Recurrent Chunking Mechanisms for Long-Text Machine Reading Comprehension

Hongyu Gong[1*]   **Yelong Shen**[2†]   **Dian Yu**[3]   **Jianshu Chen**[3]   **Dong Yu**[3]

[1]University of Illinois at Urbana-Champaign, IL, USA
[2]Microsoft Dynamics 365 AI, Redmond, WA, USA
[3]Tencent AI Lab, Bellevue, WA, USA

hgong6@illinois.edu
yeshe@microsoft.com
{yudian,jianshuchen,dyu}@tencent.com

Abstract

In this paper, we study machine reading comprehension (MRC) on long texts, where a model takes as inputs a lengthy document and a question and then extracts a text span from the document as an answer. State-of-the-art models tend to use a pretrained transformer model (e.g., BERT) to encode the joint contextual information of document and question. However, these transformer-based models can only take a fixed-length (e.g., 512) text as its input. To deal with even longer text inputs, previous approaches usually chunk them into *equally-spaced* segments and predict answers based on each segment independently without considering the information from other segments. As a result, they may form segments that fail to cover the correct answer span or retain insufficient contexts around it, which significantly degrades the performance. Moreover, they are less capable of answering questions that need cross-segment information.

We propose to let a model learn to chunk in a more flexible way via reinforcement learning: a model can decide the next segment that it wants to process in either direction. We also employ recurrent mechanisms to enable information to flow across segments. Experiments on three MRC datasets – CoQA, QuAC, and TriviaQA – demonstrate the effectiveness of our proposed recurrent chunking mechanisms: we can obtain segments that are more likely to contain complete answers and at the same time provide sufficient contexts around the ground truth answers for better predictions.

1 Introduction

Teaching machines to read, process, and comprehend natural language is a coveted goal of machine reading comprehension (MRC) problems

(Hermann et al., 2015; Hill et al., 2016; Rajpurkar et al., 2016; Trischler et al., 2017; Zhang et al., 2018; Kočiský et al., 2018). Many existing MRC datasets have a similar task definition: given a document and a question, the goal is to extract a span from the document (in most cases) or instead generate an abstractive answer to answer the question.

There is a growing trend of building MRC readers (Hu et al., 2019; Xu et al., 2019; Yang et al., 2019a; Keskar et al., 2019) based on pre-trained language models (Baker et al., 2019; Yang et al., 2019b), such as GPT (Radford et al., 2018) and BERT (Devlin et al., 2019). These models typically consist of a stack of transformer layers that only allow fixed-length (e.g., 512) inputs. However, it is often the case that input sequences exceed the length constraint, e.g., documents in the TriviaQA dataset (Joshi et al., 2017) contain 2,622 tokens on average. Some conversational MRC datasets such as CoQA (Reddy et al., 2018) and QuAC (Choi et al., 2018) often go beyond the length limit as we may need to incorporate previous questions as well as relatively long documents into the input to answer the current question.

To deal with long text inputs, a commonly used approach firstly chunks the input text into equally-spaced segments, secondly predicts the answer for each individual segment, and finally ensembles the answers from multiple segments (Devlin et al., 2019). However, there are two major limitations of this approach: first, a predetermined large stride size for chunking may result in incomplete answers, and we observe that models are more likely to fail when the answer is near the boundaries of a segment, compared to the cases when an answer is in the center of a segment surrounded by richer context (Figure 1); second, we empirically observe that chunking with a smaller stride size contributes little to (sometimes even hurts) the model performance. A possible explanation is that predicting answer

---

for each segment independently may cause incomparable answer scores across segments. A similar phenomenon is also observed in open-domain question answering tasks (Clark and Gardner, 2017).

Considering the limitations mentioned above, we propose recurrent chunking mechanisms (RCM) on top of the transformer-based models for MRC tasks. There are two main characteristics of RCM. First, it could let the machine reader learn how to choose the stride size intelligently when reading a lengthy document via reinforcement learning, so it helps prevent extracting incomplete answers from a segment and retain sufficient contexts around the answer. Second, we apply recurrent mechanisms to allow the information to flow across segments. As a result, the model can have access to the global contextual information beyond the current segment.
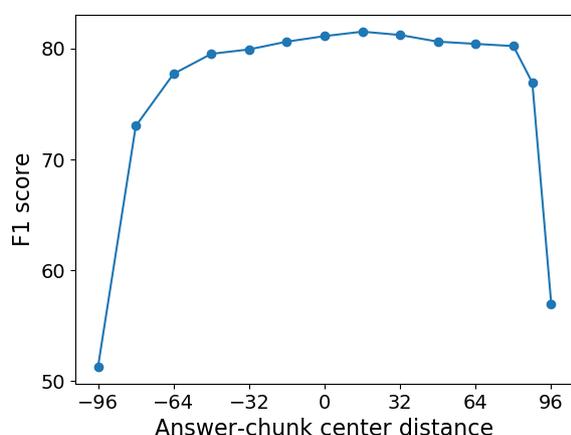

Figure 1: The influence of the distance between the center of the answer span and the center of the segment. The test performance (in F1 score) is evaluated on CoQA using a BERT-Large reader. The best performance is achieved when the chunk center coincides with the answer span center. Within the distance of ± 80 (in tokens), while 99% answers are completely covered, the performance degrades as the segment center moves away from the answer center, and the segment contains fewer relevant contexts. When the distance reaches 96, more than half of the predicted spans are incomplete.

