Dynamic Memory Network on Natural Language Question-Answering

Qian Lin
Department of Applied Physics
Stanford University
Stanford, CA 94305
linqian@stanford.edu

Hongyu Xiong
Department of Applied Physics
Stanford University
Stanford, CA 94305
hxiong2@stanford.edu

Abstract

Question-Answering (QA) is an important milestone for the research of artificial intelligence (AI). In this work, we explore the application of memory-based neural network model to reading-comprehension type QA tasks. Based on the idea of dynamic network model (DMN) [1], we re-implement both unsupervised and supervised DMN, establishing an baselin on bAbi QA dataset, and applying to a more complex natural language QA dataset - MCTest. During the process, we designed a new Option Module and a modified Answer Module to make DMN suitable for MCTest input, and experiment on supervised gating mechanism to improve its performance.

1 Introduction

Question-Answer (QA) type of problems is one of the most interesting issues in the study of natural language processing (NLP). The breakthrough of QA problems tends to help the development of other research fields. For example, the solution of QA could directly apply to Turing Test research. In addition, advancement in QA study will benefit related researches in Human-Computer Interaction (HCI) and will finally make peoples life more convenient and comfortable.

Recently memory-network based system has been making breakthroughs in a specific type of question-answering: reading comprehension [1-6]. Many of these works use a synthetic dataset, the facebook bAbI [4], which are a set of machine-generated short stories and questions that tests a variety of tasks ranging from single supporting fact to induction to path finding. While such a synthetic dataset has the advantage at separating subtasks in the complicated question-answering problem, and thus better show the capabilities and limitations of the systems, it nevertheless suffer from a few problems common to synthetic datasets. The bAbI dataset has a very small vocabulary size of only 150. Furthermore, the generated stories have far simpler structure than natural language. Thus it is not clear that systems show significant improved performance in the bAbI task can also excel in nature language QA. For example, previous CS224N project [7] shows no improvement on the MCTest [9] using the then state-of-the-art deep learning MemNN [3] compared to the baseline and LSTM.

The newly proposed dynamic memory network (DMN) [1], among its other advantages, has shown improvement in QA accuracy over previous works on the bAbI dataset. We will compare the performance of DMN with LSTM and MemNN on MCTest [9], and try out possible improvement, for example using pretrained GloVe/Word2Vec for better word representation to cope with the increased vocabulary size, and different attention mechanisms.

2 Problem Statement & Dataset

The QA tasks consist of reading a piece of text, called the story or the knowledge base, of variable length (from one sentence to a few paragraphs). Then certain questions based on the story are asked, and the AI agent are expected to produce an answer in the form of either a single word, a natural language sentence, or a most probable choice among a given set of candidates.

The following are the lists of the dataset used in our project. A detailed example for each is given afterward.

bAbI QA: https://research.facebook.com/researchers

MCTest: http://research.microsoft.com/en-us/um/redmond/projects/mctest

2.1 bAbI QA

The bAbI dataset has 20 tasks, each includes 1000 training and test questions. An example story, question and answer are given below. The test accuracy will be used as a sanity check for our LSTM, MemNN and DMN implementations.

Task 3: Three Supporting Facts
John picked up the apple.
John went to the office.
John went to the kitchen.
John dropped the apple.
Where was the apple before the kitchen? A:office

2.2 MCTest

MSTest consist of a paragraph story, a few associated questions, and multiple choices answer for each question. The dataset consist of 660 stories for training and a smaller set for testing. The evaluation metrics is the test accuracy.

The MCTest corpus contains two sets of stories, named MC160 and MC500, and containing 160 and 500 stories respectively. MC160 was gathered first, then some improvements were made before gathering MC500. MC160 and MC500 are randomly divided into train, development and test sets of 70, 30 and 60 stories and 300, 50 and 150 stories respectively.

Each story contains $150 \sim 300$ words, and four multiple choices questions each having four choices. The questions may dependent on one or multiple sentences from the story, and at least half the questions are multiple-dependent.

