RoR: Read-over-Read for Long Document Machine Reading Comprehension

Jing Zhao[1], **Junwei Bao**[1]*, **Yifan Wang**[1], **Yongwei Zhou**[2], **Youzheng Wu**[1], **Xiaodong He**[1], **Bowen Zhou**[1]
[1]JD AI Research, Beijing, China
[2]Harbin Institute of Technology, Harbin, China
{zhaojing857,baojunwei,wangyifan15}@jd.com
{wuyouzheng1,xiaodong.he,bowen.zhou}@jd.com
ywzhou@hit-mtlab.net

Abstract

Transformer-based pre-trained models, such as BERT, have achieved remarkable results on machine reading comprehension. However, due to the constraint of encoding length (*e.g.,* 512 WordPiece tokens), a long document is usually split into multiple chunks that are independently read. It results in the reading field being limited to individual chunks without information collaboration for long document machine reading comprehension. To address this problem, we propose **RoR**, a *read-over-read* method, which expands the reading field from chunk to document. Specifically, RoR includes a chunk reader and a document reader. The former first predicts a set of regional answers for each chunk, which are then compacted into a highly-condensed version of the original document, guaranteeing to be encoded once. The latter further predicts the global answers from this condensed document. Eventually, a voting strategy is utilized to aggregate and rerank the regional and global answers for final prediction. Extensive experiments on two benchmarks QuAC and TriviaQA demonstrate the effectiveness of RoR for long document reading. Notably, RoR ranks 1st place on the QuAC leaderboard [1] at the time of submission (May 17th, 2021)[2].

1 Introduction

The task of machine reading comprehension (MRC), which requires machines to answer questions through reading and understanding a given document, has been a growing research field in natural language understanding (Hermann et al., 2015; Trischler et al., 2017; Rajpurkar et al., 2016, 2018; Joshi et al., 2017; Choi et al., 2018).

Transformer-based pre-trained models have been widely proven to be effective in a range of natural language processing tasks, including the MRC task (Devlin et al., 2019; Liu et al., 2019; Yang et al., 2019; Clark et al., 2020). Typically, these models consist of a stack of transformer blocks that only encode a length-limited sequence (*e.g.,* 512). However, the input sequences in some MRC tasks may exceed the length constraint. For example, each instance in open-domain MRC usually consists of a collection of passages, such as TriviaQA (Joshi et al., 2017), one of the most popular open-domain MRC datasets, containing 6,589 tokens on average. In addition, for conversational MRC task, such as QuAC (Choi et al., 2018), existing methods incorporate conversation history by prepending the previous utterances to the current question, which is packed with the document into a length input (707 tokens on average).

To handle a long document that exceeds the length constraint, a commonly used approach is to split a document into multiple individual chunks and then predict answers from each chunk separately. The highest scoring span in these answers is selected as the final answer. This approach is straightforward but results in two problems: (1) the reading field is limited to the regional chunk instead of the complete document; and (2) the scores of the answers are not comparable as they are not globally normalized over chunks.

To address these problems, we propose **RoR**, a *read-over-read* pipeline, which is able to expand the reading field from chunk-level to document-level. RoR contains a chunk reader and a document reader, both of which are based on the pre-trained model. Specifically, the chunk reader first predicts the regional answers from each chunk. These answers are then compacted into a new document through a minimum span coverage algorithm guaranteeing that its sequence length is shorter than the limitation (*i.e.,* 512). By this means, all regional answers can be normalized in one document. This document serves as the highly-condensed version

*Corresponding Author: baojunwei001@gmail.com
[1]https://quac.ai/
[2]Our code is available at https://github.com/JD-AI-Research-NLP/RoR

of the original document, which is further read by the document reader to predict a set of global answers. As the chunk reader and global reader provide high confidence answers from different views, we fully leverage both of them for final answer prediction. Specifically, after predicting the regional and global answer spans, a voting strategy is proposed and utilized to rerank them. This voting strategy is based on the idea that a candidate regional or global answer span overlapped more with the others is more likely to be correct.

The contributions are summarized as follows:

- We propose a *read-over-read* pipeline containing an enhanced chunk reader and a document reader, which is able to solve the problem of long document reading limitation in existing models.

- We propose a voting strategy to rerank the answers from regional chunks and a condensed document, overcoming the major drawback in aggregating the answers from different sources.

- Extensive experiments on long document benchmarks are conducted to verify the effectiveness of our model. Especially on the QuAC dataset, our model achieves state-of-the-art results over all evaluation metrics on the leaderboard.

2 Related Work

MRC is a fundamental task in natural language understanding that aims to determine the correct answers to questions after reading a given passage (Hermann et al., 2015; Trischler et al., 2017; Rajpurkar et al., 2016, 2018). The best performing models in various MRC tasks are commonly based on the pre-trained language models (PLMs) within the typical encoding limit of 512 tokens. However, the input sequence in some MRC tasks usually exceeds the length limit, such as conversational MRC and open-domain MRC.

