Reinforced Mnemonic Reader for Machine Comprehension

Minghao Hu* **Yuxing Peng**
School of Computer Science
National University of Defense Technology
{huminghao09,yxpeng}@nudt.edu.cn

Xipeng Qiu
School of Computer Science
Fudan University
xpqiu@fudan.edu.cn

Abstract

In this paper, we introduce the Reinforced Mnemonic Reader for machine comprehension (MC) task, which aims to answer a query about a given context document. We propose several novel mechanisms that address critical problems in MC that are not adequately solved by previous works, such as enhancing the capacity of encoder, modeling long-term dependencies of contexts, refining the predicted answer span, and directly optimizing the evaluation metric. Extensive experiments on TriviaQA and Stanford Question Answering Dataset (SQuAD) show that our model achieves state-of-the-art results.

Introduction

Teaching machines to comprehend a given context and answer relevant queries is one of the long-term goals of natural language processing and artificial intelligence. Benefiting from the rapid development of deep learning techniques (Goodfellow, Bengio, and Courville 2016) and large-scale MC benchmark datasets (Hermann et al. 2015; Hill et al. 2016; Rajpurkar et al. 2016; Joshi et al. 2017), end-to-end neural networks have achieved promising results on MC tasks (Wang et al. 2016; Xiong, Zhong, and Socher 2017; Seo et al. 2017; Wang et al. 2017).

A common trait of these recent works is the use of an "encoder-interaction-pointer" framework (Weissenborn, Wiese, and Seiffe 2017). In such a framework, word sequences of both query and context are projected into distributed representations and encoded by recurrent neural networks. The attention mechanism (Bahdanau, Cho, and Bengio 2014) is then used to model the complex interaction between the query and the context. Finally a pointer network (Vinyals, Fortunato, and Jaitly 2015) is used to predict the boundary of the answer.

Although previous works have shown the effectiveness of the "encoder-interaction-pointer" framework for MC task, there still exists several limitations:

1. In the encoder layer, previous models, such as Bi-Directional Attention Flow (Seo et al. 2017) and R-Net (Wang et al. 2017), represent each word with word-level embeddings and character-level embeddings, but we hypothesize that many syntactic and linguistic information, such as parts-of-speech tags, named-entity tags, and the query category, are important for MC task.
2. In the interaction layer, previous methods, such as Multi-perspective Matching (Wang et al. 2016) and Dynamic Coattention Networks (Xiong, Zhong, and Socher 2017), only capture the interaction between the query and the context by using either the attention mechanism or the coattention mechanism. However, they fail to fully capture the long-distance contextual interaction between parts of the context, by only using long short-term memory network (LSTM) (Hochreiter and Schmidhuber 1997) or gated recurrent units (GRU) (Chung et al. 2014) for encoding and modeling.
3. In the pointer layer, previous models (Weissenborn, Wiese, and Seiffe 2017; Chen et al. 2017) use the pointer network to calculate two probability distributions for the start and end position of the answer. However, in complex MC tasks, multi-sentence reasoning may be required and there may exist several candidate answers. "One-hop" prediction may fail to fully understand the query and point to the wrong span. Besides, most of previous models use the boundary detecting method proposed by (Wang and Jiang 2017) to train their model, which is equivalent to optimizing the Exact Match (EM) metric (Norouzi et al. 2016). However, this optimization strategy may fail when the answer boundary is fuzzy or too long, such as the answer of the "why" query.

In this paper, we propose the Reinforced Mnemonic Reader for MC tasks, which deals with the above problems in the following aspects. First, we incorporate both syntactic and lexical features with the embedding of each word to enhance the capacity of the encoder. Additional features such as exact match binary feature, POS and NER tags, and the query category help to identify key concepts and entities of texts. Second, we iteratively align the context with the query as well as the context itself, and then efficiently fuse relevant semantic information into each context word for obtaining a fully-aware context representation. Based on such representation, we design a memory-based answer pointing mechanism that allows our model to gradually increase its reading knowledge and continuously refine the answer span. Finally,

*Most of this work was done while the author was at Fudan University.