One sunny day, Martha went on a walk through the park. While walking, she noticed something strange. No one was outside. She was the only person at the park. "How strange, where is everyone?" she thought. Martha looked everywhere. She looked inside the restrooms, under the benches, and even at the top of the slide. She was confused. Usually, she would see her friends playing with each other. She started walking again when one of her friends popped up, surprising her. Her friend asked her, "Why are you outside?" Martha asked what she meant, and explained that she always came out to the park to play. Her friend then looked at her strangely and asked, "Didn't Stephan invite you to his party?" Martha hadn't known that Stephan was holding a party. She was sad that he hadn't invited her. She walked back home, upset.
1: multiple: Who didn't invite Martha to the party?
A) The park
B) No one
C) Martha
*D) Stephan

3 Previous Work

In general, QA systems must perform both retrieval and inference. In detail, the model must store the input information or knowledge, and later it is able to search the memory and retrieve the related information that would help to answer the question posed to the system. This is the retrieval part. When analyzing the question text, there should be a way to process and get the question's representation, e.g. via the trained network parameters of a sequence predictor like RNN or LSTM model, and then connect the question with the memory the system already have. This is the inference part. Both the tasks are critical to the success of a QA system.

In order to deal with Question-answering problems ourselves, we have referred to several previous work for inspirations. In general, we have used MemNN developed by Sainbayar Sukhbaatar et al. [3], DMN developed by Ankit Kumar et al. [1], and adopted ideas from last year CS224D final project by Darshan Kapashi and Pararth Shah [7], and last year CS224N final project by Te-Lin Wu and De-An Huang [8].

The general idea of MemNN is that the memory layer remembers the facts from the story as M. To extract the information relevant to the query, the memory is extracted weighted by its similarity to the query phrase. Then an answer is generated combining the extracted memory O and the query phrase U.

The MemNN method is described based on [3]. The key steps are

- **Input memory representation:** the input set x_i of length s are convert to into memory vectors m_i ($M = AX$) using embedding matrix $A \in \mathbb{R}^{d \times V}$. The query q_i of length q are convert to internal state u_i ($U = AQ$) using the same embedding matrix A. Compute match between query and memory $P = U^T \cdot M \in \mathbb{R}^{q \times s}$.
- **Output memory representation:** The input set x_i is embedded with another matrix C to generate c_i. Sum over c_is weighted by memory match P generates the output vector $O = P(CX)$.
- **Prediction:** The output vector O and internal state U is summed and past through a final linear+softmax layer to predict the output word $a = Softmax(W(O + U))$.

The Dynamic Memory Network (DMN) is developed to significantly improve the logical inference process [1]. The model contains four parts: input module, question module, episodic memory module, and answer module. What makes DMN special is the episodic memory module. The episodic memory module runs in iterative process regarding the question passed in, with each iteration (epoch) the module absorbing inputs and previous memory to see which parts of the memory are more relevant, and then use a RNN to generate the answer accordingly. This relevance is expressed by a value called "attention", calculated by the encoded question hidden state and previous memory hidden states.

The DMN model consists of four models, as shown in Fig.1

- **Input module:** this converts the input set to a sequence of hidden state representation c_t using a RNN (gated recurrent network). For sequence of hidden state consist either of the hidden state of every words in the case of a single sentence input, or the end hidden state of each sentence of a multi-sentence input.
- **Question module:** produce the final hidden state $q = q_{T_Q}$ passing the question through a RNN.
- **Episodic memory module:** consist of episode e^i and memory m^i, where i is the iteration. During each iteration, an attention mechanism generate gates g_t for each input c_t based on its similarity to the query 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; the a new memory state is produce through $m^i = GRU(e^i, m^{i-1})$, with $m^0 = q$. The attention mechanism is a two-layer feed forward neural network. The update stops at T_M when the attention mechanism picks an end-of-pass over all inputs. [1]

[1] In preparing this writeup, we also came across an interested visualization of DMN episodic memory in yerevann.com/dmn-ui and Playground for bAbI tasks