Conversational MRC, which extends the traditional single-turn MRC, requires the models to additionally understand the conversation history (Reddy et al., 2019; Choi et al., 2018; Gao et al., 2018; Huang et al., 2019; Gupta et al., 2020) as dialog and conversational recommendation systems (Lu et al., 2021). A straightforward but effective approach of modeling the history is to prepend the previous dialogs to the current question, which will compose a lengthy input sequence with the relatively long document (Gong et al., 2020).

Open-domain MRC is a task of answering questions using a large collection of passages (Joshi et al., 2017; Dunn et al., 2017; Kwiatkowski et al., 2019). The main challenge of this task is that the sequence length of multiple passages relevant to each question far exceeds the length limit of 512 tokens. For example, documents in TriviaQA (Joshi et al., 2017) contain 6,589 tokens on average.

To enable the PLMs to encode long documents, a common approach is to chunk the document into overlapping chunks of length 512, then process each chunk separately, which inevitably causes the two problems aforementioned. Another intuitive approach is to increase the encoding length of the PLMs. For example, the recently proposed PLMs Longformer (Beltagy et al., 2020) and Big bird (Zaheer et al., 2021), specifically for long document modeling, have extended the encoding length from 512 to 4,096. However, their encoding length is fixed. The two problems caused by chunking still exist when encoding the sequences longer than 4,096. In contrast, our proposed model RoR is flexible which is able to encode sequences of arbitrary length. Moreover, RoR is assembleable and its encoder can be replaced with any PLMs, such as BERT and Longformer.

Theoretically, hierarchical models can be adapted to long document MRC (Yang et al., 2016; Wang et al., 2018; Yang et al., 2020). However, deploying the large transformer-based PLMs as the encoders of hierarchical models can be prohibitively costly. Typically, hierarchical models parallelly encode the splitted chunks of a long document with multiple transformers, which requires extremely large GPU support. In contrast, RoR only needs to read a chunk at each encoding process, then gradually predict all answers from chunk to document. Therefore, RoR is able to deal with a long document without consuming too much computing resources and can be more widely used than hierarchical models.

3 Approach

3.1 Task Formulation

Given a document P, a question q, the task of MRC is to predict an answer span y from P based on the comprehension of P and q. If q is an unanswerable question, the QuAC dataset requires the model to give an unanswerable tag as the final answer. To model the dialog history in QuAC, we prepend previous pairs of (question, answer) to the current

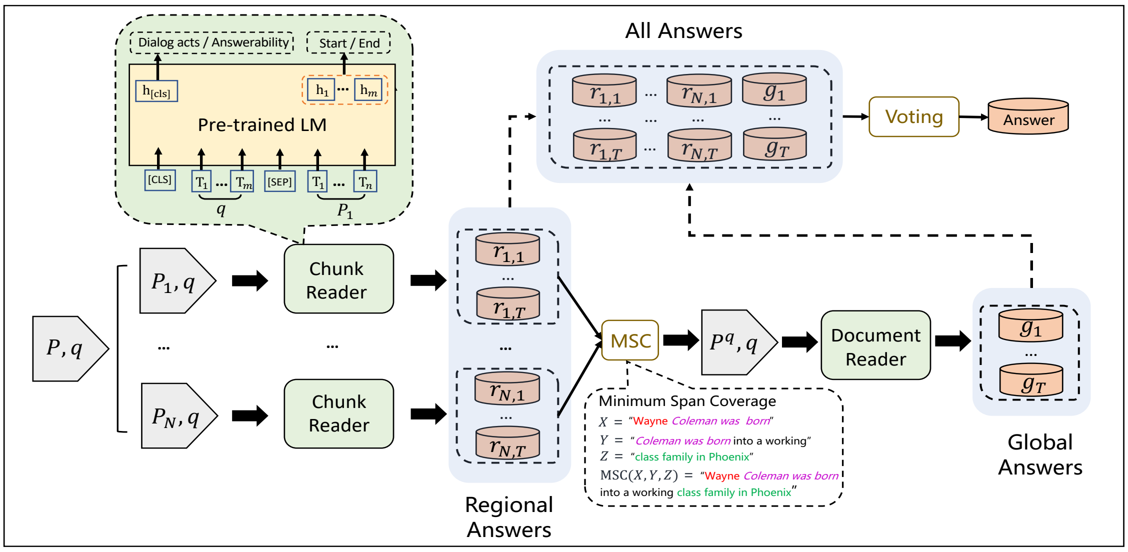
Dialog acts / Answerability
Start / End
h[cls]
h1 ... hm
Pre-trained LM
[CLS] T1 ... Tm [SEP] T1 ... Tn
q
P1
All Answers
Voting
Answer
P, q
P1, q
PN, q
Chunk Reader
Chunk Reader
Regional Answers
MSC
P^q, q
Document Reader
Global Answers
Minimum Span Coverage
X = "Wayne Coleman was born"
Y = "Coleman was born into a working"
Z = "class family in Phoenix"
MSC(X,Y,Z) = "Wayne Coleman was born into a working class family in Phoenix"

Figure 1: The architecture of the proposed *Read-over-Read* pipeline.

question to form the question q. Formally, $q = [H_k; [\texttt{SEP}]; q_k]$, where q_k is k-th question and H_k is dialog history.

