A Multi-Stage Memory Augmented Neural Network for Machine Reading Comprehension

Seunghak Yu **Sathish Indurthi** **Seohyun Back** **Haejun Lee**
Samsung Research, Seoul, Korea
{seunghak.yu, s.indurthi, scv.back}@samsung.com

Abstract

Reading Comprehension (RC) of text is one of the fundamental tasks in natural language processing. In recent years, several end-to-end neural network models have been proposed to solve RC tasks. However, most of these models suffer in reasoning over long documents. In this work, we propose a novel Memory Augmented Machine Comprehension Network (MAMCN) to address long-range dependencies present in machine reading comprehension. We perform extensive experiments to evaluate proposed method with the renowned benchmark datasets such as SQuAD, QUASAR-T, and TriviaQA. We achieve the state of the art performance on both the document-level (QUASAR-T, TriviaQA) and paragraph-level (SQuAD) datasets compared to all the previously published approaches.

1 Introduction

Reading Comprehension (RC) is essential for understanding human knowledge written in text form. One possible way of measuring RC is by formulating it as answer span prediction style Question Answering (QA) task, which is finding an answer to the question based on the given document(s). Recently, influential deep learning approaches have been proposed to solve this QA task. Wang and Jiang (2017); Seo et al. (2017) propose the attention mechanism between question and context for question-aware contextual representation. Wang et al. (2017) refine these contextual representations by using self-attention to improve the performance. Even further performance improvement is gained by using contextualized word representations for query and context (Salant and Berant, 2017; Peters et al., 2018; Yu et al., 2018).

Based on those approaches, several methods have successfully made progress towards reaching human-level performance on SQuAD (Rajpurkar et al., 2016). Each training example in the SQuAD only has the relevant paragraph with the corresponding answer. However, most of the documents present in the real-world are long, containing relevant and irrelevant paragraphs, and do not guarantee answer presence. Therefore the models proposed to solve SQuAD have difficulty in applying to real-world documents (Joshi et al., 2017). Recently, QUASAR-T (Dhingra et al., 2017) and TriviaQA (Joshi et al., 2017) datasets have been proposed to resemble real-world document. These datasets use document-level evidence as training example instead of using only the relevant paragraph and evidence does not guarantee answer presence, which makes them more realistic.

To effectively comprehend long documents present in the QUASAR-T and TriviaQA datasets, the QA models have to resolve long-range dependencies present in these documents. In this work, we build a QA model that can understand long documents by utilizing Memory Augmented Neural Networks (MANNs) (Graves et al., 2014; Weston et al., 2015b). This type of neural networks decouples the memory capacity from the number of model parameters. While there have been several attempts to use MANNs in managing long-range dependencies, applications are limited to only toy datasets (Sukhbaatar et al., 2015; Weston et al., 2015a; Kumar et al., 2016; Graves et al., 2016). Compared to the previous approaches, we mainly focus on the document-level QA task on QUASAR-T and TriviaQA. We also apply our model to SQuAD to show that our model even works well on the paragraph-level.

Our contributions in this work are as follows:

Proceedings of the Workshop on Machine Reading for Question Answering, pages 1–10
Melbourne, Australia, July 19, 2018.

(1) We develop Memory Augmented Machine Comprehension Network (MAMCN) to solve document-level RC task. (2) Our method achieves the state of the art performance compared to all the published results on both the document-level (QUASAR-T and TriviaQA) and paragraph-level (SQuAD) benchmarks. In TriviaQA we achieve 71.91 and 69.60 F1 scores for Web and Wikipedia domains respectively. Also, we achieved 86.73 F1 compared to the human performance of 91.22 F1 in SQuAD benchmark. These results show that MAMCN is a crucial component for QA task, especially they are useful in comprehending long documents.

2 Related Work

Many neural networks have been proposed to solve answer span QA task. Ranking continuous text spans within a passage was proposed by Yu et al. (2016) and Lee et al. (2016). Wang and Jiang (2017) combine match-LSTM, originally introduced in (Wang and Jiang, 2016) and pointer networks to produce the boundary of the answer. Since then, most of the models adopted pointer networks as a prediction layer and then focused on improving other layers. Some methods focused on devising more accurate attention method; Seo et al. (2017); Wang et al. (2017); Xiong et al. (2017) employ attention mechanism to match the question context mutually; In addition, Liu et al. (2017a) apply multi-layer attention and Huang et al. (2017b) expand to multi-level attention to get more enriched attention information.

