Dynamic Memory Networks for Question Answering

Arushi Raghuvanshi
Department of Computer Science
Stanford University
arushi@stanford.edu

Patrick Chase
Department of Computer Science
Stanford University
pchase@stanford.edu

Abstract

Dynamic Memory Networks (DMNs) have shown recent success in question answering. They have achieved state-of-the-art results of the Facebook bAbI dataset and performed well on sentiment analysis and visual question answering [1] [6]. In this project, we implement our own DMN in tensorflow and verify its performance quantitatively and qualitatively. We achieve very similar results to those achieved in Kumar et al. and Xiong et al. [1] [6]. In addition, we build a demo to visualize the attention placed on different input sentences in the episodic memory module, and show that the model places it's attention on the correct sentences for a variety of different tasks even without any explicit attention feedback during training. Last, we experiment with training a model to accomplish more than one babi task at the same time. We show that DMNs can successfully complete multiple babi tasks with the same model including one step reasoning, two step reasoning and yes/no questions. In addition, we illustrate through the demo that the combined model places the attention on the correct sentences when performing the different tasks.

1 Introduction

Question Answering (QA) is one of the oldest tasks in NLP. Most problems in NLP can be formulated as a question answering task, and QA has recently seen commercial popularity and media attention in applications such as Siri and Watson.

Original QA systems often involved developing a structured knowledge database that is hand-written by experts in a specific domain. In these systems, a question asked in natural language must be parsed and converted into a machine-understandable query that returns the appropriate answer.

With the massive amounts of natural language information on the web, current systems focus on extracting information from these documents. As a result, recent QA systems focus on information-retrieval based methods which include 1) a question processing module for formulating a query, 2) an information-retrieval module for selecting the appropriate document and passage, and 3) an answer processing module to generate the appropriate answer in suitable language. Many of these applications are open-domain, meaning they can answer questions about any topic.

Recently with advancements in deep learning, papers have been published that utilize recurrent neural networks for question answering. These deep networks generate latent representations of natural language text passages rather than relying on extracted features such as part of speech tagging, parsing, named entity recognition, etc. These networks require much less pre-prossesing and have recently matched and even exceeded the results of other models. The current state-of-the-art system is Dynamic Memory Networks presented by Xiong et al [6]. This system contains 4 modules: input, episodic memory, question, and answer. Each module consists of an RNN optimized for the corresponding sub-task.

We approach the question answering task using the DMN model. We implemented DMNs in Tensor Flow and train and test the model on the dataset described below.

2 Related Work

Prior to DMNs, work had been done in the related lines of attention and memory mechanisms. Wetson et al [7] first presented memory networks as a way to use a long-term memory component as a dynamic knowledge base for question answering. This memory network, unlike DMNs, requires the labeled supporting facts during training. Attention mechanisms have recently been used for a variety of applications including image captioning [8]. Stollenga et al proposed a model that allows the network to iteratively focus its internal attention on some of its convolutional filters. Similarly, DMNs use attention for QA to iteratively focus on certain sentences in the input text.

There have been a few papers published within the last year that have presented Dynamic Memory Networks and improvements on the model. DMNs gained popularity with Kumar et al's [1] publication in 2015. They present the DMN model described below and apply it to a variety of language tasks including the Facebook bAbI dataset. Xiong et al [6] showed that DMNs receive strong results when supporting facts are not marked during training, proposed improvements on the memory and input modules, and illustrated the the models do well for visual question answering in addition to textual question answering.

3 Data

3.1 Facebook bAbI Dataset

The Facebook bAbI-10k dataset has been used as a benchmark in many question answering papers. It consists of 20 tasks. Each task has a different type of question such as single supporting fact questions, two supporting fact questions, yes no questions, counting questions, etc.

