End-to-End Memory Networks for Question Answering

Zheye Deng
zdengah@ust.hk

Abstract

In this work, we compare several end-to-end memory networks and explore the application of these models to question answering tasks. We re-implement the current state-of-the-art memory network, the Dynamic Memory Network Plus(DMN+), under a weakly supervised learning approach, aiming to replicate their strong results on the Facebook bAbI dataset. We also propose some extensions on DMN+ and explore the application of DMN+ to a relatively untested multiple choice dataset, Microsoft's MCTest dataset. Ultimately, we basically replicate the state-of-the-art results and even outperform on some tasks in bAbI dataset.

1 Introduction

Question answering (QA) is a complex natural language processing task which requires an understanding of the meaning of a text and the ability to reason over relevant facts.

Currently, several state-of-the-art end-to-end memory networks exist for QA. Specifically, End-to-End Memory Networks, Dynamic Memory Networks and their variants have all been applied to this task all with reasonable degree of success.

We are aiming for a comparison among these end-to-end memory networks and try to implement Dynamic Memory Network Plus and replicate the state-of-the-art results, then explore an application of DMN on MCTest dataset.

2 Dataset

This section discusses about the two datasets and the evaluation used in our project.

2.1 The Facebook bAbI dataset

In the bAbI dataset(Jason W., 2015), a given QA task consists of a set of statements, followed by a question whose answer is typically a single word(in a few tasks, answer are a set of words). The answer is available to the model at training time but must be predicted at test time. The dataset consists of 20 different tasks with various emphases on different forms of reasoning. Here is a sample of the tasks.

```
Mary and Jeff went to the kitchen.
Then Jeff went to the park.
Q: Where is Mary? A: Kitchen
Q: Where is Jeff? A: Park
```

2.2 The Microsoft MCTest dataset

MCTest dataset(Matthew R., 2013) consists of a paragraph story, a few associated questions, and multiple choices answer for each question. While the bAbI is a synthetic dataset, MCTest is an organic dataset, mechanically turned to ensure reliability.

2.3 Evaluation

For the one word answers in the bAbI dataset and the multiple choice answers in the MCTest dataset, we frame the problem as a multi-class classification problem, and use a softmax categorical cross-entropy loss function. We can then evaluate the model by calculating the accuracy on the test set and comparing our results to the benchmarks from various published papers.

3 Previous Work

This section starts with an introduction of the primary elements of End-to-End Memory Network, or `MemN2N`. Then, we review some other end-to-end memory networks with excellent performance.

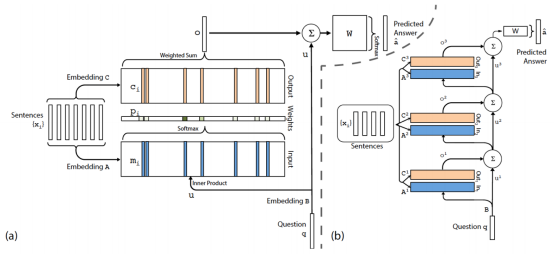

Figure 1: End-to-End Memory Networks

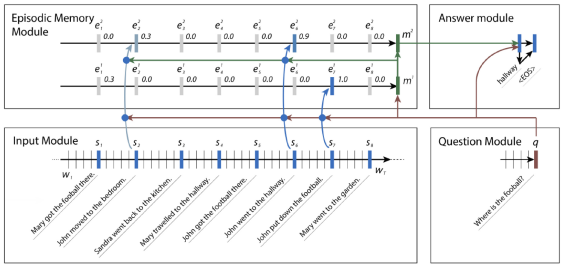

Figure 2: Dynamic Memory Networks

3.1 End-to-End Memory Networks

The MemN2N architecture, introduced by Sainbayar S. (2015), as depicted in Fig. 1, consists of two main components: supporting memories and final answer prediction.

