 15.

Adam (Kingma and Ba 2014) optimization algorithm is adopted to update the model parameters. We use a bert-base-uncased model for fine-tuning on the passage reranking datasets. The models run on an Intel(R) Xeon(R) Platinum 8163 CPU @ 2.50GHz (Mem: 330G) & 8 Tesla V100s and an Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz (Mem: 256) & 8 RTX 2080Tis.

Evaluation

This paper uses Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR) to evaluate the performance of the model. For the MARCO dataset, we use the official script to evaluate our results. This script calculates the MRR@10 which considers only the top 10 passages.

Results on the WIKIQA dataset

Table 2 presents the experimental result. Most baseline deep learning models typically design effective feature extraction schemes to better derive features from QA pairs for calculating question-answer similarity. The BERT model has a similar goal of mapping a QA pair to a valid representation. The proposed models are the following: (i) BERT base is the simple BERT model fine-tuned on the WIKIQA dataset. The inputs to the model are all QA pairs. (ii) BERT base + MaxPooling denotes the BERT model add the max pooling. We organize the QA pairs according to the document and consider all the sentences. Note that this setting cannot tackle a situation that has unlimited candidates. The max pooling extracts the document information. Because the passage number is less than 30, so we can get the max pooling. (iii) C2Retriever is the proposed model with list and adaptive context information.

We observe that our model outperforms the BERT base and improves 1.93% MAP and 0.71% MRR respectively to wGRU-sGRU-G_{12}-Cnt (Tan et al. 2018). Considering the context information by max pooling improves 2.88% MAP and 2.92% MRR respectively. This means that considering the document-level context information is helpful for each passage. We observe that our C2FRetriever improves 6.17% MAP and 6.82% MRR on the BERT base model. This is because our network considers context information from other answers. The sentences of WIKIQA come from consecutive sentences in the wiki page paragraphs, so other sentences can also provide rich contextual information. Our network softly incorporates the document context information into the sentence representation.

Results on the MARCO Dataset

Table 3 lists the results on the MARCO dataset. Compared with the WIKIQA dataset, each passage may contain two or more sentences so the passage length varies over a wide range. The passages of any question are from different documents retrieved by a search engine, so the continuity of passages is also reduced. This experiment can better test the versatility of our method. (Nogueira and Cho 2019) fine-tune the BERT base and large models and simply use the matching score of QA pairs for ranking. Note that they train their models on multiple TPUs with appropriate batch size and sequence length, which can help to better adapt the representation to the target domain. This device significantly improves model results.

Our approach potentially enables the usage of BERT based ranking model with lower equipment requirements. However, the drawback is that we compare the model performance by truncating the passage length. We further train our model by considering longer sequences (400 tokens) with multiple GPUs. Our model achieves further improvement by 1.7%. We achieve 0.377 on the development set, which improves 3% from the TPU BERT base. The full-ranking method (Yan (2019)) achieves the highest score, but

[2]This score is generated on the development set.

[3]This score is generated with full ranking, while other models are reranking model.

Table 2: Results on the WIKIQA dataset

Method	MAP	MRR
WordCnt (Yang, Yih, and Meek 2015)	0.4891	0.4924
WgtWordCnt (Yang, Yih, and Meek 2015)	0.5099	0.5132
CNN-CNt (Yang, Yih, and Meek 2015)	0.5170	0.5236
CNN_R (Severyn and Moschitti 2016)	0.6951	0.7107
ABCNN-3 (Yin et al. 2016)	0.6921	0.7108
KV-MemNN (Miller et al. 2016)	0.7069	0.7265
BiMPM (Wang, Hamza, and Florian 2017)	0.7180	0.7310
IARNN-Occam (Wang, Liu, and Zhao 2016)	0.7341	0.7418
CNN-MULT (Wang and Jiang 2016)	0.7433	0.7545
CNN-CTK (Tymoshenko, Bonadiman, and Moschitti 2016)	0.7417	0.7588
wGRU-sGRU-G_{12}-Cnt (Tan et al. 2018)	0.7638	0.7852
BERT base	**0.7831**	**0.7923**
BERT base + MaxPooling	**0.8119**	**0.8215**
C2FRetriever	**0.8448**	**0.8605**

