MemoReader: Large-Scale Reading Comprehension through Neural Memory Controller

Seohyun Back[1,2] **Seunghak Yu**[1,*] **Sathish Indurthi**[1] **Jihie Kim**[1] **Jaegul Choo**[2,*]
[1] Samsung Research, Seoul, Korea
[2] Korea University, Seoul, Korea
{scv.back, seunghak.yu, s.indurthi, jihie.kim}@samsung.com
jchoo@korea.ac.kr

Abstract

Machine reading comprehension helps machines learn to utilize most of the human knowledge written in the form of text. Existing approaches made a significant progress comparable to human-level performance, but they are still limited in understanding, up to a few paragraphs, failing to properly comprehend lengthy document. In this paper, we propose a novel deep neural network architecture to handle a long-range dependency in RC tasks. In detail, our method has two novel aspects: (1) an advanced memory-augmented architecture and (2) an expanded gated recurrent unit with dense connections that mitigate potential information distortion occurring in the memory. Our proposed architecture is widely applicable to other models. We have performed extensive experiments with well-known benchmark datasets such as TriviaQA, QUASAR-T, and SQuAD. The experimental results demonstrate that the proposed method outperforms existing methods, especially for lengthy documents.

1 Introduction

Most of the human knowledge has been stored in the form of text. Reading comprehension (RC) to understand this knowledge is a major challenge that can vastly increase the range of knowledge available to the machines. Many neural network-based methods have been proposed, pushing performance close to a human level. Nonetheless, there still exists room to improve the performance especially in comprehending lengthy documents that involve complicated reasoning processes. We identify the main bottleneck as the lack of the long-term memory and its improper controlling mechanism.

Previously, several memory-augmenting methods have been proposed to solve the long-term dependency problem. For example, in relatively simple tasks such as bAbI tasks (Weston et al., 2015), Graves et al. (2014, 2016); Henaff et al. (2017) proposed methods that handle the external memory to address long-term dependency. Inspired by these approaches, we develop a customized memory controller along with an external memory augmentation (Graves et al., 2016) for complicated RC tasks. However, we found that the memory controller is susceptible to information distortion as neural networks become deeper, this distortion can hinder the performance.

To overcome this issue, we propose two novel strategies that improve the memory-handling capability while mitigating the information distortion. We extend the memory controller with a residual connection to alleviate the information distortion occurring in it. We also expand the gated recurrent unit (GRU) (Cho et al., 2014) with a dense connection that conveys enriched features to the next layer containing the original as well as the transformed information. We conducted extensive experiments through several benchmark datasets such as TriviaQA, QUASAR-T, and SQuAD. The results show that the proposed model outperforms all the published results. We also integrated the proposed memory controller and the expanded GRU cell block with other existing methods to ensure that our proposed components are widely applicable. The results show that our components consistently bring performance improvement across various state-of-the-art architectures.

The main contributions of this work include the following: (1) We propose an extended memory controller module for RC tasks. (2) We propose a densely connected encoder block with self attention to provide rich representation of given data, reducing information loss due to deep layers of the network. (3) We present the state-of-the-art results

[*] To whom correspondence should be addressed.

in lengthy-document RC tasks such as TriviaQA and QUASAR-T as well as relatively short document RC tasks such as SQuAD.

2 Proposed Method

This section presents two of our proposed components in detail, as depicted in Figure 1.

2.1 Memory Controller

Our first proposed component is an advanced external memory controller module for solving RC tasks. We modified the recently proposed memory controller (Graves et al., 2016) by using our new encoder block and layer-wise residual connections. These modifications enable the memory controller to reason over a lengthy document, leading to the overall performance improvement.

This layer takes input as a sequence of vector representations corresponding to individual tokens, $\mathbf{d}_t \in \mathbb{R}^l$, where l is the given vector dimension. For example, such input can be the output of the co-attention layer in Section 3. The operation of this layer is defined as

$$\mathbf{o}_t, \mathbf{i}_t = \text{Controller}(\mathbf{d}_t, M_{t-1}).$$

That is, at time step t, the controller generates an interface vector $\mathbf{i}_t$ for read and write operations and an output vector $\mathbf{o}_t$ based on the input vector $\mathbf{d}_t$ and the external memory content from the previous time step, $M_{t-1} \in \mathbb{R}^{p \times q}$, where p is the memory size and q is the vector dimension of each memory.

Through this controller, we encode an input $D = \{\mathbf{d}_t\}_{t=1}^n$ to $\{\mathbf{x}_t\}_{t=1}^n$ by using the encoder block, i.e.,

$$\{\mathbf{x}_t\}_{t=1}^n = \text{EncoderBlock}^x(D) \in \mathbb{R}^{n \times k},$$

where k is the output dimension of the encoder block. In general, this block is implemented as a recurrent unit, e.g., GRU (Cho et al., 2014). In our model, we replace it with our dense encoder block with self attention (DEBS), as will be discussed in Section 2.2.