Supporting memories: They are comprised of a set of input and output memory representations with memory cells. The input and output memory cells, denoted by $\mathbf{m}_i$ and $\mathbf{c}_i$, are obtained by transforming the input context$\{x_i\}$ using two embedding matrices. Similarly, the question q is encoded using another embedding matrix, resulting in a question embedding u. The input memories $\{m_i\}$, together with the embedding of the question u, are utilized to determine the relevance of each of the stories in the context, yielding a vector of attention weights

$$p_i = softmax(\mathbf{u}^T \mathbf{m}_i) \tag{1}$$

Subsequently, the response $\mathbf{o}$ from the output memory is constructed by the weighted sum:

$$\mathbf{o} = \sum_i p_i \mathbf{c}_i \tag{2}$$

For multiple layers model, each memory layer is named a hop and the $(k+1)^{\text{th}}$ hop takes as input the output of the k^{th} hop:

$$\mathbf{u}^{k+1} = \mathbf{o}^k + \mathbf{u}^k \tag{3}$$

Answer prediction: The prediction of the answer to the question q, is performed by

$$\tilde{\mathbf{a}} = softmax(\mathbf{W}(\mathbf{o}^K + \mathbf{u}^K) \tag{4}$$

where $\tilde{\mathbf{a}}$ is the predicted answer distribution, $\mathbf{W}$ is a parameter matrix for the model to learn and K the total number of hops.

3.2 Gated End-to-End Memory Networks

Based on the elements behind residual learning and highway neural models, equation 3 can be considered as a form of residuality with $\mathbf{o}^k$ working as the residual function and $\mathbf{u}^k$ the shortcut connection. However, as discussed in Rupesh K. S. (2015), in contrast to the hard-wired skip connection in Residual Networks, one of the advantages of Highway Networks is the adaptive gating mechanism, capable of learning to dynamically control the information flow based on the current input. Therefore, Fei L. (2016) adopt the idea of the adaptive gating mechanism of Highway Networks and integrate it into MemN2N. The resulting model, named Gated End-to-end Memory Networks (GMemN2N) is capable of dynamically conditioning the memory reading operation on the controller state $\mathbf{u}^k$ at each hop. Concretely, they reformulate equation 3 into:

$$\mathbf{T}^k(u^k) = \sigma(\mathbf{W}_T^k \mathbf{u}^k + \mathbf{b}_T^k) \tag{5}$$

$$\mathbf{u}^{k+1} = \mathbf{o}^k \odot \mathbf{T}^k(\mathbf{u}^k) + \mathbf{u}^k \odot (1 - \mathbf{T}^k(\mathbf{u}^k)) \tag{6}$$

where $\mathbf{W}_T^k$ and $\mathbf{b}^k$ are the hop-specific parameter matrix and bias term for the k^{th} hop and $\mathbf{T}^k(x)$ the transform gate for the k^{th} hop.

3.3 Dynamic Memory Networks

The Dynamic Memory Networks (DMN), introduced by Ankit K. (2016), is developed to significantly improve the logical inference process. As depicted in Fig. 2, it consists of four parts: input module, question module, episodic memory module and answer module.

Input Module: The input module encodes raw text inputs from the task into distributed vector representations using a RNN(gated recurrent network). In cases where the input sequence is a

single sentence, the input module outputs the hidden states of the RNN. In cases where the input sequence is a list of sentences, the input module outputs the hidden states at each of the end-of-sentence tokens.

Question Module: Similar to the input module, the question module encodes the question into a vector representations using a RNN.

Episodic Memory Module: This module is what makes the DMN special. It consists of episode e^i and memory m^i, where i is the iteration. During each iteration, an attention mechanism, which is a two-layer feed forward neural network, generate gates g_t for each input c_t based on its similarity to the question q and previous memory m^{i-1}, then use a RNN gated by g_t over c_t to produce as final state a new episode e^i, then a new memory state is produced through $m^i = GRU(e^i, m^{i-1})$, with $m^0 = q$. The update stops at T_M when the attention mechanism picks an end-of-pass over all inputs.