Other approaches use contextualized word representations to further improve the performance. Salant and Berant (2017); Peters et al. (2018) utilize embedding from pre-trained language model as an additional feature and Yu et al. (2018) select machine translation model instead. Also, there are few attempts at augmenting memory capacity of the model (Hu et al., 2017; Pan et al., 2017). Hu et al. (2017) refine the contextual representation with multi-hops, and Pan et al. (2017) simply use the encoded query representations as a memory vector for refining the answer prediction, which are not meant to handle long-range dependency that we consider in this work.

3 Proposed Model

We propose a memory augmented reader for answer-span style QA task. Answer-span style QA task is defined as follows. Question and document can be represented by sequence of words $\boldsymbol{q} = \{w_i^q\}_{i=1}^m$, $\boldsymbol{d} = \{w_j^d\}_{j=1}^n$ respectively. Answer-span $\boldsymbol{a} = \{w_k^d\}_{k=s}^e$ (where, $1 \leq s \leq e \leq n$) should be returned, given $\boldsymbol{q}$ and $\boldsymbol{d}$. We embed the sequence of words in $\boldsymbol{q}$ and $\boldsymbol{d}$ to get contextual representations. These contextual representations are used for calculating question-aware context representation which is controlled by external memory unit and used for predicting answer-span. We describe more details of each layer in following sections and depict overall architecture of proposed model in Figure 1.

3.1 Contextual Representation

Word Embedding: We use two different kinds of embeddings to get the richer feature representation for each word in the question and document. Word-level embedding helps to have a representation explaining semantic similarities as proximity in high dimensional vector space. In addition to this, we utilize character-level embedding by applying convolution filters to address out-of-vocabulary and infrequent words problem. We concatenate both embeddings $[\boldsymbol{e}^w; \boldsymbol{e}^c]$ to represent each word embedding as $\boldsymbol{e} \in \mathbb{R}^l$.

Contextual Embedding: We compute the contextual representation for each word in the question and document by using bi-directional GRU (Cho et al., 2014) as follows:

$$\boldsymbol{c}_i^q = BiGRU_q(\boldsymbol{e}_i^q, \boldsymbol{h}_{i-1}, \boldsymbol{h}_{i+1}), \quad (1)$$

$$\boldsymbol{c}_j^d = BiGRU_d(\boldsymbol{e}_j^d, \boldsymbol{h}_{j-1}, \boldsymbol{h}_{j+1}), \quad (2)$$

where $\boldsymbol{e}_i^q, \boldsymbol{e}_j^d$ are the word embeddings for each word in the question and document, and $\boldsymbol{h}_{i-1}, \boldsymbol{h}_{i+1}$ are the hidden states of the forward and reverse GRUs respectively. The final question and document contextual representation are $\boldsymbol{C}^q \in \mathbb{R}^{m \times 2l}$ and $\boldsymbol{C}^d \in \mathbb{R}^{n \times 2l}$.

Co-attention: We compute question-aware representations for each word in the document by adopting the attention mechanism in (Clark and Gardner, 2017). To get these representations, we first compute the similarity matrix $\boldsymbol{S}$ followed by bi-directional attention between the words in the question and document as follows:

$$s_{ij} = \boldsymbol{w}_q \boldsymbol{C}_{i,:}^q + \boldsymbol{w}_d \boldsymbol{C}_{j,:}^d + \boldsymbol{w}_h (\boldsymbol{C}_{i,:}^q \odot \boldsymbol{C}_{j,:}^d) \quad (3)$$

$\boldsymbol{w}_q$, $\boldsymbol{w}_d$, and $\boldsymbol{w}_h$ are trainable weights and $\odot$ is element-wise multiplication. Each element of