In the experiments, we evaluate the proposed RCM[1] on three MRC datasets: CoQA, QuAC, and TriviaQA. Experimental results demonstrate that RCM leads to consistent performance gains on these benchmarks. Furthermore, it also generates segments that are more likely to cover the entire answer spans and provide richer contextual information around the ground truth answers.

---

[1]The code is available at https://github.com/HongyuGong/RCM-Question-Answering.git.

The primary contributions of this work are:

- We propose a chunking mechanism for machine reading comprehension to let a model learn to chunk lengthy documents in a more flexible way via reinforcement learning.

- We also apply recurrence to allow information transfer between segments so that the model can have knowledge beyond the current segment when selecting answers.

- We have performed extensive experiments on three machine reading comprehension datasets: CoQA, QuAC, and TriviaQA. Our approach outperforms two state-of-the-art BERT-based models on different datasets.

2 Method

The proposed recurrent chunking mechanisms (RCM) are built upon the pre-trained BERT models. We will briefly introduce the basic model in Section 2.1, and then the RCM approach in Section 2.2 and 2.3. More details of our model in training and testing are presented in Sections 2.4 and 2.5.

2.1 Baseline Model

Pre-trained BERT model has been shown to achieve new state-of-the-art performance on many MRC datasets (Devlin et al., 2019). Here, we introduce this basic BERT model, which is used as our baseline. As the maximum input length in BERT is restricted to be 512, a widely adopted strategy is to chunk a long document into multiple segments with a fixed stride size (i.e., 128). Following the input format of BERT, the input for each document segment starts with "CLS" token, which is followed by question tokens "Q" and document segment tokens. We use "SEP" token as a separator between the question and the segment. We also append a special "UNK" token at the end of the segment to handle unanswerable questions. If a given question is annotated as unanswerable, we mark the "UNK" token as the ground truth answer during training. Accordingly in evaluation, if "UNK" token is selected by the model from the input segment, we output the answer as "unanswerable".

Answer Extraction. Following previous work on extractive machine reading comprehension, we predict the start and the end positions of the answer span in the given document segment. BERT first generates a vector representation $\mathbf{h}_{c,i}$ for each i-th
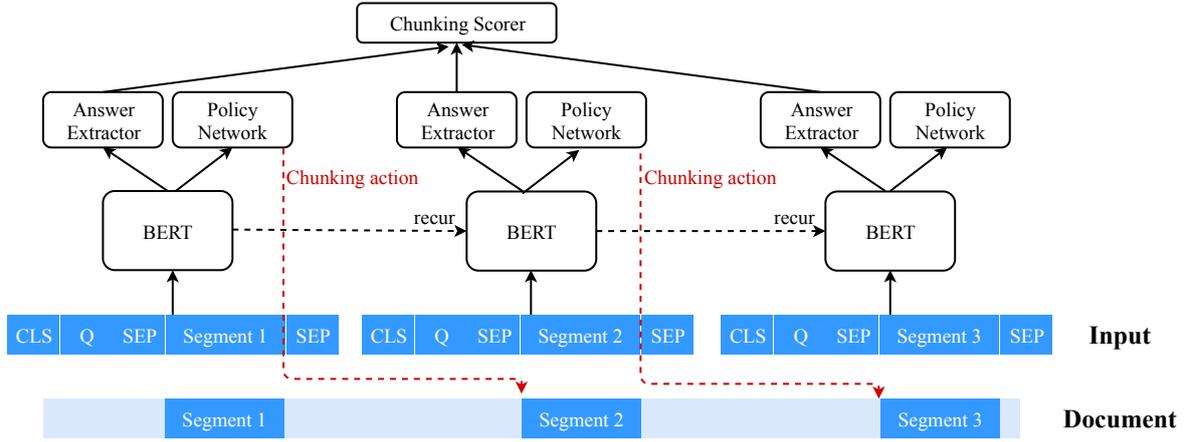
Figure 2: BERT generates representations for each input sequence, and recurrence accumulates information over segments. Based on these representations, the answer extractor extracts answers from the current segment, and the policy network takes chunking action and moves to the next segment. Chunking scorer scores each segment by estimating its likelihood of containing an answer and selects answers among predictions from multiple segments.

token in the c-th segment. Given $\mathbf{h}_{c,i}$, the model scores each token in terms of its likelihood of being the start token of the answer span.

$$l_{c,i}^{\text{start}} = \mathbf{w}_s^T \mathbf{h}_{c,i}, \qquad (1)$$

where $\mathbf{w}_s$ is the model parameter. The probability $p_{c,i}^{\text{start}}$ that the answer starts at the i-th token is computed by applying the softmax to $l_{c,i}^{\text{start}}$.

$$p_{c,i}^{\text{start}} = \text{softmax}(l_{c,i}^{\text{start}}) \qquad (2)$$

Likewise, the model scores how likely the answer ends at the j-th token in segment c using

$$l_{c,j}^{\text{end}} = \mathbf{w}_e^T \mathbf{h}_{c,j}, \qquad (3)$$

where $\mathbf{w_e}$ is the model parameter. The probability of the j-th token being the end of the answer (denoted as $p_{c,j}^{\text{end}}$) is calculated in a similar manner as Eq. (2).

