Stacked Attention Recurrent Relational Networks for Question Answering

Pei Liu*, Chao Xiang, Dan Jia, Xiaoyu Zhao, Weiye Meng and Jianxiu Wang
China Telecom Corporation Limited Beijing Research Institute, Beijing, China

*Corresponding author e-mail: liupei11@chinatelecom.cn

Abstract. Relational reasoning is ability to reason about entities and their interactions, which is not applicable for many deep neural networks. Recurrent relational networks, introduced by Palm et al. (2017), increase the complexity of the reasoning tasks they can address [1]. In this paper, we introduce Stacked Attention Recurrent Relational Networks (SARRN) to answer natural language questions from facts, which fundamentally hinge on multiple steps of relational reasoning and improve the ability of reasoning. Our model is a stacked attention model that use recurrent attention to focus on fine-grained parts of the documents. We apply our model to the BaBi tasks, which have a set of proxy tasks that evaluate reading comprehension via question answering. Our model solve 19/20 tasks and the experimental results on the test sets tasks show that our model yields great improvement. We also use qualitative analysis to show the result intuitively.

1. Introduction

Text understanding and reasoning is one of the long-term goals for artificial intelligence research, particularly for reading comprehension. A model with the ability to reason about objects and their interactions is required for the development of question answering (QA) research.

The task of question answering is automatically answering a natural language question by understanding unstructured collection of natural language documents. For the ability of reasoning, many researches introduce memory networks [2, 3, 4] and relational networks (RNs) [1, 5]. Memory networks use explicit storage to make multiple computational steps. Relational networks are a plug-and-play neural network module focusing on flexible relational reasoning, which can be implemented in many tasks. The capacity to compute relations and capture the core common properties of relational reasoning is built into RNs architecture. To implement multi-steps relational reasoning, Santoro et al. (2017) proposed Relational Networks (RNs), which can complete more tasks required more steps of computation.

Our model can be considered a continuous form of the recurrent relational networks implemented in it, which add many kinds of attention mechanisms and simplify the benchmark model. Both of the models have the multiple layers of architecture to send messages between the objects. Our model offers following improvements to the benchmark model. First, it explicitly uses iteration attention to focus on fine-grained parts of objects and reason multiple times to infer the answer progressively. The benchmark model only uses a multi-layer perceptron (MLP) to implicitly learn what kind of messages to send. Second, instead of bi-directional Long Short-Term Memory (Bi-LSTM), we use bi-directional Gated
Recurrent Unit (Bi-GRU) to encode the sentence, which can significantly simplify the model. Third, our model reduces the number of layers of MLP. The model with stacked attention is highly complex and easy to overfitting, so our simplified model gets a better result.

The remainder of the paper is organized as follows. Section 2 details related work of reasoning and attention mechanism on question answering. Section 3 defines the model structure that we employ, which contains the feature extraction and stacked attention recurrent relational networks and answer prediction. Section 4 describes the qualitative and quantitative analysis we experimented. Section 5 makes a conclusion.

2. Related work
Recently, the proposed deep neural networks for QA task have accelerated the progress of QA system. In this work we propose stacked attention recurrent relational networks and there are two works which we are related to.

2.1. Reasoning on question answering
Reasoning is an indispensable ability for understanding documents and solving complex questions. There are many QA researches concentrated on reasoning. To improve the ability of reasoning, many models perform inference over the external memories [2, 3, 4, 6, 7]. Weston et al. (2015) proposed BaBi tasks measuring understanding of question answering systems in induction, deduction and many more [8]. The benchmark model, Memory Network (MemNNs), reason with inference components combined with a long-term memory component [2, 8]. However, the model in that paper required supervision at each layer of the network and was not easy to train via backpropagation. Sukhbaatar et al. (2015) introduced End-to-End Memory Network (MemN2N) to train end-to-end from input-output pairs and so is applicable to more tasks. For further improvement, Rae et al. (2016) present a differentiable memory access scheme, Sparse Access Memory (SAM) [9]. The model can alleviate the weakness of scaling poorly in both space and time as the amount of memory grows. Apart from external memory, many researches introduce relational networks for reasoning [1, 5, 10]. Battaglia et al. (2016) introduced Interaction Network [10]. This model can reason about how entities in complex systems interact, inferences about the properties of the system and supporting dynamical predictions. Then, Relational Networks (RNs) was proposed by Santoro et al. (2017), which is a simple plug-and-play module to capture the main common properties of relational reasoning [5]. To complete the tasks required magnitude steps of relation reasoning, Palm et al. (2017) introduced Recurrent Relational Networks, a learned message passing algorithm [1].

