Recurrent Chunking Mechanisms for Conversational Machine Reading Comprehension

Hongyu Gong*
University of Illinois at Urbana-Champaign, USA
hgong6@illinois.edu

Yelong Shen, Dian Yu, Jianshu Chen, Dong Yu
Tencent AI Lab, Bellevu, WA
{yelongshen,yudian,jianshuchen,dyu}@tencent.com

Abstract

In this paper, we focus on the conversational machine reading comprehension (MRC) problem, where the input to a model could be a lengthy document and a series of interconnected questions. To deal with long inputs, previous approaches usually chunk them into *equally-spaced* segments and predict answers based on each chunk independently without considering the information from other chunks. As a result, they may form chunks that fail to cover complete answers or have insufficient contexts around the correct answer required for question answering. Moreover, they are less capable of answering questions that need cross-chunk information.

We propose to let a model learn to chunk in a more flexible way via reinforcement learning: a model can decide the next chunk that it wants to process in either direction. We also apply recurrent mechanisms to allow information to be transferred between chunks. Experiments on two conversational MRC tasks – CoQA and QuAC – demonstrate the effectiveness of our recurrent chunking mechanisms: we can obtain chunks 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

Recently, we have seen a surge of interest towards extractive and abstractive machine reading comprehension (MRC) tasks [3, 5, 17, 23, 29, 10]: given a document and questions, answers can be spans from the document or free-form texts. In this paper, we focus on conversational MRC tasks such as CoQA [1] and QuAC [18], in which a series of interconnected (instead of independent) questions are designed based on the given documents. In consequence, these questions together with their answers form conversations.

There is also a growing trend of building MRC readers [8, 25, 26, 9] based on pre-trained language models such as GPT [15] and BERT [2]. Since these models only allow a fixed-length (e.g., 512) input, it is often the case that an input sequence exceeds the length constraint. This is especially the case for conversational MRC tasks as we may need to combine previous questions to answer the current question, and these tasks have relatively long documents (e.g., 401 tokens in QuAC v.s 117 tokens in SQuAD [17]). Therefore, dealing with lengthy inputs is an important challenge in conversational MRC tasks.

* *Most of the work was done during internship at Tencent AI Lab, Bellevue.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

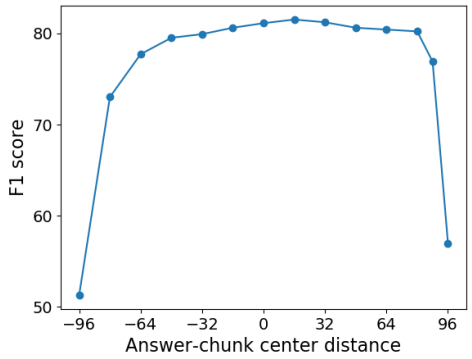

Figure 1: The influence of the distance between the center of the answer span and the center of the chunk. The test performance (in F1 scores) is evaluated on the CoQA dataset using a BERT-based reader.

There are two major limitations in previous MRC readers when dealing with lengthy documents. First, they typically chunk a lengthy document into multiple equally-spaced segments by moving the model from the current chunk to the next one using a pre-determined stride. This chunking strategy can be problematic since it may result in incomplete answers. Moreover, we also observe that a model tends to make better predictions when a chunk provides richer contexts around the ground truth answer.

To confirm our observation, we first fine-tune a BERT-based reader on CoQA and then evaluate the obtained model on chunks with different center distances from the answer span (Figure 1). 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 chunk center moves away from the answer center and the chunk contains fewer relevant contexts. When the distance reaches 96, more than half of the predicted spans are incomplete. Therefore, we argue that a good chunking policy should generate chunks that not only fully cover the correct answer span but also provide sufficient contexts around the correct answer.

Second, besides the fixed-length chunking, most existing methods predict answers by only reading the local information within each chunk. However, in practice, information from different chunks is essential to answer questions that involve global contextual information such as coreferential name mentions within a document.

We propose to let a machine reader learn how to chunk intelligently via reinforcement learning. Instead of using a fixed stride in one direction, we allow the model to decide the next chunk to be processed in either direction. Henceforth, the model is capable of making better predictions based on the carefully selected chunks (Section 2.3). We also apply recurrent mechanisms to allow the information to flow between chunks. As a result, the model can have access to information beyond the current chunk (Section 2.2).

