JOURNAL OF ARTIFICIAL INTELLIGENCE AND ROBOTICS
Jan-Mar 2024, VOL. 1, ISSUE 1, pp. 19-28
https://doi.org/10.61577/jaiar.2024.100005

RESEAPRO JOURNALS

ORIGINAL ARTICLE

OPEN ACCESS

Reranking passages with coarse-to-fine neural retriever enhanced by list-context information

Hongyin Zhu[1,2]

[1]Architecture Research Department, Inspur Electronic Information Industry Co., Ltd., Beijing, China
[2]National Key Laboratory of High-end Server System, Jinan, China

ABSTRACT

Passage reranking is a critical task in various applications, particularly when dealing with large volumes of documents. Existing neural architectures have limitations in retrieving the most relevant passage for a given question because the semantics of the segmented passages are often incomplete, and they typically match the question to each passage individually, rarely considering contextual information from other passages that could provide comparative 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 with a cache policy learning algorithm, enabling the 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.

KEYWORDS

Passage reranking; List-context attention; Coarse-to-fine neural retriever; Machine reading comprehension; Multi-stage retrieval; Two-level retrieval

ARTICLE HISTORY

Received 23 October 2023;
Revised 27 January 2024;
Accepted 6 February 2024

Introduction

Passage reranking 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 [1]. Each passage contains one or several sentences, as shown in Table 1. The most common approach is to model the question-answer (QA) pair [2], and then compute various similarity measures between obtained representations. Finally, we can choose the high-score candidates as the answer.

Table 1. An example of 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 hypertension are the most common.
D. Additionally, with ageing 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

Recent studies have improved the quality of general text embeddings for representing passages. BAAI general embedding (BGE) utilizes the RetroMAE approach for pre-training and employs contrastive learning for fine-tuning [3]. Moreover, online services such as OpenAI's text embedding and Cohere-V3 generate text embeddings through their API [4]. While previous work has often focused on enriching text embeddings or enhancing the interaction between question-answer pairs, they have rarely considered the influence of other candidates. As a result, the relationships between candidates have not been fully explored. When humans select an answer, they consider not only whether each individual passage aligns with the question but also the presence of superior alternatives. Especially when the passages are derived from the same document or related documents, their semantics are often incomplete, and other candidates may contain valuable information that can supplement and interpret the current passage. Enriching the representation of

***Correspondence:** : Dr. Hongyin Zhu, Researcher, Architecture Research Department, Inspur Electronic Information Industry Co., Ltd., Beijing, China, e-mail: hongyin_zhu@163.com

each passage by considering the context information from other candidates can lead to more confident results. We use "list-context" across different passages to differentiate the "context" that is often discussed in the same QA pair. Context-independent representations may limit passage semantics when other passages provide useful context information for the question. For example, passage A in Table 1 explains that "cardiovascular disease" refers to heart disease. Although passage C does not mention heart disease, we can still derive relevant information from passage 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 gated recurrent unit recurrent neural networks (GRU RNNs) to consider context information among sentences [5]. 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 [6]. 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 [7], 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 finetunes 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 cache policy learning (CPL) 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 cache 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 and MS MARCO 2.0 datasets [8,9]. 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 with a cache policy learning algorithm, which can select answers from a coarse to fine level in a single pass. The experimental results demonstrate the good performance of the proposed method.

Related Work

Previous work employs deep learning models to enhance sentence representations and compute their similarity. Rocktäschel et al. propose a textual entailment model that models word relations between sentences by using word-to-word attention on an LSTM-RNN [10]. Severyn and Moschitti propose CNNR to consider overlapping words to encode relational information between question and answer [2]. Yin et al. propose 3 attention methods on a CNN model (ABCNN) to encode mutual interactions between sentences [11]. Miller et al. propose key-value memory networks (KV-MemeNN) to select answers by using key-value structured facts in the model memory [12]. Wang et al. propose a bilateral multi-view matching (BiMPM) model [13], which utilizes an attention mechanism to model the mutual interaction of sentences at different scales. Bachrach et al. 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 [14]. A work close to ours is the hierarchical GRU-RNN [5], 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. propose an option comparison network (OCN) for multiple choice reading comprehension (MCRC) [15].