To generate a memory-augmented vector $\mathbf{z}_t$, we concatenate $\mathbf{x}_t$ with the vectors read from the previous time step memory, M_{t-1}, i.e.,

$$\mathbf{z}_t = [\mathbf{x}_t; \mathbf{m}_{t-1}^1; \cdots; \mathbf{m}_{t-1}^s] \in \mathbb{R}^{k+sq},$$

where s represents the number of read heads in the memory interface. We then feed the vector $\mathbf{z}_t$ to the bi-directional GRU (BiGRU) layer and obtain the output vector $\mathbf{h}_t^m$ as

$$\mathbf{h}_t^m = \text{BiGRU}(\mathbf{z}_t, \mathbf{h}_{t-1}^m, \mathbf{h}_{t+1}^m) \in \mathbb{R}^{2l}.$$

Afterwards, we generate output vector $\mathbf{v}_t$ as the weighted sum of the BiGRU output and read vectors from the memory in the current step, i.e.,

$$\mathbf{v}_t = W_h \mathbf{h}_t^m + W_m [\mathbf{m}_t^1; \cdots; \mathbf{m}_t^s] \in \mathbb{R}^{2l}.$$

Finally, we add a residual connection between the input $\mathbf{d}_t$ and the output $\mathbf{v}_t$ to mitigate any possible information distortion that can occur while accessing the memory, resulting in a the output vector that can handle long-term dependency, i.e.,

$$\mathbf{o}_t = \text{ReLU}(W_v \mathbf{v}_t + \mathbf{d}_t) \in \mathbb{R}^l.$$

For further details on how the interface vector works, we refer the readers to Graves et al. (2016) as well as our supplemental material.

2.2 Dense Encoder Block with Self Attention

The second novel component we propose is a dense encoder block with self attention (DEBS), which further improves a GRU cell. Recently, Huang et al. (2017a) proposed that adding a connection between each layer to the other layers in convolution networks can help to properly convey the information across multiple layers. Inspired by this, we add such dense connections that concatenate the input to a particular layer to its output. We also add a self-attention module to this block, to properly address long-term dependency in a length document. In this manner, our encoder block maintains the necessary information not only along the vertical direction (across layers) through dense connections but also along the horizontal direction (across time steps) through self attention.

DEBS takes the input vector sequence with its length as n and transforms each vector to an l-dimensional vector $\mathbf{p}_t$ through the fully connected layer with ReLU as a nonlinear unit and generates a contextually encoded vector $\mathbf{r}_t$ as

$$\mathbf{r}_t = \text{BiGRU}(\mathbf{p}_t, \mathbf{r}_{t-1}, \mathbf{r}_{t+1}) \in \mathbb{R}^{2l}.$$

Then we concatenate each output vector $\mathbf{r}_t$ to the projected input $\mathbf{p}_t$ to obtain $\mathbf{g}_t = [\mathbf{r}_t; \mathbf{p}_t] \in \mathbb{R}^{3l}$ and pass it to the self-attention layer. The self-attention layer then calculates the similarity map $S^g \in \mathbb{R}^{n \times n}$ using the tri-linear function as

$$s_{ij}^g = \mathbf{w}_a \cdot \mathbf{g}_i + \mathbf{w}_b \cdot \mathbf{g}_j + \mathbf{w}_f \cdot (\mathbf{g}_i \odot \mathbf{g}_j),$$