Additionally, a special task in the QuAC dataset is dialog act prediction. QuAC provides two dialog acts, namely, continuation (Follow up) and affirmation (Yes/No). The continuation dialog act consists of three possible labels (follow up, maybe follow up or don't follow up). The affirmation dialog act also consists of three labels (yes, no or neither). Both two dialog act predictions are three-label classification tasks.

3.2 Framework Overview

The architecture of our proposed *read-over-read* pipeline is illustrated in Figure 1. RoR includes a chunk reader and a document reader, both of which employ the PLM as the text encoder. Given a data sample (P, q), we split it into multiple chunk-based sample $\{(P_1, q), ..., (P_N, q)\}$ with slide-window, where N is the number of split chunks. The chunk reader first predicts a set of regional answers $\{\{r_{i,j}\}_{j=1}^{T}\}_{i=1}^{N}$ from all chunks, where T is the max number of the predicted answers for one chunk. The regional answers are then compacted to a new document P^q by a minimum span coverage algorithm (MSC). Notably, most of the answers in TriviaQA dataset are named entities that cannot reflect enough contextual information. Therefore, for the TriviaQA dataset, we use the sentences where the regional answers are located to compact to P^q. P^q is the condensed version of original document P to the question q, which is packed into the text encoder at once. P^q is further read by the document reader to predict the global answers $\{g_i\}_{i=1}^{T}$. The answers from all chunks and the condensed document are aggregated together and reranked by a voting strategy to choose the final answer.

3.3 Chunk Reader

The chunk reader predicts the regional answers for each chunk based on the contextualized representations of a given document which are obtained by the pre-trained encoder. This section introduces the components of the chunk reader in detail.

3.3.1 Text Encoder

The goal of a text encoder is to convert the input sequence into a series of contextualized feature representations $\{\mathbf{h}_i\}_{i=1}^{L}$, where L is the length of input sequence. The input sequence of the encoder contains a chunk P, the question q which are concatenated to one token sequence with a special splitter $[\texttt{SEP}]$, represented as $\texttt{X} = [[\texttt{CLS}]; q; [\texttt{SEP}]; P]$, where $\texttt{X}$ is the input token sequence. the representation of $[\texttt{CLS}]$ is treated as the sentence-level feature for the sentence classification tasks, *i.e.* answerability and dialog acts prediction.

3.3.2 Answer Prediction

The answers prediction of conversational MRC requires two levels of feature, token level feature for predicting answer span and sentence level feature for predicting dialog acts and answerability. Open-domain MRC only requires token level feature.

Token level answer. The encoder representations

$\{\mathbf{h}_i\}_{i=1}^L$ serve as the token level features, which are used to compute the probability of each token being the begin or the end of an answer span. Concretely, $\{\mathbf{h}_i\}_{i=1}^L$ are projected onto start logit and end logit through multi-layer perceptrons separately, which are then sent to a softmax function to compute the start and end probability distributions along all tokens in this sequence. The probabilities of each token being the begin and the end of predicted span are calculated as follows:

$$r_i^s = \mathbf{W}_2^s \mathtt{tanh}(\mathbf{W}_1^s \mathbf{h}_i) \tag{1}$$
$$r_i^e = \mathbf{W}_2^e \mathtt{tanh}(\mathbf{W}_1^e [\mathbf{h}_i ; \mathbf{h}_s]) \tag{2}$$
$$p^s = \mathtt{softmax}(r^s) \tag{3}$$
$$p^e = \mathtt{softmax}(r^e) \tag{4}$$

where $\mathbf{W}_1^s, \mathbf{W}_2^s, \mathbf{W}_1^e, \mathbf{W}_2^e$ are trainable parameters of the projection function. $\mathbf{h}_s$ is the token representation of the start label. p^s, p^e are the start and the end probability distributions over all tokens respectively, where $p^s \subseteq \mathbb{R}^L, p^e \subseteq \mathbb{R}^L$. Different from predicting start and end independently, we explicitly model the relation between them. As shown in Equation 2, the calculation of end distribution depends on start position. The training objective of token level prediction is defined as the cross entropy loss of start and end predictions:

$$\mathcal{L}_t = -\frac{1}{M} \sum_{j=1}^{M} [log(p_{y_j^s}^s) + log(p_{y_j^e}^e)] \tag{5}$$

where y_j^s and y_j^e are the ground-truth of start and end positions of j-th example respectively. M is the number of examples.