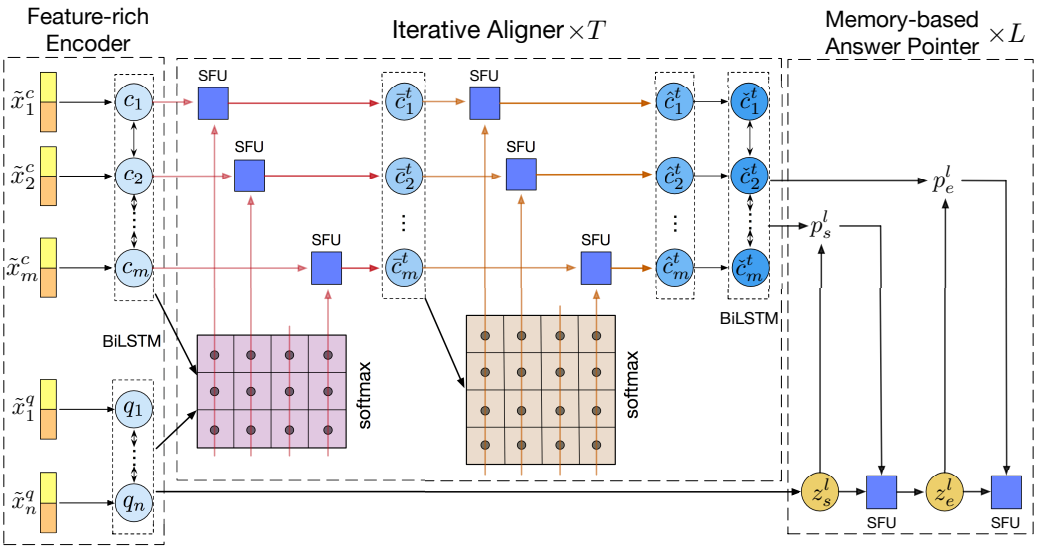

Figure 1: The high-level overview of Reinforced Mnemonic Reader. In the feature-rich encoder, $\{\tilde{x}_i^q\}_{i=1}^n$, $\{\tilde{x}_j^c\}_{j=1}^m$ are embedding matrices of query and context respectively. $\{q_i\}_{i=1}^n$ and $\{c_j\}_{j=1}^m$ are concatenated hidden states of encoding BiLSTM. In the iterative aligner, $\{\bar{c}_j^t\}_{j=1}^m$ and $\{\hat{c}_j^t\}_{j=1}^m$ are the query-aware and self-aware context representation in the t-th hop respectively, while $\{\check{c}_j^t\}_{j=1}^m$ is the fully-aware context representation. In the memory-based answer pointer, z_s^l and z_e^l are memory vectors used for predicting probability distributions of the answer span (p_s^l and p_e^l) in the l-th hop respectively. SFU refers to the *semantic fusion unit*.

to directly optimize both the EM metric and the F1 score, we introduce a new objective function which combines the maximum-likelihood cross-entropy loss with rewards from reinforcement learning. Experiments show that Reinforced Mnemonic Reader obtains state-of-the-art results on TriviaQA and SQuAD datasets.

Model Overview

For machine comprehension (MC) task, a query Q and a context C are given, our task is to predict an answer A, which is constrained as a segment of text of C (Rajpurkar et al. 2016; Joshi et al. 2017). We design the Reinforced Mnemonic Reader to model the probability distribution $p_\theta(A|C,Q)$, where θ is the set of all trainable parameters. Our model consists of three basic modules: *feature-rich encoder*, *iterative aligner* and *memory-based answer pointer*, which are depicted in Figure 1. Below we discuss them in more details.

Feature-rich Encoder

In MC task, the context and the query are both word sequences. The feature-rich encoder is responsible for mapping these word sequences to their corresponding word embeddings, and encoding these embeddings for further processing. We utilize both word-level and character-level embedding as well as additional features to enhance the capacity of the encoder, which is described as follows.

Hybrid Embedding Given a query $Q = \{w_i^q\}_{i=1}^n$ and a context $C = \{w_j^c\}_{j=1}^m$, the feature-rich encoder firstly converts each word w to its respective word embedding x_w, where n and m denote the length of query and context respectively. At a more low-level granularity, the encoder also embeds each word w by encoding its character sequence with a bidirectional long short-term memory network (BiLSTM) (Hochreiter and Schmidhuber 1997). The last hidden states are considered as the character-level embedding x_c. Each word embedding x is then represented as the concatenation of character-level embedding and word-level embedding, denoted as $x = [x_w; x_c] \in \mathbb{R}^d$, where d is the total dimension of word-level embedding and character-level embedding.

Additional Features To better identify key concepts and key entities in both the query and the context, we go beyond word embeddings and utilize several additional linguistic features. We first use a simple yet effective binary feature of exact matching (EM), which has been successfully used in MC tasks (Chen et al. 2017). This binary feature indicates whether a word in context can be exactly matched to one query word, and vice versa for a word in query. Moreover, we create additional look-up based embedding matrices for parts-of-speech tags and named-entity tags. For each word in the query and the context, we simply lookup its tag embeddings and concatenate them with the word embedding and the EM feature.

The explicit query category (i.e., *who*, *when*, *where*, and so on) is obviously a high-level abstraction of the expression of queries, providing additional clues for searching the answer. For example, a "when" query pays more attention on

temporal information while a "where" query seeks for spatial information. Following (Zhang et al. 2017), we count the key word frequency of all queries and obtain top-9 query categories: *what*, *how*, *who*, *when*, *which*, *where*, *why*, *be*, *other*. Each query category is then represented by a trainable embedding. For each query, we lookup its query-category embedding and use a feedforward neural network for projection. The result is then added into the corresponding query embedding matrix. After incorporating all of these additional features with original embeddings, we obtain $\{\tilde{x}_i^q\}_{i=1}^n$ for the query and $\{\tilde{x}_j^c\}_{j=1}^m$ for the context.

Encoding In order to model the word sequence under its contextual information, we use another BiLSTM to encode both the context and the query as follows:

$$\begin{aligned} q_i &= \text{BiLSTM}(q_{i-1}, \tilde{x}_i^q), \forall i \in [1, ..., n] \\ c_j &= \text{BiLSTM}(c_{j-1}, \tilde{x}_j^c), \forall j \in [1, ..., m] \end{aligned} \tag{1}$$

where q_i and $c_j \in \mathbb{R}^{2h}$ are concatenated hidden states of BiLSTM for the i-th query word and the j-th context word, and h is the dimension of hidden state. Finally we obtain two encoded matrices: $Q' = \{q_i\}_{i=1}^n \in \mathbb{R}^{2h \times n}$ for the query and $C' = \{c_j\}_{j=1}^m \in \mathbb{R}^{2h \times m}$ for the context.