We used the English version of the dataset with 10,000 training examples and 1000 test examples. All examples consist of an input-question-answer tuple. The input is a variable length passage of text. The type of question and answer depends on the task. For example, some tasks have yes/no answers while others are focused on positional reasoning or counting. For each question-answer pair, the dataset also gives the line numbers of the input passage that is relevant to the answer. Every answer in the bAbI dataset is one word. Examples from the dataset can be seen below.

```
Two supporting fact example:          Yes/no question example:

1 Mary got the milk there.            2 John moved to the bedroom.
2 John moved to the bedroom.          3 Is John in the kitchen?   no   2
3 Sandra went back to the kitchen.
4 Mary travelled to the hallway.
5 Where is the milk?   hallway   1 4
```

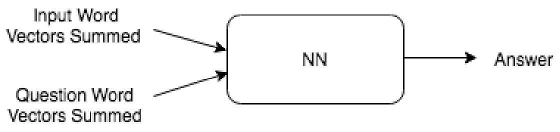
3.2 Evaluation

For the one word answers in the bAbI dataset, we frame the problem as a multi-class classificaion 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.

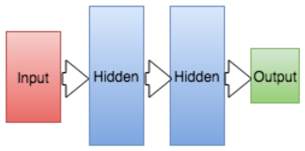
4 Approach

4.1 Simple Neural Network Baseline

As a simple baseline, we used a basic neural network. Since the input text and question have have a variable length of words, we used a simple summing heuristic of the word vectors to generate a



(a) Simple Baseline Model



(b) Neural Network Architecture

Figure 1: Simple Baseline

fixed length representation of the text. We sum all of the GloVe word vectors for each word in the input text, sum all of the GloVe word vectors for each word in the question text, concatenate the two together, and then use this as the input to the NN. The output of the network is a probability distribution on the output tokens. (Figure 1a)

We experimented with the depth of the network. Our final baseline was a NN with two deep fully connected layers with a hidden dimension of 200 followed by a softmax layer as seen in Figure 1b. We used ReLU nonliniarities and the Adam optimizer. In addition, we used l2 regularization with weight of 0.0001. The model depth, hidden dimensions, and regularization weight were tuned for optimal performance.

4.2 GRU Baseline

We then implemented a better model with three modules: input, question, and answer. The input and question modules are recurrent neural networks with gated recurrent unit (GRU) cells. This allows us to better embed the variable length sentences into a fixed length feature vector, while taking the position of the words into account. More specifically, for each word we update the state of the GRU, then after we've ingested all of the words, the final state is used as the embedding of the variable length input. We then concatenate the vector from the input module and the vector from the question module and feed them into the answer module. The answer module consists of a linear transform and a softmax layer to achieve a probability distribution over the output tokens. For this project we focused on one word answers, but the answer module could be replaced with a more complex GRU, which could be used to generate multiple word answers. A diagram of the GRU baseline can be seen in figure 2.

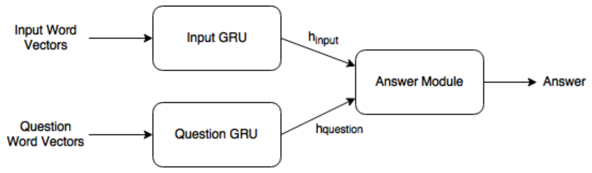


Figure 2: GRU Baseline

4.3 Dynamic Memory Networks

The final DMN model consists of 4 modules: input, question, episodic memory, and answer. This section explains the functionality of each of these building blocks. The entire architecture is displayed in Figure 3.

4.3.1 Input Module

The input module takes in the word vectors for the input and feeds then through a GRU and outputs the hidden states at the end of each sentence for the episodic memory module to reason over. More formally for a sequence of T_I words $w_1, ..., w_{T_I}$ we update the state using

$$h_t = GRU(L[w_t], h_{t-1})$$

Then say the T_I words comprise T_S sentences $s_1, ..., s_{T_S}$. We then project the hidden states corresponding to the end of each sentence. So, the final output of the input module is $h_{s_1}, ..., h_{s_{T_S}}$.

4.3.2 Question Module

The question module also runs a GRU over the word vectors, however it just outputs the final state of the GRU to encode the question. So, for a question of T_Q words $w_1, ..., w_{T_Q}$ we update the state using

$$h_t = GRU(L[w_t], h_{t-1})$$

The final output of the question module is h_{T_Q}.

4.3.3 Episodic Memory Module

The episodic memory module reasons over the sentence states from the input module as well as the question state from the question module and ultimately produces a final memory state that is sent to the answer module to generate an answer.

