# Ranking and Sampling in Open-Domain Question Answering

**Yanfu Xu[1,2], Zheng Lin[1*], Yuanxin Liu[1,2], Rui Liu[1,2], Weiping Wang[1] and Dan Meng[1]**
[1]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2]School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
{xuyanfu,linzheng,liuyuanxin,liurui1995,wangweiping,mengdan}@iie.ac.cn

## Abstract

Open-domain question answering (OpenQA) aims to answer questions based on a number of unlabeled paragraphs. Existing approaches always follow the distantly supervised setup where some of the paragraphs are wrong-labeled (noisy), and mainly utilize the paragraph-question relevance to denoise. However, the paragraph-paragraph relevance, which may aggregate the evidence among relevant paragraphs, can also be utilized to discover more useful paragraphs. Moreover, current approaches mainly focus on the positive paragraphs which are known to contain the answer during training. This will affect the generalization ability of the model and make it be disturbed by the similar but irrelevant (distracting) paragraphs during testing. In this paper, we first introduce a ranking model leveraging the paragraph-question and the paragraph-paragraph relevance to compute a confidence score for each paragraph. Furthermore, based on the scores, we design a modified weighted sampling strategy for training to mitigate the influence of the noisy and distracting paragraphs. Experiments on three public datasets (Quasar-T, SearchQA and TriviaQA) show that our model advances the state of the art.

## 1 Introduction

Different from the traditional reading comprehension (RC) task, which aims to answer a question based on an off-the-peg paragraph, the reading paragraphs on OpenQA datasets are always collected via an information retrieval (IR) system. For example, given a question as shown in Figure 1, an OpenQA system usually coarsely retrieves paragraphs that are similar to the question. Therefore, the OpenQA model has to answer a question based on numerous paragraphs. As can be seen in Figure 1, some of the negative paragraphs are similar to the question whereas the answer string is missed, e.g. Paragraph4. We consider these paragraphs as distracting ones. Besides, in the distantly supervised setup (Mintz et al., 2009), it is postulated that the paragraphs that contain the answer string are ground truths, while even some of the positive paragraphs are wrong-labeled as they do not concern the question, e.g. Paragraph3. We consider these paragraphs as noisy ones. As a result, only Paragraph1 and Paragraph2 provide relevant answer to the question and the distracting and noisy paragraphs will prevent the model from identifying the correct answer.

**Question**: Who is Donald Duck's uncle ?
**Answer**: Scrooge
**Paragraph1 (positive, relevant)**: Donald's maternal uncle Scrooge McDuck made his first appearance ...
**Paragraph2 (positive, relevant)**: This story was reissued as a single record entitled "Donald Duck and Uncle Scrooge's Money Rocket" ...
**Paragraph3 (positive, noisy)**: In Scrooge's stories, Donald Duck was cast as Scrooge's assistant ...
**Paragraph4 (negative, distracting)**: Donald Duck is the uncle of Huey, Dewey, and Louie.

Figure 1: An example of OpenQA. The key information, answers in correct-labeled and wrong-labeled contexts are marked in blue, green and red respectively.

Recent OpenQA approaches always follow the retrieve-then-read pipeline and can be roughly divided into two categories: single-paragraph approaches (Joshi et al., 2017; Wang et al., 2018a) and multi-paragraph approaches (Chen et al., 2017; Clark and Gardner, 2018; Pang et al., 2019). Single-paragraph approaches mainly focus on the most relevant paragraph during reading. Multi-paragraph approaches apply a RC model to multiple paragraphs to extract the final answer. However, these approaches still face two major issues.

First, as described above, recent methods only consider the paragraph-question relevance in selecting paragraphs but neglect the paragraph-

[*]Corresponding author

paragraph relevance, which can be exploited to associate relevant paragraphs that are useful to the downstream RC task. Second, some previous approaches only take into account the positive paragraphs, which are known to contain the answer string, at the training stage. This will make the model become too confident in heuristics or patterns that are only effective in positive paragraphs. As a result, the model will suffer from the problem of impaired generalization ability and be easily bewildered by the distracting paragraphs during testing. Such phenomenon has also been observed by Clark and Gardner (2018). Therefore, a more carefully-designed training strategy is needed.

To address the two issues, we first propose an attention based ranking model which utilizes both the question-paragraph and the paragraph-paragraph relevance to compute a more accurate confidence score for each paragraph. Through the multi-level attention mechanism, the model can utilize the word-level and sentence-level information between the paragraph and the question. Through the sentence-level self-attention mechanism, the model can aggregate the effective evidence among relevant paragraphs and thus increase their confidence scores.

Second, based on the confidence scores, we design a modified weighted sampling strategy to select training paragraphs, which simultaneously ameliorate the influence of the distracting and noisy paragraphs. Through "sampling", the model can be prevented from being too confident in heuristics that are only effective in positive paragraphs. Through "weighted", the model can be less affected by the noisy paragraphs. After that, we concatenate the selected paragraphs and feed them to a RC model to predict the final answer.

We evaluate our work on three distantly supervised OpenQA datasets including Quasar-T (Dhingra et al., 2017b), SearchQA (Dunn et al., 2017) and TriviaQA (Joshi et al., 2017). Empirical results show that our model is effective in improving the performance of OpenQA and advances the state of the art on all three datasets. We additionally perform an ablation study on our model to give insights into the contribution of each submodule. We will release our code on GitHub for further research explorations. [1]

[1]https://github.com/xuyanfu/RASAOpenQA

## 2 Methodology

### 2.1 Task Definition

OpenQA aims to answer a question based on a number of paragraphs retrieved by IR systems. Formally, given a question containing $m$ words $Q = \{q_1, q_2, ..., q_m\}$ and a set of retrieved paragraphs $D = \{P_1, P_2, ..., P_n\}$, where $P_i = \{p_i^1, p_i^2, ..., p_i^{|p_i|}\}$ is the i-th paragraph, $|p_i|$ is the number of words in the i-th paragraph, our model is supposed to extract the answer from the paragraph collection $D$.