Iterative Aligner

The iterative aligner consists of multiple horizontal hops. In each hop, the module updates the representation of each context word by attending to both the query and the context itself. A novel *semantic fusion unit* is used to perform the updating. Finally, an aggregating operation is taken to further encourage information to flow across the context at the end of each hop.

Interactive Aligning The iterative aligner totally reads the context and the query over T hops. In the t-th hop, the aligner first attends to both the query and the context simultaneously to capture the interaction between them, and generates the query-aware context representation. More specifically, we first compute a coattention matrix $B^t \in \mathbb{R}^{n \times m}$ between the query and the context: $B_{ij}^t = {q_i}^{\mathrm{T}} \cdot \check{c}_j^{t-1}$, where $\check{c}_j^{t-1}$ denotes the representation for the j-th context word in the previous hop and $\check{c}_j^0 = c_j$. The matrix element B_{ij}^t indicates the similarity between the i-th query word and the j-th context word.

For each context word, we intend to find the most relevant query word by computing an attended query vector, and fuse it back to the context word. Let $b_j^t \in \mathbb{R}^n$ denote the normalized attention distribution of query for the j-th context word, and $\tilde{q}_j^t \in \mathbb{R}^{2h}$ denotes the corresponding attended query vector. The computation is defined in Eq. 2:

$$\begin{aligned} b_j^t &= \text{softmax}(B_{:j}^t) \\ \tilde{q}_j^t &= Q' \cdot b_j^t, \forall j \in [1, ..., m] \end{aligned} \tag{2}$$

Hence $\{\tilde{q}_j^t\}_{j=1}^m$ represents attended query vectors for all context words.

To compute the query-aware representation for the j-th context word, we combine the context representation $\check{c}_j^{t-1}$ with the attended query vector $\tilde{q}_j^t$ by an effective heuristics, and feed them into a *semantic fusion unit* as follows:

$$\bar{c}_j^t = \text{SFU}(\check{c}_j^{t-1}, \tilde{q}_j^t, \check{c}_j^{t-1} \circ \tilde{q}_j^t, \check{c}_j^{t-1} - \tilde{q}_j^t) \tag{3}$$

where $\bar{C}^t = \{\bar{c}_j^t\}_{j=1}^m$ denotes the query-aware representation of context, $\circ$ stands for element-wise multiplication and $-$ means the element-wise subtraction.

Semantic Fusion Unit To efficiently fuse attended information into original word, we design a simple semantic fusion unit (SFU). A SFU accepts an input vector r and a set of fusion vectors $\{f_i\}_{i=1}^k$, and generates an output vector o. The dimension of the output vector and all of fusion vectors are the same as the input vector. The output vector is expected to not only retrieve correlative information from fusion vectors, but also retain partly unchanged as the input vector.

SFU consists of two components: *composition* and *gate*. The composition component produces a hidden state $\tilde{r}$ which is a linear interpolation of the input vector and fusion vectors. The gate component generates an update gate g to control the composition degree to which the hidden state is exposed. The output vector is calculated as follows:

$$\begin{aligned} \tilde{r} &= \tanh(W_r([r; f_1; ...; f_k]) + b_r) \\ g &= \sigma(W_g([r; f_1; ...; f_k]) + b_g) \\ o &= g \circ \tilde{r} + (1 - g) \circ r \end{aligned} \tag{4}$$

where W_r and W_g are trainable weight matrices, b_r and b_g are trainable biases, and σ is the sigmoid activation function.

Self Aligning After the interactive aligning, the iterative aligner further aligns the query-aware context representation $\bar{C}^t$ with itself to synthesize contextual information among context words. The insight is that, the limited capability of recurrent neural networks make it difficult to model long-term dependencies of contexts (Bengio, Simard, and Frasconi 1994). Each word is only aware of its surrounding neighbour and has no cues about the entire context. Aligning information between context words allows crucial clues to be fused into the context representation.

Similar to the interactive aligning, we first compute a self-coattention matrix $\tilde{B}^t \in \mathbb{R}^{m \times m}$ for the context:

$$\tilde{B}_{ij}^t = \mathbb{1}(i \neq j) \bar{c}_i^{t\mathrm{T}} \cdot \bar{c}_j^t \tag{5}$$

where B_{ij}^t indicates the similarity between the i-th context word and the j-th context word, and the diagonal of self-coattention matrix is set to be zero in case of the word being aligned with itself. Next, for each context word, we compute an attended context vector $\tilde{c}_j^t \in \mathbb{R}^{2h}$ in a similar way of Eq. 2. Hence, $\{\tilde{c}_j^t\}_{j=1}^m$ contains attended context vectors for all query-aware context words.

Finally, we use the same heuristics as in Eq. 3 to combine the query-aware context representation $\bar{c}_j^t$ with the attended context vectors $\tilde{c}_j^t$ for each context word, and fuse them through another SFU to produce the self-aware context representation $\hat{c}_j^t$. Collecting these representation for all context words yield $\hat{C}^t = \{\hat{c}_j^t\}_{j=1}^m$.