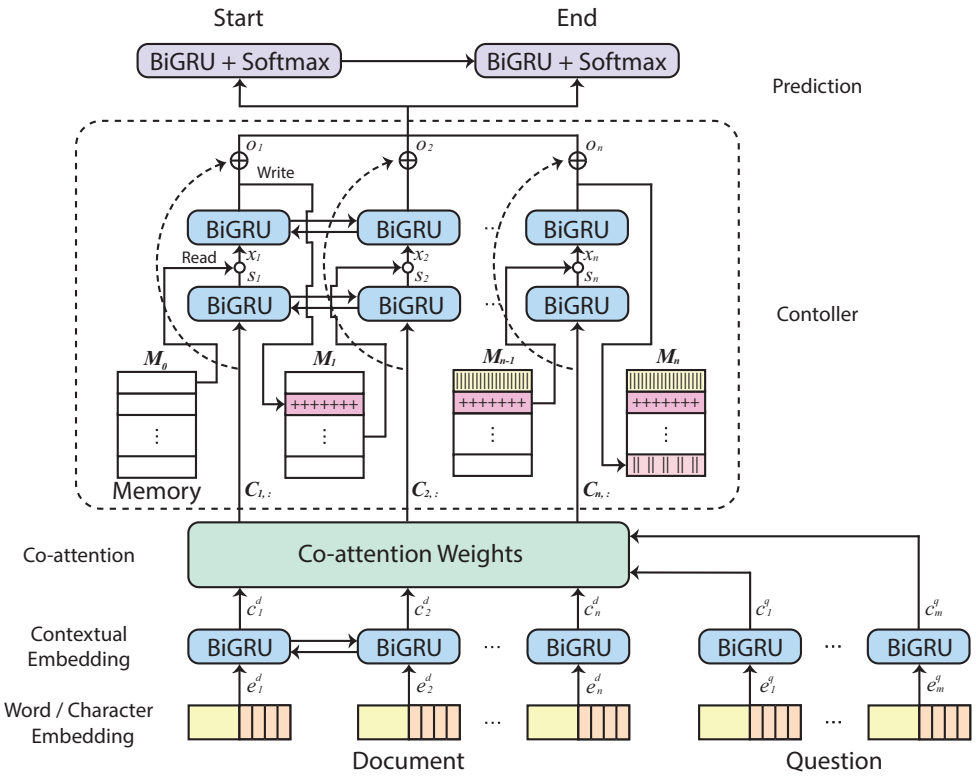

Figure 1: The architecture of Memory Augmented Machine Comprehension Network (MAMCN).

similarity matrix s_{ij} represents attention between i_{th} word in the question and j_{th} word in the document. We can get the attention weights of question words to each document word by applying column-wise softmax to $\boldsymbol{S}$.

$$a_{ij}^q = \left(\frac{e^{s_{ij}}}{\sum_{k=1}^{m} e^{s_{kj}}}\right) \tag{4}$$

$\boldsymbol{A}_{:,j}^q$ is attention weights of questions to j_{th} word in the document. We can get attended question vectors for the document words by multiplying entire attention matrix $\boldsymbol{A}^q$ to the contextual embedding of question $\boldsymbol{C}^q$.

$$\tilde{\boldsymbol{C}}^q = \boldsymbol{A}^{qT}\boldsymbol{C}^q \in \mathbb{R}^{n\times 2l} \tag{5}$$

Also, we can get attention weights on the document words for the question $\boldsymbol{a}^d$ by applying *softmax* to the column-wise max values ($\boldsymbol{v}^d$) of attention matrix $\boldsymbol{A}^q$.

$$a_j^d = \left(\frac{e^{v_j^d}}{\sum_{k=1}^{n} e^{v_k^d}}\right), \quad (v_j^d = \max_{1\le i\le m} a_{ij}^q) \tag{6}$$

This attention weight on the document words $\boldsymbol{a}^d$ is duplicated n times for each row to make $\boldsymbol{A}^d$ and applied to contextual embedding of document $\boldsymbol{C}^d$ to get attended document vectors.

$$\tilde{\boldsymbol{C}}^d = \boldsymbol{A}^d\boldsymbol{C}^d \in \mathbb{R}^{n\times 2l} \tag{7}$$

Finally, we can make question-aware contextual representation $\boldsymbol{C}$ by concatenating as follows:

$$\boldsymbol{C} = [\boldsymbol{C}^d; \tilde{\boldsymbol{C}}^q; \boldsymbol{C}^d \odot \tilde{\boldsymbol{C}}^q; \boldsymbol{C}^d \odot \tilde{\boldsymbol{C}}^d] \in \mathbb{R}^{n\times 8l} \tag{8}$$