### 2.2 Two-stage Framework

Figure 2 gives an overview of our OpenQA model which is composed of two modules including a **ranker** and a **reader**. The ranker is responsible to compute a confidence score for each paragraph. Based on the confidence scores, we design five strategies to select $k$ paragraphs, which are then concatenated and passed to the reader. Note that $k$ is a hyperparameter. Finally, a sub-phrase of the concatenated paragraph is predicted as the answer according to reader's output, which is the probability distribution of the start and end positions.

### 2.3 Paragraph Ranker

We adopt a ranker to produce a confidence score for each paragraph. The ranker is comprised of an encoding layer, a word-level matching layer, a sentence-level matching layer and a sentence-level decoding layer.

#### Encoding Layer

Given a paragraph $P_i = \{p_i^1, p_i^2, ..., p_i^{|p_i|}\}$ and a question $Q = \{q_1, q_2, ..., q_m\}$, we map each word into a vector by combining the following features:

**Word Embeddings:** We use pre-trained word vectors, GloVe (Pennington et al., 2014), to obtain the fixed embedding of each word.

**Char Embeddings:** We map the characters in a word into 20-dimensional vectors which are then passed to a convolutional layer and a max pooling layer to obtain a fixed-size vector of the word.

**Common word:** The feature is set to 1 if the word appears in both the question and the paragraph, and otherwise it is set to 0. Then the feature is mapped into a vector of fixed size.

By concatenating the above features, we obtain the encoded sequence of the paragraph $\mathbf{P}_i^{emb} = \{\mathbf{p}_i^t\}_{t=1}^{|p_i|} \in \mathbb{R}^{d_{emb} \times |p_i|}$ and the question $\mathbf{Q}^{emb} = \{\mathbf{q}^t\}_{t=1}^{m} \in \mathbb{R}^{d_{emb} \times m}$ , where $|p_i|$ is the number

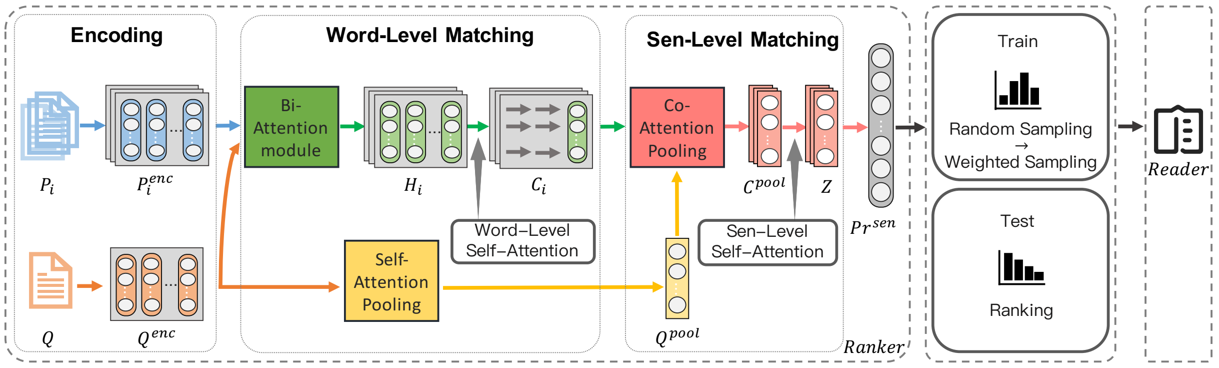


Figure 2: The framework of out model, which consists of a paragraph ranker (left) and a paragraph reader (right). The ranker first computes a confidence score for each paragraph and $k$ paragraphs are selected through the ranking and sampling strategies. Then, the reader concatenates the selected paragraphs together and computes the start score and the end score for each word in the concatenated paragraph.

of words in the i-th paragraph. After that, a bi-directional LSTM is used to calculate a contextual encoding for each word in the paragraph and the question respectively:

$$\mathbf{P}_i^{enc} = BiLSTM(\mathbf{P}_i^{emb}) \in \mathbb{R}^{d_{hid} \times |p_i|} \quad (1)$$

$$\mathbf{Q}^{enc} = BiLSTM(\mathbf{Q}^{emb}) \in \mathbb{R}^{d_{hid} \times m} \quad (2)$$

**Word-level Matching Layer**

The word-level matching layer takes $\mathbf{P}_i^{enc}$ and $\mathbf{Q}^{enc}$ as inputs and produces a new question-aware representation $\mathbf{C}_i$ for each paragraph. Following Clark and Gardner (2018), we first perform bi-directional attention between the question $\mathbf{Q}^{enc}$ and the paragraph $\mathbf{P}_i^{enc}$, which gives rise to $\mathbf{H}_i$ for $\mathbf{P}_i^{enc}$, where $\mathbf{H}_i \in \mathbb{R}^{d_{hid} \times |p_i|}$. Next, we feed $\mathbf{H}_i$ to the self-attention layer and obtain $\mathbf{E}_i \in \mathbb{R}^{d_{hid} \times |p_i|}$. We strictly implement the steps as suggested by Clark and Gardner (2018). After that, we use a bi-directional LSTM to obtain a question-aware paragraph embedding for the paragraph $P_i$:

$$\mathbf{C}_i = BiLSTM(\mathbf{E}_i) \in \mathbb{R}^{d_{hid} \times |p_i|} \quad (3)$$