In our experiments, we evaluate our model[2] on two conversational machine reading comprehension datasets: CoQA and QuAC. Experimental results demonstrate that our method can generate chunks that are more likely to cover complete answer spans and provide richer contextual information around the ground truth answers. The proposed chunking mechanisms lead to performance gains on the benchmark datasets.

2 Method

2.1 Baseline Model

As shown in Figure 2, our model consists of a pre-trained model, an answer extractor, a policy network, and a chunk scorer. We use BERT as the pre-trained model that generates representations for document chunks and questions [2]. Following the input format of BERT, each input sequence

[2]The code is available at https://github.com/HongyuGong/ConversationalQA.

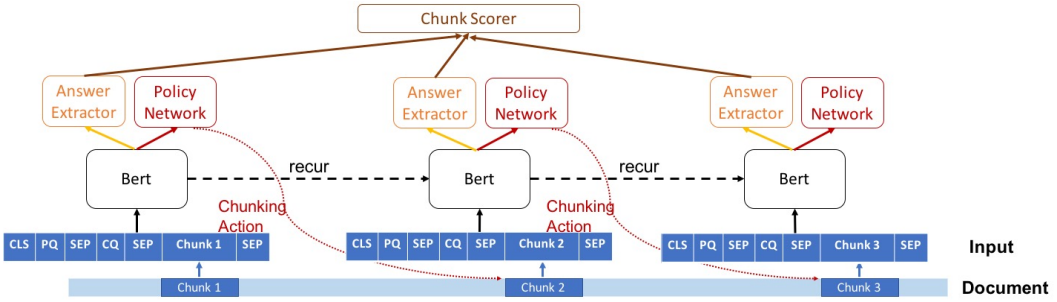

Figure 2: view: BERT generates representations for each input chunk, and recurrence accumulates information over chunks. Based on these representations, the answer extractor extracts answers from the current chunk, and the policy network takes chunking action and moves to the next chunk. Chunk scorer scores each chunk by giving its likelihood of containing an answer and selects answers among predictions from multiple chunks.

starts with "CLS" token, which is followed by previous questions (PQ), the current question (CQ), and the document chunk. The three parts are separated by "SEP" tokens.

The maximum input length in BERT is restricted to be 512. However, the document length of 14.1% of test questions in QuAC already exceeds this input length constraint. A popular approach is to segment the long document into multiple chunks [18, 1].

Answer Extractor. Following previous work on extractive machine reading comprehension, we predict the start and the end positions of the answer span in the given document. BERT first generates a vector representation $\mathbf{h_{c,i}}$ for each i-th token in the c-th chunk. Given $\mathbf{h_{c,i}}$, the model scores each token by giving the likelihood of it being the start token of the answer span:

$$l^s_{c,i} = \mathbf{w_s^T h_{c,i}}, \tag{1}$$

where $\mathbf{w_s}$ is model parameter. The probability that the answer starts at the i-th token is computed by applying the softmax operation to $l^s_{c,i}$:

$$p^s_{c,i} = \text{softmax}(l^s_{c,i}) \tag{2}$$

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

$$l^e_{c,i} = \mathbf{w_e^T h_{c,i}}, \tag{3}$$

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

2.2 Recurrent Mechanisms

Given a question, existing BERT-based models only access the local information from each chunk and predict answers independently. We argue that document-level information is essential for answer prediction, especially in conversational MRC tasks. In this work, we apply recurrent mechanisms to allow information flow between chunks, and thus a model can have access to information beyond the current chunk.

Suppose that the chunk-level representation of chunk c is v_c, which is generated without knowledge from other chunks. As mentioned earlier, "CLS" is the first token of the input sequence to the BERT model, which has been used to capture the information of the whole chunk [2]. We thus use the vector of the "CLS" token as the chunk representation $\mathbf{v}_c$ in this work.

The recurrence is applied to the chunk representation $\mathbf{v}_c$ so that the information of previous chunks can be transferred afterwards. The enriched chunk representation $\tilde{\mathbf{v}}_c$ is defined as

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

where $f(\cdot)$ is the recurrence function. We consider two recurrent mechanisms here: linear recurrence and Long Short Term Memory (LSTM) [6] recurrence. Linear recurrence is simply a weighted sum of its inputs:

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

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

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

Chunk Scorer. We extract a candidate answer from each chunk. Given multiple chunks from a lengthy document, we need to decide the answer among all the candidate answers from multiple chunks. Besides the answer prediction within each chunk, the model also assigns a confidence score to each chunk. This score predicts the probability q_c that chunk c contains an answer.

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

where $\mathbf{W}_c$ and $\mathbf{b}_c$ are model parameters, and $\sigma(\cdot)$ is sigmoid function.