Answer Ensemble. The baseline model adopts a max-pooling approach to ensemble candidate answers from multiple segments. The answer with the highest probability is selected.

2.2 Recurrent Mechanisms

The baseline model makes the answer prediction for each document segment independently, which may cause incomparable answer scores across segments due to the lack of document-level information. We propose to use a recurrent layer to propagate the information across different segments and a chunking scorer model to estimate the probability that a segment contains the answer.

For an input sequence containing the segment c, BERT's representation for its first token "CLS" is taken as the local representation v_c of the segment. The segment representation is further enriched with the representations of previously generated segments via recurrence. We denote the enriched segment representation as $\tilde{\mathbf{v}}_c$:

$$\tilde{\mathbf{v}}_c = f(\mathbf{v}_c, \tilde{\mathbf{v}}_{c-1}), \qquad (4)$$

where $f(\cdot)$ is the recurrent function. We consider two recurrent mechanisms here: gated recurrence and Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) recurrence.

Gated recurrence is simply a weighted sum of its inputs:

$$f_{\text{gated}}(\mathbf{v}_c, \tilde{\mathbf{v}}_{c-1}) = \alpha \mathbf{v}_c + \beta \tilde{\mathbf{v}}_{c-1}, \qquad (5)$$

where α and β are coefficients depending on the inputs. We have $\alpha, \beta = \text{softmax}(\mathbf{w}_r^T[\mathbf{v}_c, \tilde{\mathbf{v}}_{c-1}])$, where $\mathbf{w}_r$ is a model parameter.

The LSTM recurrence, which uses LSTM unit as the recurrence function, takes $\mathbf{v}_c$ as the current input and $\tilde{\mathbf{v}}_{c-1}$ as the previous hidden state.

$$f_{\text{LSTM}}(\mathbf{v}_c, \tilde{\mathbf{v}}_{c-1}) = \text{LSTM}(\mathbf{v}_c, \tilde{\mathbf{v}}_{c-1}). \qquad (6)$$

Chunking Scorer. Given the enriched segment representation $\tilde{\mathbf{v}}_c$ as input, the chunking scorer produces an scalar q_c by:

$$q_c = \sigma(\mathbf{W}_c \tilde{\mathbf{v}}_c + \mathbf{b}_c), \qquad (7)$$

where $\mathbf{W}_c$ and $\mathbf{b}_c$ are model parameters, and $\sigma(\cdot)$ is the sigmoid function. The scalar q_c is an estimation of the probability that an answer is included in

segment c. Then, the chunking scorer uses q_c to further refine the likelihood of the candidate answers from different segments (see Sections 2.4 and 2.5 for more details on this part of chunking scorer).

2.3 Learning to Chunk

The baseline approach divides a long document into multiple segments with a fixed stride size, from left to right. We will present an approach that could allow the model to choose the stride size flexibly by itself when reading the document. Our motivation, as mentioned in Section 1, is to prevent the answer span from being too close to the segment boundary and covering incomplete answers.

We formulate the problem of learning-to-chunk under the framework of reinforcement learning. We define the **state** s of the model to be the segments that a model has processed up to the current segment c, i.e., $s = \{1, 2, \ldots, c\}$. The **action** a is the stride size and direction (forward or backward) the model chooses to move to the next document segment. We define the **action space** A as a set of strides, e.g., $A = \{-16, 16, 32\}$, where 32 indicates moving forward with stride size 32 and -16 indicates moving backward with stride size 16. In this work, we represent the state s with the enriched segment representation $\tilde{\mathbf{v}}_c$.

Chunking Policy. The chunking policy gives the probability $p^{\text{act}}(a \mid s)$ of taking an action a at the current state s, which is modeled by a one-layer feedforward neural network:

$$p^{\text{act}}(a \mid s) = \text{softmax}(\mathbf{W}_a \tilde{\mathbf{v}}_c + \mathbf{b}_a), \quad (8)$$

where $\mathbf{W}_a$ and $\mathbf{b}_a$ are trainable parameters.

Fig. 2 gives an overview of the proposed recurrent chunking mechanisms built upon the BERT model: the chunking policy network takes the enriched segment representation as the input to generate the chunking action, which decides the next segment to be processed.

2.4 Training

In the training phase of the recurrent chunking mechanisms, the stride actions of moving to the next segment are sampled according to the probability given by the chunking policy (Sutton and Barto, 2018). Our model generates a sequence of document segments for each question. We train the answer extractor and chunking scorer network with supervised learning, and we train the chunking policy network via reinforcement learning.

Supervised Learning for Answer Extraction. Just as the baseline model, we train the answer extraction network via supervised learning. Given a question, the answer extractor classifies whether a word from a document segment is the start or the end of the answer. The cross-entropy loss can be computed given the ground-truth answer and the predictions of the answer extractor. Suppose that the i^*-th and j^*-th tokens are the answer start and end, respectively. The training objective to minimize the following cross-entropy loss, L_{ans}:

$$L_{\text{ans}} = -\sum_c \log p^{\text{start}}_{c,i*} - \sum_c \log p^{\text{end}}_{c,j*}, \quad (9)$$

Supervised Learning for Chunking Scorer. A binary variable y_c indicates whether the segment c contains an answer or not. Chunking scorer estimates the probability q_c that the segment contains an answer. Similarly, the chunking scorer network can be trained in a supervised manner by minimizing the cross-entropy loss, L_{cs}:

$$L_{\text{cs}} = -\sum_c y_c \log q_c - \sum_c (1 - y_c) \log(1 - q_c),$$
$$(10)$$

where the chunking score q_c is given in Eq. (7).