**Sentence-level Matching Layer**

Given $\mathbf{D}_c = \{\mathbf{C}_1, \mathbf{C}_2, ..., \mathbf{C}_n\}$ and $\mathbf{Q}^{enc}$, the sentence-level matching layer produces a sentence-level representation $\mathbf{Z}_i$ for each paragraph by effectively exploiting the paragraph-question and the paragraph-paragraph relevance. Firstly, we use self-attention pooling to obtain a fixed-size vector for the encoded question representation. Each position in the encoded question representation is assigned with a score computed using a two-layer multi-layer perception (MLP). This score is normalized and used to compute a weighted sum over the columns of the question representation:

$$\alpha = softmax(\mathbf{W}_2^T tanh(\mathbf{W}_1 \mathbf{Q}^{enc})) \in \mathbb{R}^m \quad (4)$$

$$\mathbf{Q}^{pool} = \sum_t \alpha_t \mathbf{Q}_{:t}^{enc} \in \mathbb{R}^{d_{hid}} \quad (5)$$

Here, $\alpha_t$ is the normalized score for the t-th word in the question and $\mathbf{W}_1 \in \mathbb{R}^{d_{hid} \times d_{hid}}$, $\mathbf{W}_2 \in \mathbb{R}^{d_{hid}}$ are trainable parameters for the MLP. Next, we utilize the co-attention pooling (CAP) to get the fixed-length summary vector for each paragraph. The score for each position in $\mathbf{C}_i$ is calculated by measuring the similarity between $\mathbf{C}_i$ and $\mathbf{Q}^{pool}$, based on which we can compute a weighted sum over the columns of the question-aware paragraph embedding $\mathbf{C}_i$:

$$\hat{\alpha} = softmax(\mathbf{Q}_{pool}^T \mathbf{C}_i) \in \mathbb{R}^{|p_i|} \quad (6)$$

$$\mathbf{C}_i^{pool} = \sum_t \hat{\alpha}_t \mathbf{C}_{i:t} \in \mathbb{R}^{d_{hid}} \quad (7)$$

$$\mathbf{C}^{pool} = \{\mathbf{C}_i^{pool}\}_{i=1}^n \in \mathbb{R}^{d_{hid} \times n} \quad (8)$$

where $\hat{\alpha}_t$ is the normalized score for the t-th position in $\mathbf{C}_i$. Let $\mathbf{C}^{pool}$ represent the sequence of summary vectors for all paragraphs. Afterwards, in order to make full use of the paragraph-paragraph relevance, we use the sentence-level self-attention (SSA) and a bi-directional LSTM to produce the sentence-level paragraph representation $\mathbf{Z}$. The SSA is calculated as:

$$\mathbf{A} = softmax({\mathbf{C}^{pool}}^T \mathbf{C}^{pool}) \in \mathbb{R}^{n \times n} \quad (9)$$

$$\mathbf{U} = \mathbf{C}^{pool} \mathbf{A} \in \mathbb{R}^{d_{hid} \times n} \quad (10)$$

Here, A is the similarity matrix and $A_{ij}$ indicates the similarity between $\mathbf{C}_{:i}^{pool}$ and $\mathbf{C}_{:j}^{pool}$. $\mathbf{U}_{:i}$

is the i-th column of $\mathbf{U}$ which is the representation of the i-th paragraph where information of all the retrieved paragraphs is encompassed. $\mathbf{U}$ and $\mathbf{C}^{pool}$ are concatenated together to yield $\mathbf{G}$, which is defined by

$$\mathbf{G}_{:i} = [\mathbf{C}^{pool}_{:i}; \mathbf{U}_{:i}; \mathbf{C}^{pool}_{:i} \circ \mathbf{U}_{:i}; \mathbf{C}^{pool}_{:i} - \mathbf{U}_{:i}] \in \mathbb{R}^{4d_{hid}} \tag{11}$$

where $\circ$ is element-wise multiplication, $-$ is element-wise subtraction, and $[;]$ is vector concatenation across rows. After that, a bi-directional LSTM is used to obtain the final sentence-level paragraph representation $\mathbf{Z}$:

$$\mathbf{Z} = BiLSTM(\mathbf{G}) \in \mathbb{R}^{d_{hid} \times n} \tag{12}$$

**Sentence-level Decoding Layer**

The sentence-level decoding layer, which consists of a linear layer and a sotfmax function, will produce a normalized score for each paragraph:

$$\mathbf{Pr}^{sen} = softmax(W_3^T \mathbf{Z}) \in \mathbb{R}^n \tag{13}$$

where $W_3 \in \mathbb{R}^{d_{hid}}$ is a trainable weight vector. $\mathbf{Pr}^{sen}_i$ is the i-th element of $\mathbf{Pr}^{sen}$ which represents the probability that the i-th paragraph contains the answer.

## 2.4 Ranking and Sampling Strategies

Based on the retrieved paragraphs and their confidence scores, we design five strategies to select paragraphs for the reader.

**Ground Truth (GT)** strategy simply chooses all positive paragraphs which contain the answer string as reading datasets and filters out the negative ones. Since we have to know which paragraph contains the answer string, this strategy can only be used while constructing the training set.

**Random Sampling (RS)** strategy samples $k$ different paragraphs from all the retrieved paragraphs based on the uniform probability distribution. Different from the GT strategy, this strategy will include paragraphs, which do not contain an answer, in the training set. As a result, the RC model is prevented from becoming too confident in heuristics or patterns that are only effective when it is guaranteed that an answer string exists.

**Ranking (RK)** strategy selects $k$ top-ranked paragraphs based on the confidence scores produced by the ranker. In this strategy, the ranker is expected to select more relevant paragraphs, providing a better start point for the reader to predict the final answer.

**Weighted Sampling (WS)** strategy samples $k$ different paragraphs from all retrieved paragraphs based on the probability distribution generated by the ranker. This strategy is an improved version of RK. It hopes to consider the effects of the distracting paragraphs while filtering out the noisy ones.