For a candidate answer span that starts with the i-th token and ends with the j-th token in chunk c, its probability $p^{\text{A}}_{i,j,c}$ is based on both the estimation within the chunk and the confidence score q_c of this chunk:

$$p^{\text{A}}_{i,j,c} = p^s_{c,i} \cdot p^e_{c,j} \cdot q_c. \tag{8}$$

Candidate answer spans selected from different chunks are ranked in terms of the predicted answer probability, and the one with the highest probability is selected as the answer by the model.

2.3 Learning to Chunk

In this section, we introduce how we learn to chunk a document. Existing approaches usually generate chunks from left to right of a document [2], and models move from the current chunk to the next chunk with a fixed stride size m (in tokens). More specifically, if the current chunk starts with the i-th token in the document, the next chunk will start at the $(i + m)$-th token. However, one problem with these kinds of methods is that a document may be inappropriately segmented. For instance, an answer span may cross the chunk boundary after segmentation. Also, the segmentation method might generate a chunk where the answer span is close to the chunk boundary. As such, a model may fail to make good predictions due to a lack of sufficient contexts around the answer.

We propose to allow the baseline model (Section 2.1) to learn to chunk. Instead of fixing its stride to be a given size and be in one direction, we enable the model to decide the next chunk that it wants to process and allow it to move back and forth. Henceforth, the model is capable of making better predictions based on the carefully selected chunks. Learning to chunk is done with reinforcement learning (RL).

To formulate the learning-to-chunk as an RL problem, we define the state and the action space as follows. The **state** s is defined to be the chunks that the model has processed up to the current time, i.e., $s = \{1, 2, \ldots, c\}$. The **action** a is the size and direction of the stride for moving to the next chunk. We define the **action space** A of chunking as a set of strides. The negative stride allows the model to look backwards at the already seen chunks, and the positive stride allows it to move forward to process the unseen chunks.

Policy Network. We use a neural network to model the policy for selecting the actions. Specifically, the policy network uses a feedforward neural network to model the probability distribution $\mathbf{p}^{\text{act}}(a \mid s)$ over all stride actions given the current state s encoded in the chunk representation $\tilde{\mathbf{v}}_c$, which is enriched with document-level information as described in Eq. (4):

$$\mathbf{p}^{\text{act}}(a \mid s) = \text{softmax}(\mathbf{W}_a \tilde{\mathbf{v}}_c + \mathbf{b}_a), \tag{9}$$

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

During training, the action is randomly sampled with the probability $\mathbf{p}^{\text{act}}(a \mid s)$. This allows a better exploration-exploitation tradeoff [22] and increases the diversity of chunks.

Rewards. The policy network sequentially selects actions to chunk a document and receives a reward that reflects the quality of model's final answer prediction. Note that this is a delayed reward problem since we do not know whether actions are good or bad until the end after the model finishes reading

all chunks. We first define the rating of a chunk in the following manner. Suppose a chunk c contains the ground truth answer, which starts at the i^*-th token and ends at the j^*-th token. Then the rating of the chunk, denoted as r_c, is equal to $p^s_{c,i^*} \cdot p^e_{c,j^*}$. Otherwise, it is zero. Formally,

$$r_c = \begin{cases} p^s_{c,i^*} \cdot p^e_{c,j^*}, & \text{answer included,} \\ 0, & \text{else.} \end{cases} \tag{10}$$

Suppose that each chunking action a results in a new chunk c. A sequence of actions generates a sequence of chunks for a document, from which we can compute the reward of each action using dynamic programming. Specifically, recall from Eq. (7) that q_c is the probability that chunk c contains an answer. We define the reward $R(s, a)$ for taking action a in state s in a recursive manner:

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

where (s', a') denotes next state-action pair.

Ideally, the chunking actions are taken so that the model can maximize the rewards of predicted answers. Mathematically, we define the expected reward as J_θ, and θ are model parameters related to the chunking decision.

$$J_\theta = \mathbb{E}_{p^{\text{act}}_\theta(a \mid s)}[R(s, a)]. \tag{12}$$

The action probability $p^{\text{act}}_\theta(a \mid s)$ is the chunking policy learned by the model. This policy guides the model to generate chunks on which the model can make good predictions of answer spans.

2.4 Training

As shown in Figure 2, our model comprises of a policy network that chunks a given document, an answer extractor that extracts a candidate answer from the current chunk, and a chunk scorer that selects the answer among chunks. The training loss of our model consists of three parts — answer loss L_{ans}, chunking policy loss L_{cp}, and chunk scoring loss L_{cs} — to take care of training all these modules. We now discuss these losses separately.