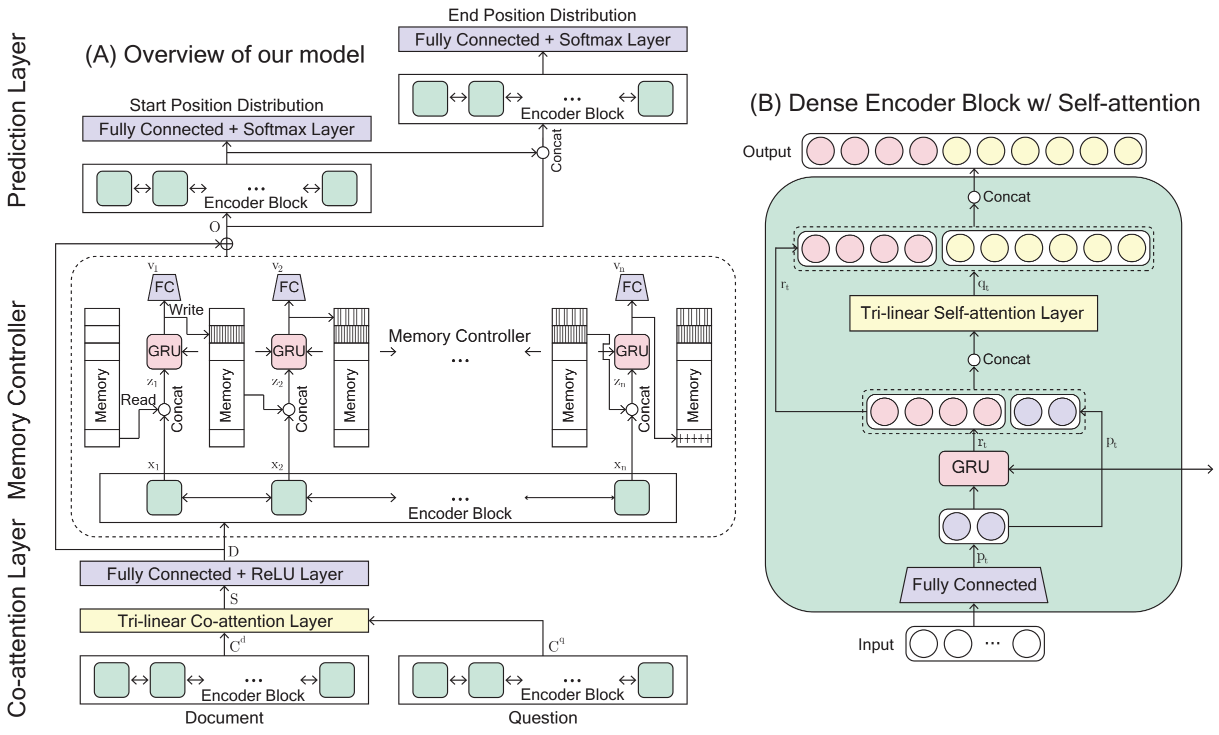

Prediction Layer
Memory Controller
Co-attention Layer
(A) Overview of our model
End Position Distribution
Fully Connected + Softmax Layer
Encoder Block
Concat
Start Position Distribution
Fully Connected + Softmax Layer
Encoder Block
O
FC
Write
GRU
Read
Memory
Concat
Memory Controller
Encoder Block
D
Fully Connected + ReLU Layer
S
Tri-linear Co-attention Layer
C^d
C^q
Encoder Block
Document
Encoder Block
Question
(B) Dense Encoder Block w/ Self-attention
Output
Concat
Tri-linear Self-attention Layer
Concat
GRU
Fully Connected
Input

Figure 1: Overview of our model (A) and dense encoder block with self attention (B).

where $i, j = 1, \ldots, n$. Finally, the self-attended representation $Q = \{\mathbf{q}_t\}_{t=1}^{n}$ is obtained by performing column-wise softmax on S^g to get the attention matrix A^g, which is further multiplied with $G = \{\mathbf{g}_t\}_{t=1}^{n}$, i.e.,

$$Q = A^g G \in \mathbb{R}^{n \times 3l}.$$

The final output is obtained as the concatenation of outputs from the recurrent layer (BiGRU) and the self-attention layer, i.e., $[\mathbf{r}_t; \mathbf{q}_t] \in \mathbb{R}^{5l}$.

3 Reading Comprehension Model with Proposed Components

We apply the proposed components to our model for RC tasks. As depicted in Figure 1, the model consists of three major layers: the co-attention layer, the memory controller, and the prediction layer. Given the embeddings of a question and a document, the co-attention layer generates query-aware contextual representations. The memory controller further refines these contextual representations using an external memory. Based on such representations, the prediction layer determines the start and the end token indices that form the answer span. In addition, we replace all the encoder block with DEBS in the three major layers.

Embedding. We incorporate both word- and character-level embedding methods to obtain the vector representation of each word in the input data. For word-level embedding $\mathbf{e}_w$, we utilize pre-trained, 300-dimensional embedding vectors from GloVe 6B (Pennington et al., 2014). The character-level word embedding $\mathbf{e}_c$ is obtained as a 100-dimensional vector by first applying a convolution layer with 100 filters to a sequence of 20-dimensional character embeddings learned during training and by further applying global max-pooling over the entire character-level sequence. Then we obtain the embedding vector of a given word token, $\mathbf{e}$, by concatenating these word- and character-level embeddings, i.e., $\mathbf{e} = [\mathbf{e}_w; \mathbf{e}_c] \in \mathbb{R}^{400}$.

Finally, we obtain the two sets of embedding vectors of question and document token sequences as $E^q = \{\mathbf{e}_u^q\}_{u=1}^{m} \in \mathbb{R}^{m \times 400}$ and $E^d = \{\mathbf{e}_t^d\}_{t=1}^{n} \in \mathbb{R}^{n \times 400}$, where m and n represent the sequence length of a question and a document, respectively.

