A BERT-Based Semantic Matching Ranker for Open-Domain Question Answering

Shiyi Xu, Feng Liu, Zhen Huang, Yuxing Peng and Dongsheng Li
College of Computer,
National University of Defense Technology,
Changsha, China
xushiyi18@nudt.edu.cn; richardlf@nudt.edu.cn; huangzhen@nudt.edu.cn; dsli@nudt.edu.cn; pengyuxing@aliyun.com

ABSTRACT
Open-domain question answering (QA) is a hot topic in recent years. Previous work has shown that an effective ranker can improve the overall QA performance by denoising irrelevant context. There are also some recent works leveraged BERT pre-trained model to tackle with open-domain QA tasks, and achieved significant improvements. Nevertheless, these BERT-based models simply concatenates a paragraph with a question, ignoring the semantic similarity of them. In this paper, we propose a simple but effective BERT-based semantic matching ranker to compute the semantic similarity between the paragraph and given question, in which three different representation aggregation functions are explored. To validate the generalized performance of our ranker, we conduct a series of experiments on two public open-domain QA datasets. Experimental results demonstrate that the proposed ranker contributes significant improvements on both the ranking and the final QA performances.

CCS Concepts
• Computing methodologies→Natural Language Processing.

Keywords
Open-domain question answering, retriever , semantic ranker , reading comprehension

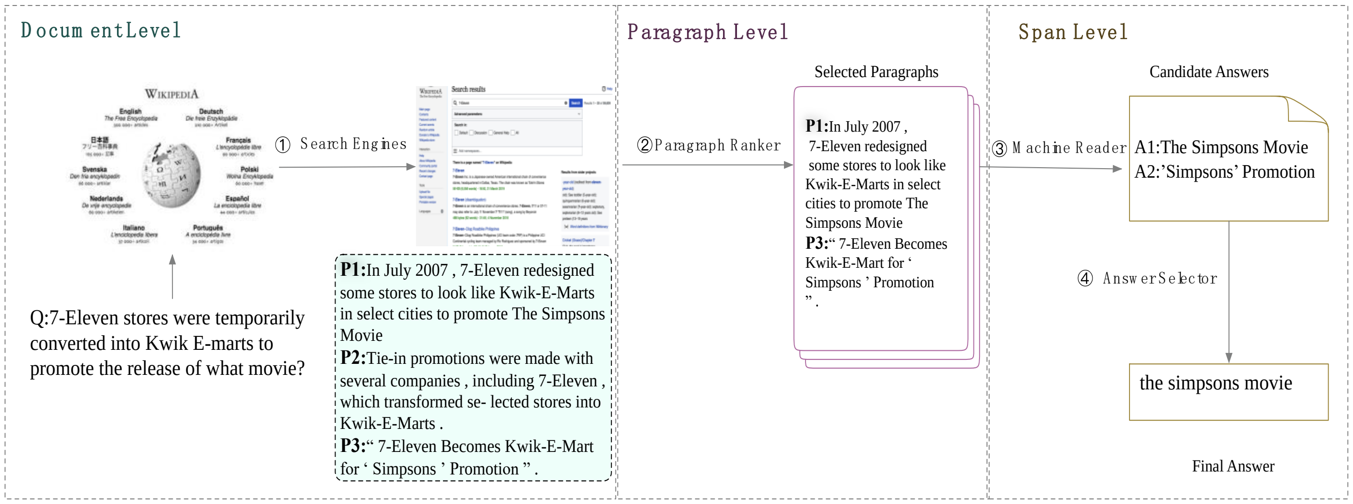
1. INTRODUCTION
Open-domain question answering (QA) has received a lot of research efforts in recent years due to the release of several large-scale datasets as well as huge advancements in neural reading comprehension. An open-domain QA system is supposed to efficiently read relevant context and effectively find correct answers from a large text corpus like the Web or Wikipedia. As described in Figure 1, a coarse-to-fine pipeline for open-domain QA consists of three level: document level, paragraph level and span level. In document level, given a question, the search engine coarsely returns a list of relevant documents using information retrieval techniques like TF-IDF or BM25. Then in paragraph level, these relevant documents are split in paragraphs and sent to a paragraph ranker to retrieve the most relevant paragraphs with the question. Finally in span level, few top ranked paragraphs are sent to the machine reader to extract candidate answers, and predict the final answer from these candidates with an answer selector.