Answer Loss. For training instances, the ground truth answer of a given question is marked in the associated document. We already know the start and end tokens of the answer in a chunk, and thus the answer extractor can be directly trained to identify answers within the chunk via supervised learning. As has been described, the answer extractor predicts the probability distribution of answer start/end over all tokens in a chunk. Suppose that the i^*-th and j^*-th token in the chunk are answer start and end, respectively. We aim to optimize the probability of the two tokens and thus use cross-entropy loss as the answer loss L_{ans}:

$$L_{\text{ans}} = -\sum_{c,i} y^s_i \log p^s_{c,i} - \sum_{c,j} y^e_j \log p^e_{c,j},$$

where $y^s_{c,i}$ is a binary label indicating whether the i-th token in chunk c is answer start, and $p^s_{c,i}$ is its predicted start probability as shown in Eq. (2). Similarly, $y^e_{c,j}$ and $p^e_{c,j}$ is the label and prediction of the j-th end token in chunk c, repectively.

Chunking Policy Loss. Chunking Policy network, which is trained with the action reward via reinforcement learning, enables more flexible document chunking. The chunking policy loss L_{cp} is the negative of expected reward J_θ in Eq. (12), i.e., $L_{\text{cp}} = -J_\theta$.

Chunk Scoring Loss. It is known whether a given chunk contains an answer in the training stage. Again we apply cross entropy loss to optimize the chunk scoring. The chunking scoing loss L_{cs} is

$$L_{\text{cs}} = -\sum_{c} y_c \log q_c, \tag{13}$$

where y_c is a binary label indicating whether chunk c contains an answer, and q_c is model's predicted probability as shown in Eq. (7).

In summary, the training loss L of our model is $L = L_{\text{ans}} + L_{\text{cp}} + L_{\text{cs}}$. Since losses L_{ans} and L_{cs} are differentiable, the model parameters can be simply updated with their gradients ∇L_{ans} and ∇L_{cs}.

As for the non-differentiable chunking policy loss L_{cp}, we optimize it by applying the idea from REINFORCE algorithm [24], which uses a sample approximation to its gradient:

$$\nabla_\theta L_{\text{ck}} = -\sum_t \mathbb{E}[\nabla_\theta \log p_\theta^{\text{act}}(a_t \mid s_t) R(s_t, a_t)],$$

where $p_\theta^{\text{act}}(a_t \mid s_t)$ is given in Eq. (9).

2.5 Testing

During testing, the model starts from the beginning of the document in its first chunk. Given the current chunk c, the model uses the chunk representation $\tilde{\mathbf{v}}_\mathbf{c}$ to select the optimal stride action a^* with the policy network, where

$$a^* =_{a \in A} \text{softmax}(\mathbf{W_a}\tilde{\mathbf{v}}_\mathbf{c} + \mathbf{b_a}). \tag{14}$$

Stride action a^* is taken to generate the next chunk c'. Similarly, a set of chunks C are sequentially extracted from a document. We score an answer span spanning from the i-th to the j-th token in chunk c with model's estimated likelihood $p^{\text{A}}_{i,j,c}$ shown in Eq. (8). The best answer span $(\bar{i}, \bar{j})$ from chunk $\bar{c}$ is the one with the highest likelihood:

$$\bar{i}, \bar{j}, \bar{c} =_{i \le j, c \in C} p^{\text{A}}_{i,j,c}. \tag{15}$$

3 Experiment

Dataset	Train			Validation		
	Question #	Avg tokens #	Max token #	Question #	Avg tokens #	Max token #
CoQA	108,647	352	1323	7,983	341	1037
QuAC	83,568	516	2310	7,354	576	2146

Table 1: Statistics of CoQA and QuAC data. We consider the number of sub-tokens generated by BERT tokenizer.

Datasets. We use two conversational machine reading comprehension datasets (i.e., CoQA [18] and QuAC [1]) in our experiments. A background document is provided for each conversation, which involves a set of questions to be answered based on the given document sequentially.

(1) Conversational Question Answering (CoQA). Answers in the CoQA dataset can be free-form 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 [27]. 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 answer that can be abstractive.

(2) Question Answering in Context (QuAC). All the answers in the QuAC dataset are text spans highlighted by annotators in the given document.

The dataset statistics is summarized in Table 1, including the data sizes and the number of sub-tokens in documents. Details of data processing are available in the supplementary material.