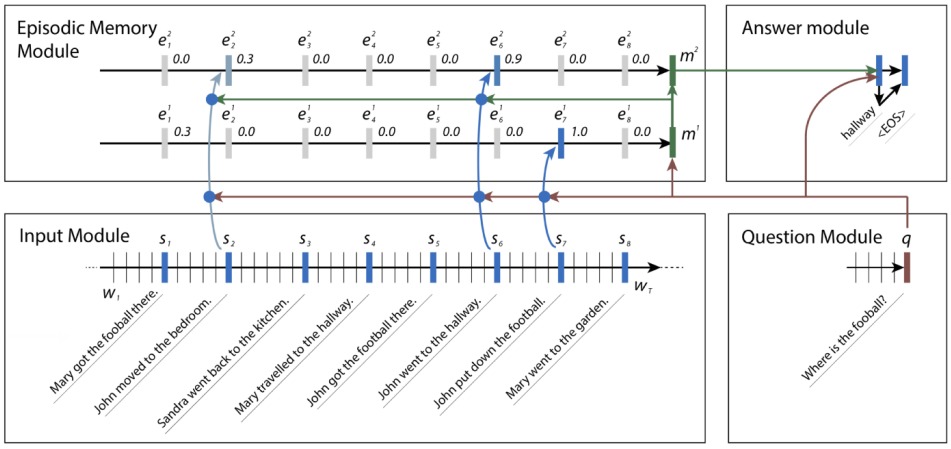


Figure 1: Dynamic memory network from [1].

- **Answer module:** use vector m^{T_M} as initial hidden state, and with query q and previous prediction as input to each step, to generate an answer using GRU.

We also learn valuable experience from last year reports studying on QA problems. Wu and Huang [8] managed to used the GloVe pretrained word vectors to represent the words in the question vocabulary of bAbi QA dataset. We have been confused for quite a while when testing bAbi QA using baseline LSTM model: importing all the word vectors from GloVe, which possess high semantic correlations with each other, does not necessarily yields a better test accuracy on the bAbi QA tasks; in fact, for most of the tasks such as qa1, qa2, and qa15, the test accuracy cannot even reach 20%. Their paper [8] addresses this issue upfront: "In general incorporating the semantic embed- ding trained on outside dataset makes performance worse. This is understandable as some words that should be discriminated are forced to have similar word vector in our model. " And the work done by Kapashi and Shah [7] makes us realize the format difference between MCTest and bAbi QA, motivating us to incorporate a new options module into DMN to cope with it.

4 Approach

The baseline models (LSTM, MemNN, DMN) we used are well described in previous works [1,3]. We re-implement these models using Theano and Keras.

Modification on DMN model in order to suit the input of MCTest dataset, In order to make DMN suitable for MCTest dataset (whose options contain sentences instead of just single words), we designed a new option module, and applied two methods to incorporate it into the original DMN architecture.

- **Option Module** The MCTest dataset has a different format than the bAbi QA dataset. Besides the format of story, the most significant difference, which directly turns down the DMN functionality on MCTest, is the format of options and answer. Unlike each question in the bAbi QA tasks, which offers one single word (for most tasks) as answer, the question in MCTest gives four options for the answer to choose from, with each option not necessarily having one word. In this sense, we have to deal with the options in this question separately, constructing a new module to read the corresponding information. As shown in Fig.2, for each sentence of the four options, we apply a GRU to read words step by step and get the hidden state (Oa, Ob, Oc, Od), and output the o as the representation of options, which put all four hidden states into a matrix.
- **Modified Attention Mechanism** After we have our newly designed option module, how do we incorporate that into the DMN structure so it can function similarly as that on bAbi QA. The first method we have tried is to incorporate the Option Module into the attention mechanism, which means that the attention calculation not only considers the question

hidden state q and memory hidden state m, but also the options hidden state. Specifically speaking, we add several terms inside the $z(c, q, m)$ function to make it $z(c, q, m, o)$.