Sentence level answer. Encoder representation of [CLS] token $\mathbf{h}_{\texttt{[CLS]}}$ is viewed as the sentence level feature, which is used to predict dialog acts and answerability by:

$$p^f = \delta(\mathbf{W}_2^f \mathtt{tanh}(\mathbf{W}_1^f \mathbf{h}_{\texttt{[CLS]}}) \tag{6}$$
$$p^y = \delta(\mathbf{W}_2^y \mathtt{tanh}(\mathbf{W}_1^y \mathbf{h}_{\texttt{[CLS]}}) \tag{7}$$
$$p^u = \sigma(\mathbf{W}_2^u \mathtt{tanh}(\mathbf{W}_1^u \mathbf{h}_{\texttt{[CLS]}}) \tag{8}$$

where $\mathbf{W}_1^f, \mathbf{W}_2^f, \mathbf{W}_1^y, \mathbf{W}_2^y, \mathbf{W}_1^u, \mathbf{W}_2^u$ are trainable parameters. δ is a softmax function. σ is a sigmoid function. p^f, p^y are prediction distributions of continuation and affirmation respectively, where $p^f \subseteq \mathbb{R}^3, p^y \subseteq \mathbb{R}^3$. p^u is the prediction score of answerability. Their corresponding cross entropy losses are defined as:

$$\mathcal{L}_{ct} = -\frac{1}{M} \sum_{j=1}^{M} [logp_{y_j^{ct}}^f] \tag{9}$$

$$\mathcal{L}_{af} = -\frac{1}{M} \sum_{j=1}^{M} [logp_{y_j^{af}}^y] \tag{10}$$

$$\mathcal{L}_{na} = -\frac{1}{M} \sum_{j=1}^{M} [y_j^{na} logp^u + (1 - y_j^{na}) log(1 - p^u)] \tag{11}$$

where $y_j^{ct}, y_j^{af}, y_j^{na}$ are the ground-truths of continuation, affirmation and answerability respectively. The training objective of sentence level prediction is defined as:

$$\mathcal{L}_s = \mathcal{L}_{ct} + \mathcal{L}_{af} + \mathcal{L}_{na} \tag{12}$$

3.3.3 Answer Calibration

The highest scoring span among the regional answers is sometimes not the span with the highest F1 score. Motivated by this issue, we introduce an answer calibration mechanism, with the goal of predicting more accurate regional answers. Particularly, given the answer candidates, we first compute their span representation, which is a weighted self-aligned vector:

$$\alpha^t = \mathtt{softmax}(\mathbf{W}_r \mathbf{h}_{s_t:e_t}) \tag{13}$$

$$\mathbf{c_t} = \sum_{j=s_t}^{e_t} \alpha_j^t \mathbf{h}_j \tag{14}$$

where $\mathbf{W}_r$ is a trainable parameter. $\mathbf{h}_{s_t:e_t}$ is a shorthand for stacking a list of vectors $\mathbf{h}_j$ $(s_t \leqslant j \leqslant e_t)$. s_t, e_t are the start and end of the t-th answer candidate. $\mathbf{c}_t$ is the span representation of the t-th answer candidate. Then, all candidate representations are transferred to a multi-head self-attention layer (Multi-SelfAtt) to capture the similarities and the differences among them, and the detailed calculation is shown as:

$$\mathbf{c}_t = \mathbf{c}_t + \mathtt{Emb}(t) \tag{15}$$
$$\mathbf{c}' = \mathtt{Multi\text{-}SelfAtt}(\mathbf{c}) \tag{16}$$
$$\beta = \mathtt{softmax}(\mathtt{tanh}(\mathbf{W}_o \mathbf{c}'_{0:T})) \tag{17}$$

where Emb is the position embedding of t and a smaller value t means a higher original prediction score, which is a valuable feature for the model to identify the different importance of candidates. $\mathbf{c}$

is list of candidate representations, formally $\mathbf{c} = [\mathbf{c}_1, ..., \mathbf{c}_T]$. $\mathbf{W}_o$ is trainable parameter. β is the distribution of calibration score over the answer candidates, $\beta \subseteq \mathbb{R}^T$. The cross entropy loss for answer calibration is given as:

$$\mathcal{L}_{ac} = -\frac{1}{T}\sum_{j=1}^{T}[log\beta_{y_j^{ac}}] \tag{18}$$

where y_j^{ac} is a manually constructed label. We define y_j^{ac} is the candidate which obtains the highest F1 score with the gold span among all candidates. If the highest F1 score is zero and the corresponding question is answerable, we randomly replace a candidate with the gold span.

3.4 Document Reader

The predicted spans from different chunks are compacted to a new text P^q with a designed Minimum Span Coverage (MSC) algorithm, as shown in Algorithm 1. MSC guarantees that P^q covers all regional spans and is sufficiently condensed to be encoded once.

Algorithm 1 Minimum Span Coverage

Input: $\mathbf{A} = \{a_i\}_{i=1}^{NT}$
A is the set of regional spans from all chunks
N is the number of chunks, T is the number of regional spans for one chunk

```
1: for $a_i$ in A do
2:     for $a_{j,j\neq i}$ in A do
3:         if overlap($a_i$, $a_j$) then
4:             A += coverage($a_i$, $a_j$)
5:             A -= $a_i$, A -= $a_j$
6: coverage($a_i$,$a_j$) is the span corresponding to (start,end) = ($\texttt{min}(s_i, s_j), \texttt{max}(e_i, e_j)$), where $(s_i, e_i)$ and $(s_j, e_j)$ are (start,end) of $a_i$ and $a_j$.
7: Recursively execute the above steps until no condition of overlap($a_i$,$a_j$)
8: Concatenate the elements in A together as $P^q$
9: return $P^q$
```

The input sequence of the text encoder is $\texttt{X} = [[\texttt{CLS}]; q; [\texttt{SEP}]; P^q]$, which is further read by the document reader to predict the answers as the global answers. The span label of the document reader is the longest common substring between P^q and the original gold span.