3.2 Memory Controller

Memory controller allows us to utilize external memory to compensate for the limited memory capacity of the recurrent layers. We develop the memory controller inspired by the memory framework of (Graves et al., 2016). The operation of the controller at time step t is given by:

$$\boldsymbol{o}_t, \boldsymbol{i}_t = Controller(\boldsymbol{C}_{t,:}, \boldsymbol{M}_{t-1}) \tag{9}$$

It takes the contextual representation vector $\boldsymbol{C}_{t,:}$ as input, and the external memory matrix of the previous time step $\boldsymbol{M}_{t-1} \in \mathbb{R}^{p\times q}$, where p is the number of memory locations, and q is the dimension of each location. We choose two layers of BiGRUs as the recurrent layer for the controller. The contextual representations are fed into the first layer to capture interactions between contexts.

$$\boldsymbol{s}_t = BiGRU(\boldsymbol{C}_{t,:}, \boldsymbol{h}_{t-1}, \boldsymbol{h}_{t+1}) \in \mathbb{R}^{2l}. \tag{10}$$

Then, $\boldsymbol{s}_t$ and the subset of memory vectors obtained from the memory matrix are concatenated to generate an input vector $\boldsymbol{x}_t$ for the second layer,

$$\boldsymbol{x}_t = [\boldsymbol{s}_t; \boldsymbol{m}_{t-1}^1; \cdots; \boldsymbol{m}_{t-1}^s] \in \mathbb{R}^{2l+sq}, \quad (11)$$

where s is the number of read vectors. After feeding the input vector $\boldsymbol{x}_t$ to the second recurrent layer, the controller uses the outputs of the layer $\boldsymbol{h}_t^m$ to emit the output vector $\boldsymbol{o}_t$ and the interface vector $\boldsymbol{i}_t$. We describe details of obtaining each vector as follows:

$$\boldsymbol{h}_t^m = BiGRU(\boldsymbol{x}_t, \boldsymbol{h}_{t-1}^m, \boldsymbol{h}_{t+1}^m) \quad (12)$$

Output Vector: The controller makes the output vector as a weighted sum $\boldsymbol{v}_t$ of hidden state of the recurrent layer and memory vectors.

$$\boldsymbol{v}_t = W_o \boldsymbol{h}_t^m + W_m[\boldsymbol{m}_t^1; \cdots; \boldsymbol{m}_t^s] \quad (13)$$

We add a residual connection between the controller's input and output to mitigate the information morphing that can occur when interacting with memory. As a result, we get a long-term dependency-aware output vector as follows:

$$\boldsymbol{o}_t = W_v \boldsymbol{v}_t + W_c \boldsymbol{C}_{t,:} \in \mathbb{R}^{2l} \quad (14)$$

Interface Vector: At the same time, the controller generates interface vector $\boldsymbol{i}_t$ for the memory interaction based on h_t^m as follows:

$$\boldsymbol{i}_t = W_i \boldsymbol{h}_t^m \in \mathbb{R}^{sq+5s+3q+3}. \quad (15)$$

We consider it as a concatenation of various functional vectors which determine the basic operation of the memory such as memory addressing, read and write. The complete list of functional vectors is described in Table 1.

Memory Addressing : We use content-based addressing mechanism for read and write operations. In content-based addressing, the memory locations required to read/write at the current time step t are obtained by using the probability distribution over the memory locations, which is a similarity function between the key vector and each memory vector.

$$c_i = softmax(cos(\boldsymbol{M}_{i,:}, \boldsymbol{k})\tilde{\boldsymbol{\alpha}}) \quad (16)$$

cos is the cosine similarity and $\tilde{\boldsymbol{\alpha}}$ is a constrained strength value to $[1, \infty)$ by $(1 + \log(1 + e^{\boldsymbol{\alpha}}))$.

Operation	Name	Vector
Read	key	$\{\boldsymbol{k}_t^{r,i}\}_{i=1}^s \in \mathbb{R}^q$
	strength	$\{\boldsymbol{\alpha}_t^{r,i}\}_{i=1}^s \in \mathbb{R}$
	mode	$\{\boldsymbol{\pi}_t^i\}_{i=1}^s \in \mathbb{R}^3$
Write	key	$\boldsymbol{k}_t^w \in \mathbb{R}^q$
	strength	$\boldsymbol{\alpha}_t^w \in \mathbb{R}$
	erase vector	$\boldsymbol{e}_t \in \mathbb{R}^q$
	write vector	$\boldsymbol{v}_t \in \mathbb{R}^q$
	free gate	$\{\boldsymbol{g}_t^{f,i}\}_{i=1}^s \in \mathbb{R}$
	allocate gate	$\boldsymbol{g}_t^a \in \mathbb{R}$
	write gate	$\boldsymbol{g}_t^w \in \mathbb{R}$

Table 1: The list of functional vectors that make up the interface vector of controller.