Although early works such as BiDAF [1], Match-LSTM [2] and DCN+ [3] have made significant improvements on span level reading comprehension, open-domain QA is much more complicated as there are no ground-truth paragraphs sent to the system. In another word, only a small part of textual context actually covers the correct answer, and the rest of retrieved document content are the noise for the reader.

In recent years, several works like [4] [5] [6]concentrate on developing effective paragraph rankers by building up neural networks to retrieve most relevant paragraphs for open-domain QA, and have proven that a neural ranker can improve the overall performance. Most of these approaches leverage fixed word-level or sentence-level vectors to encode question and paragraph representations, which may lead to semantic information loss. Some recent works leveraged BERT [7] to tackle with open-domain QA tasks. The end-to-end models [8] [9] directly applied the BERT-RC model after retrieve multiple passages without filter the irrelative ones. The pipelined models [10] [11] treated the ranker as binary classification problems and usually use the final hidden state of the [CLS] token as the final sequence representation for classification tasks. However, these methods only concatenate the question and paragraphs with coarse-grained semantic matching and ignored fine-grained context-aware semantic similarity.

In this paper, we propose a simple and effective BERT-based semantic matching ranker (BBSMR) to tackle the above problems. Our ranker is designed to build word representations on the basic of pre-trained BERT-base model. On top of it, we design various representation aggregation functions to form the paragraph representation, fully considering the importance of contextual content and semantic relevance. Furthermore, this paper also explores an open-domain QA pipeline by combining the proposed ranker with a machine reader and an answer selector. Evaluated on QUASAR-T [12] and SearchQA [13] datasets, our system achieves 11.9% and 18.8% absolute improvements in the overall QA performance on two datasets respectively.

NLPIR 2020, December 18–20, 2020, Seoul, Republic of Korea
© 2020 Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-7760-7/20/06…$15.00
DOI: https://doi.org/10.1145/3443279.3443301



Q:7-Eleven stores were temporarily converted into Kwik E-marts to promote the release of what movie?

P1: In July 2007 , 7-Eleven redesigned some stores to look like Kwik-E-Marts in select cities to promote The Simpsons Movie
P2: Tie-in promotions were made with several companies , including 7-Eleven , which transformed se- lected stores into Kwik-E-Marts .
P3:" 7-Eleven Becomes Kwik-E-Mart for ' Simpsons ' Promotion " .

P1: In July 2007 , 7-Eleven redesigned some stores to look like Kwik-E-Marts in select cities to promote The Simpsons Movie
P3:" 7-Eleven Becomes Kwik-E-Mart for ' Simpsons ' Promotion " .

A1:The Simpsons Movie
A2:'Simpsons' Promotion

the simpsons movie

Figure 1. An overview of the open-domain QA pipeline.

2. RELATED WORK

Information retrieval (IR) aims to retrieve few relevant context for reducing text noise, and thus plays an important role in open-domain QA. It is usually involved with similarity calculation and relevance matching between candidate paragraphs and given question. Neural IR models focus on semantic matching. Specifically, representation-based IR models like ARC-I [14] construct a fixed-dimensional vector representation for each text string separately and then perform matching within the latent space. Interaction-based models like DRMM [15] and RN ranker [4] compute the interaction between each individual word pairs. However, as pointed out by [5], the complex modeling of interactions between documents and questions cannot be scaled to large corpora. Therefore, we mainly focus on developing representation-based IR models in this paper.

Open-Domain QA is supposed to extract answers for factoid questions from a large text corpus. As the rapid development of reading comprehension (RC) models, many works [1][2][8] apply the RC models in open-domain QA tasks. However, different from the RC task, open-domain QA is much more challenging and complicated. To effectively tackle with open-domain QA, pointed out by [16], the general pipeline consists of three components, information retrieving, reading comprehension and answer selection. There are a series of well performing pipeline based solutions, such as DrQA [17], Reinforced Ranker Reader [18], MSReansonNet [19], BERTserini [9] and multi-passage BERT[10]. In this paper, we mainly focus on the ranker component, and apply a classical reader of DrQA to build up a lightweight model for the open-domain QA task.

3. OUR APPROACH

In this section, we build up an open-domain QA pipelined system, as shown in Figure 1, Our model consists of three components: paragraph ranker, machine reader and answer selector.