Co-attention layer. Given E^q and E^d, we feed each of them into the encoder block and obtain their contextual representations as

$$C^q = \{\mathbf{c}_u^q\}_{u=1}^{m} = \text{EncoderBlock}^q(E^q) \in \mathbb{R}^{m \times k}$$
$$C^d = \{\mathbf{c}_t^d\}_{t=1}^{n} = \text{EncoderBlock}^d(E^d) \in \mathbb{R}^{n \times k}.$$

These representations are used to calculate the pairwise similarity matrix $S \in \mathbb{R}^{m \times n}$ between tokens in the question and those in the document by a tri-linear function (Seo et al., 2017), i.e.,

$$s_{ij} = \mathbf{w}_q \cdot \mathbf{c}_i^q + \mathbf{w}_d \cdot \mathbf{c}_j^d + \mathbf{w}_c \cdot (\mathbf{c}_i^q \odot \mathbf{c}_j^d),$$

where $i = 1, \ldots, m$, $j = (1, \ldots, n)$, and $\odot$ represents the element-wise multiplication and $\mathbf{w}_q$, $\mathbf{w}_d$, and $\mathbf{w}_c$ are trainable vectors. We apply column-wise softmax to S to obtain the document-to-question attention matrix A. Afterwards, a question-attended document representation $\tilde{C}^q$ is calculate as

$$\tilde{C}^q = \{\tilde{\mathbf{c}}_t^q\}_{t=1}^n = A^T C^q \in \mathbb{R}^{n \times k}.$$

In addition to this, we obtain vector $\tilde{\mathbf{a}} \in \mathbb{R}^n$, corresponding to the attention of a question to document tokens, by applying softmax to the column-wise max values of S. Then document-attended question vector is obtained by

$$\tilde{\mathbf{c}}^d = \sum_{t=1}^{n} \tilde{\mathbf{a}}_t \mathbf{c}_t^d \in \mathbb{R}^k.$$

The final co-attended representations $\{\mathbf{d}_t\}_{t=1}^n$ is obtained by fully connected layer with ReLU as a nonlinear unit, φ, as

$$\mathbf{d}_t = \varphi([\mathbf{c}_t^d; \tilde{\mathbf{c}}_t^q; \mathbf{c}_t^d \odot \tilde{\mathbf{c}}_t^q; \mathbf{c}_t^d \odot \tilde{\mathbf{c}}_t^d]) \in \mathbb{R}^l.$$

Memory controller. This layer takes the output of the co-attention layer $\{\mathbf{d}_t\}_{t=1}^n$ as input and refine their representations using our proposed memory controller (Section 2.1). Afterwards, the resulting output vector $\{\mathbf{o}_t\}_{t=1}^n$ are given as input to the prediction layer.

Prediction layer. We feed the output of the memory controller $\{\mathbf{o}_t\}_{t=1}^n$ to the prediction layer to predict the start and the end token indices of the answer span. First, it goes through the encoder block followed by the fully connected layer with softmax over the entire sequence to compute the probability distribution of a start index. The probability distribution of the end index is calculated by concatenating the output of the encoder block for the start index with the output of the memory controller and then by feeding them as input to another encoder block. These probability distributions are used as part of the negative log-likelihood objective function.

4 Experimental Setup

Datasets and preprocessing. We perform extensive experiments with well-known benchmarks such as TriviaQA, QUASAR-T, and SQuAD, as summarized in Table 1. In most of these datasets, a question q and a document d are represented as a sequence of words, and the answer span has to be selected from the document words based on the question. SQuAD consists of crowd-sourced questions and paragraphs from Wikipedia articles containing the answer to these questions. QUASAR-T is mostly based on factoid questions with their corresponding, large-sized corpus. TriviaQA is composed of question-answer pairs obtained from 14 trivia and quiz-league websites, along with the documents collected later that are likely to contain the answer from either web search or Wikipedia. In TriviaQA dataset, we truncate each document to 1,200 words. Even with such truncation, the average word count per document (AWC) of TriviaQA is approximately four times larger than that of SQuAD. In terms of the AWC, documents in TriviaQA, QUASAR-T, and SQuAD can be viewed as large-, medium-, and small-length documents, respectively.

Dataset	Total Train / Dev / Test	AWC
SQuAD	87,599 / 10,570 / UNK	142
QUASAR-T (Short)	25,465 / 2,043 / 2,068	221
QUASAR-T (Long)	26,318 / 2,129 / 2,102	392
TriviaQA (Web)	528,979 / 68,621 / 65,509	631
TriviaQA (Wikipedia)	110,648 / 14,229 / 13,661	955

Table 1: Statistics of datasets in terms of the average word count per document (AWC). In TriviaQA, AWC was calculated after truncating each document to 1,200 words.

In TriviaQA dataset, because a document is collected separately for an already collected question-answer pair, the document does not sometimes have the information to properly infer the answer to the question. In response, Clark and Gardner (2017) attempted to solve this problem by exposing both relevant and irrelevant paragraphs together separated based on TF-IDF scores. We follow this approach in TriviaQA. In QUASAR-T, we follow the same preprocessing steps done by Dhingra et al. (2017).