2.2. Attention based models
Recently, attention mechanisms are of great important in natural networks. Many works have been done to show the effect of attention mechanisms [11, 12]. In QA models with attention mechanisms, the representation of document is always built with attention from the representation of question which is uni-directional attention mechanism. Wang et al. (2016) use uni-directional attention mechanism in its model, adjusting each word-embedding vector in the document by multiplying a relevancy weight computed against the question [3]. To get a better performance, many QA systems start to use bi-directional attention mechanism in their model, which provide complimentary information to both of documents and the questions. Seo et al. (2017) proposed the Bi-Directional Attention Flow (BIDAF) network with the use of bi-directional attention flow mechanism which obtains the attentions and the attended vectors in both directions of document-to-question and question-to-document [13]. There are many variants of attention mechanism. Self-attention, also called intra-attention is related to different positions of a single document in order to concentrate on the core position in the document. Self-attention has been used successfully in [14], which help to reduce the total computational complexity and improve path length between long-range dependencies in the network. Vaswani et al. (2017) also proposed multi-head attention [14]. They found it beneficial to allow the network to jointly attend to
information from different representation subspaces at different positions. Our model use different kinds of attention mechanism to get a better performance.

3. Model

In this section, we propose a Stacked Attention Recurrent Relational Networks (SARRN) to estimate probability distribution $P$ upon all of the answers to predict the answer of the question. Figure 1 shows the architecture of our model. Here the input of our model is a set of facts and a corresponding question which are successively passed through embedding layer, stacked attention recurrent relational networks and output layer to get an answer as the output of our model. Besides, to reduce the computational complexity, we use self-attention in stacked attention recurrent relational networks, which can simplify the model and get comparable results.

![Figure 1. Architecture of Stacked Attention Recurrent Relational Networks (SARRN). Our model is stacked to multiple layers and is set to be 2 layers in this architecture.](image)

3.1. Embedding layer

The goal of embedding layer is to represent a set of facts $s_1, s_2, ..., s_m$ , whose number is $m$ , by feature vectors $(f_1, f_2, ..., f_m)(f_i \in \mathbb{R}^e)$ and represent the question $u \in \mathbb{R}^w$ with a feature vector $q \in \mathbb{R}^e$ . The $s_1, s_2, ..., s_m$ is a discrete set where $s_i \in \mathbb{R}^w$ represent the $i$-th fact and $w$ is the maximum number of words of the facts. Each of the $s_i, u$ contains $w$ symbols coming from a dictionary which indexes every word in the dataset with a unique number. If the number of words in a sentence is less than $w$, $s_i$ and $q$ will be padded with symbol 0.