3.1 Paragraph Ranker

The ranker module is supposed to output a semantic similarity score between the question and each paragraph, indicating the relevance between them. For each question, we select top K ranked paragraphs according to the scores as the inputs of machine reader. As illustrated in Figure 2, our BBSMR model consists of three layers, as described from the ground up below.

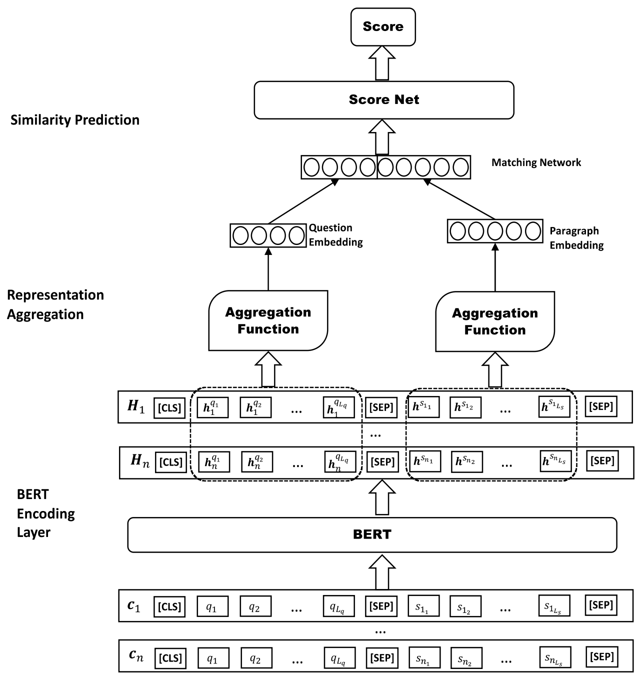

Score
Score Net
Similarity Prediction
Matching Network
Question Embedding
Paragraph Embedding
Representation Aggregation
Aggregation Function
Aggregation Function
H_1 [CLS] $h_1^{q_1}$ $h_1^{q_2}$ ... $h_1^{q_{L_q}}$ [SEP] $h^{s_{1_1}}$ $h^{s_{1_2}}$ ... $h^{s_{1_{L_s}}}$ [SEP]
H_n [CLS] $h_n^{q_1}$ $h_n^{q_2}$ ... $h_n^{q_{L_q}}$ [SEP] $h^{s_{n_1}}$ $h^{s_{n_2}}$ ... $h^{s_{n_{L_s}}}$ [SEP]
BERT Encoding Layer
BERT
c_1 [CLS] q_1 q_2 ... q_{L_q} [SEP] s_{1_1} s_{1_2} ... $s_{1_{L_s}}$ [SEP]
c_n [CLS] q_1 q_2 ... q_{L_q} [SEP] s_{n_1} s_{n_2} ... $s_{n_{L_s}}$ [SEP]

Figure 2. Architecture overview of the BBSMR model.

3.1.1 BERT encoding layer

We leverage pretrained BERT-base model to generate word embeddings of question and paragraph. Given a question Q and corresponding paragraph P, we first tokenize each Q and P as $Q_{tok} = \left\{q_1, q_2, \dots, q_{L_q}\right\}$ and $P_{tok} = \left\{p_1, p_2, \dots, p_{L_p}\right\}$, L_q and L_p denote length of the tokenized question and paragraph separately. As BERT has a constraint on the max length of a tokenized sequence, we assume the max sequence length is L_{max}, when $L_q + L_p > L_{max}$, we use a sliding window of length $l = L_{max} - L_q - 3$ with a stride r over the paragraph and produce a set of text segments $\boldsymbol{S} = \{\boldsymbol{s}_1, \boldsymbol{s}_2, \dots, \boldsymbol{s}_n\}$, where $n = \left\lceil \frac{L_p - l}{r} \right\rceil + 1$ is the number of segments. Then we concatenate question with each segment $\boldsymbol{s} = \{s_1, s_2, \dots s_{L_s}\}$ as BERT input sequence. If $L_q + L_p < L_{max}$, we treat paragraph as a segment and directly concatenates Q and each paragraph P. The input sequence $\boldsymbol{c}$ can be formulated as:

$$[CLS]\ q_1, \dots, q_{L_q}\ [SEP]\ s_1, \dots, s_{L_s}\ [SEP] \quad (1)$$

Where $[CLS]$ is the start token and $[SEP]$ is the segment token.