Implementation details. We use TensorFlow[1]

[1]http://www.tensorflow.org

Domain	Model	Full		Verified		AF
		EM	F1	EM	F1	
Web (AWC=631)	Our model (with DEBS)	**68.21**	**73.26**	**82.57**	**86.05**	
	Our model (without DEBS)	66.82	71.91	81.01	84.12	
	BiDAF + SA + SN (Clark and Gardner, 2017)	66.37	71.32	79.97	83.70	
	Reading Twice for NLU (Weissenborn, 2017)	50.56	56.73	63.20	67.97	✓
	M-Reader (Hu et al., 2017)	46.65	52.89	56.96	61.48	✓
	BiDAF + DNC	42.34	48.65	51.50	57.17	
	MEMEN (Pan et al., 2017)	44.25	48.34	53.27	57.64	✓
	BiDAF (Seo et al., 2017)	40.74	47.05	49.54	55.80	
Wikipedia (AWC=955)	Our model (with DEBS)	64.12	69.44	**71.75**	**76.91**	
	Our model (without DEBS)	**64.41**	**69.60**	70.21	75.49	
	BiDAF + SA + SN (Clark and Gardner, 2017)	63.99	68.93	67.98	72.88	
	QANet (Yu et al., 2018a)	51.10	56.60	53.30	59.20	
	Reading Twice for NLU (Weissenborn, 2017)	48.60	55.10	53.40	59.90	✓
	M-Reader (Hu et al., 2017)	46.94	52.85	54.45	59.46	✓
	BiDAF + DNC	42.57	48.30	46.23	51.61	
	MEMEN (Pan et al., 2017)	43.16	46.90	49.28	55.83	✓
	BiDAF (Seo et al., 2017)	40.32	45.91	44.86	50.71	

Table 2: Single model results on TriviaQA (Web and Wikipedia) dataset. All the results are gathered from their corresponding publications except for our models and 'BiDAF + DNC,' which we implemented on our own. 'Full' represents a complete dataset not guaranteed to contain relevant information to answer the question while 'Verified' corresponds to its subset annotated by humans so that the relevant information for the answer is guaranteed to exist. The last column indicates whether a model uses any additional feature augmentation (AF).

to build the model and Sonnet[2] to implement the memory interface. NLTK (Bird and Loper, 2004) is used for tokenizing words. In the memory controller, we use four read heads and one write head, and the memory size is set to 100×36, with all initialized as 0. The hidden vector dimension l is set to 200. We use AdaDelta (Zeiler, 2012) as an optimizer with a learning rate of 0.5. The batch size is set to 20 for TriviaQA (Joshi et al., 2017) and 30 for SQuAD (Rajpurkar et al., 2016) and QUASAR-T (Dhingra et al., 2017). We use an exponential moving average of weights with a decaying factor of 0.001. Our model does require more memory than existing methods, but a single GPU (e.g., M40 with 12GB memory) was enough to train model within a reasonable amount of time.

5 Quantitative Results

For our quantitative comparisons, we use BiDAF with self attention (Clark and Gardner, 2017) as a baseline, which maintains the best results published on both TriviaQA and SQuAD datasets. In TriviaQA and QUASAR-T dataset, we compare our model with BiDAF (Seo et al., 2017) as well as its variant called 'BiDAF + DNC,' which is augmented with an existing external memory architecture (Graves et al., 2016) just before the answer prediction layer in the BiDAF.

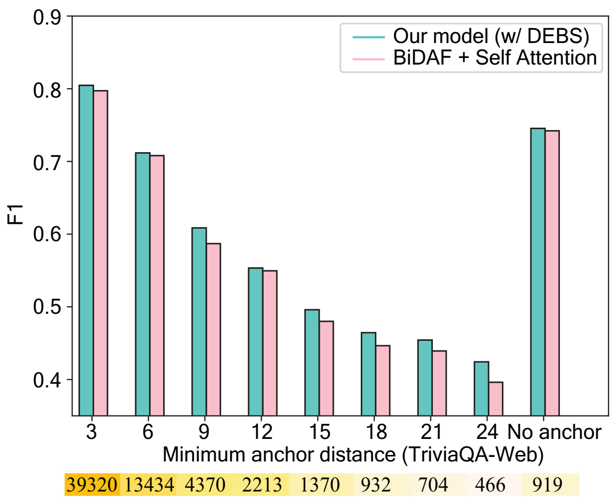
Overall, in lengthy-document cases such as TriviaQA and QUASAR-T, our model outperforms all the published results, as seen in Tables 2 and 3, while in the short-document case such as SQuAD, we mostly achieve the best results, as seen in Table 4. In the following, we present detailed analyses on each dataset.