When embedding the question, we use pre-trained word vectors, GloVe[15], to represent every symbol in $u$ with an e-dimensional continuous vector and obtain an intermediate matrix $q' \in \mathbb{R}^{wx_e}$ for the question. By this method for learning vector space representations of words, we can capture fine-
grained semantic and syntactic regularities. Next, we use bi-directional Gated Recurrent Unit (Bi-GRU) with $e$ hidden units to encode the question to utilize contextual cues from surrounding words to refine the embedding of the words. We sum the outputs of the two GRU together by which we get a matrix (of size $w \times e$). Then, in order to convert the matrix into a vector, elements of the matrix were summed in column. In this way, we convert the question $u$ into a question vector $q \in \mathbb{R}^w$. We extract features of every sentence in facts with the same way using the question and convert the set of facts $s_1, s_2, ..., s_m$ into vectors $\{d_1, d_2, ..., d_m\}(d_i \in \mathbb{R}^e)$. Besides, we concatenate a position vector $p_i$, which encode the position, with $d_i$ to represent the relative position of every sentence in the facts. The position vector is an one-hot encoding that contain $m$ elements and set the $i$-th element to be “1” of the $i$-th sentence in facts and others to be “0”. By this way, we get fact vectors $\{f_1, f_2, ..., f_m\}(f_i \in \mathbb{R}^e)$ which encoding the semantic and position features of the set of facts. The output of this layer is a question vector $q$ and fact vectors $\{f_1, f_2, ..., f_m\}$.

3.2. Stacked attention recurrent relational networks

This is the core layer for reasoning about entities and their interactions. The entities have different weights to answer, so we use attention mechanism to concern about which entities are relevant to answer. The recurrent relational networks with multiple layers are help for relational reasoning between the entities. We use nodes to represent the entities in this model. The number of nodes is equal to the number of sentence in facts. Given the question vector $q \in \mathbb{R}^w$ and fact vectors $\{f_1, f_2, ..., f_m\}$, we concatenate every $f_i$ with $q$ to be $h_i^t$ and get vectors $\{h_1^t, h_2^t, ..., h_m^t\}$, which help to initialize the hidden state of the nodes. The $t$ represent that $h_i^t$ is the hidden state of nodes in the $t$-th layer of the stacked attention recurrent relational networks.

To reason about the interactions between nodes, every node is updated through combining the messages sent by other nodes. We define the message from node $i$ to node $i$ at layer $t$ to be $m_{ij}^t$:

$$m_{ij}^t = MLP(h_i^{t-1}, h_j^{t-1}) \quad (1)$$

Where MLP is a multi-layer perceptron. This function helps the network to implicitly learn what kind of messages to send.

To explicitly concern about which nodes are relevant to answer, we have different weights $p = \{p_1, p_2, ..., p_m\}(p \in \mathbb{R}^m)$ to nodes. We get the $p_i$ by computing the match between $q$ and each fact vector $f_i$ by taking the inner product followed by a softmax:

$$p_i = \text{Softmax}(p^T f_i) \quad (2)$$

Where $\text{Softmax}(z_i) = e^{z_i}/\sum_j e^{z_j}$. We can also see the $p$ as a probability vector over the nodes.

For the $j$-th node in the $t$-th layer, we sum all the incoming messages $m_{ij}^t$, which is weighted by the probability vector $p$, and get the $m_{j}^t$:

$$m_{j}^t = \sum_i p_i m_{ij}^t \quad (3)$$

When updating the hidden state of node $j$, we use the messages $m_{j}^t$ and the previous node hidden state:

$$h_j^t = \text{LSTM}(h_j^{t-1}, m_{j}^t) \quad (4)$$

Where LSTM is Long-Short Term Memory, a learned neural networks. The dependency on the messages from other nodes helps to learn the interactions between nodes. Concerning about the previous node hidden state $h_j^t$ allows the network to iteratively work towards a solution instead of starting with a blank state at every layer.

The output of stacked attention recurrent relational networks is $\{h_1^t, h_2^t, ..., h_m^t\}$, the hidden state of the last layer where $l$ is the number of layer of the network.

3.3. Output layer

The goal of output layer is to estimate probability distribution on all of the words in the dataset. The probability of maximum value is corresponding to the answer. The sum of the hidden state in the last
layer is then passed through a multi-layer perceptron (MLP) and a softmax to get predicted probability \( P \in \mathbb{R}^v \), where \( v \) is the number of words in the dataset:

\[
P = \text{Softmax}(\text{MLP}(\sum_i h_i))
\]  

(5)

During training, the training loss (to be minimized) is defined as the standard cross-entropy loss between predicted probability \( P \) and the true probability \( P' \). All the parameters in MLP and LSTM are jointly learned by minimizing the training loss when the training is performed using stochastic gradient descent. During testing, the words with the maximum probability is chosen to be the answer, computed by the predicted probability \( P \).