Table 3: Results on the MARCO dataset

Method	MRR@10 Eval
BM25	0.167
LeToR	0.195
Official Baseline (Mitra, Diaz, and Craswell 2017)	0.2517
Conv-KNRM (Xiong et al. 2017)	0.271
IRNet	0.281
BERT base (Nogueira and Cho 2019)	0.347[2]
BERT large (Nogueira and Cho 2019)	0.359
SAN + BERT base (Liu, Duh, and Gao 2018)	0.359
Enriched BERT base + AOA index	0.368
Enriched BERT base + AOA index + CAS + Full	**0.393**[3]
C2FRetriever (200 tokens)	0.347
C2FRetriever (400 tokens)	0.364

Table 4: Model setting ablations on the WIKIQA dataset

Model	MAP	MRR
C2FRetriever	0.8448	0.8605
–List	0.8236	0.8348
–Adaptive	0.8113	0.8215
–Adaptive, List	0.8009	0.8121
–Joint training	0.8178	0.8292
–Adaptive, List, + MaxPooling	0.8119	0.8215
–Adaptive, List, + LSTM	0.8085	0.8200
–Adaptive, List, Two-level	0.7831	0.7923

the drawback is that the training process is a multi-stage pipeline. In contrast, our model only uses joint training and gets the final answer in one pass. This method significantly reduces the problem's complexity.

Compared with prior works, our approach incorporates the context information of other candidates to enhance the passage representation. Each query may have hundreds of retrieved passages from a large corpus. Our network effectively integrates the coarse- and fine-selection processes by simultaneously performing model optimization and passage selection in one pass.

Ablation Study

Table 4 shows the ablation study of the effects of different model settings. A first observation is that the list context and adaptive context information are essential for a good result. Removing the List context information slightly degrades performance. This indicates that the document-level context is necessary. When we train the pipeline model, the result drops (2.7%). This means that joint training is important for the interaction of two-level ranking.

When we replace context information with MaxPooling, the result also drops, but it is better than not considering the context. This means considering document-level context information is helpful and using MaxPooling is a straightforward choice. When we replace MaxPooling with an LSTM encoder, the result slightly drops. This is because the passages may not continuous context model and we have the long-term dependency problem because we consider all pair-wise interactions. When we remove the fine ranker, the result drops. This suggests that the two-level selection scheme is necessary.

We further analyze the impact of query/passage length and number of passages, as shown in Figure 5. We truncate the query length to query:n where n is the maximum number of tokens to query. We observe that, given the same passage length, longer query lengths generally yield better results. For shorter query lengths, i.e. n¡20, longer passage lengths do not always improve results. Since the meaning of the query is not well encoded, longer passages may contain more misleading information. Interpreting and expanding queries is important to improve results. We can conclude that an appropriate combination of query and paragraph lengths and model selection is important for the method.

To evaluate the influence of bucket size, we did extensive experiments based on different bucket sizes, as shown

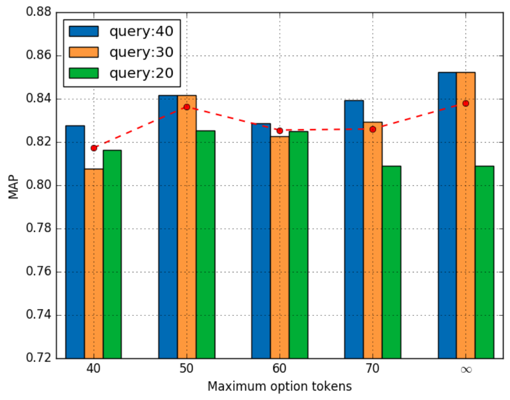


Figure 5: The influence of query/passage length and passage number in fine ranker