Read Operation: Each read head performs a read operation by weighting over entire memory.

$$\boldsymbol{c}_t^r = softmax(cos(\boldsymbol{M}_t, \boldsymbol{k}_t^r)\tilde{\boldsymbol{\alpha}}_t^r) \quad (17)$$

Along with this, we use a temporal memory linkage matrix to associate memories together, which keep track of consecutively modified locations in the memory. The multiplication of temporal link matrix and read weights from the previous time-step gives backward $\boldsymbol{b}_t$ and forward weights $\boldsymbol{f}_t$ which helps to track the temporal order of the memory. The read weights are obtained from the linear combination of corresponding weights and the mode vectors $\tilde{\boldsymbol{\pi}}$ normalized by softmax.

$$\boldsymbol{w}_t^r = \tilde{\boldsymbol{\pi}}[0]\boldsymbol{b}_t + \tilde{\boldsymbol{\pi}}[1]\boldsymbol{c}_t^r + \tilde{\boldsymbol{\pi}}[2]\boldsymbol{f}_t \quad (18)$$

These read weights are applied to memory locations to get the final read vectors as follow:

$$\boldsymbol{m}^i = \sum_i^p M_{i,:}\boldsymbol{w}_t^{r,i} \quad (19)$$

Write Operation: Similar to the read heads in read operation, a write head determines where to write by using content-based weighting.

$$\boldsymbol{c}_t^w = softmax(cos(\boldsymbol{M}_{t-1}, \boldsymbol{k}_t^w)\tilde{\boldsymbol{\alpha}}_t^w) \quad (20)$$

We adopted dynamic memory allocation as described in (Graves et al., 2016) to maintain a free list and a usage vector to track the memory freeness. Based on the memory freeness, allocation weights $\boldsymbol{a}_t$ are calculated to indicate where to write, and it is interpolated with content-based weights to get locations for writing. Write gate

Dataset	Total Train / Dev / Test	ADL
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 2: Data statistics of SQuAD, QUASAR-T, and TriviaQA. The Average Document Length (ADL) represents the average number of words. In TriviaQA, ADL was calculated after truncating the documents to the first 1200 words.

$\boldsymbol{g}_t^w$ decides whether to write or not and allocation gate $\boldsymbol{g}_t^a$ determines the degree of interpolation.

$$\boldsymbol{w}_t^w = \boldsymbol{g}_t^w[\boldsymbol{g}_t^a \boldsymbol{a}_t + (1 - \boldsymbol{g}_t^a)\boldsymbol{c}_t^w] \quad (21)$$

After finding the location to write with this weight, write operation on the memory is performed as follows:

$$\boldsymbol{M}_t = \boldsymbol{M}_{t-1} \odot (\boldsymbol{J}_{p,q} - \boldsymbol{w}_t^w \boldsymbol{e}_t^T) + \boldsymbol{w}_t^w \boldsymbol{v}_t^T \quad (22)$$

where $\odot$ is element-wise multiplication and $\boldsymbol{J}_{p,q}$ is p by q matrix of ones.

3.3 Prediction Layer

We feed output vector $\boldsymbol{o}$ from memory controller to prediction layer. First, it goes to bi-directional GRU and is then linearly projected to get probability distribution of start position of answer $Pr(a_s|\boldsymbol{q}, \boldsymbol{d})$. The end position $Pr(a_e|\boldsymbol{q}, \boldsymbol{d})$ is calculated the same way with the hidden states of the start position concatenated as an additional input. These probabilities are also used for model optimization while training in the form of negative log-likelihood probability.

4 Experimental Results

In this section, we present our experimental setup for evaluating the performance of our MAMCN model. We select different datasets based on their average document length to check the effectiveness of external memory on RC task. We compare the performance of our model with all the published results and the baseline memory augmented model. The baseline model is developed by replacing modeling layer in BiDAF (Seo et al., 2017) model with the memory controller from DNC (Graves et al., 2016).