The output of final hidden state H can be formulated as:

$$[CLS] h^{q_1}, \dots, h^{q_{L_q}}\ [SEP]\ h^{s_1}, \dots, h^{s_{L_s}}\ [SEP] \quad (2)$$

Instead of directly using the final hidden state of the [CLS] token, we aggregate the final hidden state of each word to generate question and paragraph embeddings.

3.1.2 Representation aggregation

We explore three different representation aggregation functions to summarize previous word vectors H^q and H^p into question representation E^q and paragraph representation E^p . Each embedding E can be calculated as follows:

$$\text{MAX: E} = \max\_pooling\left(\left\{\mathrm{H}^i\right\}_{i=1}^{\mathrm{n}}\right) \quad (3)$$

$$\text{MEAN: E} = mean\_pooling\left(\left\{\mathrm{H}^i\right\}_{i=1}^{\mathrm{n}}\right) \quad (4)$$

$$\text{Weighted} - \text{SUM: E} = \sum_{i=1}^{n} \boldsymbol{\alpha}_i H^i$$

$$\text{Where } \boldsymbol{\alpha}_i = SoftMax\left(\boldsymbol{w}_\alpha H^i + \boldsymbol{b}\right) \quad (5)$$

Where $\left\{\mathrm{H}^i\right\}_{i=1}^{\mathrm{n}}$ denote the hidden states of all the texts correspond to a question or a paragraph.

1) MAX operation takes the max value of all words as the sequences counterpart across all dimensions. This function assumes that for each dimension, there exists an obvious weight among text. It is suitable for text with a lot of noise.

2) MEAN operation takes the average value of all words as the embedding representation. In contrast to the MAX function, this operation is built on the assumption that every words plays a relatively balanced role in the final semantic expression. It is suitable for short texts.

3) Weighted-SUM operation applies a linear projection layer followed by the SoftMax function to produce an attention distribution among words, and then computes a weighted-sum vector. The method is supposed to learn to attend to relevant words by assigning relatively larger probabilities.

3.1.3 Similarity prediction

To compute the semantic similarity score, we utilize a variant of matching network [20], which take the question representation E^q , the paragraph representation E^q , their element-wise subtraction, and their elementwise outer product into considerations. These four vectors are concatenated and then sent into a feed-forward neural network that contains two linear projections along with a LeakyRelu activation function in between.

$$M = [E^q : E^p : E^q - E^p : E^q \odot E^p]^T \quad (6)$$

$$\boldsymbol{Score}(q,p) = \boldsymbol{W}_2 LeakyReLU(\boldsymbol{W}_1 M + \boldsymbol{b}_1) + \boldsymbol{b}_2 \quad (7)$$

where **Score**(q, p) is the scoring function calculating the similarity between q and p.

3.1.4 Loss function

We train the ranker by minimizing the standard marginal ranking loss [21] as follows:

$$L = \sum_{i=1}^{\#neg} max\left(0{,}1 - Score(q, p^{pos}) + Score\left(q, p_i^{neg}\right)\right) \quad (8)$$

where p^{pos} denotes the positive paragraph containing ground-truth answers, p^{neg} is the negative example that does not contain answers, and $\#neg$ is the number of negative paragraphs sent into the ranker model for each question during training.

3.2 Machine Reader

As the second part of overall system, machine reader is expected to extract answer spans from the ranked paragraphs. In our QA pipeline, the system is designed to be compatible to any off-the-shelf reading model. In this paper, we mostly concentrate on developing an effective ranker for the whole task. Thus, we choose the Document Reader of DrQA [17] to verify the overall QA performance.

3.3 Answer Selector

The answer selector takes the outputs of both the ranker and the reader as inputs and aims at finding the final answer. We select the top K ranked paragraphs according to similarity score, and use SoftMax over their scores to obtain a normalized probability distribution $Pr(q,p)$. The machine reader, on the other hand, produces an answer probability $Pr(a|q,p)$ given a question and a paragraph, indicating the confidence of a text span a being the correct answer. To find the final answer, a joint probability distribution of answer-question-paragraph tuple $Pr(a,q,p)$ is calculated by multiplying the above two probabilities. Then we choose the span with highest probability as the final answer of our pipeline.

$$\Pr(q,p) = SoftMax\left(\boldsymbol{Score}(q,p)\right) \quad (9)$$

$$Pr(a,q,p) = Pr(a|q,p)Pr(q,p) \quad (10)$$

4. EXPERIMENTS

4.1 Datasets

To evaluate the performance of our approach, we choose two challenging open-domain QA datasets, namely Quasar-T [12] and SearchQA [13]. Quasar-T includes two sub-sets, the short-document set and the long-document set. SearchQA provides both the whole string format and the snippet format for each given paragraph. As illustrated in Table 1, we show some statistic features for these datasets.