TriviaQA. As shown in Table 2, our model, even without DEBS, outperforms the existing state-of-the-art method such as 'BiDAF + SA + SN' by a large margin in all the cases. Our model with DEBS, which replaces BiGRU encoder blocks, performs even better than that without it in all the cases except for the combination of the 'full' and 'Wikipedia' case, which involves documents containing no relevant information for the answer. Among those methods shown in Table 2, Reading Twice for NLU (Weissenborn, 2017) uses background knowledge from ConceptNet while both M-Reader (Hu et al., 2017) and MEMEN (Pan et al., 2017) use POS and NER in-

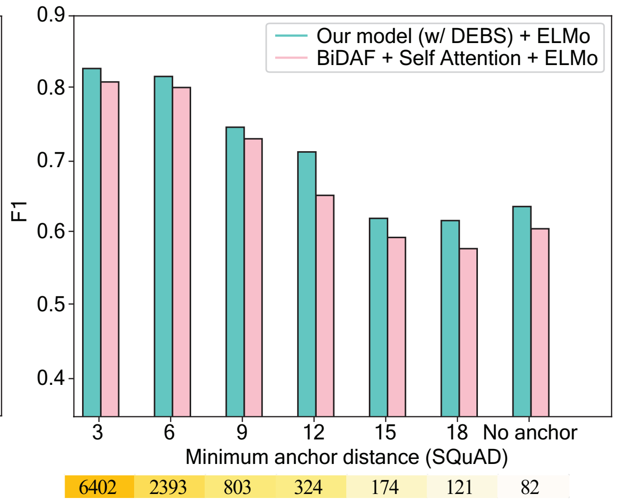
[2]https://github.com/deepmind/sonnet

Dataset	Model	Dev set EM	Dev set F1	Test set EM	Test set F1
Short documents (AWC=221)	Our model (with DEBS)	**65.06**	**69.17**	**69.11**	**71.19**
	Our model (without DEBS)	64.87	68.88	68.13	70.32
	BiDAF + DNC	51.18	54.77	54.81	58.24
	BiDAF (Seo et al., 2017)	45.40	50.90	47.60	52.40
Long documents (AWC=392)	Our model (with DEBS)	**62.08**	**65.21**	**63.54**	**66.87**
	Our model (without DEBS)	60.05	63.23	63.44	65.19
	BiDAF + DNC	48.67	52.25	52.15	54.43
	BiDAF (Seo et al., 2017)	37.00	42.50	39.50	44.50

Table 3: Performance results on QUASAR-T dataset.



(a) TriviaQA (Web).



(b) SQuAD.

Figure 2: F1 score on the development set in TriviaQA (Web) and SQuAD with respect to the minimum anchor distance.

formation as additional features. We note that our method achieves these outstanding results without any additional features.

QUASAR-T. As shown in Table 3, our simple baseline 'BiDAF + DNC,' which involves an existing memory architecture, improves performance over BiDAF, indicating the efficacy of incorporating an external memory. Moreover, our model with the proposed memory controller achieves significantly better results compared to other models. Furthermore, another proposed component, DEBS, gives an additional performance boost to our model.

SQuAD. As shown in Table 4, most of the models, if not all, use additional features such as ELMo (Peters et al.), and the self-attention mechanism to further improve the performance. We also adopt these mechanisms one by one to show that our model can also benefit from these. First, we adopt ELMo to our model (without DEBS), which uses word embedding as the weighted sum of the hidden layers of a language model with regularization as an additional feature to our word embeddings. This improves the F1 score of our model up to 85.13 and EM to 77.44, showing the highest performances among all the methods without using self attention. Due to the relatively short document length in SQuAD compared to TriviaQA and QUASAR-T, our model without DEBS performs worse than the baseline 'BiDAF + Self Attention + ELMo.' However, after applying DEBS, our model outperforms the baseline, achieving 86.73 F1 and 79.69 EM.

Minimum anchor distance. Rajpurkar et al. (2016) proposed the difficulty measure called syntactic divergence, which is computed as the edit distance between syntactic parse trees of the question and the sentence containing the answer. However, this measure has limitations that the syntactic parser does not work properly on incomplete sentences, which are common in web text. It also becomes difficult to compute this measure if the

Model	Test set EM	Test set F1	AF	SA
Our model (with DEBS) + ELMo	**79.69**	**86.73**	✓	✓
BiDAF + Self Attention + ELMo (Peters et al.)	78.58	85.83	✓	✓
Our model (without DEBS) + ELMo	**77.44**	**85.13**	✓	
RaSoR + TR + LM (Salant and Berant, 2017)	77.58	84.16	✓	
QANet (Yu et al., 2018a)	76.24	84.60	✓	✓
SAN (Liu et al., 2017b)	76.83	84.40	✓	✓
FusionNet (Huang et al., 2017b)	75.97	83.90	✓	✓
RaSoR + TR (Salant and Berant, 2017)	75.79	83.26	✓	
Conducter-net (Liu et al., 2017a)	74.41	82.74	✓	✓
Reinforced Mnemonic Reader (Hu et al., 2017)	73.20	81.80	✓	✓
BiDAF + Self Attention (Clark and Gardner, 2017)	72.14	81.05		✓
MEMEN (Pan et al., 2017)	70.98	80.36	✓	
Our model (without DEBS)	**70.99**	**79.94**		
r-net (Wang et al., 2017)	71.30	79.70		✓
Document Reader (Chen et al., 2017)	70.73	79.35	✓	
FastQAExt (Weissenborn et al., 2017)	70.85	78.86		✓
Human Performance	82.30	91.22		

Table 4: Single model results on SQuAD. All the other results than ours are those reported in their own publications. The last two column indicate whether a model uses any additional feature augmentation (AF) and self attention (SA).

answer requires multi-sentence inference.