Baselines. We have two baselines based on BERT, which have achieved state-of-the-art performance in a wide range of natural language understanding tasks including machine reading comprehension.

(1) BASIC BERT MODEL. It achieves competitive performance on extractive machine reading comprehension tasks such as SQuAD [17, 16]. It adopts a simple chunking policy – moving to the next document chunk with a fixed stride size. In the experiments, we select the stride size to be 64 in CoQA and QuAC from $(16, 32, 64, 128)$, which gives the best performance on both the two datasets, please see appendix A.2 for details.

(2) SENTENCE SELECTOR. We use a state-of-the-art sentence selector for MRC as our baseline [7]. Given a question, the selector chooses a subset of sentences that are likely to contain an answer. The selected sentence are then fed to the BERT-based baseline for answer extraction. Since a question is correlated with its previous questions within a conversation, we apply the sentence selector to select sentences based on the current question alone or the concatenation of previous and the current questions.

See the results of the two baseline implementations in the rows *Sent selector (with previous questions* and *Sent selector (only current questions)* in Table 2, respectively. Following the setting of previous

work [7], we train the selector with the margin ranking loss. The top-ranked sentences are selected under the length constraint and concatenated as the new document.

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 the harmonic mean of the precision and recall. When multiple reference answers are provided, the maximum F1 score is used for prediction.

Setting. We perform a set of experiments with different maximum sequence lengths of 192, 256, 384, and 512. Our system and the two baseline systems are built upon a pre-trained model BERT. We use the 24-layer BERT model released by [2] and tune it in each system with a learning rate of $3e-5$. Our model fixes the number of chunks read from a document for each question. It generates 4, 3, 3, and 2 chunks 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 allowed by 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 chunk always starts with the first token of the document, and the model will take stride action after the first chunk.

Dataset	CoQA				QuAC			
Max sequence length	192	256	384	512	192	256	384	512
Basic BERT [2]	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
Linear recurrence o. RL chunking	74.5	78.6	81.0	81.4	48.8	51.4	56.2	61.4
Linear recurrence w. RL chunking	**76.0**	79.2	**81.3**	**81.8**	51.6	55.2	59.9	**62.0**
LSTM recurrence o. RL chunking	74.1	78.5	81.0	81.3	49.2	51.5	56.4	61.6
LSTM recurrence w. RL chunking	75.4	**79.5**	**81.3**	**81.8**	**53.9**	**55.6**	**60.4**	61.8

Table 2: F1 score ($\%$) of different algorithms on conversational reading comprehension datasets.

Results. We experiment with a set of maximum sequence lengths to evaluate the impact of the input length on models' performance in machine reading comprehension. Table 2 presents F1 scores achieved by our methods and the baselines.

The performance of basic BERT model drops drastically as the chunk 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 chunk size decreases from 512 to 192, and more chunks are generated from documents.

Followed by the same BERT-based reader, the sentence selector baseline that only considers the current question achieves better performance than the selector fed with the combination of the current question and its previous questions. The selector with the current question performs well in selecting sentences containing answers from documents. Around 90.4% of questions in CoQA and 81.2% of questions in QuAC, the top-ranked 12 sentences in the document can include at least one complete answer. However, the selector does not improve upon basic BERT despite its high precision in sentence selection. This might be because selected sentences do not provide sufficient contexts for a reader to identify answers accurately.

Our model with recurrent chunking mechanisms performs consistently better than both basic BERT and sentence selector. On CoQA data, our chunking model with linear recurrence improves upon the basic BERT model by 3.2%, 3%, 0.3%, and 0.4% for chunk 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 data, linear 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 chunk lengths. We notice that our model is less sensitive to the chunk length, and LSTM recurrence has comparable performance to the Linear recurrence.

Our model is shown to enhance the performance on both datasets, and the gain is significant when more chunks are generated at a smaller chunk length. We note that the gain is relatively small on CoQA data under the length of 384 and 512. This is because the average document length of CoQA

data is 352. In that case, a single chunk could cover the whole document for most questions. Similarly, the gain is small on QuAC data at the input length of 512.

We show the performance of our proposed method on different length of documents in Table 3. The maximum sequence length is set as 512 for both CoQA and QuAC dataset. Although our gain is small over all documents, we note that the gain is more obvious on longer documents. For documents with more than 400 words in CoQA dataset, RL chunking with linear recurrence has an improvement of 7.3% over basic Bert, and RL chunking with LSTM recurrence enhances F1 score by 7.5%. As for QuAC data, the improvement of linear recurrence with RL chunking is 4.5%, and the improvement of LSTM recurrence is 2.6%.