Answer Module: This module generate the answer by GRU, using vector m^{T_M} as initial hidden state, and with query q and previous prediction as input to each step.

3.4 Dynamic Memory Networks+

Although DMN achieved a set of state-of-the-art results, it does have two main problems.

- The GRU only allows sentences to have context from sentences before them, but not after them. This prevents information propagation from future sentences.
- The supporting sentences may be too far away from each other on a word level to allow for these distant sentences to interact through the word level GRU.

Therefore, the appearance of Dynamic Memory Networks+ (DMN+), introduced by Caiming X. (2016), solves these problems successfully.

Basically, there are mainly two improvements made by DMN+: input representation, and attention mechanism.

Input representation: As depicted in Fig. 3, they replace the single GRU in DMN with two different components: a sentence reader which is responsible only for encoding the words into a sentence embedding and a input fusion layer allowing for interactions between sentences. They select positional encoding described in Caiming X. (2016) to for sentence reader and bi-directional GRU for the input fusion layer because it allows information from both past and future sentences to be used.

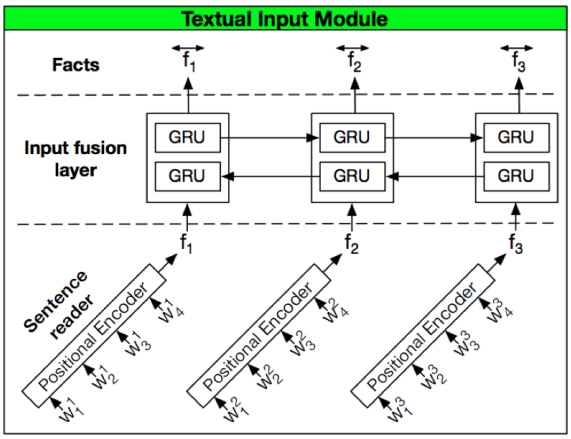

Figure 3: Input fusion layer

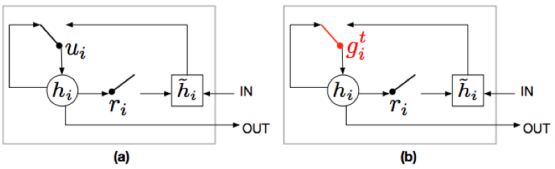

Figure 4: (a) The traditional GRU model, and (b) the proposed attention-based GRU model

Attention Mechanism: They propose a modification to the GRU architecture by embedding information from the attention mechanism, as depicted in Fig. 4. The update u_i decides how much of each dimension of the hidden state to retain and how much should be updated with the transformed input x_i from the current timestep. As u_i is computed using only the current input and the hidden state from previous timesteps, it lacks any knowledge from the question or previous episode memory. By replacing the update gate u_i in the GRU with the output of the attention gate g_i^t, the GRU can now use the attention gate for updating its internal state, which improves performance.

4 Approach

We re-implement Dynamic Memory Networks Plus introduced by XXX using Tensorflow and propose some small modifications in order to improve the performance and to comfort to the input

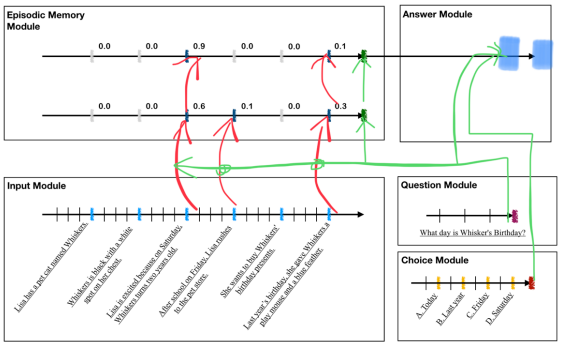

Figure 5: DMN+ with an added Choice Module in order to conform the MCTest data

format of MCTest dataset.