**RS→WS** strategy is a combination of RS and WS. In the training process, in order to make full use of the dataset and mitigate the impact of the distracting paragraphs, we generate the initial training set for the reader via RS. After that, WS is applied to select paragraphs with higher accuracy, which reduces the influence of the distracting and noisy paragraphs.

During training, we can utilize all the five strategies to generate reading data, while in testing, considering that the paragraphs should be as clean as possible, we only use RK for paragraph selection. After that, we feed the selected paragraphs to the reader to generate the final answer.

## 2.5 Paragraph Reader

The reader is a traditional RC model, where the inputs are the $k$ selected paragraphs $D^{top} = \{P_1, P_2, ..., P_k\}$ and we regard $k$ as a hyperparameter. We concatenate these paragraphs together to obtain $\hat{P} = [P_1; P_2; ...; P_k]$ which aggregates the information from the selected paragraphs in word level. Given $\hat{P}$ and $Q$, the output of the reader, i.e, $\mathbf{Pr}^s, \mathbf{Pr}^e \in \mathbb{R}^{|\hat{P}|}$, are the probability distributions of the start index and the end index over the concatenated paragraph, where $|\hat{P}|$ is the number of words in the concatenated paragraph. Our reader is the same as in Clark and Gardner (2018) and we do not give the details here due to the space limitation. After that, we split $\mathbf{Pr}^s$ and $\mathbf{Pr}^e$ into $k$ vectors according to the length of each paragraph and obtain $\{\mathbf{Pr}^s_1, \mathbf{Pr}^s_2, ..., \mathbf{Pr}^s_k\}$ and $\{\mathbf{Pr}^e_1, \mathbf{Pr}^e_2, ..., \mathbf{Pr}^e_k\}$, where $\mathbf{Pr}^s_i$/ $\mathbf{Pr}^e_i$ represent the start/end probabilities of the i-th paragraph.

## 2.6 Training and Testing

Our model is trained in two stages. We follow the distantly supervised setup that all the paragraphs that contain the answer span are ground truths.

For the ranker, each paragraph in $\{P_i\}_{i=1}^n$ is associated with a label $\mathbf{y}_i \in \{0, 1\}$, and $\mathbf{y}_i$ equals to 1 if the paragraph contains the answer string. We follow the S-Net (Tan et al., 2018) to design the loss function of the ranker. Given the paragraph probabilities $\mathbf{Pr}^{sen} \in \mathbb{R}^n$, the ranker is trained by minimizing the loss:

$$\mathcal{L}_s = -\sum_{i=1}^{n}(\mathbf{y}_i log(\mathbf{Pr}_i^{sen}) + (1-\mathbf{y}_i) log(1-\mathbf{Pr}_i^{sen})) \tag{14}$$

For the reader, we first select $k$ paragraphs based on the RS→WS strategy and then obtain the final scores $\mathbf{Pr}_i^s$ and $\mathbf{Pr}_i^e$ for each paragraph, where $\mathbf{Pr}_i^s(t)$/$\mathbf{Pr}_i^e(t)$ represent the start/end scores of the t-th word in the i-th paragraph. As described above, the answer string can appear multiple times in a paragraph. Therefore, The reader is trained using a summed objective function that optimizes the negative log probability of selecting any correct answer span:

$$\mathcal{L}_r = -(log(\sum_{i=1}^{k}\sum_{t=1}^{|a_i|}(\mathbf{Pr}_i^s(x_i^t))) + log(\sum_{i=1}^{k}\sum_{t=1}^{|a_i|}(\mathbf{Pr}_i^e(z_i^t)))) \tag{15}$$

where $\{(x_i^t, z_i^t)\}_{t=1}^{|a_i|}$ is the set of the start and end positions of the answer strings that appear in the paragraph $P_i$.

During testing, we first utilize the ranker to predict a confidence score for each retrieved paragraph and utilize the RK strategy to select $k$ paragraphs. Next, the reader calculates the start and the end score of each word for all the selected paragraphs. Based on the reader-produced scores, there are two answer choosing methods:

**MAX** method is usually adopted in the traditional RC task. It simply chooses an answer span which has the maximum span score from all selected paragraphs. This span score is the product of the start score of the first word and the end score of the last word of the answer span.

**SUM** method first extracts an answer candidate $A_i$ that has the maximum span score from each paragraph. Next, if the answer candidates from different paragraphs refer to the same answer, we will sum up the scores of these answer candidates and choose the answer candidate with the maximum score as the final prediction. The similar method has also been adopted by recent approaches (Wang et al., 2018b; Lin et al., 2018; Pang et al., 2019).

## 3 Experiments

### 3.1 Datasets and Baselines

We evaluate our model on three OpenQA datasets, namely Quasar-T (Dhingra et al., 2017b), SearchQA (Dunn et al., 2017) and the unfiltered version of TriviaQA (Joshi et al., 2017).

**Quasar-T.** It consists of 43K open-domain trivia question-answer pairs, and about 100 paragraphs are provided for each question-answer pair by using the Solr search engine.

**SearchQA.** It has a total of more than 140k question-answer pairs, and roughly 50 webpage snippets are provided by the Google search engine as background paragraphs for each question.

**TriviaQA.** It contains approximately 95K open-domain question–answer pairs. The unfiltered version of TriviaQA is used in our experiment.

To better demonstrate the effectiveness of our model, we carefully select some recent approaches as baselines, including traditional RC models, i.e., BIDAF (Seo et al., 2016) , DECAPROP (Tay et al., 2018), single-paragraph models, i.e., R3 (Wang et al., 2018a), confidence-based models, i.e., DrQA (Chen et al., 2017) , DS-QA (Lin et al., 2018), S-Norm (Clark and Gardner, 2018), HAS-QA (Pang et al., 2019), MSR (Das et al., 2019) and answer re-ranking models, i.e., Re-Ranker (Wang et al., 2018b), Joint (Wang et al., 2018c).