- **Modified Answer Module** Another approach is that we further change the structure of the Answer Module. When applying DMN on bAbi QA dataset, since the answer in most cases is a single word, the output of answer module could directly compare with it. However, for MCTest the answer is probably a sentence, and so it would not be a good way to compare the raw output directly. Here we add a new matrix U in the answer module to train on, and the hidden state m_a (original output) generate by m and q could be combined with o through a matrix multiplication $m_a^T U o$, which yields a 1×4 vector, the maximum value element indicates the option with largest probability, and we use that option as the answer.

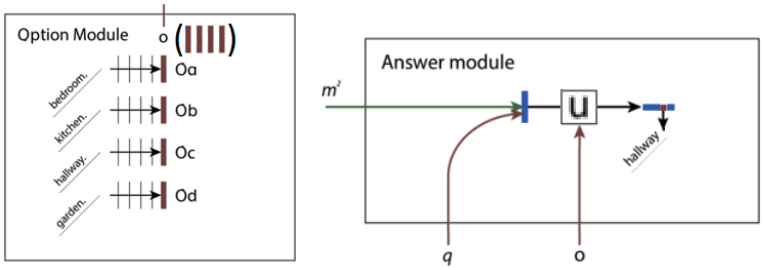


Figure 2: Additional options module processes each option sentence using a GRU and take the final hidden state o as representation of options. The options are used in the modified answer module to predict the choice using a metric similar to cosine-similarity.

Supervised attention training and model transfer MCTest dataset is usually considered difficult for statistical training, due to its small size compare to the complexity of task it presents (large vocabulary, long stories). During training we observe significant overfitting to the training dataset. Therefore we also consider using a model transfer approach to improve our performance.

More specifically, we attempt to import the weights for calculating the attention gate from supervised pre-train model on bAbi. On the original DMN paper [1], it was reported that using the provided relevant sentence information in the bAbi dataset significantly improves the accuracy, and encourages the sparsity of the episodic attention. We perform supervised training on bAbi, then import the trained weight matrix W_1, W_2, b_1, b_2, W_b and retrain other parameters for MCTest.

5 Experiments

5.1 bAbi test

At the very beginning, we have implemented the baseline LSTM, MemNN and unsupervised DMN. As a sanity check, we test on the three tasks qa1, qa2, and qa15 from bAbI QA, to confirm the models function well. The result is listed in Table.1.

Table 1: LSTM, MemNN and DMN on bAbI task 1,2,15. The MemNN and DMN we used is the weakly supervised version. MemNN is a one-hop model.

task	LSTM	MemNN	DMN
1	51%	47%	100%
2	25%	18%	33%
15	23%	48%	53%

As expected, the unsupervised DMN on bAbi do not perform very well for complicated tasks. As shown in the two subplots in Fig. 3, the unsupervised approach yields very diffused gate (attention) and little shifts over episodes. This results in poor incorporation of input information.

Figure 3: Visualization of attentions mechanism at each episode. (Top) Without supervised gate training, attention shift over episodes are not apparent. (Bottom) With supervised gate training as described in [1], the gate is more sparse and show significant shift over time.

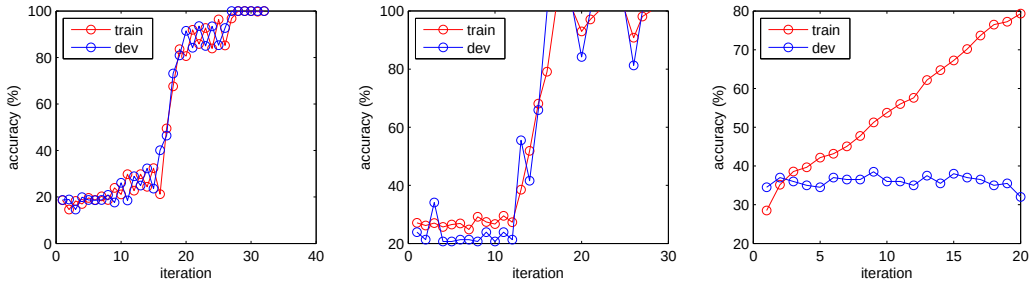


Figure 4: Accuracy vs epoch for supervised DMN training on bAbi task 2 and 15, and MCTest500 with pre-trained attention weights.