Dataset	CoQA				QuAC			
Document len	<=200	(200, 300]	(300, 400]	>400	<=300	(300, 450]	(450, 600]	>=600
Query percentage (%)	15.3	63.3	18.9	2.5	20.5	52.0	19.7	7.8
Basic Bert	81.0	81.9	81.8	67.2	66.2	62.8	62.2	38.7
Linear reccurrence w. RL chunking	81.1	82.1	82.3	74.5	66.1	62.6	63.6	**43.2**
LSTM recurrence w. RL chunking	81.1	82.0	82.3	**74.7**	66.4	62.6	63.0	41.3

Table 3: F1 Score on Documents of Different Lengths.

Ablation Analysis. We have shown the performance gains brought by the combination of recurrent mechanisms and chunking policy. Here we evaluate the improvements brought by the chunking policy alone. By comparing rows *LSTM recurrence w. RL chunking* and *LSTM recurrence o. RL chunking* in Table 2, we can find that RL chunking alone improves F1 score by 1.3%, 1.0%, 0.3%, and 0.5% under the chunk length constraint of 192, 256, 384, and 512 respectively on the CoQA dataset. Its improvements are 4.7%, 4.1%, 4.0%, and 0.2% on the QuAC dataset. The learned chunking policy brings non-trivial gains.

Recurrence. We study the effect of recurrence alone without RL chunking here. As shown in rows *basic BERT* and *Linear recurrence o. RL chunking* in Table 2, linear recurrence alone can improve F1 score by 2.4%, and LSTM recurrence gives an improvement of 2.3% without RL chunking when the maximum chunk length is 256. We provide more discussions and quantitative analysis on the learned chunking policy in Appendix A.4

4 Related Work

4.1 Conversational Reading Comprehension

There is a growing interest in conversational MRC tasks that require the understanding of conversations, which either contain a series of questions and answers [19, 1, 18, 25] or serve as documents [12, 14, 21]. In this paper, we focus on large-scale extractive and abstractive conversational MRC tasks with background documents: QuAC [1] and CoQA [18]. Very recently we see significant improvements in performance on conversational MRC tasks by leveraging additional extractive non-conversational datasets such as SQuAD [16] and NewsQA [23], which is beyond the scope of this paper.

4.2 Addressing Long Contexts in Machine Reading Comprehension Tasks

To deal with lengthy documents in machine reading comprehension tasks, some previous work skips certain tokens [28, 20] or selects a set of sentences as input based on the given questions [4, 13, 11]. However, they mainly focus on tasks in which most of the questions are formed by a single informative sentence.

5 Conclusion

We propose to let a model learn to chunk in a more flexible way via reinforcement learning: a model can decide the next chunk that it wants to process in either direction. We also apply recurrent mechanisms to allow information transfer between chunks. Experiments on two conversational machine reading comprehension tasks – CoQA and QuAC – demonstrate the effectiveness of our mechanisms. We can obtain chunks that are more likely to contain complete answers and at the same time cover sufficient contexts around the correct answer for better answer predictions.

References

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

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

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

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

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

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

[7] Phu Mon Htut, Samuel R Bowman, and Kyunghyun Cho. Training a ranking function for open-domain question answering. In *Proceedings of the NAACL-HLT*, 2018.

[8] Minghao Hu, Yuxing Peng, Zhen Huang, Nan Yang, Ming Zhou, et al. Read+ verify: Machine reading comprehension with unanswerable questions. In *Proceedings of the AAAI*, 2018.

[9] Nitish Shirish Keskar, Bryan McCann, Caiming Xiong, and Richard Socher. Unifying question answering and text classification via span extraction. *arXiv preprint*, cs.CL/1904.09286v1, 2019.

[10] Tomáš Kočiskỳ, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gáabor Melis, and Edward Grefenstette. The narrativeqa reading comprehension challenge. *Transactions of the Association of Computational Linguistics*, 6:317–328, 2018.

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

[12] Kaixin Ma, Tomasz Jurczyk, and Jinho D Choi. Challenging reading comprehension on daily conversation: Passage completion on multiparty dialog. In *Proceedings of the NAACL-HLT*, pages 2039–2048, New Orleans, LA, 2018.

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

[14] Nikita Moghe, Siddhartha Arora, Suman Banerjee, and Mitesh M Khapra. Towards exploiting background knowledge for building conversation systems. In *Proceedings of the EMNLP*, pages 2322–2332, 2018.