### 3.2 Implementation Details

In the experiments, we adopt the same data preprocessing scheme as Clark and Gardner (2018). For the ranker, we use the 300-dimensional word embeddings pre-trained by GloVe (Pennington et al., 2014), which are fixed during training, and regard the 20-dimensional character embeddings as learnable parameters. The common word feature is mapped into a 4-dimensional vector and it is updated during training. We set the hidden size of LSTM to 150 and the number of LSTM layers to 1. We use Adam with learning rate 5e-4 to optimize the model. The batch size is set to 8 and dropout is applied to the outputs of all LSTM layers at a rate of 0.2. As for RS→WS, we first train the reader until convergence by RS. After that, we utilize WS to select paragraphs for training in order to further facilitate the performance of the reader. Our reader adopts the same hyperparameters as S-Norm (Clark and Gardner, 2018). While training the reader, the parameters of the ranker are fixed. The average length of paragraphs in TriviaQA is greater than that of Quasar-T and SearchQA, so we select 30, 30 and 15 paragraphs for Quasar-T, SearchQA and TriviaQA respectively.

| Models | Quasar-T | | SearchQA | | TriviaQA | | Average | |
|---|---|---|---|---|---|---|---|---|
| | EM | F1 | EM | F1 | EM | F1 | EM | F1 |
| BIDAF (Seo et al., 2016) | 25.9 | 28.5 | 28.6 | 34.6 | 41.1 | 47.4 | 31.8 | 36.8 |
| DECAPROP (Tay et al., 2018) | 38.6 | 46.9 | 56.8 | 63.6 | - | - | - | - |
| R3 (Wang et al., 2018a) | 35.3 | 41.7 | 49.0 | 55.3 | 47.3 | 53.7 | 43.8 | 50.2 |
| Re-Ranker (Wang et al., 2018b) | 42.3 | 49.6 | 57.0 | 63.2 | 50.6 | 53.7 | 49.9 | 55.5 |
| Joint (Wang et al., 2018c) | 45.9 | 53.9 | 58.3 | 64.2 | - | - | - | - |
| DrQA (Chen et al., 2017) | 37.7 | 44.5 | 41.9 | 48.7 | 32.3 | 38.3 | 37.3 | 43.8 |
| DS-QA (Lin et al., 2018) | 42.2 | 49.3 | 58.8 | 64.5 | 48.7 | 56.3 | 49.9 | 56.7 |
| S-Norm (Clark and Gardner, 2018) | 38.6 | 45.4 | 59.8 | 67.1 | 61.3 | 67.2 | 53.2 | 59.9 |
| HAS-QA (Pang et al., 2019) | 43.2 | 48.9 | 62.7 | 68.7 | 63.6 | 68.9 | 56.5 | 62.1 |
| MSR(BiDAF) (Das et al., 2019) | 40.6 | 47.0 | 56.2 | 61.4 | 55.9 | 61.7 | 50.9 | 56.7 |
| Our (RS, RK, MAX) | 46.6 | 56.5 | 60.4 | 68.0 | 63.8 | 70.4 | 56.9 | 64.9 |
| Our (RS→WS, RK, MAX) | 48.2 | 57.5 | 61.1 | 68.8 | 64.6 | 71.0 | 57.9 | 65.7 |
| Our (RS→WS, RK, SUM) | **48.6** | **57.8** | **62.9** | **69.8** | **65.2** | **71.3** | **58.9** | **66.3** |

Table 1: Experimental results on three OpenQA datasets: Quasar-T, SearchQA and TriviaQA. The results of 'BIDAF', 'DrQA' and 'S-Norm' are replicated from Pang et al. (2019). The unfiltered version of TriviaQA does not provide the answer of the test set. Therefore, we follow previous works to report the results on the development set of TriviaQA. Average: The average score of the three datasets.

### 3.3 Overall Results

In this section, we focus on the performance of the whole model. Table 1 presents the F1 and Exact Match (EM) scores of our model and the baseline models. These evaluation metrics are widely-adopted for OpenQA. Our (RS, RK, MAX) model utilizes RS and RK to select paragraphs for training and testing respectively and uses MAX to generate answer.

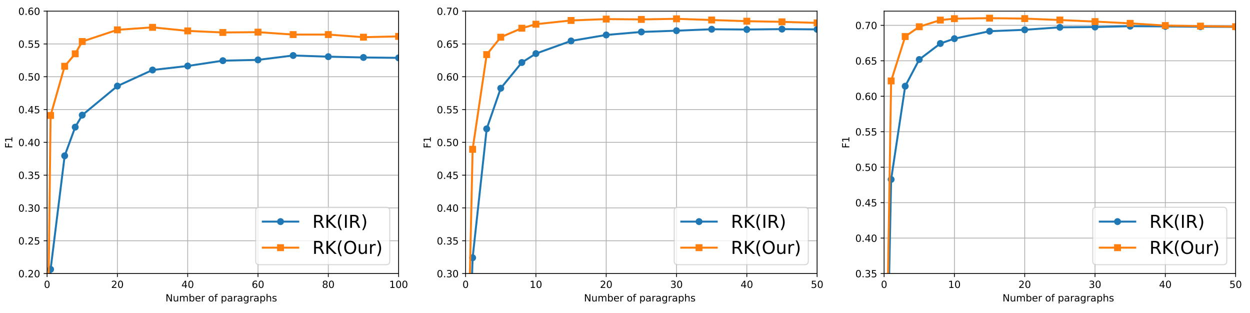
Our (RS, RK, MAX) model achieves better results on most of the datasets compared to the baselines. The main reason is that RS can select distracting paragraph to train the reader, thereby preventing the reader from being too confident in patterns that are only effective in positive paragraphs. Besides, compared with the paragraph-question relevance, the paragraph-paragraph relevance can further enhance the evidence among the relevant paragraphs. By jointly considering those relevance, RK can select more relevant paragraphs during testing to provide a better start point for the reader.