Guo et al. propose the deep dependent matching model (DRMM), which introduces a histogram pooling technique to summarize the translation matrix [16]. Xiong et al. propose KNRM, which uses a kernel pooling layer to softly compute the frequency of word pairs at different similarity levels [17]. MARCO's official baseline uses two separate DNNs to model query-document relevance using local and distributed representations, respectively [18]. Conv-KNRM enhances KNRM by utilizing CNN to compose n-gram embeddings from word embeddings and cross-matched n-grams of different lengths [19]. SAN + BERT base maintains a state and iteratively refines its predictions [20].

Previous work mainly uses a multi-stage retrieval architecture in web search systems [7]. A set of candidate documents is generated using a series of increasingly expensive machine-learning techniques. present a method for optimizing cascaded ranking models [21]. BERT is a pre-trained language model based on the bidirectional Transformer architecture [22-26]. It has been demonstrated to be highly effective in generating rich representations for pairs of sentences. BERT offers a means to directly model the interactions between passages. Sentence-BERT (SBERT) is an improved model based on BERT and RoBERTa that is used to efficiently calculate the similarity between sentences [27]. It is trained through siamese

https://doi.org/10.61577/jaiar.2024.100005

and triplet network architecture, making the representation of sentences in vector space more suitable for cosine-similarity calculations. KEPLER proposes a method to jointly optimize knowledge embedding (KE) and pre-trained language model (PLM) [28]. This means that the model simultaneously learns how to extract relational facts from the knowledge graph and how to understand language patterns from text during training.

General text embedding techniques are commonly employed for web search and question answering, and they are also used to enhance the capabilities of large language models [1,29,30]. BGE adopts the RetroMAE method for pre-training and leverages contrastive learning for fine-tuning [3]. OpenAI text embedding produces high-quality text and code vector representations through contrastive pre-training on large amounts of unsupervised data [4]. Cohere-v3[1] improves the quality of text embeddings by evaluating how well the query matches the document's topic and assessing the overall quality of the content. However, when the document is segmented, it is easy to cause semantic incompleteness. The above state-of-the-art method only enhances the text embeddings of a single passage and cannot obtain supplementary semantic information from other passages belonging to the same document. Our method provides complementary information between passages by modeling the relationship between passages in the same document. 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.

[1]https://huggingface.co/Cohere

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 (1~1000). Each passage O_i contains one or several sentences (passage) which consists of o_i tokens $\{w_1^o, w_2^o, ..., w_i^o\}$. 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 the 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.

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 to generate representations of QA pairs and a single-layer neural network to compute matching scores [22].

Passage O_i is concatenated with question Q to form a complete sequence, denoted as ⟨[CLS]; Q; [SEP]; O_i; [SEP]⟩.

$$O_i^{(c)} = F_{bert}(\langle [\text{CLS}]; Q; [\text{SEP}]; O_i; [\text{SEP}] \rangle) \quad (1)$$

where $O_i^{(c)} \in R^d$ is the QA representation. F_{bert} denotes the network defined in Devlin et al. [22]. The representations are

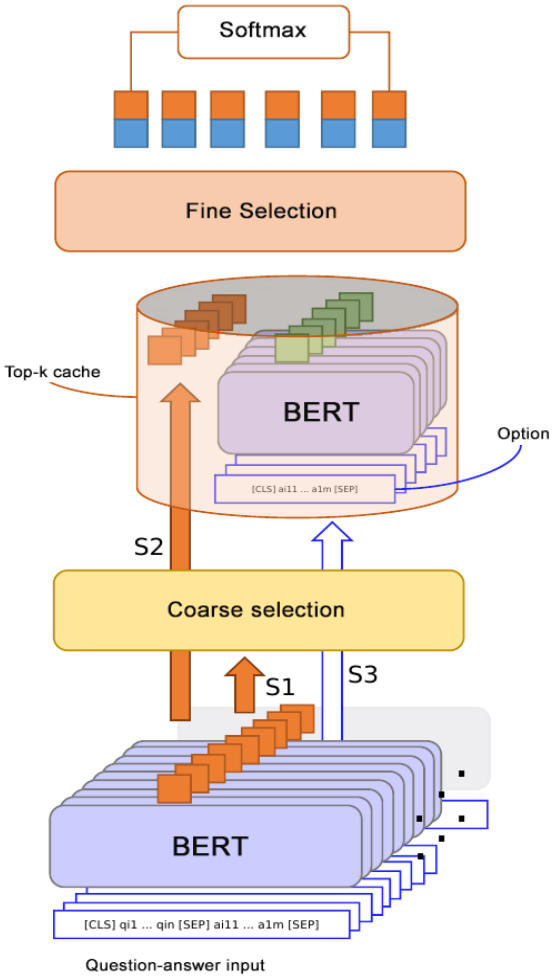

Figure 1. Schematic diagram of the model structure.

then projected to matching scores using a single layer neural network.