In contrast, we observe significant improved accuracy (Table. 2) after implementing the supervised training for the attention gate as suggested in [1]. We again visualize the episodic memory in Fig. 3. With supervision, the gate function is quite sparse, and show significant shift between episodes. From Fig.4, we can see the training accuracy and dev accuracy evolving as the iteration increases.

Another interesting thing to note is that for certain tasks such as qa9, qa13, and qa16, the supervised DMN generates even lower test accuracy than the unsupervised one we used. One possible speculation is that the training process is not complete and the model stucks at some local maxima, which is quite reasonable because for those tasks the training accuracy are also not high. Another possible speculation is that: these three types of questions related to induction and coreference, which are highly dependent on the correctness of supervised gating. We intend to dig into this problem further for future work.

5.2 MCTest

As a very first attempt to use DMN for MCTest, we ignore the choices provided in the dataset, and attempt to use a recurrent answer module to generate the correct answer sentence. This yields very low accuracy ($< 1\%$). We then attempt to restrict ourselves to only questions with one-word

Table 2: Comparison among the unsupervised and supervise DMN we implemented and the one used in [1] on all bAbI tasks. Some of the supervised DMN tasks have not been run to conversion due to time limitation.

task	DMN Implemented	Supervised DMN	DMN in [1]
1	100%	100%	100%
2	33.0%	100%	98.2%
3	36.5%	61.4%	95.2%
4	96.6%	100%	100%
5	83.4%	99.5%	99.3%
6	93.4%	99.6%	100%
7	76.6%	92.9%	96.9%
8	76.3%	94.0%	96.5%
9	91.7%	63.8%	100%
10	90.9%	92.5%	97.5%
11	64.8%	100%	99.9%
12	72.9%	100%	100%
13	88.6%	18.3%	99.8%
14	75.8%	100%	100%
15	53.1%	100%	100%
16	47.1%	25.6%	99.4%
17	58.0%		59.6%
18	92.0%	99.1%	95.3%
19	8.10%		34.5%
20	98.0%		100%

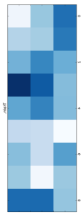
answer. In the MC500 training set that includes 300 stories and 1200 questions, there are 410 one-word-answer questions. One this subset our accuracy are still very low (3%). This is partially due to trying to use a dense layer to produce a vocabulary-size (~ 4000) prediction. Such a large size vocabulary dense layer will be very difficult for training on such a small corpus like MCTest dataset.

We implement the options module o as described in Section 4. Each option is representing by the end state of running the option sentence through a GRU (weights identical to the one used for question and input). This end state is (1) used in the gate function $z(c, m, q, o)$; (2) modify the answer module to produce a cosine-similarity metric $m^T U o$. Both obtain a testing accuracy of above 35% after a one epoch. The second method show slightly high test accuracy of 37.3% (but may not be statistically significant). Significant overfitting, with training accuracy approaching 90% while development and test accuracy remains around 35%, is still observe even with dropout of 0.5 or with L2 regularization. We also attempt to reduce model capacity by reducing internal states (c, q, m) dimension from 40 to 20 and 10 and did not observe any improvement.

As a next step, we load the pre-train attentions weights W_1, W_2, b_1, b_2, W_b obtained from supervised training on bAbi task 3 with 5 memory hops to initialized a DMN model for MCTest, and continue training from there (with other weight parameters initialized as usually). We obtain high initial accuracy and a small 1% improvement for the final test accuracy after ~ 10 epoch.

Although the test accuracy generated by MemNN and DMN is better than random guess (25%), it still cannot beat the test accuracy of the baseline model (SW+D) [9]. 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 supervised DMN is better than the unsupervised DMN, which is better than MemNN, as we expected (Table.3). We can see how the training accuracy and dev accuracy evolving as iteration increases from Fig.4, and visualize the episodic memory attentions from Fig.5.