4.1 Data Set

We perform experiments with recently proposed QUASAR-T and TriviaQA datasets to see the performance of our model on the long documents. The QUASAR-T dataset consists of factoid question-answer pairs and a corresponding large background corpus (Callan et al., 2009) to comprehend. In TriviaQA, question-answers pairs are collected from trivia and quiz-league websites. The evidence documents for these QA pairs are collected from Wikipedia articles and Web search results. The average length of the documents in these datasets are much longer than SQuAD and question-documents pairs are collected in a decoupled way, making them more difficult to comprehend. In the case of TriviaQA, we truncate the documents to 1200 words for training and 2000 words for the test. Even these truncated documents are 3 to 5 times longer than SQuAD documents. The average length of original documents in TriviaQA is about 3,000 words, so there is no guarantee that above truncated documents contain the answer for a given question.

We also conduct experiments on SQuAD dataset to show our model can even work well on paragraph-level data. The SQuAD contains a collection of Wikipedia articles and crowd-sourced question-answer pairs. Even though SQuAD dataset pushed existing models to achieve significant performance improvements in solving RC task, the document length does not resemble the length of the real-world document.

The statistics of all these datasets are shown in Table 2. We use official train, dev, and test splits provided in all these datasets for experiments.

4.2 Implementation Details

We develop MAMCN using Tensorflow[1] deep learning framework and Sonnet[2] library. For the word-level embedding, we tokenize the documents using NLTK toolkit (Bird and Loper, 2004) and substitute words with GloVe 6B (Pennington et al., 2014) 300-dimensional word embeddings. We also use 20-dimensional character-level embeddings which are learned during training. The

[1] www.tensorflow.org
[2] https://github.com/deepmind/sonnet

Dataset	Model	Dev set		Test set	
		EM	F1	EM	F1
Short-documents (ADL=221)	**MAMCN**	**64.87**	**68.88**	**68.13**	**70.32**
	BiDAF + DNC	51.18	54.77	54.81	58.24
	BiDAF	45.40	50.90	47.60	52.40
Long-documents (ADL=392)	**MAMCN**	**60.05**	**63.23**	**63.44**	**65.19**
	BiDAF + DNC	48.67	52.25	52.15	54.43
	BiDAF	37.00	42.50	39.50	44.50

Table 3: Performance results on QUASAR-T dataset.

Domain	Model	Full		Verified	
		EM	F1	EM	F1
Web (ADL=631)	**MAMCN**	**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 (ADL=955)	**MAMCN**	**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., 2018)	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 4: Single model results on TriviaQA dataset[3] (Web and Wikipedia). SA: Self-attention, SN: Shared normalization.

hidden size is set to 200 for QUASAR-T and TriviaQA, and 100 for SQuAD. In the memory controller, we use 100 x 36 size memory initialized with zeros, 4 read heads and 1 write head. The optimizer is AdaDelta (Zeiler, 2012) with an initial learning rate of 0.5. We train our model for 12 epochs, and batch size is set to 30. During the training, we keep the exponential moving average of weights with 0.001 decay and use these averages at test time.

4.3 Results

We use Exact Match (EM) and F1 Score as evaluation metrics for all the datasets. EM measures the percentage of the predictions that exactly matches with the corresponding ground truth answers. The F1 score measures the overlap between the predictions and corresponding ground truth answers.

QUASAR-T: The results on QUASAR-T are shown in Table 3. As described in Table 3, the baseline (BiDAF + DNC) results in a reasonable gain, however, our proposed memory controller gives more performance improvement. We achieve 68.13 EM and 70.32 F1 for short documents and 63.44 and 65.19 for long documents which are the current best results.

TriviaQA: We compare proposed model with all the previously suggested approaches as shown in Table 4. We perform the experiments on both Web and Wikipedia domains and report evaluation results for "Full" and "Verified" cases. "Full" is not guaranteed to have all the supporting factors to answer the question, however, it is the entire dataset selected with distant supervision. The "Verified" is a subset of the "Full" dataset cleaned by the human annotators to guarantee the presence of supporting facts to answer.