3.4. Attention function

In stacked attention recurrent relation networks, the weight \( p \) is computed by a compatibility attention function of the question with the facts. Additive attention[16] and dot-product attention are the two most commonly used attention functions. Dot-product attention is used in the model we describe above. We also try additive attention in our model and get the weight \( p \) by using a feed-forward network with a single hidden layer:

\[
p_i = V^T \tanh(W^T [p; f_i])
\]  

(6)

Where \( V \) and \( W \) are two learned parameters, \([\cdot]\) represents the two vectors are concatenated and \( \tanh \) is an activation function.

In theory, dot-product attention and additive attention are similar in complexity. However, in practice, the latter can be implemented faster and more efficiently by using matrix multiplication. Both variants perform similar for small dimensionality of the weighted vectors, but additive attention performs better for larger dimensions.

We also try the variant of multi-head attention in our model. We concatenate the weighted vector by different attention functions and then sum them to be the input of next layer:

\[
m^d_{ij} = \sum_i [p^d_i m^d_{ij} ; p^a_i m^a_{ij}]
\]  

(7)

Where \( p^d \) and \( p^a \) are weights respectively computed by the dot-product attention and additive attention.

3.5. Self-attention

When calculating the weight \( p \) used in stacked attention recurrent relational networks, we also use self-attention, which is different from the way used in [14]. Self-attention, also called intra-attention is related to different positions of a set of single facts without the related question in order to concentrate on the core position in facts. We get the weight \( p \in \mathbb{R}^m \) by the computing method of additive attention, which using a feed-forward network with a single hidden layer:

\[
p_i = A^T \tanh(B^T [f_i])
\]  

(8)

Where \( A \) and \( B \) are two learned parameters and \( \tanh \) is an activation function. By this way, we can reduce the total computational complexity per layer and shorter the paths between the combination of facts and the question. The experiment shows that this way can get a competitive result as other attention mechanisms do.

4. Experiment

We conducted our experiments on the BaBi task, a text based QA dataset to evaluate the performance of our model named SARRN.

4.1. Dataset

BaBi tasks are toy tasks from Facebook, introduced by [17]. They considered a prerequisite to agents which can “read” a set of facts and the related question and then reason and understand it to answer the question. They are synthetically general language tasks which are designed to test a wide variety of reasoning abilities, such as deduction, induction, co-referencing, spatial and temporal reasoning, etc. Some questions can be answer by reasoning one of the offered facts in the form of short sentences. Some
questions are preceded by a number of facts. For example, given a question: Where was the apple before the bathroom?, we should reason about the answer: office from three sentence: John went to the office. John journeyed to the bathroom. John discarded the apple there. The tasks have a vocab of about 150 words and answer is typically a single word.

4.2. Model Setup
In embedding layer, we encode the facts and questions using GRU with 32 hidden units and each word in facts and questions is represented by a 32-dimensional vector, the last hidden state of GRU. All the MLP using in this model are 2 ReLU layers or 1 ReLU layer by a linear layer with 128 hidden units for all layers. The position vectors concatenated with the facts are of 40-dimension. Because of the limit of the memory of our server, we use a batch size of 64 in all training, and the maximum gradient norm is 40. The gradient in training is clip to this norm. We use the Adam [18] optimizer, with an initial learning rate of $\eta = 0.01$ and an epsilon value of 2e-4. When training, we use L2 regularization with the rate of 1e-4 in our model.

Because of the random initialization, the result of every training is different. To remedy this, we repeated each training 5 times and picked the best result as the final result.