Aggregating At the end of each hop, we utilize another BiLSTM as in Eq. 1 to model the self-aware context representation $\hat{C}^t$ with its contextual interaction. Notice that this step is different from the encoding step in that this step captures the short-term information among context words conditioned on entire query and context. The output is the fully-aware context representation $\check{C}^t = \{\check{c}_j^t\}_{j=1}^m$, which is used for the next hop of alignment or the answer prediction.

Memory-based Answer Pointer

The MC task requires the model to find a sub-phrase of the context to answer the query, which is derived by predicting the start position i and the end position j of the answer phrase A,conditioned on both the context C and the query Q:

$$p_\theta(A|C,Q) = p_s(i|C,Q) \cdot p_e(j,|i,C,Q) \tag{6}$$

Here we denote $p_s(i|C,Q)$ as $p_s(i)$ and $p_e(j,|i,C,Q)$ as $p_e(j)$ for abbreviation.

We propose a memory-based answer pointing module which maintains a memory vector to record necessary reading knowledge for continuously refining the predicted answer span. The memory vector is initialized as a query summary $\vec{q} \in \mathbb{R}^{2h}$ and is allowed to be updated with relevant evidences during the prediction. In our experiment we found that simply using last hidden states of encoded query representations as $\vec{q}$ results in good performance.

The answer pointing module totally consists of L hops. In the l-th hop, the module firstly attends over the fully-aware context representation with the memory vector z_s^l to produce the probability distribution of the start position p_s^l, by using a pointer network (Vinyals, Fortunato, and Jaitly 2015):

$$\begin{aligned} s_i^l &= \text{FN}(\check{c}_i^T, z_s^l, \check{c}_i^T \circ z_s^l) \\ p_s^l(i) &= \text{softmax}(w_s^l s_i^l) \end{aligned} \tag{7}$$

where the abbreviation FN means a feedforward neural network that provides a non-linear mapping of its input. $w_s^l \in \mathbb{R}^h$ is a trainable weight vector.

The normalized probability $p_s^l \in \mathbb{R}^m$ points out the potential start position of the answer, and can be viewed as an attention distribution for aggregating information of the current prediction, yielding an evidence vector $u_s^l \in \mathbb{R}^{2h}$: $u_s^l = \check{C}^T \cdot p_s^l$. The memory vector can retrieve relevant cues from the evidence vector to refine itself. Here we use the SFU for the refinement, which takes the memory vector z_s^l and the evidence vector u_s^l as inputs and outputs the new memory vector z_e^l: $z_e^l = \text{SFU}(z_s^l, u_s^l)$.

The probability distribution of the end position p_e^l is computed similarly as Eq. 7, by using z_e^l instead of z_s^l:

$$\begin{aligned} e_j^l &= \text{FN}(\check{c}_j^T, z_e^l, \check{c}_j^T \circ z_e^l) \\ p_e^l(j) &= \text{softmax}(w_e^l e_j^l) \end{aligned} \tag{8}$$

If the l-th hop is not the last hop, then the normalized probability $p_e^l \in \mathbb{R}^m$ is also used as an attention distribution to generate an evidence vector $u_e^l \in \mathbb{R}^{2h}$, which is further fed with the memory vector z_e^l through another SFU to yield the memory vector z_s^{l+1} for the next-hop prediction.

Training Procedure

In this section, we propose two different ways for training our model. Specifically, we use the standard maximum-likelihood estimation (MLE) for maximizing the sum of log probabilities of the ground-truth answer span, and we propose a reinforcement learning approach which aims at directly optimizing the evaluation metric of MC task.

Supervised Learning with Boundary Detecting The moset widely used training method for MC task is the boundary detecting method (Wang and Jiang 2017), which minimizes the sum of negative log probabilities of the true start and end position based on predicted distributions:

$$J_{MLE}(\theta) = -\sum_{i=1}^{N} \log p_s^L(y_i^s) + \log p_e^L(y_i^e) \tag{9}$$

where y_i^s and y_i^e are the ground-truth start and end position of the i-th example, and N is the number of examples in the dataset.

Minimizing $J_{MLE}(\theta)$ can be viewed as optimizing the Exact Match (EM) metric, since the log-likelihood objective is equivalent to a KL divergence between a delta distribution $\delta(A|A^*)$ and the model distribution $p_\theta(A|C,Q)$, where $\delta(A|A^*) = 1$ at $A = A^*$ and 0 at $A \neq A^*$ (Norouzi et al. 2016), and A^* denotes the ground-truth answer. However, directly determining the exact boundary may be difficult in some situations where the answer boundary is fuzzy or too long. For example, it is quite hard to define the answer boundary of a "why" query, where the answer usually is not a distinguishable entity.

Reinforcement Learning for Machine Comprehension One way to tackle this problem is to directly optimizing the F1 score with reinforcement learning. The F1 score measures the overlap between the predicted answer and the ground-truth answer, serving as a "soft" metric compared to the "hard" EM. Taking the F1 score as reward, we use the REINFORCE algorithm (Williams 1992) to maximize the model's expected reward. For each sampled answer $\hat{A}$, we define the loss as:

$$J_{RL}(\theta) = -\mathbb{E}_{\hat{A} \sim p_\theta(A|C,Q)}[R(\hat{A}, A^*)] \tag{10}$$

where p_θ is the policy to be learned, and $R(\hat{A}, A^*)$ is the reward function for a sampled answer, computed as the F1 score with the ground-truth answer A^*. $\hat{A}$ is obtained by sampling from the predicted probability distribution $p_\theta(A|C,Q)$.