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

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

Modeling all candidates directly will result in out-of-memory issues due to the large amount of list-context information that needs to be processed simultaneously. The model parameter size is very large, and the gradient state retained during the optimization process will double the memory usage. To solve this problem, we propose using different paths and caching mechanisms. Each path represents a different function that processes the data in a way that reduces memory usage. The coarse ranker creates a cache used to store the top-k passages it has encountered previously. According to p_i, passages are dynamically ranked through path S1, which contains the function listed below.

$$h_t = K(p_i, h_{t-1}) \quad (3)$$

where h_0 is the initial state of the empty cache. h_t is the cache state after t step update and contains the selected passages. K denotes the ranking function.

Prior early-stage models typically process all the candidates

https://doi.org/10.61577/jaiar.2024.100005

and then retain the top-k candidates which are also written to disk. Different from them, we incrementally maintain the cache memory which only retains the top-k scoring QA pairs. k is a hyperparameter. Then, residual path S2 is used to connect the QA representation to the fine ranker, and it contains the function of the equation (4).

$$O_{h_t}^{(f)} = C(O_{:t}^{(c)}, h_t) \tag{4}$$

Where $O_{:t}^{(c)}$ represents the $[O_1^{(c)}, O_2^{(c)}, \ldots, O_t^{(c)}]$. $O_{h_t}^{(f)}$ is the matrix for top-k QA pairs. *(f)* denotes the fine ranker. *C* is the selection function.

The residual path S3 indicates that it is used to transfer the top-k paragraphs selected in the coarse ranker to the fine ranker layer and generate a vector representation of the passage. Unlike the vector representation in the coarse ranker, it only contains the content of the passage, not the question. These passages are derived from the top-k candidate paragraphs to generate list context information. The sorting result is the result of the borrowed coarse ranker without repeated calculations. This path can be described by equation (5). Finally, the selected passages and representations in the cache are sent to the fine ranker.

$$P_{h_t}^{(f)} = C(P_{:t}^{(c)}, h_t) \tag{5}$$

Where $P_{h_t}^{(f)}$ is the matrix for top-k passages. $P_i^{(c)}$ is the representation for *i*-th passage, which can be calculated by equation (6).

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

Fine ranker

The inputs to the fine ranker are vectors of top-k QA pairs $O_{h_t}^{(f)}$ and top-k passages $P_{h_t}^{(f)}$. The architecture of the fine ranker is depicted in Figure 2. It incorporates a list-context attention mechanism that combines both static and adaptive attention.

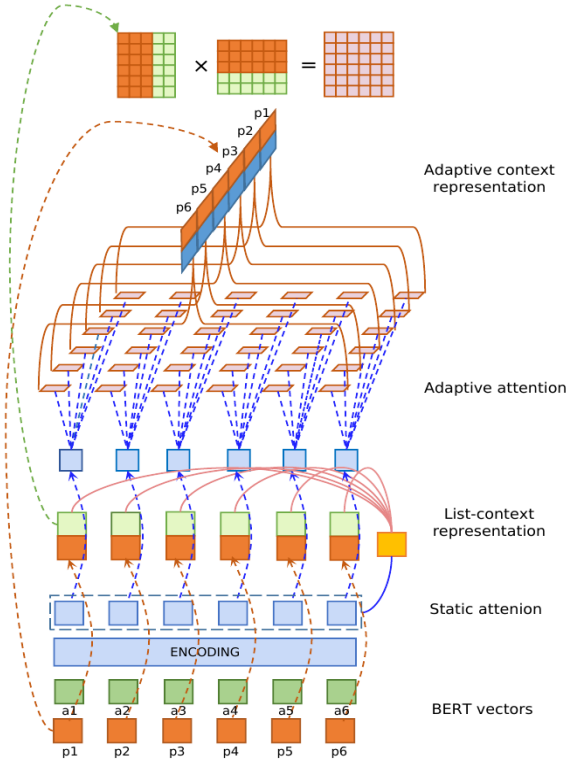

Figure 2. Schematic diagram of the fine ranker.

Static attention

To capture and compose the context semantics from the answer list, our model uses an attention mechanism which achieves the best performance in different alternatives to obtain the list-context representation [31,32]. This method 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 $C_l \in \mathrm{R}^{\mathrm{d}}$ 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.