Our (RS→WS, RK, MAX) model consistently outperforms Our (RS, RK, MAX) model across the three datasets. The improvement is attributed to the adoption of the weighted sampling strategy. In the first stage of training, RS prevents excessive focus on the positive paragraphs. On this basis, WS further alleviates the influence of noisy paragraphs in the second stage without hurting the model's generalization ability.

We can observe that the SUM method (Wang et al., 2018b) helps our model better extract the correct answer. As we discussed, different from the traditional RC task, the answer string will appear multiple times in different paragraphs in OpenQA. By summing up the answer span scores which refer to the same answer candidate, we can extract a more accurate answer.

On SearchQA dataset, Our (RS→WS, RK, MAX) model achieves a close performance compared to HAS-QA model. 35% of the paragraphs in the development set of SearchQA contain the answer string, and for Quasar-T and TriviaQA the percentage of such paragraphs in the development set are 13% and 29% respectively. This indicates that SearchQA contains more positive paragraphs than other datasets. Since our model is better at filtering out the irrelevant data, the improvement in SearchQA is not as significant as in other datasets. However, Our (RS→WS, RK, SUM) model still achieves the SOTA result by adopting the SUM method and HAS-QA, which is a strong baseline, also adopts the similar method.

In the following sections, we give further analyses about the ranker, the selection strategies, the hyperparameter, the case study and the ablation study of our model.



(a) Quasar-T (b) SearchQA (c) TriviaQA

Figure 3: Impact of the number of the selected paragraphs during testing on Quasar-T, SearchQA and TriviaQA. RK (IR), RK (Our): Utilizing a information retrieval model with BM25 or our ranker to select top $k$ paragraphs.

| | **Quasar-T** | | | **SearchQA** | | |
|---|---|---|---|---|---|---|
| **Models** | **Top1** | **Top3** | **Top5** | **Top1** | **Top3** | **Top5** |
| IR | 21.9 | 38.5 | 47.6 | 48.5 | 74.0 | 82.3 |
| R3 | 40.3 | 51.3 | 54.5 | - | - | - |
| DS-QA | 27.7 | 36.8 | 42.6 | 59.2 | 70.0 | 75.7 |
| FSE | 42.8 | 55.2 | 58.9 | 67.9 | 81.1 | 86.0 |
| MSR | 42.9 | 55.5 | 59.3 | - | - | - |
| Our | **49.9** | **58.8** | **62.6** | **75.0** | **85.3** | **89.1** |

Table 2: The performance of the rankers on the Quasar-T and SearchQA. We re-implement 'IR' and 'FSE'.

| | **Quasar-T** | | **SearchQA** | | **TriviaQA** | |
|---|---|---|---|---|---|---|
| **Train** | **EM** | **F1** | **EM** | **F1** | **EM** | **F1** |
| GT | 33.6 | 41.6 | 40.2 | 47.9 | 59.8 | 66.4 |
| RK | 46.1 | 55.2 | 59.7 | 67.3 | 61.9 | 68.5 |
| RS | 46.6 | 56.5 | 60.4 | 68.0 | 63.8 | 70.4 |
| WS | 47.2 | 55.9 | 60.9 | 68.6 | 64.3 | 70.6 |
| RS$\rightarrow$WS | **48.2** | **57.5** | **61.1** | **68.8** | **64.6** | **71.0** |

Table 3: The performance of different training selection strategies on Quasar-T, SearchQA and TriviaQA.

## 3.4 Performance of the Ranker

We separately evaluate our ranker by comparing it with a typical IR model [2] and SOTA models which also have a ranking submodule to select paragraphs, i.e. DS-QA (Lin et al., 2018), R3 (Wang et al., 2018a), FSE (Zhang et al., 2018), MSR (Das et al., 2019). This evaluation simply focuses on whether the ground-truth appears in the Top 1, 3, 5 paragraphs. We follow the previous works to report the results on Quasar-T and SearchQA. As shown in Table 2, we can observe that our ranker surpasses the IR model with a large margin. The result implies that, the question-paragraph relevance is effective in selecting the relevant paragraphs. Compared with R3, DS-QA, FSE and MSR, which also adopt a neural network to select paragraphs, our model still works better. It suggests that exploiting the paragraph-question and paragraph-paragraph relevance with multi-level attention and self-attention mechanisms can further improve the performance of the ranker.

## 3.5 Performance of Selection Strategies

In this section, we compare the performance of different selection strategies for training. For fair comparison, a common scheme for testing is applied, where we use RK to select testing paragraphs and MAX to extract the answer. As shown in Table 3, the performance of RS is much better than GT and RK. The reason is that both GT and RK tend to train the reader on the positive paragraphs which inevitably impairs the generalization ability of the reader. It demonstrates that mainly choosing the positive paragraphs for training will hurt the performance of the model. Compared with RS, WS leads to a slight improvement thanks to the denoising effect. However, WS still reveals a bias toward the positive paragraphs during training, which may explain for the reason why the improvement is trivial. We can see that RS$\rightarrow$WS gains the best score on three datasets. The reason is that the RS$\rightarrow$WS strategy can not only ameliorates the influences of the distracting paragraphs but also filter out the noisy paragraphs to further strengthen our model.

## 3.6 Impact of the Number of Paragraphs

In order to further investigate the influences of different rankers and the hyperparameter $k$ during testing, we compare the final performance of RK (Our) with that of RK (IR). We utilize RS$\rightarrow$WS to select paragraphs for training and and extract answers through the MAX method.