Table 1. Statistics of the datasets.

datasets	#q(train)	#q(test)	#p(test)	#avg_p
Quasar-T-short	37,012	3,000	299,389	22.18
Quasar-T-long	37,012	3,000	59,979	360.49
searchQA	99,820	27,248	1,351,527	37.10

#q represents the number of questions, #p represents the number of paragraphs. #avg_p means the average token number of paragraph. We choose six paragraphs for each question to calculate the ranking loss for training, and use all paragraphs in the datasets for testing.

4.2 Baselines

We consider the following models as our baselines, including two ranking models and five general open-domain QA systems.

4.2.1 Ranker Baselines

- InferSent Ranker (IS Ranker) [4] is a sentence-embedding based neural network to evaluate semantic similarity in ranking. The ranker has been proven helpful in the short sub-set of Quasar-T datasets.

- Ranker from $\mathbf{R^3}$ ($\mathbf{R^3}$ Ranker) [18] is a neural ranker that uses Match-LSTM for encoding. The ranker and the reader are trained jointly using reinforcement learning to optimize the expectation of extracting ground-truth answers from retrieved passages.

4.2.2 Open-Domain QA Baselines

- DS-QA [22] is an end-to-end model consists of a paragraph ranker and a machine reader, which makes full use of all informative paragraphs and tries to alleviate the distant supervision problem.
- MemoReader [23] is a powerful machine reading model to tackle with a long-range dependency problem for the open-domain QA task. They apply an advanced memory-augmented architecture and an expanded gated recurrent unit with dense connections that mitigates potential information distortions occurring in the memory.
- MSReasoner [19] designs a framework that the ranker and the reader iteratively interacts with each other. They generate a new query vector through the reader, passing it back into the retriever model for documents ranking. This allows the model to read more paragraphs and combine new evidences across multiple paragraphs.
- HAS-QA [24] is a three-level hierarchical structure designed for open-domain QA, namely the question level, the paragraph level and the answer span level. They design an end-to-end model to fuse the above probabilities so as to predict the final answer.
- MAMCN[25] proposed a multi-stage memory augmented neural network to deal with the problem of long-range dependencies for reading comprehension tasks. They stored representations of documents and questions in external memory units, and build a memory controller which can read and write these external storage units so as to compensate the limitation of recurrent neural network.
- Multi-passage BERT [10] is an open-domain QA pipelined model based on BERT, it implemented a passage ranker with BERT reading model and explored various method to improve the overall performance for open-domain QA.

4.3 Experimental Settings

We implement our approach using Pytorch framework. We initialize our model using uncased versions of BERT-base model with default hyper-parameters, and use AdamW optimizer with a learning rate of 5e-5 and train for 5 epochs. We pre-train the model with 2 2080Ti GPUs. The total batch size is 48 for Quasar-T-short and searchQA, and 96 for Quasar-T-long (with 4 gradient accumulation steps). During the preprocessing of Quasar-T-long datasets, as the length of most of the tokenized paragraphs are greater than L_{max}, we need to split paragraphs into snippets using sliding window. We try several set of parameters and finally chose the sliding window with $L_{max} = 384$ and $r = 234$. For Quasar-T-short and SearchQA, we directly use the snippet format provided in the dataset. To train our ranker, we consider the paragraph that contains ground-truth answers as positive examples, and randomly choose five negative examples. Notice that we have not used the IR_{score} information which is provided in the Quasar-T dataset to identify the relevance roughly.

5. RESULTS AND ANALYSIS

5.1 Main Results

Firstly, we evaluate the ranker model based on the recall@K, indicating the percentage of top-K ranked paragraphs containing the ground-truth answer. As shown in Table 2, our approach outperforms other competitive baselines by a large margin, reaching 67.0% on the Quasar-T short set. Moreover, we find that the recall on top-3 of our ranker is only a little bit worse than the one on top-5, but is much higher than the result of top-1. Consequently, we can choose to return top-3 paragraphs instead of top-5 when a high recall is required with limited computing resources.