The global answers and the regional answers are aggregated as final predictions. For answerability prediction, the document reader predicts a global no answer score U_g and the chunk reader predicts a series of no answer score $U_r = \{u_k\}_{k=1}^{N}$, where u_k is the predicted no answer score of k-th chunk. The final no answer score $\mathcal{S}_{na}$ is defined as:

$$\mathcal{S}_{na} = \lambda U_g + (1-\lambda)\texttt{min}(U_r) \tag{19}$$

where λ is a hyperparameter to tune the weights of the global answers and the regional answers.

3.5 Voting Strategy

After aggregating the answer spans, we re-score them with a voting strategy that is based on a hypothesis: the spans predicted by both the chunk reader and the document reader are more likely to be correct. This strategy allows all spans to vote with each other to choose the most common span. Concretely, the voting score of each span is obtained by:

$$\texttt{Voting}(x_i) = \frac{1}{T\text{-}1}\sum_{j=1}^{T}(\texttt{F1}(x_i, x_{j,j\neq i})) \tag{20}$$

$$\texttt{F1}(x_i, x_j) = \frac{2R(i,j)P(i,j)}{R(i,j)+P(i,j)} \tag{21}$$

$$R(i,j) = \frac{|x_i \cap x_j|}{|x_j|} \tag{22}$$

$$P(i,j) = \frac{|x_i \cap x_j|}{|x_i|} \tag{23}$$

where | | denotes the number of words, $|x_i \cap x_j|$ denotes the number of the common words between x_i and x_j. The function of $\texttt{F1}(x_i, x_j)$ represents the sequence similarity between x_i and x_j. A larger voting score means that the corresponding span is similar to more candidates than the others. Finally, the voting strategy reranks answer spans according to the original prediction score $\mathcal{S}(x)$ and the voting score:

$$\texttt{score}(x) = \gamma\mathcal{S}(x) + (1-\gamma)\texttt{Voting}(x) \tag{24}$$

γ is the weight of two scores.

3.6 Training and Inference

We adopt the multi-task learning idea to jointly learn the predictions of answer span, answerability and dialog acts. All parameters are trained with an end-to-end manner. The training loss of the chunk reader L_c and the document reader L_d are:

$$\mathcal{L}_c = \mathcal{L}_t + \mathcal{L}_s + \mathcal{L}_{ac} \tag{25}$$

$$\mathcal{L}_d = \mathcal{L}_t + \mathcal{L}_s \tag{26}$$

The detailed training and inference processes are given in Algorithm 2.

Algorithm 2 Training and Inference Process

```
Input: D = training set, d = test set
Initialize: $\Theta_1, \Theta_2 \leftarrow$ pre-trained parameters
1: Training
2: Train $\Theta_1$ on D with $\mathcal{L}_c$, then predict answers
   on D to construct a new dataset $\mathbf{D}'$ through
   Algorithm 1
3: Train $\Theta_2$ on $\mathbf{D}'$ with $\mathcal{L}_d$
4: Inference
5: $\Theta_1$ predict the regional answers set $A$ on d and
   construct a new dataset $\mathbf{d}'$ through Algorithm 1
6: $\Theta_2$ predict the global answers set $A'$ on $\mathbf{d}'$
7: Aggregate and rerank the final answers set $\widetilde{A} =$
   $A \cup A'$ with Equation 24
```

3.7 Experiment Setup

3.7.1 Dataset

Our experiments are mainly conducted on two long document datasets QuAC (Question Answering in Context) (Choi et al., 2018) and TriviaQA (Joshi et al., 2017). QuAC is a large-scale dataset created for simulating information-seeking conversations. Its questions are often more open-ended, unanswerable, or only meaningful within the dialog context. TriviaQA is a large-scale open-domain MRC dataset, which requires cross sentence reasoning to find answers. It contains data from Wikipedia and Web domains, where Wikipedia subset is used in our work. The statistic information of these two dataset is summarized in Table 1.

	Train	**Dev**	**Test**
TriviaQA			
# questions	61,888	7,993	7,701
# tokens / input	11,222	11,382	-
QuAC			
# questions	83,568	7,354	7,353
# tokens / input	641	707	-
# dialogs	11,567	1000	1002
# question / dialog	7.2	7.4	7.4
% unanswerable	20.2	20.2	20.1

Table 1: Statistics of two datasets. # denote the number of each item. % denote a percentage value.