To further stabilize training and prevent the model from overwriting its earlier training, we integrate the maximum-likelihood estimation with the reinforcement learning by using a linear interpolation:

$$J(\theta) = \lambda J_{MLE}(\theta) + (1-\lambda) J_{RL}(\theta) \tag{11}$$

where λ is a scaling hyperparameter tuned in experiments. Minimizing this loss is equivalent to optimize both the EM metric and the F1 score.

		Train	Dev	Test	Avg.L
SQuAD	Contexts	18.8K	2.0K	-	122
	Queries	87K	10K	-	11
TriviaQA Wikipedia	Contexts	110K	14K	13K	495
	Queries	61.8K	7.9K	7.7K	15
TriviaQA Web	Contexts	528K	68K	65K	458
	Queries	76.5K	9.9K	9.5K	13

Table 1: Data statistics for SQuAD and TriviaQA. Both of SQuAD and the Wikipedia domain are evaluated over queries while the Web domain of TriviaQA is evaluated over contexts. Avg.L refers to average length.

Model	Domain	Full		Verified	
		EM	F1	EM	F1
Classifier[1]	Wiki	22.5	26.5	27.2	31.4
BiDAF[2]		40.3	45.9	44.9	50.7
MEMEN[3]		43.2	46.9	49.3	55.8
M-Reader		**46.9**	**52.9**	**54.5**	**59.5**
Classifier[1]	Web	24.0	28.4	30.2	34.7
BiDAF[2]		40.7	47.1	49.5	55.8
MEMEN[3]		44.3	48.3	53.3	57.6
M-Reader		**46.7**	**52.9**	**57.0**	**61.5**

Table 2: The performance of Mnemonic Reader and other competing approaches on TriviaQA test set: Joshi et al.(2017)[1], Seo et al.(2017)[2], and Pan et al.(2017)[3].

Evaluation

Datasets

We mainly focus on two large-scale machine comprehension datasets to train and evaluate our model. One is the SQuAD and the other is the recently released TriviaQA. SQuAD (Rajpurkar et al. 2016) is a machine comprehension dataset, totally containing more than 100K queries manually annotated by crowdsourcing workers on a set of Wikipedia articles.

TriviaQA (Joshi et al. 2017) is a newly available machine comprehension dataset consisting of over 650K context-query-answer triples. The contexts are automatically generated from either Wikipedia or Web search results. Table 1 contains the statistics of both datasets. Notice that the query-answer tuple in the Wikipedia domain may refer to multiple evidence contexts. Therefore we perform the prediction for each context separately and choose the candidate answer with the highest confidence score.

The length of contexts in TriviaQA (average 2895 words) is much more longer than the one in SQuAD (average 122 words). To efficiently train our model on TriviaQA, we first approximate the answer span by finding the first match of answer string in contexts. Then we truncate around the answer sentence in a window size of 8 for the train set and development set, and we truncate the context to the first 800 words for the test set. After the preprocessing, the average length of contexts is rescaled under 500 words.

Model	EM	F1
Logistic Regression Baseline[1]	40.4	51.0
Match-LSTM with Ans-Ptr (Boundary)*[2]	67.9	77.0
FastQAExt [3]	70.9	78.9
Document Reader [4]	70.7	79.4
Ruminating Reader [5]	70.6	79.5
M-Reader+RL	**73.2**	**81.8**
Dynamic Coattention Networks*[6]	71.6	80.4
Multi-Perspective Matching*[7]	73.8	81.3
jNet *[8]	73.0	81.5
BiDAF*[9]	73.7	81.5
SEDT*[10]	74.1	81.7
ReasoNet*[11]	75.0	82.6
MEMEN*[12]	75.4	82.7
R-Net*[13]	76.9	84.0
M-Reader+RL*	**77.7**	**84.9**

Table 3: The performance of Mnemonic Reader and other competing approaches on SQuAD test set: Rajpurkar et al.(2016)[1], Wang & Jiang(2017)[2], Weissenborn et al.(2017)[3], Chen et al.(2017)[4], Gong et al. (2017)[5], Xiong et al.(2017)[6], Wang et al.(2016)[7], Zhang et al.(2017)[8], Seo et al.(2017)[9], Liu et al.(2017)[10], Shen et al.(2016)[11], Pan et al.(2017)[12] and Wang et al.(2017)[13]. $*$ indicates ensemble models.

Experimental Configuration

We evaluate the Mnemonic Reader (M-Reader) and the Reinforced Mnemonic Reader (M-Reader+RL) by running the following experiments on both datasets. We first run maximum-likelihood estimation (MLE) to train M-Reader until convergence by optimizing Eq. 9. We then finetune this model with reinforcement learning (RL) by optimizing Eq. 11, until the F1 score on the development set no longer improves. We use a $\lambda = 0.01$ for the M-Reader+RL during RL training.