Reinforcement Learning for Chunking Policy. Since the selection of the stride actions is a sequential decision-making process, it is natural to train the chunking policy via reinforcement learning.

First of all, the accumulated reward for taking action a at state s is denoted as $R(s, a)$, which is derived in a recursive manner:

$$R(s, a) = q_c r_c + (1 - q_c) R(s', a'), \quad (11)$$

where q_c is the probability that segment c contains an answer as given in Eq. (7), and (s', a') denotes the next state-action pair. The value of r_c indicates the probability of the correct answer being extracted from the current segment c. The mathematical definition of r_c is given as:

$$r_c = \begin{cases} p^{\text{start}}_{c,i*} \cdot p^{\text{end}}_{c,j*}, & \text{if answer included,} \\ 0, & \text{else.} \end{cases} \quad (12)$$

The first term in Eq. (11) is the reward of the answer being correctly extracted from the current segment. The answer is included in the current segment c with probability q_c, and thus the first term is weighted by q_c in reward $R(s, a)$. The second term in Eq. (11) indicates that $R(s, a)$ also

Dataset	Train			Validation		
	Question #	Avg tokens #	Max token #	Question #	Avg tokens #	Max token #
CoQA	108,647	352	1,323	7,983	341	1,037
QuAC	83,568	516	2,310	7,354	576	2,146
TriviaQA (wiki)	61,888	2,622	5,839	7,993	2,630	6,690

Table 1: Statistics of the CoQA, QuAC and TriviaQA datasets. We report the number of sub-tokens generated by the BERT tokenizer.

relies on the accumulated reward $R(s', a')$ of the next state when the answer is not available in the current segment.

The chunking policy network can be trained by maximizing the expected accumulated reward (as shown in Eq. (13)) through the policy gradient algorithm (Williams, 1992; Sutton et al., 2000; Gong et al., 2019).

$$J = \mathbb{E}_{p^{\text{act}}(a\,|\,s)}[R(s, a)]. \tag{13}$$

To be consistent with the notations in answer extraction and chunking scorer modules, we denote the loss function of chunking policy as L_{cp}, which is the negative expected accumulated reward J in Eq. (13): $L_{\text{cp}} = -J$. Thus, the stochastic gradient of L_{cp} over a mini-batch of data $\mathcal{B}$ is given by:

$$\nabla L_{\text{cp}} = -\sum_{(s,a)\in\mathcal{B}} \nabla \log p^{\text{act}}(a\,|\,s)R(s, a), \tag{14}$$

where $p^{\text{act}}(a\,|\,s)$ is the chunking policy in Eq. (8).
Training procedure. The overall training loss L is an sum of all three losses: $L = L_{\text{ans}} + L_{\text{cs}} + L_{\text{cp}}$. In addition, we initialize the bottom representation layers with a pre-trained BERT model and initialize other model parameters randomly. We use the Adam optimizer with peak learning rate 3×10^{-5} and a linear warming-up schedule.

2.5 Testing

In the testing phase, the model starts from the beginning of the document as its first segment. Later on in state s, the model takes the best stride action a^* according to the chunking policy:

$$a^* = \underset{a \in A}{\arg\max} \, p^{act}(a\,|\,s) \tag{15}$$

After the stride action a^* is taken, a new segment is taken from the given document, and so on untill the maximum number of segments C is reached. Now for a document segment c, we score its candidate answer spanning from the i-th to the j-th token by $p_{i,j,c}^{\text{A}}$:

$$p_{i,j,c}^{\text{A}} = p_{c,i}^{\text{start}} \cdot p_{c,j}^{\text{end}} \cdot q_c. \tag{16}$$

The best answer span $(\bar{i}, \bar{j})$ across multiple segments can be obtained by selecting the one with the highest score $p_{i,j,c}^{\text{A}}$.

$$\bar{i}, \bar{j} = \underset{i \le j, 1 \le c \le C}{\arg\max} \, p_{i,j,c}^{\text{A}}, \tag{17}$$

where dynamic programming is used to find $(\bar{i}, \bar{j})$ efficiently in linear time.

3 Experiment

3.1 Datasets

We use three MRC datasets, CoQA (Reddy et al., 2018), QuAC (Choi et al., 2018) and TriviaQA (Joshi et al., 2017)) in our experiments.
(1) **CoQA**. Answers in the CoQA dataset can be abstractive texts written by annotators. It is reported that an extractive MRC approach can achieve an upper bound as high as 97.8% in F1 score (Yatskar, 2019). Therefore, We preprocess the CoQA training data and select a text span from the document as the extractive answer that achieves the highest F1 score compared with the given ground truth.
(2) **QuAC**. All the answers in the QuAC dataset are text spans, which are highlighted by annotators in the given document.
(3) **TriviaQA**. TriviaQA is a large-scale MRC dataset, containing data from Wikipedia and Web domains. We use its Wikipedia subset in this work. It is reported to be challenging in its variability between questions and documents as well as its requirement of cross-sentence reasoning. Documents in TriviaQA contain more than 2,000 words on average, which is suitable for evaluating the capability of a model to deal with long documents.