Instead, we develop our own metric called a minimum anchor distance, which is simple and robust to noisy text. To compute this metric, we first identify for all the co-occurring words (anchor words) between a document and a question while ignoring stop words. Then, we compute the number of words found between the answer and all the possible anchor words and select the minimum number from these.

In Figure 2, we show F1 scores of our model with DEBS and the baseline with respect to the minimum anchor distance. The scores are obtained from the development set of TriviaQA(Web) and SQuAD. The heat map at the bottom of the figure indicates the number of samples in each interval of the minimum anchor distance. One can see that our model performs increasingly better than the baseline as the minimum anchor distance gets larger. The examples shown in Table 5 indicate that documents with long dependencies tend to have a large minimum anchor distance. These examples show that our model predicts the remotely placed answer from the anchor word relatively well when anaphora resolution and negation are involved.

Ablation study with an encoder block. We assume that the concatenation of the layer outputs in DEBS helps the memory controller store contextual representations clearly. To see how DEBS affects the memory controller depending on different positions in the entire network, we conducted an ablation study by replacing the encoder block with DEBS on SQuAD. As can be seen in Table 6, using DEBS in all the places improves the performance most, and furthermore, the memory controller with DEBS gives the largest performance margin. This implies that DEBS can generally work as a better alternative to a BiGRU module, and DEBS is critical in maintaining the high performance of our proposed memory controller.

Adding our proposed modules to other models. To show the wide effectiveness of our proposed approaches, we choose two well-known baseline models in SQuAD: R-net (Wang et al., 2017) and 'BiDAF + Self Attention' (Clark and Gardner, 2017). These models have similar architectures where the model first pairs a given question and document pair using an attention and afterward applies a self-attention mechanism. We use the publicly available implementation of these models[3,4]. In Table 7, replacing all the recurrent units with DEBS and adding our memory controller between the question-document pairing

[3]https://github.com/HKUST-KnowComp/R-Net

[4]https://github.com/allenai/document-qa

Dataset	Example
TriviaQA (Web)	**Question** : What claimed the life of singer Kathleen Ferrier? **Context** : (omit) $\cdots$ Kathleen Ferrier (22.III.1912 Higher Walton, Lancashire- 8.X. 1953 London, England) was an English contralto singer[*] who achieved an international reputation with a repertoire extending from folksong and popular ballads to the classical works. Her death from $\boxed{\textit{cancer}}$, at the height of her fame, was a shock to the musical world and particularly to the general public, which was kept in ignorance of $\cdots$ (omit)
SQuAD	**Question** : What did Mote think the Yuan class system really represented? **Context** : The historian Frederick W.Mote wrote that the usage of the term "social classes" for this system was misleading and that the position of people within the four -class system[*] was not an indication of their actual social power and wealth, but just entailed $\boxed{\textit{"degrees of privilege"}}$ to which they were entitled institutionally and legally, so a person's standing within the classes was not a guarantee of their standing, $\cdots$ (omit)

[*]A word with an asterisk indicates an anchor word closest to the ground truth answer.

Table 5: Examples in TriviaQA (Web) and SQuAD. *Italic* means the ground truth answer, $\boxed{\text{frame}}$ indicates the prediction of our model (with DEBS) and underline shows the prediction of 'BiDAF + Self Attention' model.

Adding DEBS			Dev	
C	M	P	EM	F1
			77.22	85.01
✓			77.31	85.22
✓		✓	77.75	85.34
✓	✓		78.70	86.12
✓	✓	✓	**78.93**	**86.26**

Table 6: Ablation study of replacing an encoder block with DEBS in the co-attention layer (C), the memory controller (M), and the prediction layer (P) in SQuAD. ✓means that DEBS is used. Otherwise, BiGRU is used.

Base model	Adapt-ation	Dev	
		EM	F1
R-net[3]	-	70.71 (0.07)	79.48 (0.08)
	✓	**71.12** (0.12)	**79.99** (0.11)
BiDAF +SA[4]	-	71.61 (0.07)	80.78 (0.08)
	✓	**72.82** (0.15)	**81.33** (0.09)

Table 7: Effects of our proposed components added to R-net and 'BiDAF + Self Attention (SA)' on SQuAD. The values in parentheses represent the standard deviation from 6 runs. The first row of each base model indicates the result of the original methods. When adding the proposed component, DEBS is used in the place of all the recurrent layers while the memory controller is added between the co-attention and the self-attention layers.

layer and the self-attention layer increases the F1 score by around 0.5 compared to the baseline.

6 Related Work

Numerous neural network-based methods have been proposed, pushing the performance nearly up to a human level. Although slight differences exist, (Wang et al., 2017; Seo et al., 2017; Xiong et al., 2017) mostly leverage the question-document co-attention based on their pairwise similarity of word-level vector representations. These models currently work as the backbone architecture for many other models. Furthermore, Wang et al. (2017) suggest utilizing a self attention mechanism between tokens within a document to refine contextual representations.