We use the Adam optimizer (Kingma and Ba 2014) with an initial learning rate of 0.0008 for MLE training, which is halved whenever meeting a bad iteration. We use the SGD optimizer with a learning rate of 0.001 for RL training. The batch size is set to be 45. A dropout rate (Srivastava et al. 2014) of 0.2 is used to prevent overfitting. Word embeddings are initialized with 100-dimensional Glove vectors (Pennington, Socher, and Manning 2014) and remain fixed during training. Out of vocabulary words are randomly sampled from Gaussian distributions. The sizes of character embedding and character hidden state are 50. The dimension of hidden state is 100. The number of hops is set to be 2 for both the aligner and the answer pointer. We set the max length of context to be 300 for SQuAD and 500 for TriviaQA.

Overall Results

Two metrics, Exact Match (EM) and F1 score, are used to evaluate the performance of the model on both SQuAD and TriviaQA. Table 2 shows the performance comparison on the test set of TriviaQA. M-Reader outperforms competitive baselines such as BiDAF and MEMEN on both the

Model	EM	F1
M-Reader+RL	**72.1**	**81.6**
M-Reader	71.8	81.2
- feature-rich encoder	70.5	80.1
- interactive aligning	65.2	74.3
- self aligning	69.7	78.9
- memory-based answer pointer	70.1	79.8

Table 4: Abalation results on SQuAD dev set.

Hops	Aligner		Answer pointer	
	EM	F1	EM	F1
0	62.4	71.6	70.1	79.8
1	70.7	80.3	71.4	80.6
2	**71.8**	**81.2**	**71.8**	**81.2**
3	71.5	81.1	71.3	80.9
4	71.7	80.9	71.2	80.6

Table 5: Performance of M-Reader across different number of hops on SQuAD dev set. Note that for the 0-hop case, the iterative aligner is discarded while the memory-based answer pointer is replaced by Pointer Network (Vinyals, Fortunato, and Jaitly 2015).

Wikipedia domain and the Web domain.

Table 3 shows the performance comparison of our models and other competing models on the test set of SQuAD. M-Reader+RL achieves an EM score of 73.2% and a F1 score of 81.8%. Since SQuAD is a very competitive machine comprehension benchmark, we also build an ensemble model which consists of 16 single models with the same architecture but initialized with different parameters. The answer with the highest score is chosen among all models for each query. Our ensemble model improves the EM score to 77.7%, and the F1 score to 84.9%.

Ablation Results

To evaluate the individual contribution of each component of our model, we run an ablation study on the the SQuAD development set, which is shown in Table 4. The M-Reader+RL obtain the highest performance on both metrics, demonstrating the usefulness of RL training. Ablating addtional features of the encoder results a performance drop of nearly 1.2%. The interactive aligning is most critical to performance as ablating it results a drop of nearly 7% on both metrics. The self aligning accounts for about 2.3% of performance degradation on F1 score, which clearly shows the effectiveness of aligning context words against themselves. Finally, we replace the memory-based answer pointer with the standard pointer network. The result shows that memory-based answer pointer outperforms the pointer network by nearly 1.5% on both metrics.

Table 5 shows the performance under different number of hops. We set the default number of hops as 2 in both the iterative aligner and the memory-based answer pointer, and change the number separately to compare the performance. As we can see, both metrics increase sharply as the number of hops enlarges to the default value. The model with 2 hops achieves the best performance. The larger number of hops potentially result in overfitting on the training set, therefore harming the performance.

Visualization

We provide a qualitative inspection of our model on SQuAD development set. We visualize the attention matrices as well as the estimated probability distributions in Figure 2, which are all extracted from the last hop of the aligner and the answer pointer.

Left subfigure shows the coattention matrix, in which several phrases in the query have been successfully aligned with the corresponding context phrases. Middel subfigure presents the self-coattention matrix. We can see that this matrix is symmetric since the context is aligned with itself, and the diagonal weight is low because the word is not allowed to attend to itself. Besides, several semantically similar phrases have been successfully aligned with each other. One of the most substantial aligned pairs is "Denver Broncos" and "Carolina Panthers", both of which are highly related candidate answers. Right figure shows the estimated distributions of answer spans. In the first hop, the model is not confident in choosing "Denver Broncos" since the score of "Carolina Panthers" is also high. But in the subsequent hop, the model adjusts the answer span by lowering the probability of "Carolina Panthers". This demonstrates that the model is capable of gradually locating the correct span by incorporating knowledge of candidate answers with the query memory.

Related Work

Reading comprehension. The significant advance on reading comprehension has largely benefited from the availability of large-scale datasets. Large cloze-style datasets such as CNN/DailyMail (Hermann et al. 2015) and Childrens Book Test (Hill et al. 2016) were first released, make it possible to solve MC tasks with deep neural architectures. The SQuAD (Rajpurkar et al. 2016) and the TriviaQA (Joshi et al. 2017) are more recently released datasets, which take a segment of text instead of a single entity as the answer, and contain substantial syntactic and lexical variability between the query and the context.

Attention mechanism. The coattention mechanism (Xiong, Zhong, and Socher 2017; Seo et al. 2017) has been widely used in end-to-end neural networks for machine comprehension (MC). Unlike the original attention mechanism (Bahdanau, Cho, and Bengio 2014) that uses a summary vector of the query to attend to the context (Hermann et al. 2015), the coattention is computed as an alignment matrix corresponding to all pairs of context words and query words, which can model complex interaction between the query and the context (Cui et al. 2016; Seo et al. 2017; Xiong, Zhong, and Socher 2017).