Our implementation of DMN for MCTest can be found online at DMN-MCTest. The repo contains our implementation of supervised attention training for babi, DMN for MCTest including generative answer module, option module and attention-weight-transfer. It also contains visualization for episodic memory.



...Story...
on a monday a mother was in the kitchen cooking ham for the family .
she was in quite the hurry because she had to pick the kids up from school .
without knowing she stuck a spoon in the ham and put the ham in the oven .
she wasn ' t paying attention because of being in a hurry and also because the dog was playing with a ball .
the ball flew past her head as she had finished putting the ham in the oven .
she yelled at the dog and made him play with the ball outside .
she saw clouds in the sky as she let the dog out .
seeing that the clouds were so pretty that she wanted to take a picture of them .
she got her camera and took a few pictures as the dog played happily in the yard .
...Question...
what did the mother do when she saw the clouds in the sky ?
...Options...
*0. take pictures of them
1. throw a ball
2. cook them
4. pick the kids up from school
...Predict: 0 ...Confidence: 0.73

Figure 5: An example of episodic memory attentions (g) for DMN on MCTest.

Table 3: Baseline, LSTM, MemNN and DMN on MCTest MC500. The MemNN and DMN used are the weakly supervised version.

Set	baseline (SW+D) [9]	MemNN [7]	DMN	DMN w spv
Train	72.5%	92.9%	92.5%	80.1%
Test	66.2%	36%	37.3%	38.5%

Our baseline LSTM and MemNN implementation is based on Keras. Our DMN implementation is based on YerevaNN's DMN in Theano.

6 Conclusion & Perspectives

Unsupervised memory network based system (MemMM, DMN) are very versatile architecture, but training them can be hard. For example, the DMN episodic memory without attention supervision doesnt seem to actually shift focus over episode. In this work, we have extended the capability of the DMN on natural language question-answering problems. Specifically, we have contributed:

- New Option Module and modified Answer Module to make DMN suitable for MCTest dataset and exploration on attention mechanism;
- Supervised attention training on bAbi and application to MCTest;
- Model reduction to reduce over-fitting (L2 and dropout doesnt seem to help).

Furthermore, we also discovered something very interesting along the way, and probably look into them in more details in the future:

- The MCTest dataset is too small for decent good training, and we hope to study some models that specifically suit for small dataset;
- More understanding on the supervised attention training, due to the fact that our implementation yields some very low test accuracy on certain types of tasks.

References

[1] Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, Richard Socher. Ask Me Anything: Dynamic Memory Networks for Natural Language Processing. arXiv:1506.07285.

[2] Weston, J., Bordes, A., Chopra, S., and Mikolov, T. Towards ai-complete question answering: A set of prerequisite toy tasks. CoRR, abs/1502.05698, 2015.

[3] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston and Rob Fergus. End-To-End Memory Networks. arXiv:1503.08895.

[4] Mohit Iyyer,Jordan Boyd-Graber,Leonardo Claudino,Richard Socher, Hal Daum e III. Neural Network for Factoid Question Answering over Paragraphs. In EMNLP, 2014.

[5] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory Networks. ICLR 2015.

[6] Bowman, S. R., Potts, C., and Manning, C. D. Recursive neural networks for learning logical semantics. CoRR, abs/1406.1827, 2014.

[7] Darshan Kapashi and Pararth Shah. Answering Reading Comprehension Using Memory, past CS224D project. 2015.

[8] Te-Lin Wu and De-An Huang. CS224N Final Project: QA. 2015.

[9] MatthewRichardson, Christopher J.C. Burges, and Erin Renshaw. Mctest: A challenge dataset for the open-domain machine comprehension of text. EMNLP 2013.

[10] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation.

[11] Baolin Peng, Zhengdong Lu, Hang Li, Kam-Fai Wong. Towards Neural Network-based Reasoning. arXiv:1508.05508.