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

where $\max(u)$ gets the max value of $[u_0, u_1, \ldots, u_k]$ 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.

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

Where V_l is the list-context representation.

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.

$$w_{ij} = [O_i^{(c)^t}, V_l^T]\,[{}^{O_i^{(c)}}_{V_l}] \tag{10}$$

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.

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

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 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 cache can be calculated as below.

$$p^{(f)} = softmax(W^{(f)}\left[P_{h_t}^{(f)}; Z\right]) \tag{13}$$

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

https://doi.org/10.61577/jaiar.2024.100005

Training algorithm

Prior multi-stage retrieval methods typically cascade different machine learning techniques. Different from the cascaded ranking architecture, we introduce the CPL 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 cache 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 cache 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 cache, 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 cache 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 backpropagation using an optimization algorithm.

Algorithm 1 The CPL algorithm

Input: QA pairs pair, Label l, Cache 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: $O_j^{(c)} = \mathrm{F}_{bert}(pair[j])$

5: $p_j^{(c)} = \sigma\left(Linear\left(O_j^{(c)}\right)\right)$

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 lists (p_i, p_v, p_{bert}) and (n_i, n_v, n_{bert}) into cache 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 $loss_c(p^{(c)}, l[c_i]), loss_f(p^{(f)}, l[c_i])$

After getting the top-k sentences, this model will carry out the fine selection process as described in the Fine Ranker subsection. This model is trained using the log loss of two-level selection as shown below.

$$L^{(c)} = -\sum_j \left[y_j . \log p_j^{(c)} + (1 - y_j) . \log(1 - p_j^{(c)})\right] \quad (14)$$

$$L^{(f)} = -\sum_j \left[y_j . \log p_j^{(f)} + (1 - y_j) . \log(1 - p_j^{(f)})\right] \quad (15)$$

where y_j is the label of the j-th passage. $p_j^{(c)}$ and $p_j^{(f)}$ 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.

$$\min_\theta L = \sum_i^{|D|} (L_i^{(c)} + \lambda L_i^{(f)}) \quad (16)$$

where λ (We set λ=1.0) 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 ($\rho\times1$) 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 cache, we need a top-k sort operation which takes $O(nk)$ at the worst case. 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 [8], 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.

https://doi.org/10.61577/jaiar.2024.100005

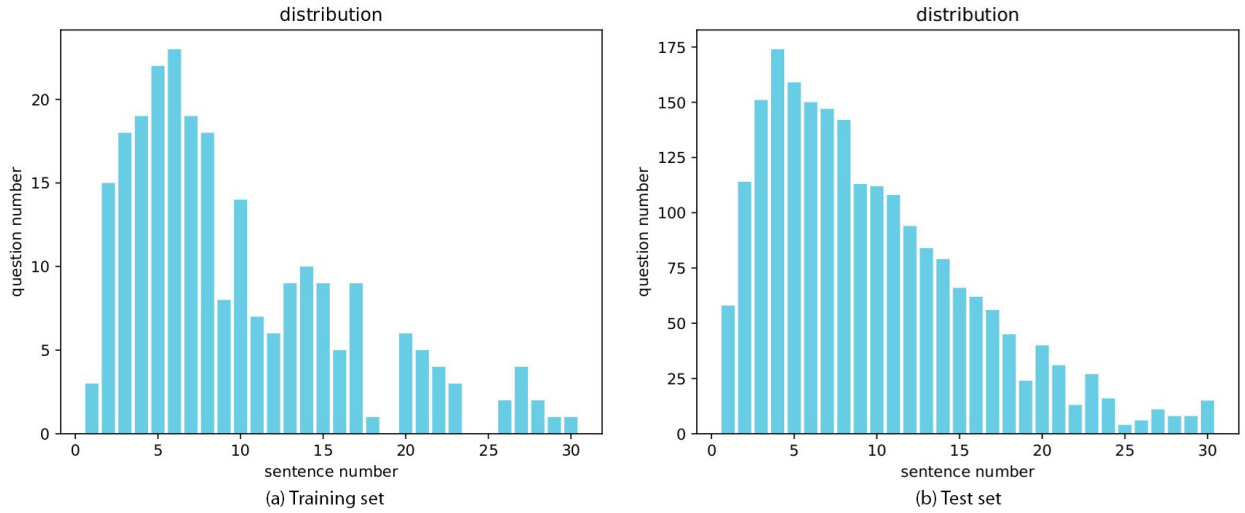

Figure 3. Passage distribution of the WIKIA dataset.