Episode Update Mechanism

Each episode reasons over the sentences and produces a final state for that pass over the data. Here is how it is updated for a new input sentence state c_t:

$$z_t^i = [c_t, m, q, c_t \circ q, c_t \circ m, |c_t - q|, |c_t - m|]$$

$$Z_t^i = W^{(2)} \tanh\left(W^{1)} z_t^i + b^{(1)}\right) + b^{(2)}$$

$$g_t^i = \frac{\exp(Z_t^i)}{\sum_{k=1}^{M_i} \exp(Z_k^t)}$$

$$h_t^i = g_t^i GRU(c_t, h_{t-1}^i) + (1 - g_t^i) h_{t-1}^i$$

So, the current sentence state c_t, the current memory state m, and the question state q are collectively used to determine if the current sentence is important or not to the answer and encoded in g_t^i. We see that if $g_t^i \approx 0$ then the previous state will be copied through and the sentence will be ignored, but if $g_t^i \approx 1$ the past will be ignored and a lot of attention will be placed on the current sentence.

It is also important to note that we use the softmax function to determine the value of g_t^i. This is an update proposed in Xiong et al. to enable the attention to be visualized more easily, since it forces the sum of all attention gates to be 1 [6].

The final state for the episode is the state of the GRU after all the sentences have been seen is $e^i = h_{T_S}^i$

Memory Update Mechanism

The memory is then updated using the current episode state and the prevous memory state.

$$m^t = GRU(e^t, m^{t-1})$$

The final state of the memory after the maximum allowed passes over the data is then sent to the answer module to generate an answer.

4.3.4 Answer Module

The answer module is simple linear layer with a softmax activation to produce a probability distribution over the answer tokens. This could be extended to an RNN for multiword answers, however we kept it as a simple softmax since the bAbI dataset only has one word answers.

4.4 Implementation Details

Everything was implemented in TensorFlow, except for the simple baseline which was implemented in Keras. The demo was implemented using Flask.

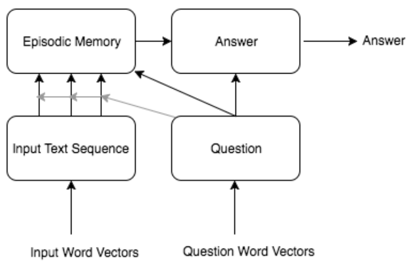


Figure 3: Dynamic Memory Network

We wrote a script to distribute runs on 30 of Stanford's FarmShare computers using CPUs in parallel. This configuration allowed us to experiment with many different hyper-parameters. Once we narrowed down the range of hyper-parameters, we trained or final models on AWS using GPU for computational power and speed.

4.5 Visualizing Attention

Implementation and debugging were two of the significant challenges of this project. We implemented a demo which allowed us to visualize the attention in each episode in order to see if the network was performing as expected. See attached video for a demo our the web app we built to visualize the attention as well as the screenshots from the demo in the qualitative analysis sections.

5 Experiments

5.1 Single Task Results

First we trained and tested our model on different bAbI tasks individually. We trained on the the (input, question, answer) tuples from a single task and then tested on that task. We did not use any of the explicit attention signals (sentence numbers that contain answer) when training.

We found that the GRU baseline improved on the simple baseline, and the DMN imporoved on the GRU baseline. These results can be seen in table 1.

Table 1: DMN bAbI-10k validation accuracies for each model on Task 6

Model	Task	Val
NN Baseline	Yes/No questions	0.78
GRU Baseline	Yes/No questions	0.85
DMN	Yes/No questions	1.0

5.1.1 Tuning DMN

As a starting point for tuning the DMN, we used the best parameters given in Xiong et al [6]. The l2 regularization value was not given, so we spent some time tuning this parameter. As seen from the plots in Figure 4, the correct regularization values were important in preventing underfitting and overfitting.