Table 1. The results of our model and competing approaches on Babi task.

| TASK                        | LSTM  | MemNN | MemN2N | SDNC  | RNs  | RRN  | SARRN(our) |
|-----------------------------|-------|-------|--------|-------|------|------|------------|
| 1 - Single Supporting Fact  | 0.481 | 1.0   | 0.952  | 1.0   | 0.998| 0.998| 1.0        |
| 2 - Two Supporting Facts   | 0.195 | 1.0   | 0.581  | 0.992 | 0.912| 0.982| 0.998      |
| 3 - Three Supporting Facts | 0.211 | 1.0   | 0.322  | 0.985 | 0.811| 0.952| 0.972      |
| 4 - Two Arg. Relations     | 0.551 | 0.732 | 0.931  | 1.0   | 1.0  | 1.0  | 1.0        |
| 5 - Three Arg. Relations   | 0.721 | 0.860 | 0.811  | 0.998 | 0.965| 0.981| 0.997      |
| 6 - Yes/No Questions       | 0.482 | 1.0   | 0.654  | 1.0   | 0.992| 0.999| 1.0        |
| 7 - Counting               | 0.453 | 0.832 | 0.800  | 0.982 | 0.997| 1.0  | 0.999      |
| 8 - Lists/Sets             | 0.550 | 0.942 | 0.651  | 0.995 | 1.0  | 1.0  | 0.995      |
| 9 - Simple Negation        | 0.612 | 1.0   | 0.721  | 1.0   | 1.0  | 1.0  | 1.0        |
| 10 - Indefinite Knowledge  | 0.420 | 0.974 | 0.420  | 0.995 | 0.974| 1.0  | 1.0        |
| 11 - Basic Coreference     | 0.621 | 1.0   | 0.899  | 1.0   | 0.968| 0.999| 0.999      |
| 12 - Conjunction           | 0.625 | 1.0   | 0.912  | 0.984 | 0.971| 0.999| 1.0        |
| 13 - Compound Coref.       | 0.821 | 1.0   | 0.937  | 0.993 | 0.984| 0.999| 1.0        |
| 14 - Time Reasoning        | 0.172 | 1.0   | 0.765  | 0.999 | 0.981| 0.995| 1.0        |
| 15 - Basic Deduction       | 0.217 | 0.775 | 0.965  | 1.0   | 1.0  | 1.0  | 1.0        |
| 16 - Basic Induction       | 0.114 | 1.0   | 0.323  | 0.420 | 0.972| 0.412| 0.482      |
| 17 - Positional Reasoning  | 0.554 | 0.573 | 0.354  | 0.982 | 1.0  | 0.999| 1.0        |
| 18 - Size Reasoning        | 0.565 | 0.546 | 0.900  | 0.992 | 1.0  | 0.996| 0.995      |
| 19 - Path Finding          | 0.082 | 0.155 | 0.110  | 0.964 | 1.0  | 0.999| 1.0        |
| 20 - Agent’s Motivations   | 0.891 | 1.0   | 1.0    | 1.0   | 0.999| 1.0  | 1.0        |
| mean accuracy              | 0.467 | 0.869 | 0.700  | 0.964 | 0.976| 0.966| 0.971      |
| Solved tasks               | 0     | 11    | 3      | 19    | 18   | 19   | 19         |

4.3. Results and Analysis
The results of our model and competing approaches are show in Table 1. All the models are trained jointly. We evaluated our model using test accuracy on our 20 Tasks using 1k training examples. A task is considered solved if a model achieves greater than 95% accuracy. We also use the mean accuracy and the number of solved tasks to evaluate our model in the last two lines. In Table I, the LSTM and MemNNs is the benchmark model proposed by [17]. The LSTMs are popular method for sequence prediction [19] and outperform standard RNNs (Recurrent Neural Networks) for tasks similar to ours. MemN2N can be seen as an extension of MemNNs, which perform inference over the external memories and can be trained end-to-end [3]. SDNC is proposed by [9], which is a differentiable compute and can be adapted for models that maintain temporal associations between memories. RNs is Relation Networks which is a simple plug-and-play module to obtain the ability to reason about the relations between entities and their properties [5]. RRN increases the suite of solvable tasks to those that require an order of magnitude more steps of relational reasoning [1]. As we can see, the number of tasks of our model,
which is training jointly, is achieved state-of-art. Our model and RRN solve 19 tasks and our model improves the accuracy by 7%. Our model gets a competitive result with RNs on mean accuracy and achieve more tasks.