[2]The IR model ranks the paragraph with BM25

| **Question**: Who was born on Krypton ? |
|---|
| **Answer**: Superman |
| **5 top-ranked paragraphs selected by our ranker**: |
| **P1 (positive, relevant)** : With a premise that taps into adolescent fantasy, Superman is born Kal-El on the alien planet Krypton, before being rocketed to Earth as an infant by his scientist father moments before the planet exploded. |
| **P2 (positive, relevant)** : The original story of Superman relates that he was born Kal-El on the planet Krypton, before being rocketed to Earth as an infant by his scientist father Jor-El, moments before Krypton's destruction. |
| **P3 (positive, relevant)** : Superman was born on the planet Krypton and was sent to Earth by his father moments before his planet exploded. |
| **P4 (positive, relevant)** : As of the early 1960s, (this story has undergone many revisions) Superman was born as Kal-el on the planet Krypton. |
| **P5 (positive, relevant)** : Superman was born Kal-El on the planet Krypton. |
| **5 top-ranked paragraphs selected by RKN**: |
| **P1 (positive, noisy)** : Television in the pilot episode of the 1950s television program Adventures of Superman Jor-El, portrayed by Robert Rockwell, was Krypton 's leading scientist, who tried to warn the ... |
| **P2 (negative, distracting)** : They did not tell anyone of his alien origins, letting family and friends believe he was born on the farm as their biological son, whom they named Clark Joseph Kent. |
| **P3 (positive, relevant)** : "Superman will be the only Kryptonian who survived the destruction of Krypton"- John Byrne on The Man of Steel. |
| **P4 (positive, noisy)** : In 1985, writer Alan Moore gave a somewhat darker glimpse into the world of Krypton in his story "For the Man Who Has Everything" (in Superman Annual), the premise being an ... |
| **P5 (positive, noisy)** : George Reeves, star of the Adventures of Superman television series, born in 1914. |

Table 4: An example from Quasar-T to illustrate the necessity of the paragraph-paragraph relevance. The key information, answers in correct-labeled and wrong-labeled contexts are marked in blue, green and red respectively. RKN: a naive version of our ranker which only utilizes the paragraph-question relevance.

RK (Our) and RK (IR) denote utilizing our ranker and a typical IR model, respectively, to select the $k$ top-ranked paragraphs for testing. As shown in Figure 3, it is obvious that the performance of the two models first increases and then remains stable or reduces as $k$ rises on three datasets. As the number of paragraphs increases, the chances are better that the answer is included in these paragraphs. However, the difficulty and running time for the reader also increase. As can be seen in Figure 3, the peak value of our ranker exceeds that of the IR model on three datasets, suggesting that our ranker can discover more useful and relevant paragraphs during testing, providing a better start point for the reader to predict the final answer.

## 3.7 Case Study

Table 4 shows an example from Quasar-T. The second row contains 5 top-ranked paragraphs which are produced by our ranker and the paragraphs in the third row are selected by a naive version of our ranker (RKN) which replaces the co-attention pooling (CAP) with max pooling for $\mathbf{C}_i$ and removes the sentence-level self-attention (SSA). For the question "Who was born on Krypton?", our ranker can select more related paragraphs and all of them support "Superman" as the answer. These paragraphs are inherently relevant to each other and provide enhanced evidence for the correct answer. However, most of the paragraphs in the third row are noisy and distracting, even though some of them are related to the question to some extent. Compared with RKN, which only uses the paragraph-question relevance, our completed ranker can further utilize the paragraph-paragraph relevance to aggregate the evidence among relevant paragraphs, thereby increasing their confidence scores. As a result, the reader is provided with more useful paragraphs to enhance the performance of our model.

| Models | P@1 | P@3 | P@5 | EM | F1 | ΔF1 |
|---|---|---|---|---|---|---|
| Our | 49.9 | 49.2 | 47.7 | 48.6 | 57.8 | 0 |
| -SUM | 49.9 | 49.2 | 47.7 | 48.2 | 57.5 | -0.3 |
| -RS→WS | 49.9 | 49.2 | 47.7 | 46.8 | 55.7 | -2.1 |
| -CAP | 47.8 | 46.9 | 45.8 | 47.6 | 57.0 | -0.8 |
| -SSA | 42.3 | 39.5 | 38.4 | 47.2 | 56.7 | -1.1 |
| -ranker | 21.9 | 20.1 | 19.1 | 44.2 | 52.6 | -5.2 |

Table 5: Ablation results on Quasar-T.

## 3.8 Ablation Study

Finally, we conduct an ablation study to analyze the effect of all proposed methods and report the results on Quasar-T. As shown in Table 5, we evaluate the performance of the ranker according to the precision at 1, 3 and 5. In comparison with the evaluation which assesses whether the ground-truth appears in the top-ranked paragraphs, this metric can better evaluate the performance of the ranker, as we expect to find as many relevant paragraphs in the $k$ top-ranked paragraphs as possible. We start with our completed model which utilizes RS→WS to select 30 paragraphs for training and RK to select 30 top-ranked paragraphs for testing. As shown in Table 5, the '-SUM' denotes using MAX to choose the answer and the performance drop indicates that SUM can further enhance our model. Replacing RS→WS with RK for training accounts for a 2.1% performance drop, which demonstrates the effectiveness of our proposed selection strategy. The '-CAP' means replacing the co-attention pooling with max pooling for $\mathbf{C}_i$ and the '-SSA' means removing the sentence-level self-attention , and from the results, we can see that both paragraph-question and paragraph-paragraph relevance are integral to our model. The '-ranker' denotes replacing our ranker with the IR model in training and testing stages. The replacement accounts for a performance drop of 5.2%, which illustrates the indispensable role of our ranker in the final architecture.

## 4 Related Work

Most recent OpenQA approaches extract answer from distantly supervised paragraphs which are retrieved from the web (Chen et al., 2017; Dhingra et al., 2017a; Buck et al., 2018).

