An Improved Strategy for Machine Comprehension Based upon Dynamic Memory Networks

Yuchen Sun and Min Zhu*
School of Computer Science and Software Engineering, East China Normal University, Shanghai, 200062, China
*yc_sun1002@foxmail.com

Abstract. Recently many complicated reading comprehension models have been proposed. Most of these models are constantly improved from a basic model. However, most of the innovations are at the interaction layer. In this work, we analyse the shortcomings of Dynamic Memory Networks and its extended model. Then we decide to put forward some different opinions that making some improvements in the embedding layer based upon these models. First of all, this method uses pre-training to get character embedding. Our model uses CNN to train the character vector, and then we splicing the character vector and the word vector. Meanwhile, we validate the model with the bAbI dataset. Our results show that this method can improve the interaction between context and question words.

1. Introduction

Machine reading comprehension is actually similar to the problems encountered by human reading comprehension. However, in order to reduce the difficulty of the task, many of the current machine reading comprehension studies exclude world knowledge, use artificially constructed relatively simple data sets and answer some relatively simple questions. Give an article that requires machine understanding and the corresponding questions and answers. More common forms of tasks include synthetic question answering (QA), Cloze-style queries, and multiple-choice questions.

Synthetic question answering is an artificially constructed article formed by several simple facts, and gives corresponding questions. It requires the machine to read and understand the content of the article and make some reasoning to get the correct answer. The correct answer is a keyword or entity in the article. The question answering system is designed to accurately find the information the user is looking for, not the document or fragment that contains the answer. A special form of question answering system, the extractive QA, is the answer to the question directly extracted from the document.

Neural network based approaches have made great strides in text classification and image (Krizhevsky et al., 2012; Socher et al., 2013)[1][2]. However, it was not until these years that progress was made on more complex tasks that required logical reasoning. This success is mostly based on the addition of memory and attention components to complex neural networks. For example, a memory network can reason about facts written in natural language or (subjects, relation, objects) triples. The attention mechanism is a successful component in machine translation (Bahdanau et al., 2015; Luong et al., 2015)[3][4].

Dynamic Memory Network+ (DMN+) is a neural network model with memory components and attention mechanisms (Xiong et al., 2016)[5]. The DMN+ has the not bad results in question
answering that require reasoning based on facts during training, sentiment analysis and part-of-speech tagging.

We improve the accuracy of question and answer by analysing DMN components, especially input modules. We propose a new input module that uses a convolutional neural network to train character sequences. The goal is to allow the model to model the word-by-word interaction between the question and the context.

2. Model

Although the original model baseline can achieve a good result, it still has several shortcomings. First, it does not capture the combination of language, making it more difficult to detect reasonable answers. Second, the semantics of the problem is greatly simplified by the representation of its expected answer type and scalar word problem characteristics.

We propose and compare the embedding-layer: the input representation. The final extended model achieved higher precision on the bAbI-10k dataset (Weston et al., 2015)[6] without supporting the facts. The purpose of this design choice is to improve the accuracy of the model in processing the data set. Embedding are described and computed in detail in the following.

2.1. Input Model

In the following, we denote the hidden dimensionality of the model by \( n \), the question tokens by \( Q = (q_1, \ldots, q_{L_Q}) \), and the context tokens by \( X = (x_1, \ldots, x_{L_X}) \). The task of embedding layer is to represent tokens \( x \) as their corresponding \( n \)-dimensional representations \( x \). Typically, this is typically done by mapping each word \( x \) to its corresponding word embedding \( x^w \) (lookup-embedding) using the embedding matrix \( E \), s.t. \( x^w = Ex \). Then each word is encoded by encoding their corresponding character sequence \( x^c = (c_1, \ldots, c_{L_Q}) \) with \( C \), s.t. \( x^c = C(x_1) \) (char-embedding). In this work, we use a convolutional neural network for \( C \) of filter width 5 with max-pooling over time (Seo et al., 2017)[7]. This method splicing the word vector and the word vector, and finally embedding becomes \( x = [x^w; x^c] \in \mathbb{R}^d \).