Modification on Answer Module: In the original DMN+, they employ a GRU whose initial state is initialized to the last memory $a_0 = m^{T_M}$. At each timestep, it takes as input the question q, last hidden state a_{t-1}, as well as the previously predicted output y_{t-1}.

$$y_t = softmax(W^{(a)} a_t) \quad (7)$$
$$a_t = GRU([y_{t-1}, q], a_{t-1}) \quad (8)$$

where they concatenate the last generated word and the question vector as the input at each time step.

In this way, the weighted on the last memory is always decreasing, in other words, it may lose some memories. Therefore, at each timestep, we let the model take the last memory m^{T_M} into account as well, which means to reformulate the equation 8 to

$$a_t = GRU([y_{t-1}, q, m^{T_M}], a_{t-1}) \quad (9)$$

Modification for MCTest dataset Because the main feature of the MCTest dataset is that each question has four multiple choice answers, we need a choice module to encode the choices information, as depicted in Fig. 5. Also, when producing memory vector representation in the episodic memory module and making prediction in the answer module, we need to take into account the choices as well. Because the modification is kind of slight, we will not discuss about the details. Note that a simple way to implement this is to concatenate question q, an end-of-question token and four choices o_1, o_2, o_3, o_4 to get a new vector representation $\tilde{q}$ to replace the original q.

Task	DMN+	MyDMN+
2: 2 Supporting Facts	**99.7**	98.3
3: 3 Supporting Facts	**98.9**	81.5
5: 3 Arg. Relations	99.5	**100**
7: Counting	97.6	**98.4**
14: Time Reasoning	99.8	**100**
16: Basic Induction	54.7	**57.4**
17: Positional Reasoning	95.8	**97.5**
18: Size Reasoning	97.9	**99.2**
Mean Accuracy(%)	**97.2**	96.6

Table 1: Test accuracy on the bAbI-10k dataset. DMN+ numbers taken from Caiming et al. (Caiming X., 2016). Tasks where both models achieved 100% accuracy are skipped.

5 Experiments

5.1 My implementation

For all datasets we use either the official train, development, test splits or if no development set was defined, we used 10% of the training set for development. Hyperparameter tunning and model selection (with early stopping) is done on the development set. The DMN+ is trained via backpropagation and Adam optimizer (Diederik P. K., 2014) with a learning rate of 0.002 and batch size of 128. We employ L_2 regularization, and dropout on the context generation, keeping the input with probability $p = 0.9$. Most of the parameters are copied from Caiming X.

We use Tensorflow as the deep learning framework. The hardware we use is the NVIDIA GTX 960 GPU which has 2GB of memory.

We are able to reproduce results similar to those in Caiming X. as shown in Table 1. We outperform the DMN+ on task 5,7,14,16,17,18, and are close to the performance of the DMN+ on task 2, although far away from the DMN+ on task 3, which leads to a slightly lower mean accuracy. For example, as depicted in Fig. 6, we achieve a better accuracy with a gain of 1.7% on accuracy. We believe the reason for the improvement over the DMN+ is the modification on answer module, while the reason for the poor performance on task 2 and 3 is that there may be some training tricks not mentioned in Caiming X..