Table 2. Recall of different rankers on Quasar-T-short test sets (%).

Models	Top-1	Top-3	Top-5
$\boldsymbol{R^3}$ Ranker [18]	40.3	51.3	54.5
IS Ranker [4]	36.1	52.8	56.7
BBSMR	**49.9**	**62.5**	**67.0**

Next, we compare our overall QA pipeline with baseline systems by evaluating the Exact Match (EM) and F1 scores on predicted answers. For each question, we feed top-5 paragraphs selected by the ranker to the DrQA reading model, and use the proposed answer selector module to choose the final answer. As shown in Table 3, the proposed pipelined system achieves state-of-the-art results on both datasets, indicating that integrating our ranker into an existing pipeline can boost the overall performance for the open-domain QA task. Specifically, our approach obtains 71.9%/78.8%/89.4% F1 on Quasar-T(short), Quasar-T(long) and SearchQA, achieving 0.7%/11.9%/18.8% absolute gains respectively as compared to published strong baselines. Moreover, MemoReader outperforms our approach by 8.2 EM on the short set of Quasar-T, but performs worse than ours on the long counterpart, suggesting that their approach may be more suitable on the short dataset rather than the long one since no ranker is used to filter out irrelevant context.

Table 3. Exact Match (EM) and F1-score of different models on Quasar-T and SearchQA test sets (%).

	Quasar-T-short		Quasar-T-long		searchQA	
	EM	F1	EM	F1	EM	F1
ISRanker+DrQA [4]	31.2	37.6	-	-	-	-
$\boldsymbol{R^3}$ [18]	35.3	41.7	-	-	49.0	55.3
DS-QA [22]	42.2	49.3	-	-	58.8	64.5
MemoReader [23]	**69.1**	71.2	63.5	66.9	-	-
MSReasoner [19]	40.6	47.0	-	-	56.3	61.4
HAS-QA [24]	43.2	48.9	-	-	62.7	68.7
MAMCN[25]	68.1	70.3	65.2	63.4	-	-
Multi-passage BERT [10]	51.3	59.0	-	-	65.2	70.6
BBSMR+DrQA	60.9	**71.9**	**66.1**	**78.8**	**84.3**	**89.4**

Table 4. Multiple Aggregation Function experiment results on datasets (%).

	Quasar-T-short			Quasar-T-long			searchQA		
	Recall	EM	F1	Recall	EM	F1	Recall	EM	F1
Max	66.23	59.47	70.63	75.07	**66.40**	**79.03**	92.90	84.29	89.39
Mean	**66.97**	**60.97**	**71.93**	75.30	66.13	78.75	**92.92**	**84.31**	**89.41**
Weighted-Sum	66.47	60.37	71.58	75.30	66.20	78.83	92.70	84.09	89.22

5.2 Effects of Representation Aggregation Functions

In order to figure out the effects of different representation aggregation functions, we deliver a series of comparable experiments on two datasets. We compare the results of retrieving top-5 paragraphs for reading, the as illustrated in Table 4, Mean function performs the best on short datasets. For long text, the performance of three methods is similar, with Max function having the highest performance.

6. CONCLUSION

In this paper, we propose a simple and effective BERT based semantic matching ranker (BBSMR) for open-domain QA. Our ranker utilizes BERT pre-trained model to generate embeddings, performs information integration with both question-side and paragraph-side, and applies an aggregation function to effectively summarize representations. Moreover, we use a classical machine reading model and take multiple probabilities into considerations to predict the final answer. To evaluate the generalizations and effectiveness of the proposed ranker, we conduct several experiments on the two open-domain QA datasets. Results show that BBSMR can make significant improvements on both the ranking performance and the overall open-domain QA result. For future work, we will continue this research on three aspects: (i) developing an end-to-end model based on BBSMR to further improve the overall QA performance (ii) integrating our ranker with more advanced machine reading models to further improve the overall performance and (iii) designing a method to speeding the ranking process and reduce memory consumption for the open-domain QA task.

7. REFERENCES

[1] Seo, M., Kembhavi, A., Farhadi, A., & Hajishirzi, H. 2017. Bidirectional Attention Flow for Machine Comprehension. In International Conference on Learning Representations.

[2] Wang, S., & Jiang, J. 2017. Machine comprehension using match-lstm and answer pointer. In International Conference on Learning Representations.