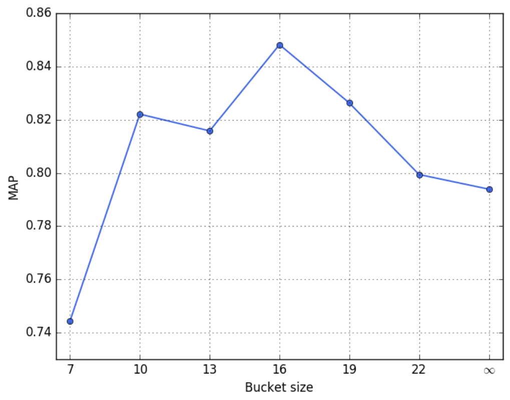


Figure 6: The influence of bucket size

in Figure 6. We observe that this model achieves higher performance with the bucket size=16. As the bucket size increases, the performance improves as it will allow the model to consider more contextual information. When we consider many passages, it can hurt performance because it introduces noise, so considering all candidates is not always a good option. The choice of bucket size is important to the result.

Conclusion and Future work

In this paper, we propose a passage reranking Model with list-context integration for improving the representation of passages in various contexts. Previous studies have often overlooked the importance of list-context information from other candidates. We enhance the passage representation by softly integrating list-context information. Our model addresses the challenge of two-stage joint retrieval by seamlessly integrating coarse and fine rankers. Our model can be trained in a single process, simultaneously optimizing all components, and generating the final answer in a single pass, significantly reducing the problem's complexity.

This paper primarily focuses on addressing the challenge of incorporating list-context information from other candidates. The model has the potential to be extended to other cascaded tasks, such as information extraction (Zhu et al. 2022) and downstream applications, in the future.

References

[Bachrach et al. 2017] Bachrach, Y.; Zukov-Gregoric, A.; Coope, S.; Tovell, E.; Maksak, B.; Rodriguez, J.; McMurtie, C.; and Bordbar, M. 2017. An attention mechanism for neural answer selection using a combined global and local view. In *Proceedings of ICTAI 2017*. IEEE.

[Chen et al. 2017] Chen, R.; Gallagher, L.; Blanco, R.; and Culpepper, J. S. 2017. Efficient cost-aware cascade ranking in multi-stage retrieval. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017*.

[Dai et al. 2018] Dai, Z.; Xiong, C.; Callan, J.; and Liu, Z. 2018. Convolutional neural networks for soft-matching n-grams in ad-hoc search. In *Proceedings of WSDM 2018*. ACM.

[Devlin et al. 2018] Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[Guo et al. 2016] Guo, J.; Fan, Y.; Ai, Q.; and Croft, W. B. 2016. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM.

[Kingma and Ba 2014] Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Liu, Duh, and Gao 2018] Liu, X.; Duh, K.; and Gao, J. 2018. Stochastic answer networks for natural language inference. *arXiv preprint arXiv:1804.07888*.

[Mackenzie et al. 2018] Mackenzie, J.; Culpepper, J. S.; Blanco, R.; Crane, M.; Clarke, C. L.; and Lin, J. 2018. Query driven algorithm selection in early stage retrieval. In *Proceedings of WSDM 2018*. ACM.

[Miller et al. 2016] Miller, A.; Fisch, A.; Dodge, J.; Karimi, A.-H.; Bordes, A.; and Weston, J. 2016. Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126*.

[Mitra, Diaz, and Craswell 2017] Mitra, B.; Diaz, F.; and Craswell, N. 2017. Learning to match using local and distributed representations of text for web search. In *Proceedings of WWW 2017*.

[Nguyen et al. 2016] Nguyen, T.; Rosenberg, M.; Song, X.; Gao, J.; Tiwary, S.; Majumder, R.; and Deng, L. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

[Nogueira and Cho 2019] Nogueira, R., and Cho, K. 2019. Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*.

[Parikh et al. 2016] Parikh, A. P.; Täckström, O.; Das, D.; and Uszkoreit, J. 2016. A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*.