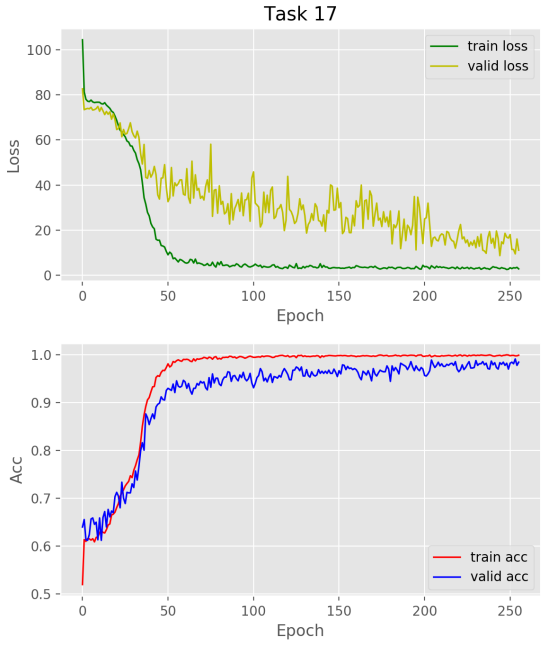

Figure 6: Training and validation accuracy and loss on task 17

5.2 Comparison among end-to-end memory networks

Up to now, we have introduced four end-to-end memory networks. In order to understand how the performance of models changes, we analyze the performance of these four end-to-end memory networks on bAbI dataset. We copy the results of the networks we mentioned on the same dataset using strongly supervised training and add our results, trying to make a complete comparison. It should be noted that MemNN is the baseline method, which is not end-to-end but is a strongly supervised AM+NG+NL Memory Networks approach, proposed in Jason W. (2015). We list the result in Table 2.

5.2.1 (G) MemN2N vs. DMN (+)

It is clear that compared with the MemN2N and GMemN2N, DMN and DMN+ make a significant improvement on overall result, particularly on task 3, 7, 9, 10, 19. We believe it is the episodic memory module that improves the result, because these tasks require the model to iteratively retrieve facts and store them in a representation that slowly incorporates more of the relevant information of the input sequence.

5.2.2 MemN2N vs. GMemN2N

GMemN2N achieves substantial improvements on task 5 and 17, which is a gain of more than 10 in absolute accuracy. We believe the adaptive gating mechanism really works.

5.2.3 DMN vs. DMN+

In general, DMN+ has a better performance. Specifically, DMN+ makes some significant improvements on task 3 and 19, which is a gain of 5% accuracy. Due to the improvements of bidirectional GRU and input fusion layers in the input module and the Attention based GRU in the episodic memory module, DMN+ does create a new set of state-of-the-art results.

5.2.4 More details

To see the differences among these models in detail, we list the methods they use in several aspects in Table 3.

MemN2N vs. DMN Compare with MemN2N, DMN replaces lots of modules by GRU, in order to retrieve more information. Because the methods such as BoW, Linear Regression and Soft Attention have a big disadvantage, which is that they will lose both positional and ordering information. Whilst multiple attention passes can retrieve some of this information, this is inefficient.

DMN vs. DMN+ Compared with DMN, DMN+ replaces the GRU with Bidirectional GRU in the Input Module and replace the GRU with Attention-based GRU in the attention mechanism, in order to allow for distant supporting sentences to have a more direct interaction. Moreover, DMN+ replaces the GRU in memory update session with ReLU reduce the probability of overfitting and get the accuracy improved.

5.3 Training on MCTest

As a kind of very first attempt to use DMN+ for MCTest, we implement the choice module o as described in Section 4. Although the test accuracy generated by DMN+ is better than random guess (25%), it still cannot beat the test accuracy of the baseline model (SW+D)(Matthew R., 2013). The most convincing reason we think is due to the fact that the MCTest dataset is too small to complete the training. However, we can still see that the result of DMN+ is better than the result of DMN on MCTest proposed in Qian L. from Table 4. We