The dataset statistics are summarized in Table 1, including the data sizes, the average and maximum number of sub-tokens in documents.

3.2 Baselines and Evaluation Metric

Baselines. We have two strong baselines based on the pre-trained BERT, which has achieved state-of-the-art performance in a wide range of NLP tasks

Dataset	CoQA				QuAC			
Max sequence length	192	256	384	512	192	256	384	512
BERT-Large (Devlin et al., 2019)	72.8	76.2	81.0	81.4	34.5	50.6	56.7	61.5
Sent-Selector (with previous questions)	54.5	63.8	75.3	79.4	33.9	38.8	47.6	55.4
Sent-Selector (only current questions)	57.5	66.5	76.5	79.5	34.3	39.1	47.6	56.4
BERT-RCM								
- Gated recurrence (no RL chunking)	74.5	78.6	81.0	81.4	48.8	51.4	56.2	61.4
- Gated recurrence	**76.0**	79.2	**81.3**	**81.8**	51.6	55.2	59.9	**62.0**
- LSTM recurrence (no RL chunking)	74.1	78.5	81.0	81.3	49.2	51.5	56.4	61.6
- LSTM recurrence	75.4	**79.5**	**81.3**	**81.8**	**53.9**	**55.6**	**60.4**	61.8

Table 2: Comparison of F1 scores (%) achieved by different algorithms.

including machine reading comprehension.
(1) BERT-LARGE MODEL. It achieves competitive performance on extractive MRC tasks such as SQuAD (Rajpurkar et al., 2016, 2018). It adopts a simple sliding window chunking policy – moving to the next document segment with a fixed stride size from left to right. We also analyze the performance of the Large BERT model with different stride sizes in training and testing (see Section 4.1 for details). The best performance is obtained by setting stride size as 64 in CoQA and QuAC, and 128 in TriviaQA.

(2) SENTENCE SELECTOR. Given a question, the sentence selector chooses a subset of sentences that are likely to contain an answer. The selected sentences are then concatenated and fed to the BERT-Large model for answer extraction. For conversational datasets CoQA and QuAC, since a question is correlated with its previous questions within the same conversation, we apply the sentence selector to select sentences based on the current question alone or the concatenation of the previous questions and the current question. We only use the current question as the input to the sentence selector for TriviaQA, which does not involve any conversational history. The sentence selector we used in experiments is released by Htut et al. (2018).

Evaluation Metric. The main evaluation metric is macro-average word-level F1 score. We compare each prediction with the reference answer. Precision is defined by the percentage of predicted answer tokens that appear in the reference answer, and recall is the percentage of reference answer tokens captured in the prediction. F1 score is the harmonic mean of the precision and recall. When multiple reference answers are provided, the maximum F1 score is used for evaluation.

3.3 Results on CoQA and QuAC

We first perform experiments on two conversational MRC datasets, CoQA and QuAC.
Setting. We perform a set of experiments with different maximum sequence lengths of 192, 256, 384, and 512. Our model fixes the number of segments read from a document for each question. It generates 4, 3, 3, and 2 segments under the length limit of 192, 256, 384, and 512, respectively.

Considering that questions are highly correlated due to the existence of coreferential mentions across questions, we concatenate each question with as many of its previous questions as possible up to the length limit of 64 question tokens. The action space of the model strides is set as $[-16, 16, 32, 64, 128]$ for CoQA and $[-16, 32, 64, 128, 256]$ for QuAC considering that documents in CoQA documents are shorter than those in QuAC. The first segment always starts with the first token of the document, and the model will take stride action after the first segment.

Results. In Table 2, we present F1 scores achieved by our methods and the baselines. The performance of the BERT-Large model drops drastically as the maximum sequence length decreases. We see a drop of 8.6% in F1 score on the CoQA dataset and a drop of 27.0% on the QuAC dataset when the maximum input length decreases from 512 to 192.

Followed by the same BERT-Large reader, the sentence selector baseline that only considers the current question achieves better performance than the selector fed with the concatenation of the current question and its previous questions. The selector with the current question performs well in selecting sentences containing answers from documents. For 90.4% of questions in CoQA and 81.2% of questions in QuAC, the top-ranked 12 sentences in the documents can include complete

Dataset	CoQA				QuAC			
# of Doc Tokens	<=200	(200, 300]	(300, 400]	>400	<=300	(300, 450]	(450, 600]	>600
Percentage (%)	15.3	63.3	18.9	2.5	20.5	52.0	19.7	7.8
BERT-Large	81.0	81.9	81.8	67.2	66.2	62.8	62.2	38.7
BERT-RCM								
- Gated recurrence	81.1	82.1	82.3	74.5	66.1	62.6	63.6	**43.2**
- LSTM recurrence	81.1	82.0	82.3	**74.7**	66.4	62.6	63.0	41.3

Table 3: F1 Score (%) on documents with different numbers of tokens (max sequence length is 512).

answers. However, the selector does not improve upon BERT-Large despite its high precision in sentence selection. This might be because selected sentences do not provide sufficient contexts for a model to identify answers accurately.