[Ran et al. 2019] Ran, Q.; Li, P.; Hu, W.; and Zhou, J. 2019. Option comparison network for multiple-choice reading comprehension. *arXiv preprint arXiv:1903.03033*.

[Rocktäschel et al. 2015] Rocktäschel, T.; Grefenstette, E.; Hermann, K. M.; Kočiskỳ, T.; and Blunsom, P. 2015. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*.

[Severyn and Moschitti 2016] Severyn, A., and Moschitti, A. 2016. Modeling relational information in question-answer pairs with convolutional neural networks. *arXiv preprint arXiv:1604.01178*.

[Swayamdipta, Parikh, and Kwiatkowski 2017] Swayamdipta, S.; Parikh, A. P.; and Kwiatkowski, T. 2017. Multi-mention learning for reading comprehension with neural cascades. *arXiv preprint arXiv:1711.00894*.

[Tan et al. 2018] Tan, C.; Wei, F.; Zhou, Q.; Yang, N.; Du, B.; Lv, W.; and Zhou, M. 2018. Context-aware answer sentence selection with hierarchical gated recurrent neural networks. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 26(3):540–549.

[Tymoshenko, Bonadiman, and Moschitti 2016] Tymoshenko, K.; Bonadiman, D.; and Moschitti, A. 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*, 1268–1278.

[Vaswani et al. 2017] Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*.

[Wang and Jiang 2016] Wang, S., and Jiang, J. 2016. A compare-aggregate model for matching text sequences. *arXiv preprint arXiv:1611.01747*.

[Wang, Hamza, and Florian 2017] Wang, Z.; Hamza, W.; and Florian, R. 2017. Bilateral multi-perspective matching for natural language sentences. *arXiv preprint arXiv:1702.03814*.

[Wang, Liu, and Zhao 2016] Wang, B.; Liu, K.; and Zhao, J. 2016. Inner attention based recurrent neural networks for answer selection. In *Proceedings of ACL 2016*.

[Xiong et al. 2017] Xiong, C.; Dai, Z.; Callan, J.; Liu, Z.; and Power, R. 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. ACM.

[Yang et al. 2016] Yang, Z.; Yang, D.; Dyer, C.; He, X.; Smola, A.; and Hovy, E. 2016. Hierarchical attention networks for document classification. In *Proceedings of NAACL 2016*.

[Yang, Yih, and Meek 2015] Yang, Y.; Yih, W.-t.; and Meek, C. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of EMNLP 2015*.

[Yin et al. 2016] Yin, W.; Schütze, H.; Xiang, B.; and Zhou, B. 2016. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics* 4:259–272.

[Zhu et al. 2021] Zhu, H.; Tiwari, P.; Ghoneim, A.; and Hossain, M. S. 2021. A collaborative ai-enabled pretrained language model for aiot domain question answering. *IEEE Transactions on Industrial Informatics* 18(5):3387–3396.

[Zhu et al. 2022] Zhu, H.; Tiwari, P.; Zhang, Y.; Gupta, D.; Alharbi, M.; Nguyen, T. G.; and Dehdashti, S. 2022. Switchnet: A modular neural network for adaptive relation extraction. *Computers and Electrical Engineering* 104:108445.

[Zhu 2022a] Zhu, H. 2022a. Financial data analysis application via multi-strategy text processing. *arXiv preprint arXiv:2204.11394*.

[Zhu 2022b] Zhu, H. 2022b. Metaaid: A flexible framework for developing metaverse applications via ai technology and human editing. *arXiv preprint arXiv:2204.01614*.

[Zhu 2023a] Zhu, H. 2023a. Fqp 2.0: Industry trend analysis via hierarchical financial data. *arXiv preprint arXiv:2303.02707*.

[Zhu 2023b] Zhu, H. 2023b. Metaaid 2.0: An extensible framework for developing metaverse applications via human-controllable pre-trained models. *arXiv preprint arXiv:2302.13173*.