Salant and Berant (2017); Chen et al. (2016); Pan et al. (2017); Weissenborn (2017); Peters et al. focus on augmenting feature representations in the word embedding layer to provide rich information. Salant and Berant (2017); Weissenborn (2017); Peters et al. extract and use additional features from other neural models trained for another task or external resources. Chen et al. (2016); Pan et al. (2017) utilize additional syntactic or semantic features through part-of-speech tagging or named-entity recognition, etc.

Enriching the input representation from pre-trained external models has been shown to be useful in improving RC task performances. Yu et al. (2018a) have also improved the performance by leveraging self attention for context encoding based on convolutional neural networks. Hu et al. (2017) refine the contextual representation with multiple hops, and Pan et al. (2017) use the encoded query for refining the answer prediction as a memory, which are different from our work in terms of handling long-range dependency.

7 Conclusion

This paper proposed two novel, crucial components for deep neural network-based RC tasks, (1) an advanced memory controller architecture and (2) a densely connected encoder block with self attention. We showed the effectiveness of these approaches in handling long-range dependencies using three benchmark RC datasets such as TriviaQA, QUASAR-T, and SQuAD. Our proposed modules are widely applicable to other models to improve their performance. Future work includes developing a scalable read/write accessing mechanism to handle a large-scale external memory to reason over multiple documents.

Acknowledgement

We thank to reviewers for helpful feedback. This work is the extended version of Yu et al. (2018b). This work is partially supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIP) (No. NRF2016R1C1B2015924).

References

Steven Bird and Edward Loper. 2004. Nltk: the natural language toolkit. In *Proc. the ACL 2004 Interactive poster and demonstration sessions*, page 31.

Danqi Chen, Jason Bolton, and Christopher D Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. In *Proc. the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (ACL)*, volume 1, pages 2358–2367.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proc. the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Christopher Clark and Matt Gardner. 2017. Simple and effective multi-paragraph reading comprehension. *arXiv preprint arXiv:1710.10723*.

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

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. 2016. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471.

Mikael Henaff, Antoine Bordes Jason Weston, Arthur Szlam, and Yann LeCun. 2017. Tracking the world state with recurrent entity networks. *arXiv preprint arXiv:1612.03969*.

Minghao Hu, Yuxing Peng, and Xipeng Qiu. 2017. Reinforced mnemonic reader for machine comprehension. *arXiv preprint arXiv:1705.02798*.

Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. 2017a. Densely connected convolutional networks. In *Proc. the IEEE conference on computer vision and pattern recognition (CVPR)*, volume 1, page 3.

Hsin-Yuan Huang, Chenguang Zhu, Yelong Shen, and Weizhu Chen. 2017b. Fusionnet: Fusing via fully-aware attention with application to machine comprehension. *arXiv preprint arXiv:1711.07341*.

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

Rui Liu, Wei Wei, Weiguang Mao, and Maria Chikina. 2017a. Phase conductor on multi-layered attentions for machine comprehension. *arXiv preprint arXiv:1710.10504*.

Xiaodong Liu, Yelong Shen, Kevin Duh, and Jianfeng Gao. 2017b. Stochastic answer networks for machine reading comprehension. *arXiv preprint arXiv:1712.03556*.

Boyuan Pan, Hao Li, Zhou Zhao, Bin Cao, Deng Cai, and Xiaofei He. 2017. Memen: Multi-layer embedding with memory networks for machine comprehension. *arXiv preprint arXiv:1707.09098*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proc. the conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

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

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proc. the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2383–2392.

Shimi Salant and Jonathan Berant. 2017. Contextualized word representations for reading comprehension. *arXiv preprint arXiv:1712.03609*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *Proc. the International Conference on Learning Representations (ICLR)*.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proc. the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (ACL)*, volume 1, pages 189–198.

Dirk Weissenborn. 2017. Dynamic integration of background knowledge in neural nlu systems. *arXiv preprint arXiv:1706.02596*.

Dirk Weissenborn, Georg Wiese, and Laura Seiffe. 2017. Making neural qa as simple as possible but not simpler. In *Proc. the Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 271–280.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. Towards AI-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.

Caiming Xiong, Victor Zhong, and Richard Socher. 2017. Dynamic coattention networks for question answering. In *Proc. the International Conference on Learning Representations (ICLR)*.

Adams Wei Yu, David Dohan, Quoc Le, Thang Luong, Rui Zhao, and Kai Chen. 2018a. Qanet: Combining local convolution with global self-attention for reading comprehension.

Seunghak Yu, Sathish Reddy Indurthi, Seohyun Back, and Haejun Lee. 2018b. A multi-stage memory augmented neural network for machine reading comprehension. In *Proc. the Workshop on Machine Reading for Question Answering*, pages 21–30. Association for Computational Linguistics.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.