Our model with recurrent chunking mechanisms BERT-RCM performs consistently better than both BERT-Large and BERT-Sent-Selector. On the CoQA dataset, BERT-RCM with gated recurrence improves upon the BERT-Large model by 3.2%, 3%, 0.3%, and 0.4% with maximum sequence length of 192, 256, 284, and 512, respectively. The improvement brought by LSTM recurrence and RL chunking is 2.6%, 3.3%, 0.3%, 0.4% on CoQA. As for the QuAC dataset, gated recurrence combined with RL chunking leads to improvements of 17.1%, 4.6%, 3.2%, 0.5%, and LSTM recurrence has gains of 19.4%, 5.0%, 3.7%, 0.3% under different maximum sequence lengths. On the two datasets, the gains of BERT-RCM over BERT-Large are statistically significant at $p = 0.05$ with both gated and LSTM recurrence. We notice that our model is less sensitive to the maximum sequence length, and LSTM recurrence has comparable performance to the gated recurrence.

The gain is more obvious with maximum sequence length (192, 256, 384), and relatively small under the length of 512. This is perhaps because most document lengths are smaller than 512 in CoQA and QuAC. Therefore, we report the performance of our proposed method on documents of different lengths in Table 3, where the maximum sequence length is set as 512. We observe that the gain is more obvious on longer documents. For documents with more than 400 words in the CoQA dataset, RL chunking with gated recurrence has an improvement of 7.3% over BERT-Large, and RL chunking with LSTM recurrence improves F1 score by 7.5%. As for QuAC, the improvement of gated recurrence with RL chunking is 4.5%, and the improvement of LSTM recurrence is 2.6%.

Ablation Analysis. We further study the effect of recurrence alone without RL chunking here. As shown in rows *BERT-Large* and *Gated recurrence (no RL chunking)* in Table 2, gated recurrence alone can improve F1 score by 2.4%, and LSTM recurrence leads to an improvement of 2.3% without RL chunking when the maximum sequence length is 256. However, we do not observe any improvement when the maximum sequence length is set to 384 or 512.

Algorithms	F1
BERT-Large	61.3
Sent-Selector	59.8
BERT-RCM	
- Gated recurrence	**62.9**
- LSTM recurrence	62.3

Table 4: F1 score (%) of different algorithms on the TriviaQA dataset.

3.4 Results on TriviaQA

We further evaluate the ability of our model in dealing with extremely long documents on the TriviaQA Wikipedia dataset.

Setting. We set the maximum sequence length as 512 for all models. The action space of our BERT-RCM model is set to $[-64, 128, 256, 512, 1024]$. The stride sizes are larger than those in CoQA and QuAC, since TriviaQA provides much longer documents. During training, the maximum number of segments our model can extract from a document is set to three in the TriviaQA dataset. Note that our model reads no more than $512 \cdot 3 = 1536$ tokens from these three segments, which are much fewer than the average document length.

Results. We filter a small number of questions whose answers cannot be extracted from documents and keep 7,251 questions from a total of 7,993 questions. In Table 4, we present the F1 scores of different algorithms. Compared with

Dataset	CoQA				QuAC			
BERT-Large (Devlin et al., 2019)	Prediction Stride Size				Prediction Stride Size			
Training Stride Size	16	32	64	128	16	32	64	128
16	80.8	80.9	80.8	80.7	60.6	60.7	60.7	60.8
32	81.1	81.1	81.1	81.1	60.7	60.7	60.9	61.0
64	**81.4**	**81.4**	**81.4**	81.3	61.0	61.0	**61.4**	**61.4**
128	81.0	81.1	81.1	81.1	60.8	60.8	60.8	61.2

Table 5: F1 score (%) of the BERT-Large model with different training/prediction stride sizes on the CoQA and QuAC datasets.

BERT-Large, the BERT-RCM model achieves 1.6% gain with gated recurrence and 1% gain with LSTM recurrence. Also, both BERT-RCM and BERT-Large models beat the Sent-Selector model.

4 Discussion

In this section, we will analyze the performance of the baseline BERT-Large model and our proposed recurrent chunking mechanisms.

4.1 Analysis of different Stride Sizes in BERT-Large

In Table 5, we give an analysis of how the performance varies with different stride sizes in BERT-Large model (the baseline) training and prediction. An interesting observation is that smaller stride size in prediction does not always improve the performance, sometimes even hurts as can be seen on the QuAC dataset. It suggests that BERT-Large performs badly on selecting good answers from multiple chunks. Smaller stride size in model training also leads to worse performance. A possible explanation is that smaller stride size would cause the significant distortion of training data distribution, since the longer question-document pairs produces more training samples than short ones.

4.2 Discussions of Recurrent Chunking Mechanisms

We now provide an insight into the recurrent mechanisms and chunking policy learned by our proposed model using quantitative analysis. For the clarity of our discussions, we use the following setting on the CoQA and QuAC datasets: the maximum chunk length is set to 256, and the stride size of BERT-Large model is 128.

Segment-Hit Rate. With the ability of chunking policy, BERT-RCM is expected to focus on those document segments that contain an answer.

To evaluate how well a model can capture good segments, we use *hit rate*, i.e., the percentage of segments that contain a complete answer among all extracted segments, as evaluation metric.

Hit rate	CoQA	QuAC
BERT-Large	54.0	34.1
BERT-RCM		
- Gated recurrence	73.1	44.9
- LSTM recurrence	79.7	42.8

Table 6: Comparison of BERT-Large and BERT-RCM on segment-hit rate (%).