The process of embedding character vectors is as follows. The char representation is obtained by convolving the neural network by treating the characters of the word as a sequence. Then concat with the corresponding word embedding. For example, the word "one", \([0, n, c] \) constitutes a sequence. This sequence maps characters to \( n \)-dimensional embedding after the char embedding layer. Enter embedding into the convolutional neural network to get an \( m \)-dimensional char representation. Then it is contacted together with the word embedding corresponding to the word ‘one’. Word embedding is pre-trained, and the char representation is randomly initialized and adjusted to follow the network training. We build simple convolutional character based embedding for words with a fixed filter and size. After the convolution max-pooling over characters is employed for each filter. If word sequences are given, these will be embedded with the newly created embedding.

The structure of the input module is shown in Figure 1. The sentence reader uses a position encoder and the input fusion layer uses a bi-directional GRU. The output of an encoding scheme is entered into the sentence encoding \( f_i \) taking the final embedding \([w_1^i, \ldots, w_{M_i}^i]\), where \( M_i \) is the length of the sentence.

The sentence reader can adapt to any form of coding scheme. So we chose the position code (Sukhbaatar et al., 2015)[8] and compared with their work. GRU and LSTM are also considered, but require more computing resources. And if no pre-processing is done (such as reconstructing the original sentence) then it is easy to over-fitting.

The scheme of position coding is as follows, sentence representation is procedure by \( f_i = \sum_{j=1}^{M} l_j \circ w_j^i \), where \( \circ \) is an element-by-element multiplication. \( l_j \) is a column vector having a structure \( l_{jd} = (1 - j/M) - (d/D)(1 - 2j/M) \), where \( d \) represents an embedded index and \( D \) represents a dimensional embedding.
The input module with a "fusion layer". The sentence reader encodes the sentence, and the information flows between the sentences through the bi-directional GRU.

The input fusion layer takes these input facts. And use bi-directional GRU to ensure the exchange of information between them.

\[ f_i = GRU_{fwd}(f_i, f_{i-1}) \]  
\[ \vec{f_i} = GRU_{bwd}(f_i, f_{i+1}) \]  
\[ \vec{f_i} = f_i + \vec{f_i} \]

Where \( f_i \) represents the input fact of the time step \( i \), the hidden state of the forward GRU of the time step \( i \) is \( f_i \), and the hidden state of the backward GRU of the time step \( i \) is \( \vec{f}_i \). This allows \( \vec{f}_i \) to be influenced by contextual information from future and past facts.

We have studied a number of coding schemes for sentence readers, including but not limited to GRU, LSTM and the position coding schemes (Sukhbaatar et al., 2015)[8]. In order to make the experiment simple and effective, we finally chose the position coding scheme. GRU and LSTM are also taken into account. But more computational resources are needed, and if no pre-processing is done (such as reconstructing the original sentence) then it is easy to over-fitting.

2.2. Semantic Memory Module
Semantic memory consists of two parts, the stored word concepts and the facts about them. We initialize the embedding to a vector as described in the previous section. This module includes a gazetteer or other form of explicit knowledge base, but we do not intend to use them in the work described in this article.

2.3. Question Module
This module maps the problem to the representation, which can then be entered into the input module to query for a specific fact. We have questions that consist of sequences of \( T_Q \) words \( W_t^Q \). We compute a hidden state for each via \( q_t = GRU(L[w_t^Q], q_{t-1}) \), where the GRU and the embedding weight use the same input module. The question vector is finally defined as \( q = q_{T_Q} \).

2.4. Episodic Memory Module
As shown in Figure 1, the story memory module retrieves information from the input facts \( F = \)
\[ \vec{f}_1, \ldots, \vec{f}_N \] provided to it because of the focus on a subset of these facts. We associate a single scalar value (attention gate \( g^t_i \)) with each fact \( \vec{f}_i \) one by one to achieve this concern, during the pass \( t \). This is calculated by allowing the interaction between facts and problem representations and episode memory states.

\[
\begin{align*}
    z^t_i &= [\vec{f}_i ^{\circ} q; \vec{f}_i ^{\circ} m^{t-1}; |\vec{f}_i - m^{t-1}|] \\
    Z^t_i &= W^{(2)} \tanh(W^{(1)} z^t_i + b^{(1)}) + b^{(2)} \\
    g^t_i &= \frac{\exp(Z^t_i)}{\sum_{k=1}^{M_i} \exp(Z^t_k)}
\end{align*}
\]

(4) (5) (6)

Formula above, where \( m^{t-1} \) is the previous episode memory, \( q \) is the original question, \( \vec{f}_i \) is the \( i \)th fact, \( ^{\circ} \) is the element-wise product, \( |\cdot| \) is the element-wise absolute value, and \( ; \) represents concatenation of the vectors.