MS MARCO 2.0[2]

This is a large-scale machine reading comprehension dataset sampled from Bing's search query logs [9]. We choose the dataset for the passage re-ranking task. Given a set of 1000 passages that have been retrieved using the BM25 algorithm, re-rank these passages on their relevance to the query. This dataset was the primary focus of the 2020 and 2019 TREC Deep Learning Track. It has also been utilized as a teaching resource for the ACM SIGIR/SIGKDD AFIRM Summer School, which offers courses on Machine Learning for Data Mining and Search. 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.

[2]https://microsoft.github.io/MSMARCO-Passage-Ranking-Submissions/leaderboard/

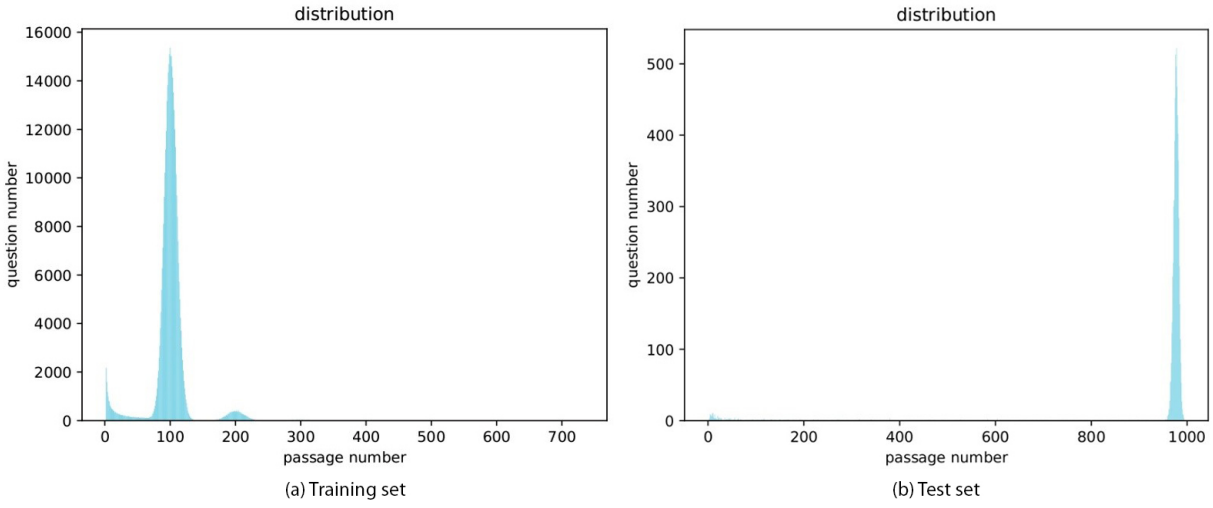

Figure 4. Passage distribution of the MS MARCO 2.0 dataset.

Hyper-parameters

This paper employs the BERT-base-uncased model to generate representations for sentence pairs. We utilize the output of the "[CLS]" token from the final layer of BERT as a representation of the QA pair. The projection layer employs a single-layer neural network with the hyperbolic tangent activation function to generate 200-dimensional vector representations. We use 40 tokens as the maximum length for questions and 200 tokens for answers. The cache size is set to 15.

The AdamW optimization algorithm is used to update the model parameters [33]. We fine-tune the BERT-base-uncased model 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 MS MARCO 2.0 dataset, we use the official script to evaluate our results. This script calculates the MRR@10 which considers only the top 10 passages.

https://doi.org/10.61577/jaiar.2024.100005

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:

Table 2. Results on the WIKIQA dataset.

Method	MAP	MRR
WordCnt [8]	0.4891	0.4924
WgtWordCnt [8]	0.5099	0.5132
CNN-CNt [8]	0.517	0.5236
CNN_R [2]	0.6951	0.7107
ABCNN-3 [11]	0.6921	0.7108
KV-MemNN [12]	0.7069	0.7265
BiMPM [13]	0.718	0.731
IARNN-Occam [34]	0.7341	0.7418
CNN-MULT [35]	0.7433	0.7545
CNN-CTK [36]	0.7417	0.7588
wGRU-sGRU-G_{l2}-Cnt [5]	0.7638	0.7852
BERT base	0.7831	0.7923
BERT base + MaxPooling	0.8119	0.8215
C2FRetriever	0.8448	0.8605

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_{l2}-Cnt [5]. 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 MS MARCO 2.0 dataset

Table 3 lists the results on the MS MARCO 2.0 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 fine-tune the BERT base and large models and simply use the matching score of QA pairs for ranking [37]. 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 achieves the highest

Table 3. Results on the MARCO dataset.