As shown in Table 6, BERT-RCM significantly outperforms BERT-Large, which indicates that the learned chunking policy is more focused on informative segments.
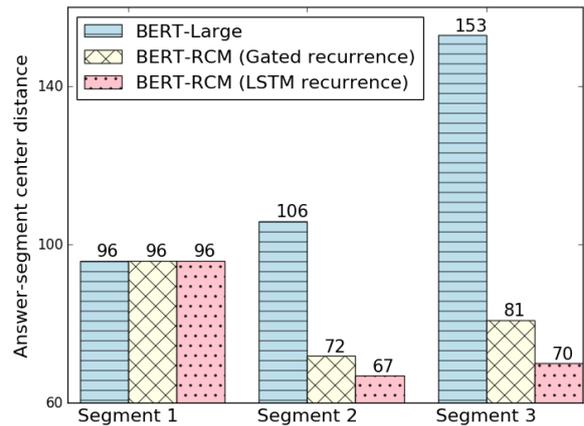

Figure 3: The answer-segment center distance.

Answer-Chunk Center Distance. As discussed in Fig. 1, the answer's position with respect to a document segment is important for answer prediction. When an answer is centered within the document segment, sufficient contexts on both sides help a model make better predictions. In

Previous question: What has helped us communicate?

Question: What kind of dishes does she bring?

Document: My doorbell rings. on the step, I find the elderly chinese lady, small and slight, holding the hand of a little boy. In her other hand, she holds a paper carrier bag. I know this lady. It is not her first visit. She is the boy 's grandmother , [Chunk 1] and her daughter bought the house next door last october . her daughter , nicole , speaks fluent english . but she is now in shanghai , and her parents are here with the little boy . nicole has obviously told her mother that i am having heart surgery soon , so her mother has decided i need more nutrients . i know what is inside the bag - - a the ##rm ##os with *hot soup* *and a stainless - steel container with rice , vegetables and either chicken , meat or shrimp , sometimes with a kind of pan* [Chunk 3] *##cake* . this has become an almost - daily practice . communication between us is somewhat affected by the fact that she doesn ' t speak english and all i can say in chinese is hello . once , she brought an ipad as well as the food . she pointed to the screen , which displayed a message from her daughter telling me that her mother wanted to know if the food was all right and whether it was too salty . i am not used to ipad ##s , so she indicated i should go with her to her house . then , she handed the ipad to her husband and almost immediately i found myself looking at nicole in shanghai and discussing [Chunk 2] her mother ' s cooking and salt intake . instantly , tears well ##ed in my eyes . " your mother just can ' t be bringing me meals like this all the time , " i insisted . " i can hardly do dishes in return . " " oh , no , lucy . " nicole said . " [......]

Figure 4: Example of generated document segments by BERT-RCM from a CoQA document.

Fig. 3, it presents the averaged center distances of the first three segments generated by BERT-Large and BERT-RCMs on the CoQA validation dataset. Since all models start from the beginning of a document in the first segment, their first answer-chunk center distances are the same: 96 tokens. But for the second and third segments generated by BERT-RCMs, the answer-chunk center distances are much smaller than BERT-Large.

In this section, we also illustrate the working flow of BERT-RCM with a case study.

Case Study. We show an example from a CoQA document in Figure 4 to illustrate the chunking mechanism of our BERT-RCM model with LSTM recurrence. The model starts with the beginning of the document as the first segment, where the answer span is close to its right boundary. The model moves forwards 128 tokens to include more right contexts and generates the second chunk. The stride size is a bit large since the answer is close to the left boundary of the second segment. The model then moves back to the left by 16 tokens and obtains its third segment. The chunking scorer assigns the three segments with the scores 0.24, 0.87, and 0.90, respectively. It suggests that the model considers the third segment as the most informative chunk in answer selection.

5 Related Work

There is a growing interest in MRC tasks that require the understanding of both questions and reference documents (Trischler et al., 2017; Rajpurkar et al., 2018; Saeidi et al., 2018; Choi et al., 2018; Reddy et al., 2018; Xu et al., 2019). Recent studies on pre-trained language models (Radford et al., 2018; Devlin et al., 2019; Baker et al., 2019; Yang et al., 2019b) have demonstrated their great success in fine-tuning on MRC tasks. However these pre-trained NLP models (e.g., BERT) only take as input a fixed-length text. Variants of BERT are proposed to process long documents in tasks such as text classification (Chalkidis et al., 2019). To deal with lengthy documents in machine reading comprehension tasks, some previous studies skip certain tokens (Yu et al., 2017; Seo et al., 2018) or select a set of sentences as input based on the given questions (Hewlett et al., 2017; Min et al., 2018; Lin et al., 2018). However, they mainly focus on tasks in which most of the answers to given questions are formed by a single informative sentence. These previous approaches are less applicable to deal with those complicated questions that demand cross-sentences reasoning or have much lexical variability from their lengthy documents.

6 Conclusion

In this paper, we propose a chunking policy network for machine reading comprehension, which enables a model learn to chunk lengthy documents in a more flexible way via reinforcement learning. We also add a recurrent mechanism to allow the information to flow across segments so that the model could have knowledge beyond the current segment when selecting answers. We have performed extensive experiments on three public datasets of machine reading comprehension: CoQA, QuAC, and TriviaQA. Our approach outperforms benchmark models across different datasets.