3.7.2 Evaluation Metrics

For answer span prediction, the QuAC challenge provides two evaluation metrics, the word-level F1 and the human equivalence score (HEQ). The word-level F1 measures the overlap of the prediction and the gold span after removing stopwords. HEQ measures the percentage of examples for which model F1 score is higher than the average human F1 score. HEQ contains two variants HEQ-Q and HEQ-D. HEQ-Q is 1 if model performance exceeds the human performance for each question. HEQ-D is 1 if model performance of all the questions in the dialog exceeds human. For dialog act prediction, the accuracy is adopted as evaluation metric. For the TriviaQA dataset, word-level F1 score and exact match (EM) are used as evaluation metrics.

3.7.3 Implementation Details

We tried three different PLMs, BERT-large [3], ELECTRA-large [4] and Longformer-large [5] as initialization parameters of text encoder to verify the effectiveness of RoR comprehensively. The max sequence length of questions is set to 128 and the answer length is set to 64. The stride of the sliding window for splitting documents is set to 128. The batch size is set to 12. The model is optimized using Adam (Kingma and Ba, 2015) with learning rate = 2e-5, maximal gradient clipping = 1.0. The hyperparameter λ is set to 0.9, γ is set to 0.5. In the inference process, we use beam search to predict end position based on start position and the beam size is 5. The decision of answerability depends on the numerical comparison between the no answer score $\mathcal{S}_{na}$ in equation 19 and a threshold ζ, which is set to 0.3. If $\mathcal{S}_{na}$ is higher than ζ, the corresponding question is unanswerable.

MSC algorithm is important in RoR which guarantees the condensed form of an arbitrarily long input sequence shorter than 512 tokens through limiting the total number and length of the regional answers. In practice, we perform the following four operations on TriviaQA and QuAC:

- The max number of answer candidates for each chunk T is set to 5.
- A document is split up to 7 chunks in QuAC and 15 chunks in TriviaQA.
- The regional answer is truncated if longer than 15 tokens.
- Many regional answers overlap or even differ by a few words. MSC algorithm removes the duplicate words to ensure the condensed document shorter.

[3]https://github.com/google-research/BERT
[4]https://github.com/google-research/electra
[5]https://github.com/allenai/Longformer

Model	F1	HEQ-Q	HEQ-D	Answerability	Continuation	Affirmation
Longformer	74.3	71.5	14.5	76.5	64.4	89.8
BERT	67.4	63.7	7.9	68.3	63.3	88.4
BERT-RoR	**69.6**	**65.8**	**9.8**	**74.7**	**63.5**	**88.6**
ELECTRA	73.8	71.2	14.3	76.1	64.9	89.7
ELECTRA-RoR	**75.7**	**73.4**	**17.8**	**78.2**	**65.0**	**90.0**

Table 2: Results on the development set of QuAC.

After the operations above, the longest condensed document contains 471 tokens in TriviaQA and 184 tokens in QuAC. When RoR is adapted to other datasets, the length of the condensed documents can be guaranteed to be shorter than 512 as long as the parameters in the above four operations are adjusted correspondingly.

In order to improve the model performance, some data augmentations are applied to better train the model. Specifically, ELECTRA is fine-tuned on other MRC datasets before fine-tuned on QuAC, such as SQuAD (Rajpurkar et al., 2018) and CoQA (Reddy et al., 2019), hoping to transfer the knowledge in other datasets to our model. Experimental results show that CoQA has a much higher lifting effect than SQuAD. This is because both CoQA and QuAC are conversational MRC datasets, while SQuAD is a single-turn MRC dataset. The answers in CoQA are free-form and generally short (average answer length = 2.7), which is quite different from QuAC (average answer length = 15.1). As a result, we choose the rationale sentence of the gold span in CoQA as the prediction target.

3.8 Main Results

Results on QuAC. Table 2 displays the experimental results on the development set of QuAC. In view of the fact that the sequence length in QuAC dataset does not exceed the encoding length limit of Longformer (*i.e.,* 4096), we did not apply RoR to the Longformer. The results show that RoR significantly improves the performance of PLMs on all evaluation metrics, illustrating the effectiveness of RoR on long document modeling. Among the three PLMs, Longformer, which can encode longer sequences, performs best, followed by ELECTRA. Nevertheless, with the enhancement of RoR, ELECTRA-RoR outperforms Longformer and achieves state-of-the-art results over all metrics on the dev set of QuAC.

Model	F1	HEQ-Q	HEQ-D
Human	81.1	100	100
ELECTRA-RoR	**74.9**	**72.2**	**16.4**
EL-QA	74.6	71.6	16.3
History QA	74.2	71.5	13.9
TR-MT	74.4	71.3	13.6
GraphFlow (Chen et al., 2020)	64.9	60.3	5.1
HAM (Qu et al., 2019b)	65.4	61.8	6.7
FlowDelta (Yeh and Chen, 2019)	65.5	61.0	6.9
HAE (Qu et al., 2019a)	62.4	57.8	5.1
FlowQA (Huang et al., 2019)	64.1	59.6	5.8
BiDAF++ *w/ 2-Context*	60.1	54.8	4.0
BiDAF++ (Peters et al., 2018)	50.2	43.3	2.2

Table 3: Test results on QuAC with sample methods on the leaderboard `https://quac.ai/`.