2.4.1. Attention based GRU. We want to note that the mechanism is sensitive to the location and ordering of input facts \( \vec{F} \) under more complex query conditions. In this case, RNN has a big advantage, except that they cannot be used in Equation (6).

\[
h_i = g^t_i ^{\circ} \vec{h}_i + (1 - g^t_i ^{\circ}) h_{i-1}
\]

(7)

As shown in Figure 2, an important consideration is that \( g^t_i \) is a scalar and its activation function is softmax. In contrast, \( u_i \in \mathbb{R}^{nH} \) is a vector whose activation function is sigmoid. Although no further research was done, the softmax activation in Equation (6) was replaced with sigmoid. This will result in \( g^t_i \in \mathbb{R}^{nH} \) generating contextual vector \( c^t \) which can be used to update the context memory state \( m^t \), we use the final hidden state of GRU based-attention.

2.4.2. Episode Memory Updates. Whenever we pass the attention mechanism, we hope to update the episode memory \( m^{t-1} \) with the newly constructed context vector \( c^t \), which in turn produces \( m^t \). The ReLU layer is adopted for the memory update, and compute the new episode memory state by

\[
m^t = \text{ReLU}(W^t [m^{t-1}; c^t; q] + b)
\]

(8)

Formula above, where \( ; \) is the concatenation operator, \( W^t \in \mathbb{R}^{nH \times nH} \), \( b \in \mathbb{R}^{nH} \), and \( nH \) is the hidden size.

3. Experiment

3.1. Dataset

In order to analyse our proposed model improvement and compare them to the original model architecture, we use the bAbI dataset.
We use the bAbI-10k English dataset (Weston et al., 2015)[6] to assess the accuracy of the improved model. This comprehensive dataset contains 20 different tasks. Each example contains a set of facts, a question, an answer, and a supporting fact that can reason the answer. The dataset has two different sizes, which refers to the number of training samples per task: bAbI-1k and bAbI-10k. The early experiment found that if a smaller bAbI-1k dataset was used (Sukhbaatar et al., 2015)[8], the lowest error rate was on average three times higher than bAbI-10k.

3.2. Analysis of improvement

Most reading comprehension systems are now built in the form of top-down. The structure of a typical neural network model contains an embedding-, encoding-, interaction- and answer layer (Wang and Jiang, 2017; Yu et al., 2017; Xiong et al., 2017; Seo et al., 2017; Yang et al., 2017; Wang et al., 2017)[9][10][11][7][12][13]. That is to say, a very complicated structure was proposed from the beginning. Then use the method called “ablation study” to continuously reduce some module configurations to verify the idea.

So we reproduce the experiment in DMN+ (Xiong et al., 2016)[5]. And some improvements have been made in the embedding-layer. The details of the improvements are shown in the previous section. We found that additional character embedding has a significant improvement on the results.

3.3. Comparison to the baseline using bAbI-10k

![Figure 3. (a) The loss of baseline model, (b) the loss of our improved model, and (c) overlapping (a) and (b).](image-url)
The original DMN architecture is proposed (Kumar et al. 2016)[14]. The original model did not make any improvements. Based on the DMN model, someone made some improvements (Xiong et al. 2016)[5]. DMN+ uses a unique set of weights for each pass and a linear layer with a ReLU activation to compute the memory update. Finally, DMN+ (Extended model), based upon DMN+, make some improvements in the embedding-layer.

As shown in the figure 3, the loss of our model is smaller than the loss of the baseline. This result also verifies from the side that the improved model is better than baseline.

Table 1. Test error rates of various models on the bAbI-10k dataset. The other unmentioned bAbI task achieved 0 error across all models.

| Task  | DMN | DMN+ | DMN+ (Extended model) |
|-------|-----|------|------------------------|
|       | bAbI English 10k dataset |
| QA2   | 36.0 | 6.1  | 5.5                    |
| QA3   | 42.2 | 22.8 | 15.8                   |
| QA5   | 0.1  | 0.5  | 0.4                    |
| QA6   | 35.7 | 0.0  | 0.0                    |
| QA7   | 8.0  | 2.4  | 2.2                    |
| QA8   | 1.6  | 0.0  | 0.0                    |
| QA9   | 3.3  | 0.0  | 0.0                    |
| QA10  | 0.6  | 0.0  | 0.0                    |
| QA14  | 3.6  | 8.8  | 1.1                    |
| QA16  | 55.1 | 45.3 | 42.5                   |
| QA17  | 39.6 | 4.2  | 3.8                    |
| QA18  | 9.3  | 2.1  | 1.9                    |
| QA20  | 1.9  | 0.0  | 0.1                    |
| Mean error | 11.8 | 7.1  | 5.6                    |