Our model achieves the state of the art performance over the existing approaches as shown in

[3]https://competitions.codalab.org/competitions/17208

Model	Test set EM	Test set F1	AF	SA
MAMCN + ELMo + DC	**79.69**	**86.73**	**O**	**O**
BiDAF + Self-attention + ELMo (Peters et al., 2018)	78.58	85.83	O	O
MAMCN + ELMo	**77.44**	**85.13**	**O**	**-**
RaSoR + TR + LM (Salant and Berant, 2017)	77.58	84.16	O	-
QANet (Yu et al., 2018)	76.24	84.60	O	O
SAN (Liu et al., 2017b)	76.83	84.40	O	O
FusionNet (Huang et al., 2017b)	75.97	83.90	O	O
RaSoR + TR (Salant and Berant, 2017)	75.79	83.26	O	-
Conducter-net (Liu et al., 2017a)	74.41	82.74	O	O
Reinforced Mnemonic Reader (Hu et al., 2017)	73.20	81.80	O	O
BiDAF + Self-attention (Clark and Gardner, 2017)	72.14	81.05	-	O
MEMEN (Pan et al., 2017)	70.98	80.36	O	-
MAMCN	**70.99**	**79.94**	**-**	**-**
r-net (Wang et al., 2017)	71.30	79.70	-	O
Document Reader (Chen et al., 2017)	70.73	79.35	O	-
FastQAExt (Weissenborn et al., 2017)	70.85	78.86	-	O
⋮	⋮	⋮	⋮	⋮
Human Performance	82.30	91.22		

Table 5: Single model results on SQuAD dataset[4]. The last two columns in tables indicate whether models use additional feature and self-attention. AF: Additional feature augmentation for word embedding, SA: Self-attention, DC: Densely connected embedding block.

Table 4. In the case of full evaluation, we get 66.82 EM, 71.91 F1 for web domain, and 64.41 EM, 69.60 F1 for Wikipedia domain. The proposed model works best in noisy data with distant supervision. In the case of human verified data, performance increases further. We get 81.01 EM and 84.12 F1 for web domain and 70.21 EM and 75.49 F1 for Wikipedia. It is encouraging that these results were obtained without any help of additional feature augmentation, such as utilizing hidden states of pre-trained language model or additional semantic/syntactic features which are commonly used in other models. Also, our model does not need self-attention layer which is prevalently used in previous models.

SQuAD: The results on SQuAD are shown in Table 5. In the longer documents case (QUASAR-T and TriviaQA), MAMCN with external memory performed well because the controller can aggregate information effectively from the long sequences, however, due to the small length of the documents in SQuAD, the existing methods based on the recurrent layers without external memory are also sufficient to achieve reasonable performance. The last two columns in the Table 5 indicate whether each model uses any additional feature augmentation and/or self-attention. All the models with the help of these additional feature augmentation and/or self-attention achieve further performance gain on SQuAD dataset.

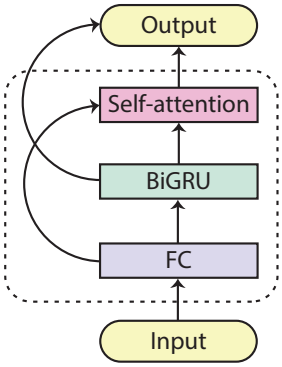

Figure 2: Densely connected embedding block

Our vanilla model (MAMCN) achieved the best performance among the models which are not using additional feature augmentation and/or self-attention layer mechanisms. We also adopted these mechanisms one by one to show that our model is compatible with them. First, we add

[4]https://rajpurkar.github.io/SQuAD-explorer/

No.	Question-Document-Answer triplet
1	**Question** : In which activity are banderillas used ? **Context** : Semi-naked animal rights activists staged a fake - bloody protest outside the European Parliament on Thursday to draw attention to the suffering of bulls during bullfights . Around 30 people taking part in the protest organized by PETA (People for the Ethical Treatment of Animals) lay on the ground with banderillas, the traditional darts used to wound and weaken bulls in the fight, attached to their backs, some spattered with fake blood. Bullfighting - ever popular in Spain - and the European Union's ... **Ground truth**: Bullfight, Bullfights
2	**Question** : What boxer was stripped of his heavyweight boxing titles when he refused his US army induction in April, 1967 ? **Context** : ... This slideshow consists mostly of boxers who have continued to fight on, despite the hindrance of being in their senior years. It also includes two or three boxers who have launched stellar career comebacks from the brink of failure, or exile. George Foreman Al Bello retirement after being defeated by Jimmy Young in Puerto Rico in 1977 shocked the boxing community, the announcement of his return in 1987 sent the sport into raptures. In what ... Muhammad Ali made several comebacks in his career, but the one that stands out has to be the rebuilding of his career after being stripped of his titles for refusing to go to war in Vietnam ... After the U.S army found him to have sub-par reading and spelling skills, he was deemed unsuitable for service in 1966. One year later, however, they revised their criteria, making the champion eligible for national service. Ali refused, on moral grounds, and was consequently stripped of his boxing license and titles permanently. After three years of legal proceedings ... **Ground truth**: Muhammad Ali

Table 6: Examples are from devset of TriviaQA (Web). The solid and dashed rectangular text boxes indicate predictions from the MAMCN and 'BiDAF + Self-attention' models respectively.