5.1.2 Final Parameters

Our final model parameters are a learning rate of 0.001, l2 regularization value of 0.001, a dropout keep probability of 0.9 and a batch size of 100. In addition, we trained with 250 epochs with early stopping. We also used a max of 3 passes over the input for all tasks except for tasks 7 and 8, which we used 5 passes over the data. This was shown to improve accuracy on these tasks in Kumar et al [1]. We used l2 regularization on all the weights in the model and used dropout on the word

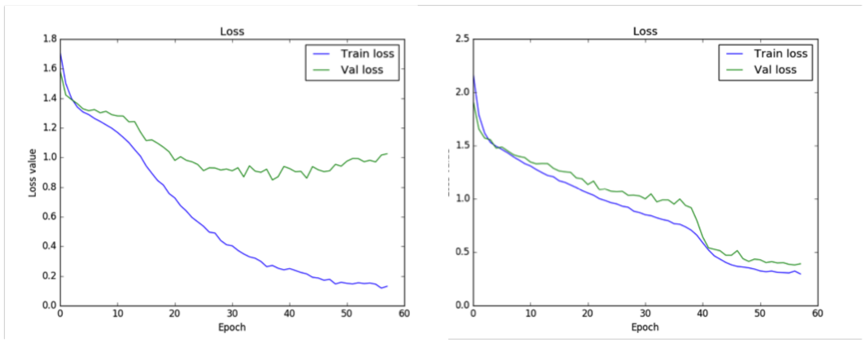


Figure 4: Training and Validation Loss for on Task 2 for 2e-5 and 1e-3 L2 Regularization

vectors and on the final memory states consumed in the answer module. Last, we did change the regularization parameter to 7e-5 for task 6 as we found this lead to better and more stable results.

5.1.3 Quantitative Results

We were able to reproduce results similar to those in Xiong et al [6] as shown in Table 2b. We outperform the original DMN on tasks 2,6,7, and 8, and are close to the performance of the DMN+ on all the tasks. We believe the reason for this improvement over the original DMN is the use of the softmax attention function instead of a sigmoid function, which was proposed in Xiong et al [6]. We did not implement any of the other improvements the paper uses, such as the bidirectional GRU in the input module and the Attention based GRU in the episodic memory module [6].

Table 2: DMN Results on bAbI tasks

(a) DMN bAbI-10k train and validation accuracies for our DMN, which for comparison purposes we call PA-DMN.

Task	Train	Val
1	1.0	1.0
2	0.981	0.966
6	1.0	1.0
7	0.960	0.967
8	0.993	0.995

(b) DMN bAbI-10k test accuracies. Here our implementation is PA-DMN, ODMN is the original DMN from Kumar et al. and DMN+ is the DMN from Xiong et al. The results here are $((100 - error\_rate)/100)$ as the error rates were reported in Xiong et al.

Task	PA-DMN	ODMN	DMN+
1	1.0	1.0	1.0
2	0.95	0.64	0.989
6	1.0	0.643	1.0
7	0.963	0.92	0.976
8	0.995	0.984	1.0

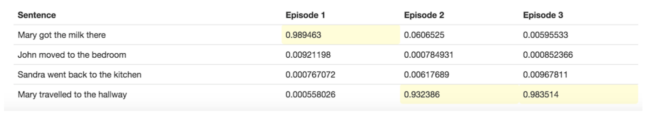
5.1.4 Qualitative Analysis

In addition to the quantitative results, we built a demo to visualize the attention in the episodic memory module. The attention on the different sentences for various test examples (never seen by the model) can be seen in figures 5, 6, and 7. Again it is important to note that we never train on any explicit attention signals.

We can see that in figure 5 the model correctly puts its attention on the first sentence that establishes that Mary has the milk in the first episode. The second time over the data it understands that Mary has the milk and looks for where Mary takes it, so it puts all of its attention on the last sentence. In episode 3, it correctly understands that the last sentence is the last relevant sentence to the position of the milk, so its attention doesn't change. In addition, figure 6 shows the attention on a counting example.

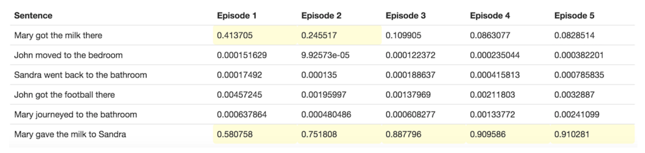
Last, figure 7, shows a one step example from task 1 fed into the model trained for two step examples. It completely misses the relevant sentences which is somewhat expected, since it hasn't seen a single step example. This is part of the motivation for the combined model we explore in the next section.