Table 2. The result of all ablation models.

| TASK                                             | No stacked attention (Bi-GRU + 2 ReLU layers of MLP) | Bi-GRU (stacked attention + 2 ReLU layers of MLP) | 2 ReLU layers of MLP (stacked attention + Bi-GRU) | SARRN(our) |
|--------------------------------------------------|-----------------------------------------------------|--------------------------------------------------|--------------------------------------------------|------------|
| 1 - Single Supporting Fact                        | 0.921                                               | 1.0                                              | 1.0                                              | 1.0        |
| 2 - Two Supporting Facts                          | 0.785                                               | 0.989                                            | 0.986                                            | 0.998      |
| 3 - Three Supporting Facts                        | 0.921                                               | 0.952                                            | 0.964                                            | 0.972      |
| 4 - Two Arg. Relations                            | 0.962                                               | 1.0                                              | 0.999                                            | 1.0        |
| 5 - Three Arg. Relations                          | 0.941                                               | 1.0                                              | 1.0                                              | 0.997      |
| 6 - Yes/No Questions                              | 0.952                                               | 1.0                                              | 1.0                                              | 1.0        |
| 7 - Counting                                      | 0.963                                               | 0.999                                            | 1.0                                              | 0.999      |
| 8 - Lists/Sets                                    | 0.986                                               | 0.998                                            | 0.996                                            | 0.995      |
| 9 - Simple Negation                               | 0.932                                               | 1.0                                              | 0.998                                            | 1.0        |
| 10 - Indefinite Knowledge                         | 1.0                                                 | 1.0                                              | 1.0                                              | 1.0        |
| 11 - Basic Coreference                            | 0.964                                               | 1.0                                              | 0.999                                            | 0.999      |
| 12 - Conjunction                                  | 0.984                                               | 1.0                                              | 1.0                                              | 1.0        |
| 13 - Compound Coref.                              | 0.956                                               | 1.0                                              | 1.0                                              | 1.0        |
| 14 - Time Reasoning                               | 0.972                                               | 0.995                                            | 0.998                                            | 1.0        |
| 15 - Basic Deduction                              | 0.985                                               | 1.0                                              | 1.0                                              | 1.0        |
| 16 - Basic Induction                              | 0.351                                               | 0.453                                            | 0.466                                            | 0.482      |
| 17 - Positional Reasoning                         | 0.971                                               | 0.996                                            | 0.992                                            | 0.995      |
| 18 - Size Reasoning                               | 0.935                                               | 0.996                                            | 1.0                                              | 0.995      |
| 19 - Path Finding                                 | 0.942                                               | 0.965                                            | 0.971                                            | 1.0        |
| 20 - Agent’s Motivations                          | 1.0                                                 | 1.0                                              | 1.0                                              | 1.0        |
| Mean accuracy                                     | 0.921                                               | 0.967                                            | 0.968                                            | 0.971      |
| Solved tasks                                      | 12                                                  | 19                                               | 19                                               | 19         |

4.3.1. Model Ablation. We also propose an ablation subtask to evaluate the effectiveness of various improvements in our SARRN model. In introduction of this paper, we have proposed three improvements of our model. We remove one improvement at a time to perform the experiment. Table II shows the result of all ablation models and our full model on BaBi tasks. As can see, when using the stacked attention in our model, replacing the Bi-LSTM with Bi-GRU and reducing the layer of MLP can be an improvement for the model. However, without the stacked attention, this change can reduce the accuracy in a certain extent. It indicates that adding the stacked attention in the model increases the complexity of the model and lead to over-fitting. Therefore, when we simplify the model, it can get a better performance.