Method	MRR@10 Eval
BM25	0.167
LeToR	0.195
Official Baseline [18]	0.2517
Conv-KNRM [17]	0.271
IRNet	0.281
BERT base [37]	0.3472*
BERT large [37]	0.359
SAN + BERT base [20]	0.359
Enriched BERT base + AOA index	0.368
Enriched BERT base + AOA index + CAS + Full	0.3933**
C2FRetriever (200 tokens)	0.347
C2FRetriever (400 tokens)	0.364

*This score is generated on the development set.

**This score is generated with full ranking, while other models are reranking model.

https://doi.org/10.61577/jaiar.2024.100005

score, but 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.

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	0.82
–Adaptive, List, Two-level	0.7831	0.7923

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 limit the maximum length of queries to n tokens. We find that, for the same length of passage, longer query lengths generally lead to better results. However, for shorter query lengths, specifically when n < 20, longer passage lengths do not always result in improved outcomes. Since the meaning of the query is not well encoded, longer passages may contain more misleading information. Therefore, it is important to interpret and expand queries to improve results. We can conclude that an appropriate combination of query and paragraph lengths and model selection is crucial for the method.

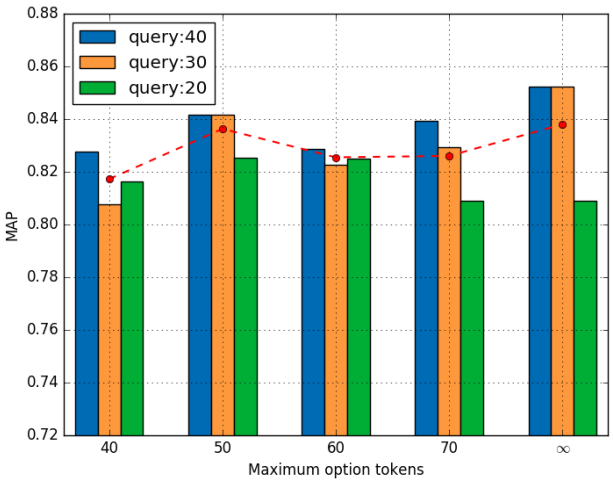

Figure 5. The influence of query/passage length and passage number in fine ranker.

To evaluate the influence of cache size, we did extensive experiments based on different cache sizes, as shown in Figure 6. We observe that this model achieves higher performance with the cache size=16. As the cache 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 cache size is important to the result.

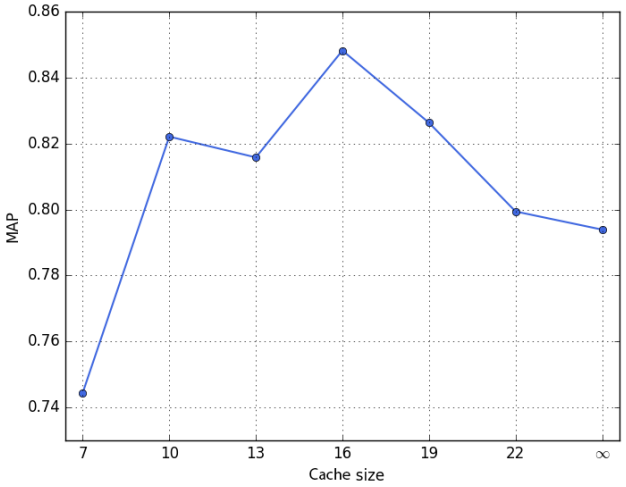

Figure 6. The influence of cache size.

Conclusions and Future Work

In this paper, we present a novel approach to passage reranking that incorporates list-context information to enhance the representation of passages across different contexts. Unlike previous studies, we recognize the significance of list-context information from other candidate passages in addressing the challenge of incomplete passage semantics and develop a method to integrate it effectively. Our model addresses the limitation of out-of-memory issues by leveraging a cache policy learning approach to represent list context. Additionally, we address the challenge of two-stage joint retrieval by integrating coarse and fine rankers in a seamless manner. Our model is

https://doi.org/10.61577/jaiar.2024.100005

trained by optimizing all components simultaneously, leading to the generation of the final answer in a single pass, which significantly reduces the complexity of the problem.