ELMo (Peters et al., 2018) which is the weighted sum of hidden layers of language model with regularization as an additional feature to our word embeddings. This helped our model (MAMCN + ELMo) to improve F1 to 85.13 and EM to 77.44 and is the best among the models only with the additional feature augmentation.

Secondly, we add self-attention with dense connections to our model. Recently, Huang et al. (2017a) have shown that adding a connection between each layer to every other layer in convolutional networks improves the performance by a huge margin. Inspired by this, we build densely connected embedding block along with self-attention to increase performance further in the case of SQuAD. The suggested embedding block is shown in Figure 2. Each layer concatenates all the inputs from the previous layers directly connected to it. We replace all the BiGRU units with this embedding block except the controller layer in our model (MAMCN + ELMo + DC). We achieve the state of the art performance, 86.73 F1 and 79.69 EM, with the help of this embedding block.

To show the effectiveness of our method in addressing long-term dependencies, we collected two examples from the devset of TriviaQA, shown in Table 6. Finding an answer in these examples require resolving long-term dependencies. The first example requires understanding dependency present between two sentences while answering. The 'BiDAF + Self-attention' model predicts incorrect answer by shallow matching a sentence which is syntactically close to the question. Our model predicts the correct answer by better combining the information from the two sentences. In the second example, the answer is present remotely in the document, the 'BiDAF + Self-attention' model without external memory face difficulty in comprehending this long document and predicts the wrong answer whereas our model predicts correct answer.

5 Conclusion

We proposed a multi-stage memory augmented neural network model to comprehend long docu-

ments in QA task. The proposed model achieved the state of the art results on the recently released large-scale QA benchmark datasets such as QUASAR-T, TriviaQA, and SQuAD. The results suggest that proposed method is helpful for addressing long-range dependencies in QA task. The future work involves implementing scalable read/write heads to handle larger size external memory to reason over multiple documents.

Acknowledgments

We would like to thank Percy Liang and Pranav Samir Rajpurkar for helping us in the SQuAD submissions.

References

Steven Bird and Edward Loper. 2004. Nltk: the natural language toolkit. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*. Association for Computational Linguistics, page 31.

Jamie Callan, Mark Hoy, Changkuk Yoo, and Le Zhao. 2009. Clueweb09 data set.

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. *arXiv preprint arXiv:1406.1078* .

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.

Minghao Hu, Yuxing Peng, and Xipeng Qiu. 2017. Reinforced mnemonic reader for machine comprehension. *CoRR, abs/1705.02798* .

Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. 2017a. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 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 *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 1601–1611.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *International Conference on Machine Learning*. pages 1378–1387.

Kenton Lee, Shimi Salant, Tom Kwiatkowski, Ankur Parikh, Dipanjan Das, and Jonathan Berant. 2016. Learning recurrent span representations for extractive question answering. *arXiv preprint arXiv:1611.01436* .

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 *Proceedings of the 2014 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. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365* .

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

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 Proceedings of the International Conference on Learning Representations* .

Sainbayar Sukhbaatar, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Advances in neural information processing systems*. pages 2440–2448.

Shuohang Wang and Jing Jiang. 2016. Learning natural language inference with lstm. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 1442–1451.

Shuohang Wang and Jing Jiang. 2017. Machine comprehension using match-lstm and answer pointer. *In Proceedings of the International Conference on Learning Representations* .

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 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 *Proceedings of the 21st 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. 2015a. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698* .

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015b. Memory networks.

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

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

Yang Yu, Wei Zhang, Kazi Hasan, Mo Yu, Bing Xiang, and Bowen Zhou. 2016. End-to-end reading comprehension with dynamic answer chunk ranking. *arXiv preprint arXiv:1610.09996* .

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