Self-attention is an attention mechanism aiming at aligning the sequence with itself, which has been successfully used in a variety of tasks inculding textual entailment

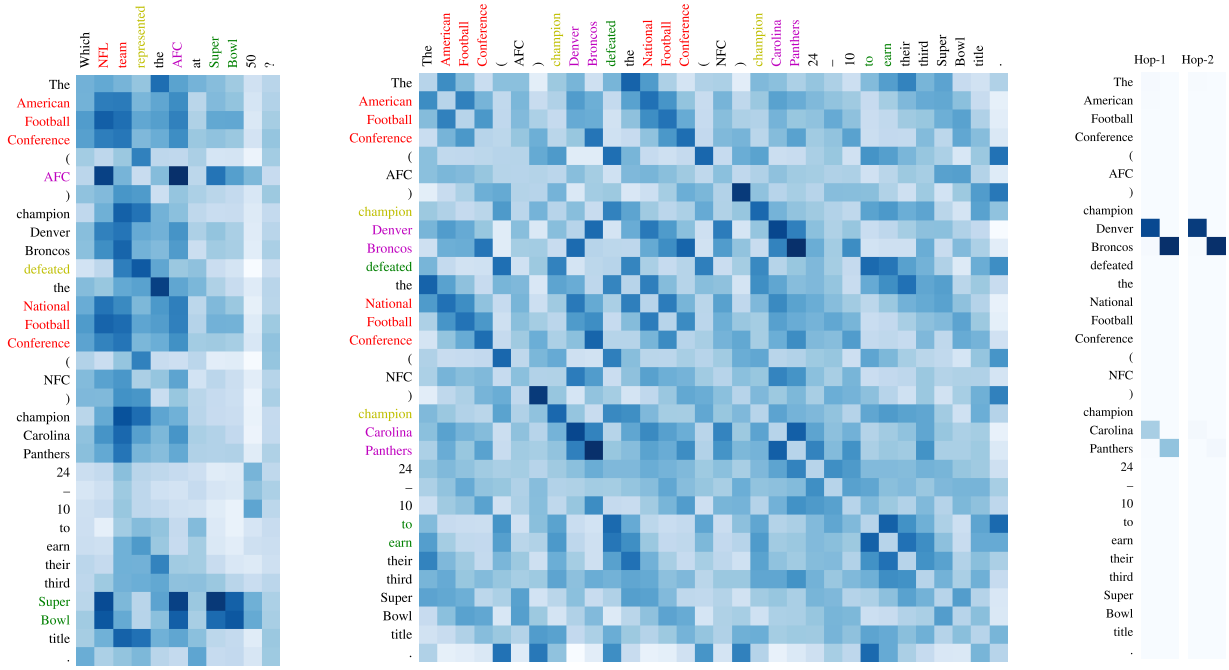

Figure 2: A visualized example of the attention mechanism and the memory-based answer pointing. Left: The coattention matrix (each row is a context word, each column is a query word). Middel: The self-coattention matrix (both rows and columns are context words). Right: The estimated probability distributions of answer spans in two hops. Different colors represent different aligned text pairs. The ground-truth answer is "Denver Broncos".

(Cheng, Dong, and Lapata 2016), neural machine translation (Vaswani et al. 2017) and sentence embedding (Lin et al. 2017). In MC, R-Net (Wang et al. 2017) utilizes self-attention to refine the representation of the context, while Reinforced Mnemonic Reader uses iterative self-coattention to model the long-term dependencies of the context. Besides, Jia et al. (2017) shows that our model outperforms previous models by about 6 F1 points under adversarial attacks, demonstrating the effectiveness of the self-coattention mechanism.

Reasoning mechanism. Inspired by the phenomenon that human increase their understanding by reread the context and the query, multi-hop reasoning models have been proposed for MC tasks (Shen et al. 2016; Xiong, Zhong, and Socher 2017). These models typically maintains a memory state which incorporates the current information of reasoning with the previous information in the memory, by following the framework of Memory Networks (Sukhbaatar et al. 2015). ReasoNet (Shen et al. 2016) utilizes reinforcement learning to dynamically determine when to stop reading. In contrast to their model, Reinforced Mnemonic Reader contains a memory-based answer pointer which is able to continuously refine the answer span.

Reinforcement learning. Reinforcement learning has been successfully used to solve a wide variety of problems in the field of NLP, including abstractive summarization (Paulus, Xiong, and Socher 2017), question generation (Yuan et al. 2017) and dialogue system (Li et al. 2016). The basic idea is to consider the evaluation metrics (like BLEU or ROUGE) which are not differentiable as the reward, and apply REINFORCE algorithm (Williams 1992) to maximize the expected reward. Previous models for MC only utilize maximum-likelihood estimation to optimize the EM metric, and they may fail when the answer span is too long or fuzzy. However, Reinforced Mnemonic Reader uses reinforcement learning to directly optimize the F1 score.