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 and downstream applications, in the future [38].

Disclosure statement

No potential conflict of interest was reported by the author.

References

1. Zhu H, Tiwari P, Ghoneim A, Hossain MS. A collaborative AI-enabled pretrained language model for AIoT domain question answering. IEEE Trans Industr Inform. 2021;18(5):3387-3396. https://doi.org/10.1109/TII.2021.3097183
2. Severyn A, Moschitti A. Modeling relational information in question-answer pairs with convolutional neural networks. arXiv preprint arXiv:1604.01178. 2016. https://doi.org/10.48550/arXiv.1604.01178
3. Xiao S, Liu Z, Zhang P, Muennighof N. C-pack: packaged resources to advance general Chinese embedding. arXiv preprint arXiv:2309.07597. 2023. https://doi.org/10.48550/arXiv.2309.07597
4. Neelakantan A, Xu T, Puri R, Radford A, Han JM, Tworek J, et al. Text and code embeddings by contrastive pre-training. arXiv preprint arXiv:2201.10005. 2022. https://doi.org/10.48550/arXiv.2201.10005
5. Tan C, Wei F, Zhou Q, Yang N, Du B, Lv W, et al. Context-aware answer sentence selection with hierarchical gated recurrent neural networks. IEEE/ACM Trans Audio, Speech, Language Process. 2017;26(3):540-549. https://doi.org/10.1109/TASLP.2017.2785283
6. Swayamdipta S, Parikh AP, Kwiatkowski T. Multi-mention learning for reading comprehension with neural cascades. In Proceedings of International Conference on Learning Representations (ICLR). 2018 (pp. 1-12). https://openreview.net/forum?id=HyRnez-RW
7. Mackenzie J, Culpepper JS, Blanco R, Crane M, Clarke CL, Lin J. Query driven algorithm selection in early stage retrieval. In Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining. 2018 (pp. 396-404). https://doi.org/10.1145/3159652.3159676
8. Yang Y, Yih WT, Meek C. WikiQA: a challenge dataset for open-domain question answering. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. 2015 (pp. 2013-2018). https://aclanthology.org/D15-1237.pdf
9. Nguyen T, Rosenberg M, Song X, Gao J, Tiwary S, Majumder R, et al. MS MARCO: a human generated machine reading comprehension dataset. Choice. 2016;2640:660. https://ceur-ws.org/Vol-1773/CoCoNIPS_2016_paper9.pdf
10. Rocktäschel T, Grefenstette E, Hermann KM, Kočiský T, Blunsom P. Reasoning about entailment with neural attention. arXiv preprint arXiv:1509.06664. 2015. https://doi.org/10.48550/arXiv.1509.06664
11. Yin W, Schütze H, Xiang B, Zhou B. ABCNN: attention-based convolutional neural network for modeling sentence pairs. Trans Assoc Comput Ling. 2016;4:259-272. https://doi.org/10.1162/tacl_a_00097
12. Miller A, Fisch A, Dodge J, Karimi AH, Bordes A, Weston J. Key-value memory networks for directly reading documents. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. 2016 (pp. 1400–1409). https://aclanthology.org/D16-1147/
13. Wang Z, Hamza W, Florian R. Bilateral multi-perspective matching for natural language sentences. In Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17). 2017 (pp. 4144-4150). https://doi.org/10.24963/ijcai.2017/579
14. Bachrach Y, Zukov-Gregoric A, Coope S, Tovell E, Maksak B, Rodriguez J, et al. An attention mechanism for neural answer selection using a combined global and local view. In 2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI). 2017 (pp. 425-432). https://doi.org/10.1109/ICTAI.2017.00072
15. Ran Q, Li P, Hu W, Zhou J. Option comparison network for multiple-choice reading comprehension. arXiv preprint arXiv:1903.03033. 2019. https://doi.org/10.48550/arXiv.1903.03033
16. Guo J, Fan Y, Ai Q, Croft WB. A deep relevance matching model for ad-hoc retrieval. In Proceedings of the 25th ACM international on conference on information and knowledge management. 2016 (pp. 55-64). https://doi.org/10.1145/2983323.2983769
17. Xiong C, Dai Z, Callan J, Liu Z, Power R. 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. 2017 (pp. 55-64). https://doi.org/10.1145/3077136.3080809
18. Mitra B, Diaz F, Craswell N. Learning to match using local and distributed representations of text for web search. In Proceedings of the 26th International Conference on World Wide Web. 2017 (pp. 1291-1299). https://doi.org/10.1145/3038912.3052579
19. Dai Z, Xiong C, Callan J, Liu Z. Convolutional neural networks for soft-matching n-grams in ad-hoc search. In Proceedings of the eleventh ACM international conference on web search and data mining. 2018 (pp. 126-134). https://doi.org/10.1145/3159652.3159659
20. Liu X, Duh K, Gao J. Stochastic answer networks for natural language inference. arXiv preprint arXiv:1804.07888. 2018. https://doi.org/10.48550/arXiv.1804.07888
21. Chen RC, Gallagher L, Blanco R, Culpepper JS. 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. 2017 (pp. 445-454). https://doi.org/10.1145/3077136.3080819
22. Devlin J, Chang MW, Lee K, Toutanova K. 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. 2019 (pp. 4171–4186). https://doi.org/10.18653/v1/N19-1423
23. Zhu H. MetaAID: a flexible framework for developing metaverse applications via ai technology and human editing. arXiv preprint arXiv:2204.01614. 2022. https://doi.org/10.48550/arXiv.2204.01614
24. Zhu H. MetaAID 2.0: an extensible framework for developing metaverse applications via human-controllable pre-trained models. arXiv preprint arXiv:2302.13173. 2023. https://doi.org/10.48550/arXiv.2302.13173
25. Zhu H. MetaAID 2.5: a secure framework for developing metaverse applications via large language models. arXiv preprint arXiv:2312.14480. 2023. https://doi.org/10.48550/arXiv.2312.14480
26. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need. Advances in Neural Information Processing Systems. 2017;30. https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html
27. Reimers N, Gurevych I. Sentence-BERT: sentence embeddings using siamese bert-networks. 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). 2019 (pp. 3980-3990). https://doi.org/10.18653/v1/D19-1410
28. Wang X, Gao T, Zhu Z, Zhang Z, Liu Z, Li J, et al. KEPLER: a unified model for knowledge embedding and pre-trained language representation. Trans Assoc Comput Ling. 2021;9:176-194. https://doi.org/10.1162/tacl_a_00360
29. Zhu H. Financial data analysis application via multi-strategy text processing. arXiv preprint arXiv:2204.11394. 2022. https://doi.org/10.48550/arXiv.2204.11394
30. Zhu H. FQP 2.0: industry trend analysis via hierarchical financial data. arXiv preprint arXiv:2303.02707. 2023. https://doi.org/10.48550/arXiv.2303.02707