Acknowledgments

We would like to thank the anonymous reviewers for their constructive comments and suggestions.

References

Henrietta Baker, Matthew R. Hallowell, and Antoine J.-P. Tixier. 2019. Automatically learning construction injury precursors from text. *arXiv preprint*, cs.CL/1907.11769v3.

Ilias Chalkidis, Ion Androutsopoulos, and Nikolaos Aletras. 2019. Neural legal judgment prediction in english. In *Proceedings of the ACL*, pages 4317–4323, Florence, Italy.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen-tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. 2018. QuAC: Question answering in context. In *Proceedings of the EMNLP*, pages 2174–2184, Brussels, Belgium.

Christopher Clark and Matt Gardner. 2017. Simple and effective multi-paragraph reading comprehension. *arXiv preprint*, cs.CL/1710.10723v2.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proeedings of the NAACL-HLT*, pages 4171–4186, Minneapolis, MN.

Hongyu Gong, Suma Bhat, Lingfei Wu, JinJun Xiong, and Wen-mei Hwu. 2019. Reinforcement learning based text style transfer without parallel training corpus. In *Proceedings of the NAACL-HLT*, pages 3168–3180, Minneapolis, MN.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Proceedings of the NeurIPS*, pages 1693–1701, Montreal, Canada.

Daniel Hewlett, Llion Jones, Alexandre Lacoste, et al. 2017. Accurate supervised and semi-supervised machine reading for long documents. In *Proceedings of the EMNLP*, pages 2011–2020, Copenhagen, Denmark.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The goldilocks principle: Reading children's books with explicit memory representations. In *Proceedings of the ICLR*, Caribe Hilton, Puerto Rico.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Phu Mon Htut, Samuel R Bowman, and Kyunghyun Cho. 2018. Training a ranking function for open-domain question answering. In *Proceedings of the NAACL-HLT*, pages 120–127, New Orleans, LA.

Minghao Hu, Yuxing Peng, Zhen Huang, Nan Yang, Ming Zhou, et al. 2019. Read+ verify: Machine reading comprehension with unanswerable questions. In *Proceedings of the AAAI*, pages 6529–6537, Honolulu, HI.

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 ACL*, pages 1601–1611, Vancouver, Canada.

Nitish Shirish Keskar, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Unifying question answering and text classification via span extraction. *arXiv preprint*, cs.CL/1904.09286v2.

Tomáš Kočiskỳ, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gáabor Melis, and Edward Grefenstette. 2018. The narrativeqa reading comprehension challenge. *TACL*, 6:317–328.

Yankai Lin, Haozhe Ji, Zhiyuan Liu, and Maosong Sun. 2018. Denoising distantly supervised open-domain question answering. In *Proceedings of the ACL*, pages 1736–1745, Melbourne, Australia.

Sewon Min, Victor Zhong, Richard Socher, and Caiming Xiong. 2018. Efficient and robust question answering from minimal context over documents. In *Proceedings of the ACL*, pages 1725–1735, Melbourne, Australia.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. In *Preprint*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. In *Proceedings of the ACL*, pages 784–789, Melbourne, Australia.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the EMNLP*, pages 2383–2392, Austin, TX.

Siva Reddy, Danqi Chen, and Christopher D Manning. 2018. CoQA: A conversational question answering challenge. *TACL*, 7:249–266.

Marzieh Saeidi, Max Bartolo, Patrick Lewis, Sameer Singh, Tim Rocktäschel, Mike Sheldon, Guillaume Bouchard, and Sebastian Riedel. 2018. Interpretation of natural language rules in conversational machine reading. In *Proceedings of the EMNLP*, pages 2087–2097, Brussels, Belgium.

Minjoon Seo, Sewon Min, Ali Farhadi, and Hannaneh Hajishirzi. 2018. Neural speed reading via Skim-RNN. In *Proceedings of the ICLR*, New Orleans, LA.

Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of the NeurIPS*, pages 1057–1063, Denver, CO.

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 RepL4NLP*, pages 191–200, Vancouver, Canada.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Hu Xu, Bing Liu, Lei Shu, and Philip S. Yu. 2019. Review conversational reading comprehension. *arXiv preprint*, cs.CL/1902.00821v2.

Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2019a. End-to-end open-domain question answering with bertserini. In *Proceedings of the NAACL-HLT*, pages 72–77, Minneapolis, MN.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019b. XLNet: Generalized autoregressive pretraining for language understanding. In *Proceedings of the NeurIPS*, pages 5754–5764, Vancouver, Canada.

Mark Yatskar. 2019. A qualitative comparison of CoQA, SQuAD 2.0 and QuAC. In *Proceedings of the NAACL-HLT*, pages 2318–2323, Minneapolis, MN.

Adams Wei Yu, Hongrae Lee, and Quoc Le. 2017. Learning to skim text. In *Proceedings of the ACL*, pages 1880–1890, Vancouver, Canada.

Sheng Zhang, Xiaodong Liu, Jingjing Liu, Jianfeng Gao, Kevin Duh, and Benjamin Van Durme. 2018. Record: Bridging the gap between human and machine commonsense reading comprehension. *arXiv preprint*, cs.CL/1810.12885v1.