As can be seen from the table 1 on mean error, our improved model is better than the DMN model in all tasks, and is superior to the DMN+ model in most tasks. The degradation of results in a few tasks is subject to further research and improvement.

4. Conclusion
We do some analysis of the erroneous results of DMN+. Most of the errors found are due to the lack of understanding of the syntactic structure and the distinction between fine-grained semantics of words with different semantic similarities. Many other errors come from manual annotation preferences.

The extra character embedding has a significant improvement on the results. It is possible to strengthen the interrelationship between words and words. Once the encoder has some knowledge of the question, the encoder can selectively track the problem-related information. It can let the sentence get the context information. It is also very helpful for the next step fusion.

5. Related work
The early model dynamic memory network was proposed, a unified neural network framework that processes input sequences and problems, and then generates relevant answers (Kumar et al., 2015)[14]. Some improvements to its memory and input modules, based upon dynamic memory network (DMN+) (Xiong et al., 2016)[5].

The embedding layer of FastQA connects word vectors to word vectors (Weissenborn et al., 2017)[15]. We have improved the DMN+ with reference to this method. For comparison with baseline, we still use bAbI-10k English (Weston et al., 2015)[6], a synthetic dataset which features 20 different tasks.
References

[1] Krizhevsky A, Sutskever I and Hinton G E 2012 Imagenet classification with deep convolutional neural networks *Advances in Neural Information Processing Systems* pp 1097-1105

[2] Socher R, Perelygin A, Wu J, Chuang J, Manning C D, Ng A and Potts C 2013 Recursive deep models for semantic compositionality over a sentiment treebank *Conference on Empirical Methods in Natural Language Processing* pp 1631-1642

[3] Bahdanau D, Cho K and Bengio Y 2015 Neural machine translation by jointly learning to align and translate *International Conference of Learning Representation*

[4] Luong M T, Pham H and Manning C D 2015 Effective approaches to attention-based neural machine translation *Conference on Empirical Methods in Natural Language Processing*

[5] Xiong C, Merity S and Socher R 2016 Dynamic memory networks for visual and textual question answering *International conference on machine learning* pp 2397-2406

[6] Weston J, Bordes A, Chopra S and Mikolov T 2015 Towards ai-complete question answering: a set of prerequisite toy tasks *arXiv preprint arXiv:1502.05698*

[7] Seo M, Kembhavi A, Farhadi A and Hajishirzi H 2017 Bi-directional attention flow for machine comprehension *International Conference of Learning Representation*

[8] Sukhbaatar S, Weston J and Fergus R 2015 End-to-end memory networks *Advances in neural information processing systems* pp 2440-2448

[9] Wang S and Jiang J 2017 Machine comprehension using match-lstm and answer pointer *International Conference of Learning Representation*

[10] Yu Y, Zhang W, Hasan K, Yu M, Xiang B and Zhou B 2017 End-to-end reading comprehension with dynamic answer chunk ranking *arXiv preprint arXiv:1610.09996*

[11] Xiong C, Zhong V and Socher R 2017 Dynamic coattention networks for question answering *International Conference of Learning Representation*

[12] Yang Z, Duingra B, Yuan Y, Hu J, Cohen W W and Salakhutdinov R 2017 Words or characters? fine-grained gating for reading comprehension *International Conference of Learning Representation*

[13] Wang Z, Mi H, Hamza W and Florian R 2017 Multi-perspective context matching for machine comprehension *arXiv preprint arXiv:1612.04211*

[14] Kumar A, Irsoy O, Ondruska P, Iyyer M, Bradbury J, Gulrajani I and Socher R 2016 Ask me anything: dynamic memory networks for natural language processing *International Conference on Machine Learning* pp 1378-1387

[15] Weissenborn D, Wiese G and Seiffe L 2017 Making neural qa as simple as possible but not simpler *Conference on Computational Natural Language Learning* pp 271-280