Conclusion

In this paper, we propose the Reinforced Mnemonic Reader, an enhanced attention reader with mnemonic information such as syntactic and lexical features, information of interactive alignment and self alignment, and evidence-augmented query memory. We further combine maximum-likelihood estimation with reinforcement learning for directly optimizing the evaluation metrics. Experiments on TriviaQA and SQuAD showed that these mnemonic information and the new training strategy lead to significant performance improvements. For future work, we will introduce more useful mnemonic information, such as knowledge or common sense, to further augment the attention reader.

Acknowledgments

We thank Pranav Rajpurkar for help in SQuAD submissions and Mandar Joshi for help in TriviaQA submissions.

References

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Bengio, Y.; Simard, P.; and Frasconi, P. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks* 5(2):157–166.

Chen, D.; Fisch, A.; Weston, J.; and Bordes, A. 2017. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*.

Cheng, J.; Dong, L.; and Lapata, M. 2016. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*.

Chung, J.; Gulcehre, C.; Cho, K.; and Bengio, Y. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *Proceedings of NIPS*.

Cui, Y.; Chen, Z.; Wei, S.; Wang, S.; Liu, T.; and Hu, G. 2016. Attention-over-attention neural networks for reading comprehension. *arXiv preprint arXiv:1607.04423*.

Gong, Y., and Bowman, S. R. 2017. Ruminating reader: Reasoning with gated multi-hop attention. *arXiv preprint arXiv:1704.07415*.

Goodfellow, I.; Bengio, Y.; and Courville, A. 2016. *Deep learning*. MIT Press.

Hermann, K. M.; Kocisky, T.; Grefenstette, E.; Espeholt, L.; Kay, W.; Suleyman, M.; ; and Blunsom, P. 2015. Teaching machines to read and comprehend. In *Proceedings of NIPS*.

Hill, F.; Bordes, A.; Chopra, S.; and Weston, J. 2016. The goldilocks principle: Reading childrens books with explicit memory representations. In *Proceedings of ICLR*.

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

Jia, R., and Liang, P. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of EMNLP*.

Joshi, M.; Choi, E.; Weld, D. S.; and Zettlemoyer, L. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of ACL*.

Kingma, D. P., and Ba, L. J. 2014. Adam: A method for stochastic optimization. In *CoRR, abs/1412.6980*.

Li, J.; Monroe, W.; Ritter, A.; Galley, M.; Gao, J.; and Jurafsky, D. 2016. Deep reinforcement learning for dialogue generation. In *Proceedings of EMNLP*.

Lin, Z.; Feng, M.; dos Santos, C. N.; Yu, M.; Xiang, B.; Zhou, B.; and Bengio, Y. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Liu, R.; Hu, J.; Wei, W.; and Nyberg, E. 2017. Structural embedding of syntactic trees for machine comprehension. *arXiv preprint arXiv:1703.00572*.

Norouzi, M.; Bengio, S.; Chen, Z.; and Schuurmans, D. 2016. Reward augmented maximum likelihood for neural structured prediction. In *Proceedings of NIPS*.

Pan, B.; Li, H.; Zhao, Z.; Cao, B.; Cai, D.; and He, X. 2017. Memen: Multi-layer embedding with memory networks for machine comprehension. *arXiv preprint arXiv:1707.09098*.

Paulus, R.; Xiong, C.; and Socher, R. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*.

Rajpurkar, P.; Zhang, J.; Lopyrev, K.; and Liang, P. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of EMNLP*.

Seo, M.; Kembhavi, A.; Farhadi, A.; and Hajishirzi, H. 2017. Bidirectional attention flow for machine comprehension. In *Proceedings of ICLR*.

Shen, Y.; Huang, P.-S.; Gao, J.; and Chen, W. 2016. Reasonet: Learning to stop reading in machine comprehension. *arXiv preprint arXiv:1609.05284*.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 1929–1958.

Sukhbaatar, S.; Szlam, A.; Weston, J.; and Fergus, R. 2015. End-to-end memory networks. In *Proceedings of NIPS*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; and Jones, L. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Vinyals, O.; Fortunato, M.; and Jaitly, N. 2015. Pointer networks. In *Proceedings of NIPS*.

Wang, S., and Jiang, J. 2017. Machine comprehension using match-lstm and answer pointer. In *Proceedings of ICLR*.

Wang, Z.; Mi, H.; Hamza, W.; and Florian, R. 2016. Multi-perspective context matching for machine comprehension. *arXiv preprint arXiv:1612.04211*.

Wang, W.; Yang, N.; Wei, F.; Chang, B.; and Zhou, M. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of ACL*.

Weissenborn, D.; Wiese, G.; and Seiffe, L. 2017. Fastqa: A simple and efficient neural architecture for question answering. *arXiv preprint arXiv:1703.04816*.

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

Xiong, C.; Zhong, V.; and Socher, R. 2017. Dynamic coattention networks for question answering. In *Proceedings of ICLR*.

Yuan, X.; Wang, T.; Gulcehre, C.; Sordoni, A.; and Trischler, A. 2017. Machine comprehension by text-to-text neural question generation. *arXiv preprint arXiv:1705.02012*.

Zhang, J.; Zhu, X.; Chen, Q.; Dai, L.; Wei, S.; and Jiang, H. 2017. Exploring question understanding and adaptation in neural-network-based question answering. *arXiv preprint arXiv:1703.04617*.