Figure 5: Task 2 (2 supporting facts). Q: Where is the milk? A: hallway



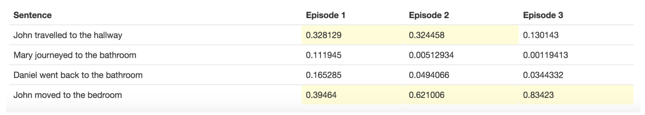
Sentence	Episode 1	Episode 2	Episode 3
Mary got the milk there	0.989463	0.0606525	0.00595533
John moved to the bedroom	0.00921198	0.000784931	0.000852366
Sandra went back to the kitchen	0.000767072	0.00617689	0.00967811
Mary travelled to the hallway	0.000558026	0.932386	0.983514

Figure 6: Task 7 (counting). Q: How many objects is Mary carrying? A: none



Sentence	Episode 1	Episode 2	Episode 3	Episode 4	Episode 5
Mary got the milk there	0.413705	0.245517	0.109905	0.0863077	0.0828514
John moved to the bedroom	0.000151629	9.92573e-05	0.000122372	0.000235044	0.000382201
Sandra went back to the bathroom	0.00017492	0.000135	0.000188637	0.000415813	0.000785835
John got the football there	0.00457245	0.00195997	0.00137969	0.00211803	0.0032887
Mary journeyed to the bathroom	0.000637864	0.000480486	0.000608277	0.00133772	0.00241099
Mary gave the milk to Sandra	0.580758	0.751808	0.887796	0.909586	0.910281

Figure 7: Task 1 (1 supporting fact) with two step model. Q: Where is Mary? A: bedroom (incorrect)



Sentence	Episode 1	Episode 2	Episode 3
John travelled to the hallway	0.328129	0.324458	0.130143
Mary journeyed to the bathroom	0.111945	0.00512934	0.00119413
Daniel went back to the bathroom	0.165285	0.0494066	0.0344332
John moved to the bedroom	0.39464	0.621006	0.83423

5.2 Multiple Task Results

After verifying that our implementation of DMNs was correct by reproducing the results presented in [6] and visualizing the attention, we trained the model on multiple tasks (1,2 and 6) which included one step reasoning, two step reasoning, and yes/no questions at the same time to generate a more general QA system.

We trained our multitask system on the training sets from tasks 1,2, and 6 and tested it on each tasks test set individually. We used the parameters from section 5.1.2 when training the model with a max of 3 passes over the input.

5.2.1 Quantitative Results

As a baseline we use the model trained on task 2 and apply it to the test set of task 1. The performance is very poor and the model for task 2 only achieves a test accuracy of 0.187 on the task 1. This is somewhat expected because it has never seen any single step examples.

In contrast, the model we trained to perform multiple tasks achieved good test accuracies and got an accuracy of over 0.98 for each sub-task as shown in table 3. It achieved the exact same test accuracy as the single task model for tasks 1 and 6 and even improved on the accuracy for task 2.

The training accuracy for this model on the combined dataset was 0.993, while the validation accuracy was 0.995.

5.2.2 Qualitative Analysis

Here we show that the multitask model has the correct attention gates on the three different tasks. Again, this wasn't trained with any explicit feedback on the gates and has no knowledge of the type of question. The attention for the different tasks can be seen in figures 8, 9, and 10. We can see that the single model now has correct attention for one step reasoning, two step reasoning and yes/no questions.

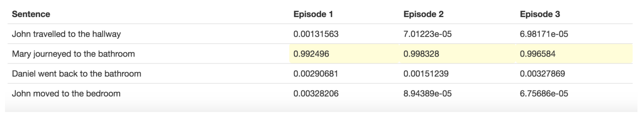
6 Conclusion

In this project we built a DMN and evaluated it on various tasks in the bAbI dataset. We verified the implementation of our DMN by achieving results very similar to those achieved by previously published works and evaluating its attention qualitatively. In addition, we experimented with training multiple models at the same time, and show that there is no drop in performance when training