Official leaderboard results on QuAC. QuAC challenge provides a hidden test set, where the dialog acts prediction is not the main task of the leaderboard and their evaluation scores are not considered in the final model ranking. Table 3 displays the span prediction results of all baselines and our model, from which we can see that our model ELECTRA-RoR outperforms the previous best performing model EL-QA and achieves new state-of-the-art on all three metrics. From the results of the leaderboard, we observe that the top ranking models almost all use the advanced pre-trained models, such as ELECTRA, RoBERTa and BERT. Although some models have a well-designed model structure, they still lag behind the models that uses the pre-trained models as encoder, showing the powerful modeling capabilities of the pre-trained model. For example, FlowDelta boosts the F1 score of FlowQA from 64.1 to 65.5 with the help of BERT. Compared to BiDAF++, BiDAF++ *w/ 2-Context* incorporates two turns of previous dialog history and significantly improves the performance of BiDAF++, verifying the importance of historical information in the conversational MRC.

Model	F1	HEQ-Q	HEQ-D	Answerability	Continuation	Affirmation
chunk reader	73.8	71.2	14.3	76.1	64.9	89.7
w/ calibration	74.1	71.4	15.3	76.9	64.6	89.7
w/ document reader	74.8	72.1	15.7	77.4	64.7	89.7
w/ voting strategy	75.4	72.9	16.9	77.4	64.7	89.7
w/ knowledge transfer	75.7	73.4	17.8	78.2	65.0	90.0

Table 4: Ablation study of ELECTRA-RoR on the development set of QuAC.

Results on TriviaQA. Table 5 displays the experimental results on the TriviaQA dataset. The experimental results show that RoR is able to comprehensively improve the score of the PLMs and the average F1 gain is 2.1, proving that RoR is also highly effective in the task of open-domain MRC.

Similar to the performance on the QuAC dataset, among the three PLMs, Longformer performs best on the TriviaQA dataset, ELECTRA followed, and the worst is BERT. Although Longformer has such excellent performance which used to be state-of-the-art model on the TriviaQA dataset, RoR still further improve its performance (F1 score from 77.8 to 80.0). The test results show that all improvements of RoR in Table 3 and Table 5 are statistically significant (paired t-test, p-value < 0.01).

Model	F1	EM
BERT	68.4	60.7
BERT-RoR	**70.3**	**62.1**
ELECTRA	70.6	65.3
ELECTRA-RoR	**72.9**	**67.8**
Longformer	77.8	73.0
Longformer-RoR	**80.0**	**75.0**

Table 5: Results on the TriviaQA dataset.

3.9 Ablation Test

We conduct an ablation analysis on development set of QuAC to investigate the contributions of each module of the best model ELECTRA-RoR.

Table 4 displays the results of ablated systems, where we gradually add the proposed modules to the model structure. It can be observed that answer calibration mechanism boosts the performance of the chunk reader in all three span evaluation metrics. An interesting phenomenon is the calibration mechanism has the ability to improve the prediction accuracy of unanswerable questions. This may be because we mask some training instances during calculating the calibration loss $\mathcal{L}_{ac}$ if their questions are unanswerable, since all span candidates are not correct. This may provide some supervision signals for model training to identify unanswerable questions. Next we analyze the contributions of the document reader. It can be seen that the evaluation scores of span prediction are further improved when adding the document reader. Especially the F1 score is improved by 0.7. Moreover, the accuracy of answerability prediction is also improved. This is because the document reader predicts a global no answer score which contributes to the final decision of answerability through equation 19. Afterwards, the ablation results show that the voting strategy yields substantial improvement over the RoR model. Finally, we can see that the transfer module could also comprehensively enhance the model performance, proving the efficiency of knowledge transfer in MRC task.

In terms of continuity and affirmation, the accuracy scores do not change much in the ablation experiments. Nevertheless, we cannot completely ignore them when training our model, as we find that the performance of RoR will drop if remove the losses of $\mathcal{L}_{ct}$ and $\mathcal{L}_{af}$. This reflects that the dialog acts can indeed encourage the encoder to produce more generic representations that benefit the answer span prediction task.

3.10 Discussion on Answer Calibration

As shown in Table 4, the answer calibration mechanism improves the performance of the chunk reader. This section further discusses its other promoting effects on RoR. The predicted regional answers determine the quality of global answers that will be predicted by the document reader. An ablation test on the condensed document P^q is conducted on the dev set. We find that the average F1 score of P^q drops from 30.3 to 29.7 if without the answer calibration. This test validates that the calibration mechanism is able improve the input accuracy of the document reader. Meanwhile, the quality of global answers are further improved.

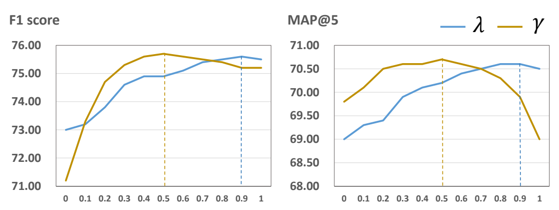

Figure 2: The influence of weight.