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

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

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

[18] Siva Reddy, Danqi Chen, and Christopher D Manning. Coqa: A conversational question answering challenge. *TACL*, 2018.

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

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

[21] Kai Sun, Dian Yu, Jianshu Chen, Dong Yu, Yejin Choi, and Claire Cardie. DREAM: A challenge dataset and models for dialogue-based reading comprehension. *Transactions of the Association for Computational Linguistics*, 2019.

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

[23] Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. NewsQA: A machine comprehension dataset. In *Proceedings of the RepL4NLP*, pages 191–200, Vancouver, Canada, 2017.

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

[25] Hu Xu, Bing Liu, Lei Shu, and Philip S. Yu. Review conversational reading comprehension. cs.CL/1902.00821v1, 2019.

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

[27] Mark Yatskar. A qualitative comparison of coqa, squad 2.0 and quac. *arXiv preprint arXiv:1809.10735*, 2018.

[28] Adams Wei Yu, Hongrae Lee, and Quoc V Le. Learning to skim text. *arXiv preprint*, cs.CL/1704.06877v2, 2017.

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

A Appendix

A.1 Data Processing

We adopt data processing techniques [2] to deal with some questions whose answers do not exist in the document. We add a special token "unknown" to the end of each document. This token is the target answer span to unanswerable questions in both datasets. Since CoQA also contains questions with "yes/no" answers, we add a yes-no classifier to the answer extractor in the model, which predicts an answer to be "yes", "no" or a text span from a document. The classification loss is added to the answer loss L_{ans} at the training stage.

A.2 Analysis of Stride Size in Basic Bert

We give how the performance varied by choosing different stride sizes in basic bert model training and prediction, as in Table 4. An interesting observation is that smaller stride size in prediction doesn't always improve the performance, sometimes even hurts as shown in QuAC. It shows that Basic Bert performs badly on ensemble the answers from multiple chunks. Smaller stride size in model training also lead to the worse performance. A possible explanation is that smaller stride size would cause the significant distortion of training data distribution, the longer question-document pairs produces more training samples than short ones.

Dataset	CoQA				QuAC			
Basic-Bert [2]	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 4: F1 score (%) of Basic Bert with different training/prediction stride sizes on the CoQA and QuAC Datasets.

	Good set		Bad set		Overall	
	size	acc (%)	size	acc (%)	size	acc (%)
Linear recurrence	6154	90.8	1829	77.5	7983	87.8
LSTM recurrence	6285	90.0	1698	76.5	7983	87.1

Table 5: Chunk scoring accuracy on CoQA data.

A.3 Quantitative analysis of recurrence

We quantitatively evaluate how the recurrence alone influences the model performance. The maximum input length is set as 256, and the maximum query length is 64. We report the accuracy of chunk scoring for the model with recurrent mechanism and fixed chunking stride size of 128. The accuracy is defined to be the percentage of questions whose chunks assigned with the highest score by the chunk scorer contain correct answers. We put examples in the set of good examples if the model has better performance than basic BERT. Otherwise, the examples are put in the bad set.

As can be seen in Table 5, the chunking scoring is quite accurate with overall accuracy around 87% for both linear and LSTM recurrence. We note that there is a drop of around 10% in accuracy in chunking accuracy on bad examples compared with that on good examples. Inaccurate chunk selection partly accounts for the degraded performance of answer extraction in the set of bad examples.

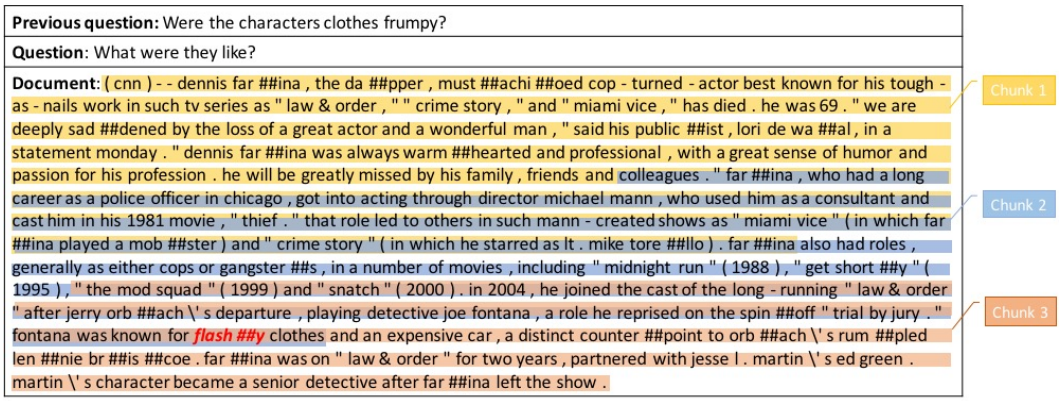

Figure 3: An example of chunks our model generated from a CoQA document.