Task	MemNN	MemN2N	GMemN2N	DMN	MyDMN+
1: 1 Supporting Facts	100	100	100	100	**100**
2: 2 Supporting Facts	100	91.7	91.9	98.2	**100**
3: 3 Supporting Facts	100	59.7	61.2	95.2	**100**
4: 2 Arg. Relations	100	97.2	99.6	100	**100**
5: 3 Arg. Relations	98	86.9	99.0	99.3	**100**
6: Yes/No Questions	100	92.4	91.6	100	**100**
7: Counting	85	82.7	82.2	96.9	96.4
8: Lists/sets	91	90.0	87.5	96.5	**99.0**
9: Simple negation	100	86.8	89.3	100	**100**
10: Indefinite knowledge	98	84.9	83.5	97.5	95.3
11: Basic Coreference	100	99.1	100	100	**100**
12: Conjunction	100	99.8	100	100	**100**
13: Compound Coreference	99	99.6	100	98.8	**100**
14: Time Reasoning	100	98.3	98.8	100	**100**
15: Basic Deduction	100	100	100	100	**100**
16: Basic Induction	100	98.7	99.9	99.4	**100**
17: Positional Reasoning	65	49.0	58.3	59.6	61.0
18: Size Reasoning	95	88.9	90.8	95.3	95.0
19: Path finding	36	17.2	11.5	34.5	**40.1**
20: Agent's Motivation	100	100	100	100	**100**
Mean Accuracy(%)	93.3	86.1	87.3	93.6	**94.3**
Failed tasks(Acc. $< 95\%$)	4	11	10	2	**2**

Table 2: Test accuracy on the 20 QA tasks for models using 1k training examples.

	(G)MemN2N	DMN	DMN+
Sentence Encoding	BoW/Positional Encoding+Weighted Sum	GRU	Positional Encoding+Bi-GRU
Attention Computing	Linear Regression	Gating Function	Gating Function+Softmax
Attention Mechanism	Soft Attention	GRU	Attention based GRU
Memory Update	Concatenation	GRU	ReLU
Memory Weights	Tied	Tied	Untied
Answer Module	Softmax	Softmax+GRU	Softmax+GRU

Table 3: A complete comparison among four kinds of end-to-end memory networks we have mentioned.

Method	Accuracy(%)
Baseline(SW+D)(Matthew R.)	66.2
DMN(Qian L.)	37.3
My DMN+	39.2

Table 4: Test accuracy on the MCTest

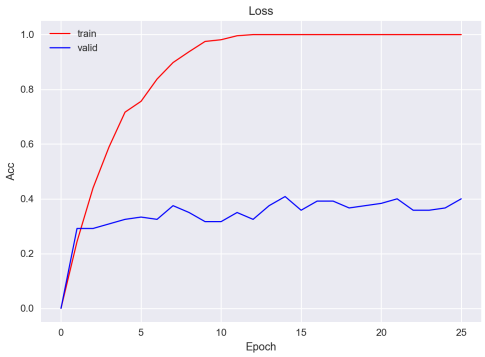

Figure 7: Training and dev accuracy and loss on MCTest

can see how the training accuracy and dev accuracy evolving as iteration increases from Fig.7. It is clear to see it suffers from overfitting.

6 Conclusion and Future work

In this project, we have compared the four end-to-end memory networks clearly and thoroughly. We believe that we have replicated the state-of-the-art techniques on DMN+, and with our proposed improvements, we create a new set of state-of-the-art results. However, there are still some problems remaining to be solved in the future. For example, as the MCTest dataset is too small for decent good training, we need some kind of model which particularly suit for small dataset. And some more complicated episodic memory module is needed for weakly supervised training. Moreover, it must be promising to explore the application of the DMN+ on other NLP tasks like sentiment analysis, POS tagging, etc.

References

Peter O. Mohit I. James B. Ishaan G. Victor Z. Romain P. Richard S. Ankit K. 2016. Ask me anything: Dynamic memory networks for natural language processing .

Stephen M. Richard S. Caiming X. 2016. Dynamic memory networks for visual and textual question answering .

Jimmy B. Diederik P. K. 2014. Adam: A method for stochastic optimization .

Julien P. Fei L. 2016. Gated end-to-end memory networks .

Antoine B. Sumit C. Alexander M. R. Bart M. Armand J. Tomas M. Jason W. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks .

Christopher J.C.B. Erin R. Matthew R. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of test. .

Hongyu X. Qian L. 2016. Dynamic memory network on natural language question-answeing .

Klaus G. Jrgen S. Rupesh K. S. 2015. Highway networks .

Arthur S. Jason W. Rob F. Sainbayar S. 2015. End-to-end memory networks .