The interest of research roughly falls into two categories, namely the single-paragraph approaches and the multi-paragraph approaches, both of which follow the retrieve-then-read pipeline. Single-paragraph approaches (Joshi et al., 2017; Wang et al., 2018a) select a most relevant paragraph from all the retrieved paragraphs and then feed it to a RC model to extract the answer. The dependence on a single paragraph renders the RC model vulnerable to selection errors and a large amount of information in the remaining paragraphs is consequently being neglected.

Multi-paragraph approaches can be further divided into answer-rerank methods and confidence-based methods. For answer-rerank methods, Wang et al. (2018b) adopted a single-paragraph approach to extract an answer candidate from each retrieved paragraph and then used an extra neural network to rerank the candidates. Wang et al. (2018c) jointly trained the candidates extraction and reranking steps with reinforcement learning to further improve the performance. But these methods still rely heavily on the candidates extraction results.

Chen et al. (2017) proposed the first confidence method which retrieved multiple paragraphs with an IR system and applied a RC model to each paragraph for extracting the answer with the highest confidence. Das et al. (2019) introduced a new framework to make use of the result of the reader for improved paragraph retrieval. However, these models suffer from the problem of impaired generalization ability as a result of too much focus on the positive paragraphs in training. Clark and Gardner (2018) presented a new method, S-Norm. They used the sampling strategy and the shared-norm objective function to teach the model to ignore non-answer containing paragraphs. However, the retrieved paragraphs are always noisy and this will hurt the performance of the model.

Zhang et al. (2018) and Lee et al. (2018) adopted a neural network to rank the retrieved paragraphs. Lin et al. (2018) and Pang et al. (2019) also ranked the paragraphs and took into account both the ranking results and the scores produced by the reader to predict the final answer. However, they mainly utilize the paragraph-question relevance to rank the paragraphs and train the reader mainly on the positive paragraphs.

## 5 Conclusion

In this paper, we tackle the issues caused by the noisy and distracting paragraphs on OpenQA via ranking and sampling. By jointly considering the question-paragraph and the paragraph-paragraph relevance, our ranking model can calculate a more accurate confidence score for each paragraph. Through the modified weighted sampling strategy, our model can make full use of the dataset and mitigate the influence of the distracting and noisy paragraphs. Experimental results on three challenging public OpenQA datasets (Quasar-T, SearchQA and TriviaQA) show that our model advances the state of the art.

## 6 Acknowledgement

We thank the anonymous reviewers for their valuable suggestions. This work was supported by National Natural Science Foundation of China (No. 61976207, No. 61906187).

## References

Christian Buck, Jannis Bulian, Massimiliano Ciaramita, Wojciech Gajewski, Andrea Gesmundo, Neil Houlsby, and Wei Wang. 2018. Ask the right questions: Active question reformulation with reinforcement learning. In *Proceedings of International Conference on Learning Representations*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 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)*, pages 1870–1879. Association for Computational Linguistics.

Christopher Clark and Matt Gardner. 2018. Simple and effective multi-paragraph reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 845–855. Association for Computational Linguistics.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, and Andrew McCallum. 2019. Multi-step retriever-reader interaction for scalable open-domain question answering. In *Proceedings of International Conference on Learning Representations*.

Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. 2017a. Gated-attention readers for text comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1832–1846. Association for Computational Linguistics.

Bhuwan Dhingra, Kathryn Mazaitis, and William W. Cohen. 2017b. Quasar: Datasets for question answering by search and reading. *Computing Research Repository*, arXiv:1707.03904.

Matthew Dunn, Levent Sagun, Mike Higgins, V. Ugur Güney, Volkan Cirik, and Kyunghyun Cho. 2017. Searchqa: A new q&a dataset augmented with context from a search engine. *Computing Research Repository*, arXiv:1704.05179.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611. Association for Computational Linguistics.

Jinhyuk Lee, Seongjun Yun, Hyunjae Kim, Miyoung Ko, and Jaewoo Kang. 2018. Ranking paragraphs for improving answer recall in open-domain question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 565–569. Association for Computational Linguistics.

Yankai Lin, Haozhe Ji, Zhiyuan Liu, and Maosong Sun. 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)*, pages 1736–1745. Association for Computational Linguistics.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1003–1011. Association for Computational Linguistics.

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Lixin Su, and Xueqi Cheng. 2019. HAS-QA: hierarchical answer spans model for open-domain question answering. In *Proceedings of Association for the Advancement of Artificial Intelligence*.

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

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. In *Proceedings of International Conference on Learning Representations*.

Chuanqi Tan, Furu Wei, Nan Yang, Bowen Du, Weifeng Lv, and Ming Zhou. 2018. S-net: From answer extraction to answer synthesis for machine reading comprehension. In *Proceedings of Association for the Advancement of Artificial Intelligence*.

Yi Tay, Anh Tuan Luu, Siu Cheung Hui, and Jian Su. 2018. Densely connected attention propagation for reading comprehension. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018*, pages 4911–4922.

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerry Tesauro, Bowen Zhou, and Jing Jiang. 2018a. R 3: Reinforced ranker-reader for open-domain question answering. In *Proceedings of Association for the Advancement of Artificial Intelligence*.

Shuohang Wang, Mo Yu, Jing Jiang, Wei Zhang, Xiaoxiao Guo, Shiyu Chang, Zhiguo Wang, Tim Klinger, Gerald Tesauro, and Murray Campbell. 2018b. Evidence aggregation for answer re-ranking in open-domain question answering. In *Proceedings of International Conference on Learning Representations*.

Zhen Wang, Jiachen Liu, Xinyan Xiao, Yajuan Lyu, and Tian Wu. 2018c. Joint training of candidate extraction and answer selection for reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1724. Association for Computational Linguistics.

Chen Zhang, Xuanyu Zhang, and Hao Wang. 2018. A machine reading comprehension-based approach for featured snippet extraction. In *Proceedings of IEEE International Conference on Data Mining*, pages 1416–1421.