https://doi.org/10.61577/jaiar.2024.100005

31. Parikh AP, Täckström O, Das D, Uszkoreit J. A decomposable attention model for natural language inference. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. 2016 (pp. 2249-2255). https://doi.org/10.18653/v1/D16-1244
32. Yang Z, Yang D, Dyer C, He X, Smola A, Hovy E. Hierarchical attention networks for document classification. In Proceedings of the 2016 Conference of the North American chapter of the Association for Computational Linguistics: Human Language Technologies. 2016 (pp. 1480-1489). https://aclanthology.org/N16-1174.pdf
33. Loshchilov I, Hutter F. Decoupled weight decay regularization. In Proceedings of International Conference on Learning Representations (ICLR). 2019 (pp. 1-19). https://openreview.net/forum?id=Bkg6RiCqY7
34. Wang B, Liu K, Zhao J. Inner attention based recurrent neural networks for answer selection. InProceedings of the 54th Annual Meeting of the Association for Computational Linguistics. 2016 (pp. 1288-1297). https://aclanthology.org/P16-1122.pdf
35. Wang S, Jiang J. A compare-aggregate model for matching text sequences. In Proceedings of International Conference on Learning Representations (ICLR). 2017 (pp. 1-11). https://openreview.net/forum?id=HJTzHtqee
36. Tymoshenko K, Bonadiman D, Moschitti A. 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. 2016 (pp. 1268-1278). https://aclanthology.org/N16-1152.pdf
37. Nogueira R, Cho K. Passage Re-ranking with BERT. arXiv preprint arXiv:1901.04085. 2019. https://doi.org/10.48550/arXiv.1901.04085
38. Zhu H, Tiwari P, Zhang Y, Gupta D, Alharbi M, Nguyen TG, et al. SwitchNet: a modular neural network for adaptive relation extraction. Comput Electr Eng. 2022;104:108445. https://doi.org/10.1016/j.compeleceng.2022.108445

https://doi.org/10.61577/jaiar.2024.100005