4.3.2. Analysis of Attention Function. To show the effect of attention mechanism, we output results of different kinds of attention in our model. Table 3 shows the test accuracy on different mechanism. As we can see, dot-product attention is the most crucial improvement of the three. Though the performance on self-attention is not better than others, this kind of attention can reduce the total computational complexity per layer.

Table 3. Test Accuracy on different kinds of attention.

| Model                 | Mean accuracy |
|-----------------------|---------------|
| Dot-product attention | 0.971         |
| Additive attention    | 0.970         |
| multi-head attention  | 0.968         |
| Self-attention        | 0.968         |
5. Conclusion and future work
In this paper, we proposed Stacked Attention Recurrent Relational Networks (SARRN), which fundamentally hinge on multiple steps of relational reasoning and improve the ability of reasoning. The model explicitly add the attention mechanism and simplify the architecture by replacing the Bi-LSTM with Bi-GRU and reducing the layer of MLP. Experimental result on the test datasets of BaBi tasks shows that our model yields improved results. The ablation analyses show the importance of each improvement in our model. In the future, we can use the reinforcement learning in our model to learn the weights of attention mechanism.

References
[1] Palm R B, Paquet U, Winther O. Recurrent Relational Networks for Complex Relational Reasoning[J]. 2018.
[2] Weston J, Chopra S, Bordes A. Memory Networks[J]. Eprint Arxiv, 2014.
[3] Sukhbaatar S, Szlam A, Weston J, et al. End-To-End Memory Networks[J]. Computer Science, 2015.
[4] Kumar A, Irsoy O, Ondruska P, et al. Ask me anything: dynamic memory networks for natural language processing[J]. 2015:1378-1387.
[5] Santoro A, Raposo D, Barrett D G T, et al. A simple neural network module for relational reasoning[J]. 2017.
[6] Hill F, Bordes A, Chopra S, et al. The Goldilocks Principle: Reading Children's Books with Explicit Memory Representations[J]. Computer Science, 2015.
[7] Miller A, Fisch A, Dodge J, et al. Key-Value Memory Networks for Directly Reading Documents[J]. 2016:1400-1409.
[8] Weston J, Bordes A, Chopra S, et al. Towards AI-Complete Question Answering: A Set of Prerequisite Toy Tasks[J]. Computer Science, 2015.
[9] Rae J W, Hunt J J, Harley T, et al. Scaling Memory-Augmented Neural Networks with Sparse Reads and Writes[J]. 2016.
[10] Battaglia P W, Pascanu R, Lai M, et al. Interaction Networks for Learning about Objects, Relations and Physics[J]. 2016.
[11] Wang B, Liu K, Zhao J. Inner Attention based Recurrent Neural Networks for Answer Selection[C]// Meeting of the Association for Computational Linguistics. 2016:1288-1297.
[12] Zhang W, Zhang C, Liu P, et al. Two-Step Joint Attention Network for Visual Question Answering[C]// International Conference on Big Data Computing and Communications. IEEE Computer Society, 2017:136-143.
[13] Seo M, Kembhavi A, Farhadi A, et al. Bidirectional Attention Flow for Machine Comprehension[J]. 2016.
[14] Vaswani A, Shazeer N, Parmar N, et al. Attention Is All You Need[J]. 2017.
[15] Pennington J, Socher R, Manning C. Glove: Global Vectors for Word Representation[C]// Conference on Empirical Methods in Natural Language Processing. 2014:1532-1543.
[16] Bahdanau D, Cho K, Bengio Y. Neural Machine Translation by Jointly Learning to Align and Translate[J]. Computer Science, 2014.
[17] Weston J, Bordes A, Chopra S, et al. Towards AI-Complete Question Answering: A Set of Prerequisite Toy Tasks[J]. Computer Science, 2015.
[18] Kingma D P, Ba J. Adam: A Method for Stochastic Optimization[J]. Computer Science, 2014.
[19] Sutskever I, Vinyals O, Le Q V. Sequence to Sequence Learning with Neural Networks[J]. 2014, 4:3104-3112.