We also attempt to integrate the answer calibration mechanism into the document reader, but the results show that the improvement is not significant. We speculate the reason is that the predictions of the document reader is sufficiently accurate.

3.11 Influence of Weight Parameter

In the voting strategy, the parameter γ weights the prediction score and the voting score. In the process of answers aggregation, λ weights the score of the regional answers and the global answers. This section explores the influence of these two parameters on the model.

Figure 4 displays the F1 and mean average precision (MAP) curves of ELECTRA-RoR on QuAC with respect to different λ and γ. The results suggest that the model performance are sensitive to the weights. As the weights increases, both curves show a trend of increasing at first and then decreasing, reaching peaks at 0.5 and 0.9 respectively. For the changing degree of the curves, γ actually has a greater influence on RoR than λ. We notice that MAP score dramatically declines when reducing the weight of the voting score (*i.e.,* γ from 0.5 to 1.0), indicating that the voting score is a reliable basis to rerank the final prediction results.

4 Conclusion

In this work, a *read-over-read* (RoR) pipeline is proposed for long document MRC, which contains an enhanced chunk reader to predict the regional answers and a document reader to predict the global answers. Moreover, a voting strategy is designed to optimize the process of answer aggregation in RoR. Comprehensive empirical studies on QuAC and TriviaQA demonstrate the effectiveness of RoR, which comprehensively improves the performances of the PLMs on long document reading. Meanwhile, ELECTRA-RoR achieves state-of-the-art over all evaluation metrics on QuAC leaderboard.

5 Acknowledge

The authors would like to thank three anonymous reviewers for their useful feedback. This work is supported by the National Key Research and Development Program of China under Grant No. 2020AAA0108600.

References

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer.

Yu Chen, Lingfei Wu, and Mohammed J. Zaki. 2020. Graphflow: Exploiting conversation flow with graph neural networks for conversational machine comprehension. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 1230–1236. ijcai.org.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wentau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. 2018. QuAC: Question answering in context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2174–2184, Brussels, Belgium. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: pretraining text encoders as discriminators rather than generators. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

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

Matthew Dunn, Levent Sagun, Mike Higgins, V. Ugur Guney, Volkan Cirik, and Kyunghyun Cho. 2017. Searchqa: A new qa dataset augmented with context from a search engine.

Jianfeng Gao, Michel Galley, and Lihong Li. 2018. Neural approaches to conversational AI. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*, pages 1371–1374. ACM.

Hongyu Gong, Yelong Shen, Dian Yu, Jianshu Chen, and Dong Yu. 2020. Recurrent chunking mechanisms for long-text machine reading comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6751–6761, Online. Association for Computational Linguistics.

Somil Gupta, Bhanu Pratap Singh Rawat, and Hong Yu. 2020. Conversational machine comprehension: a literature review. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2739–2753, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1693–1701.

Hsin-Yuan Huang, Eunsol Choi, and Wen-tau Yih. 2019. Flowqa: Grasping flow in history for conversational machine comprehension. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Mandar Joshi, Eunsol Choi, Daniel 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, Vancouver, Canada. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv preprint*, abs/1907.11692.

Yu Lu, Junwei Bao, Yan Song, Zichen Ma, Shuguang Cui, Youzheng Wu, and Xiaodong He. 2021. RevCore: Review-augmented conversational recommendation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1161–1173, Online. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Chen Qu, Liu Yang, Minghui Qiu, W. Bruce Croft, Yongfeng Zhang, and Mohit Iyyer. 2019a. BERT with history answer embedding for conversational question answering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019*, pages 1133–1136. ACM.

Chen Qu, Liu Yang, Minghui Qiu, Yongfeng Zhang, Cen Chen, W. Bruce Croft, and Mohit Iyyer. 2019b. Attentive history selection for conversational question answering. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019, Beijing, China, November 3-7, 2019*, pages 1391–1400. ACM.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Siva Reddy, Danqi Chen, and Christopher D. Manning. 2019. CoQA: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. NewsQA: A machine comprehension dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 191–200, Vancouver, Canada. Association for Computational Linguistics.

Yizhong Wang, Kai Liu, Jing Liu, Wei He, Yajuan Lyu, Hua Wu, Sujian Li, and Haifeng Wang. 2018. Multi-passage machine reading comprehension with cross-passage answer verification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1918–1927, Melbourne, Australia. Association for Computational Linguistics.

Liu Yang, Mingyang Zhang, Cheng Li, Michael Bendersky, and Marc Najork. 2020. Beyond 512 tokens: Siamese multi-depth transformer-based hierarchical encoder for long-form document matching.

In *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020*, pages 1725–1734. ACM.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 5754–5764.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. 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*, pages 1480–1489, San Diego, California. Association for Computational Linguistics.

Yi-Ting Yeh and Yun-Nung Chen. 2019. FlowDelta: Modeling flow information gain in reasoning for conversational machine comprehension. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 86–90, Hong Kong, China. Association for Computational Linguistics.

Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2021. Big bird: Transformers for longer sequences.