Case study. We show another example from CoQA dataset. The question and its document is presented in Fig. 3. The ground truth answer "flashy" (tokenized as "flash ##y" by the BERT tokenizer) is colored in red. The chunks are generated by our chunking model with LSTM recurrence. The model finds the first chunk irrelevant to the given question, and moves with the large stride size of 128 tokens to the second chunk. The second chunk does not contain an answer, and the model again moves by 128 tokens to the right. The answer is now captured by the third chunk, and have sufficient contexts around it given it lies at the center of the chunk. The confidence assigned by the chunk scorer to the three chunks are 0.09, 0.09 and 0.91 respectively. Our model extracts an answer span from the third chunk.

A.4 Discussions of Recurrent Chunking Mechanism

In this section, we explore an insight into the recurrent mechanisms and chunking policy learned by our proposed model with both quantitative and qualitative analysis. For the clarity of our discussions, we use the following setting: the maximum chunk length is set as 256, and stride size of basic BERT model is 128.

Question	So I guess her wasn't interested in farmer roast?
Chunk	" A different roast every day. " Jack said. "Let me finish." Alan said, " On the fourth day a farmer died and **I didn't want to stay there for dinner.** "

Table 6: Example of a question and its chunk.

One benefit of recurrence is information transfers across chunks. We show an example of the question and the chunk with answer bolded in Table 6. While this short chunk contains the correct answer, it does not mention *"farmer roast"* that appears in the question, and it is unlikely to be selected without document-level information. Recurrence conveys the information that *"dinner"* in the current chunk refers to *"roast"* mentioned in previous chunks, and thus the model is able to select the right chunk for predictions. We also present quantitative analysis of the recurrent mechanisms in the supplementary material.

Hit rate (%)	CoQA	QuAC
Basic BERT	54.0	34.1
Linear recurrence w. chunking	73.1	44.9
LSTM recurrence w. chunking	79.7	42.8

Table 7: The hit rate of chunks in different models.

Chunk Hit Rate. With the ability to learn to chunk, our model is expected to focus on those chunks that contain an answer. To evaluate how well a model can capture good chunks, we use *hit rate*, i.e., the percentage of chunks that contain a complete answer as evaluation metric. As shown in Table 7, both chunking with linear recurrence and chunking with LSTM recurrence outperform the sequential chunking with fixed stride. It indicates that the learned chunking policy is more focused on informative chunks.

Chunk Position. As discussed in Fig. 1, a chunk's position with respect to the answer is critical for answer prediction in machine reading comprehension. When an answer is centered within the chunk, sufficient contexts on both sides help a model make better predictions.

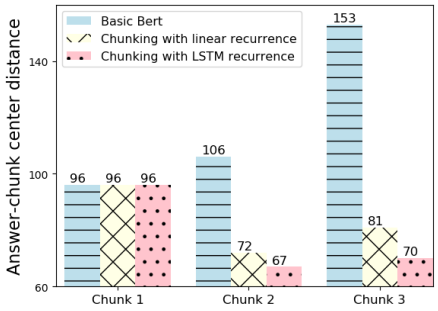

Figure 4: The answer-chunk center distance.

For chunks sequentially generated from a document, we measure the distance between the center of answer and the center of each chunk. Fig. 4 presents the center distances of three chunks generated by basic BERT, our chunking model with linear and LSTM recurrence on the CoQA data.

Since the three models start from the beginning of the document in the first chunk, their first chunks have the same distance of 96 tokens. Both chunk 2 and chunk 3 generated by our models move closer to the ground truth answer while the chunks of basic BERT move farther.

We conduct a case study using an example from CoQA and keep track of three chunks the model has generated. As shown in Fig. 3, the model starts with the beginning of the document in the first chunk,

where the answer 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 chunk. The model then moves back to the left by 16 tokens and obtains its third chunk. The confidence scores assigned by model to each chunk are 0.24, 0.87, and 0.90, respectively. As can be seen, the model moves along the document to include informative contexts in its chunk and make answer predictions with higher confidence.