Table 3: Multiple Task (1,2,6) Test Accuracy

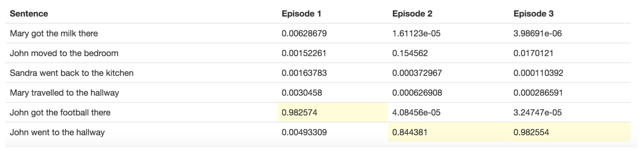
Task	**Test**
1	1.0
2	0.983
6	1.0

Figure 8: Task 1 (1 supporting fact) with combined model. Q: Where is Mary? A: bedroom (incorrect)



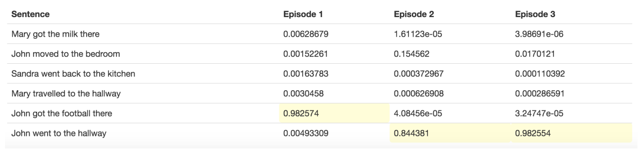
Sentence	Episode 1	Episode 2	Episode 3
John travelled to the hallway	0.00131563	7.01223e-05	6.98171e-05
Mary journeyed to the bathroom	0.992496	0.998328	0.996584
Daniel went back to the bathroom	0.00290681	0.00151239	0.00327869
John moved to the bedroom	0.00328206	8.94389e-05	6.75686e-05

Figure 9: Task 2 (2 supporting facts) with combined model. Q: Where is the football? A: hallway



Sentence	Episode 1	Episode 2	Episode 3
Mary got the milk there	0.00628679	1.61123e-05	3.98691e-06
John moved to the bedroom	0.00152261	0.154562	0.0170121
Sandra went back to the kitchen	0.00163783	0.000372967	0.000110392
Mary travelled to the hallway	0.0030458	0.000626908	0.000286591
John got the football there	0.982574	4.08456e-05	3.24747e-05
John went to the hallway	0.00493309	0.844381	0.982554

Figure 10: Task 6 (yes/no) with combined model. Q: Is John in the kitchen? A: no



Sentence	Episode 1	Episode 2	Episode 3
Mary got the milk there	0.00628679	1.61123e-05	3.98691e-06
John moved to the bedroom	0.00152261	0.154562	0.0170121
Sandra went back to the kitchen	0.00163783	0.000372967	0.000110392
Mary travelled to the hallway	0.0030458	0.000626908	0.000286591
John got the football there	0.982574	4.08456e-05	3.24747e-05
John went to the hallway	0.00493309	0.844381	0.982554

multiple tasks at the same time. This suggests that DMNs are a very powerful architecture for more general QA tasks that would encompass all of the different types of reasoning found in the bAbI dataset. As we move closer to more advanced AI systems it will be essential for the models to perform many different types of reasoning when answering questions.

7 Future Work

The bAbI dataset was useful in verifying that our implementation was correct. However, as a synthetic dataset, it may not accurately represent some of the difficulties of training on human generated dataset. It would be interesting to see how DMNs perform on more diverse datasets such as the DeepMind reading comprehension dataset [4]. We have started experimenting with this dataset and are still working on training and testing a model.

In addition, some modifications of the model that we are interested in considering is making the RNNs bidirectional, using LSTMs, or adding an additional layer to the GRUs.

Last, we are working to clean up our code and make it the first publicly available implementation of a DMN in tensorflow.

Acknowledgments

We would like to thank Richard Socher for his advice on developing DMNs and the cs224d course TAs for their guidance throughout the course.

References

[1] Kumar, Ankit et al. (2016) Ask Me Anything: Dynamic Memory Networks for Natural Language Processing. *ArXiv e-prints*

[2] Banko, Michele et al. (2002) AskMSR: Question Answering Using the Worldwide Web. *AAAI Spring Symposium on Mining Answers*

[3] Iyyer, Mohit et al. (2014) A Neural Network for Factoid Question Answering over Paragraphs. *Emperical Methods in Natural Language Processing*

[4] Hermann, Karl Moritz et al. Teaching Machines to Read and Comprehend. *ArXiv e-prints*

[5] Strzalkowski, Tomek & Sanda Harabagiu. *Advances in Open Domain Question Answering*. Springer.

[6] Xiong, Caiming et al. (2016) Dynamic Memory Networks for Visual and Textual Question Answering. *ArXiv e-prints*

[7] Wetson, Jason et al. (2015) Memory Networks. *ICLR 2015*

[8] Stollenga, Marjin F. et al. (2014) *NIPS*