[3] Xiong, C., Zhong, V., & Socher, R. 2018. DCN+: Mixed Objective And Deep Residual Coattention for Question Answering. In *International Conference on Learning Representations*.

[4] Htut, P. M., Bowman, S., & Cho, K. 2018. Training a Ranking Function for Open-Domain Question Answering. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop, 120-127.

[5] Min, S., Zhong, V., Socher, R., & Xiong, C. 2018. Efficient and Robust Question Answering from Minimal Context over Documents. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* ,1725-1735.

[6] Nie, Y., Wang, S., & Bansal, M. 2019. Revealing the Importance of Semantic Retrieval for Machine Reading at Scale. 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)*, 2553-2566.

[7] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. 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, 4171-4186.

[8] Hu, M., Peng, Y., Huang, Z., & Li, D. 2019. Retrieve, Read, Rerank: Towards End-to-End Multi-Document Reading Comprehension. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics , 2285-2295.

[9] Yang, W., Xie, Y., Lin, A., Li, X., Tan, L., Xiong, K., ... & Lin, J. 2019. End-to-End Open-Domain Question Answering with BERTserini. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations) ,72-77.

[10] Wang, Z., Ng, P., Ma, X., Nallapati, R., & Xiang, B. 2019. Multi-passage BERT: A Globally Normalized BERT Model for Open-domain Question Answering. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, 5881-5885.

[11] Qiao, Y., Xiong, C., Liu, Z., & Liu, Z. 2019. Understanding the Behaviors of BERT in Ranking. arXiv preprint arXiv:1904.07531.

[12] Dhingra, B., Mazaitis, K., & Cohen, W. W. 2017. Quasar: Datasets for Question Answering by Search and Reading. *arXiv preprint* arXiv:1707.03904. [Online].Available: http://arxiv.org/abs/1707.03904

[13] Dunn, M., Sagun, L., Higgins, M., Guney, V. U., Cirik, V., & Cho, K. (2017). Searchqa: A new q&a dataset augmented with context from a search engine. *arXiv preprint arXiv:1704.05179*. [Online].Available: http://arxiv.org/abs/1704.05179

[14] Hu, B., Lu, Z., Li, H., & Chen, Q. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in neural information processing systems*, 2042-2050.

[15] Ai, Q., Yang, L., Guo, J., & Croft, W. B. 2016. Analysis of the paragraph vector model for information retrieval. In *Proceedings of the 2016 ACM international conference on the theory of information retrieval* , 133-142.

[16] Voorhees, E. M., & Tice, D. M. 2000. Building a question answering test collection. In Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval , 200-207.

[17] Chen, D., Fisch, A., Weston, J., & Bordes, A. 2017. Reading Wikipedia to Answer Open-Domain Questions. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 1870-1879.

[18] Wang, S., Yu, M., Guo, X., Wang, Z., Klinger, T., Zhang, W., Chang, S., Tesauro, G., Zhou, B., Jiang, J. 2018. R 3 : Reinforced reader-ranker for open-domain question answering.

[19] Das, R., Dhuliawala, S., Zaheer, M., & McCallum, A. 2018. Multi-step Retriever-Reader Interaction for Scalable Open-domain Question Answering. In *International Conference on Learning Representations*.

[20] Vinyals, O., Blundell, C., Lillicrap, T., & Wierstra, D. 2016. Matching networks for one shot learning. In *Advances in neural information processing systems* , 3630-3638.

[21] Bai, B., Weston, J., Grangier, D., Collobert, R., Sadamasa, K., Qi, Y., ... & Weinberger, K. 2010. Learning to rank with (a lot of) word features. *Information retrieval*, *13*(3), 291-314.

[22] Lin, Y., Ji, H., Liu, Z., & Sun, M. 2018. Denoising distantly supervised open-domain question answering. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* , 1736-1745.

[23] Back, S., Yu, S., Indurthi, S. R., Kim, J., & Choo, J. 2018. Memoreader: Large-scale reading comprehension through neural memory controller. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2131-2140.

[24] Pang, L., Lan, Y., Guo, J., Xu, J., Su, L., & Cheng, X. 2019. Has-qa: Hierarchical answer spans model for open-domain question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence* , 6875-6882.

[25] Yu, S., Indurthi, S. R., Back, S., & Lee, H. 2018. A multi-stage memory augmented neural network for machine reading comprehension. In *Proceedings of the workshop on machine reading for question answering*, 21-30.