Dynamic Inference: Using Dynamic Memory Networks for Question Answering

Allan Jiang
Department of Computer Science
Stanford University
Stanford, CA 94305
jiangts@cs.stanford.edu

Chaitanya Asawa
Department of Computer Science
Stanford University
Stanford, CA 94305
casawa@cs.stanford.edu

Abstract

Question Answering is an incredibly important task in Natural Language Processing (NLP), and we aim to experiment with models in Machine Comprehension to attempt to perform well on Question Answering tasks. We specifically work with the Faceboook bAbi dataset, and aim to achieve strong results using Dynamic Memory Networks, which are known to have strong performance for this task. Dynamic Memory Networks process questions and inputs, and with episodic memories, try to determine the answer for a particular question.

1 Motivation

Many tasks in NLP can be seen as question answering – i.e. asking, "What is the sentiment of this sentence?" to perform sentiment analysis. Hence, if we can build a system that can perform general question answering well, it can potentially serve as a joint model for NLP.

2 Introduction

The general problem we're investigating is machine comprehension. In particular, given a story that represents a series of facts, can we have the machine correctly answer a question that is inferred on these facts? The ability to make inferences and reason based on facts will help advance the state of machine question answering and comprehension. Combined with an appropriate information retrieval system, we think such a system could advance the quality of domain-specific question answering tasks, such as in customer support. To solve this problem, we would like to use Dynamic Memory Networks, which have been shown to have state-of-the-art performance on question answering by processing the questions and inputs and using episodic memories to determine answers to questions [1].

3 Related Work

Our Dynamic Inference model is derived from closely reading the "Ask Me Anything" paper by Kumar et al [1]. The main innovation in their Dynamic Memory Network model is in the attention mechanism and episodic memory module. This module iterates over facts represented as distributed vectors, computing a gate for each fact to determine whether or not the fact is relevant to the reservoir of knowledge the module has already built up in prior iterations, known as a "memory" in the paper. The facts and their computed gates are fed through a specialized weighted GRU to compute an episode, which is then used to compute the next memory. The purpose of the model is to iteratively retrieve more and more information relevant to the original query, using newly discovered knowledge found in previous iterations to inform on what facts should be considered important for the next iteration. This model was able to achieve state of the art results on many tasks in the bAbi dataset offered by Facebook Research [2], and achieved greater than 95% accuracy on 18 tasks.

Prior to Dynamic Memory Networks, the best results achieved for question answering on the bAbi dataset were achieved by Memory Networks by Weston et al. [2]. Memory networks also have multiple components: an input component, response component, generalization component, and output feature map. The generalization component and output feature map, which also aim to iteratively retrive facts from the set of input facts, are functionally replaced by the episodic memory module in Dynamic Memory Networks. However, Dynamic Memory Networks have achieved superior performance presumably due to using sequence models for each module of the entire model, helping it better capture position and temporality of natural language.

4 Problem Statement

4.1 Data

We plan to use the bAbi dataset offered by Facebook Research [2]. This dataset has 20 toy tasks that test the ability to reason and understand. For each task, there are 1000 questions for training and 1000 for testing. Additionally, a set with 10,000 questions for training and testing for each task is available as well. A sample question from Task 1 (Single Supporting Fact) in their dataset is:

Mary went to the bathroom.
John moved to the hallway.
Mary travelled to the office.
Where is Mary? A:office

4.2 Evaluation

We will mainly evaluate our results using the percent of questions our model correctly answers on the various tasks proposed in the bAbi dataset.

5 Technical Approach and Models

We have implemented a Dynamic Memory Network from scratch. This model is based on the "Ask Me Anything" paper by Kumar et. al [1], and we are trying our best to achieve the results of the paper.

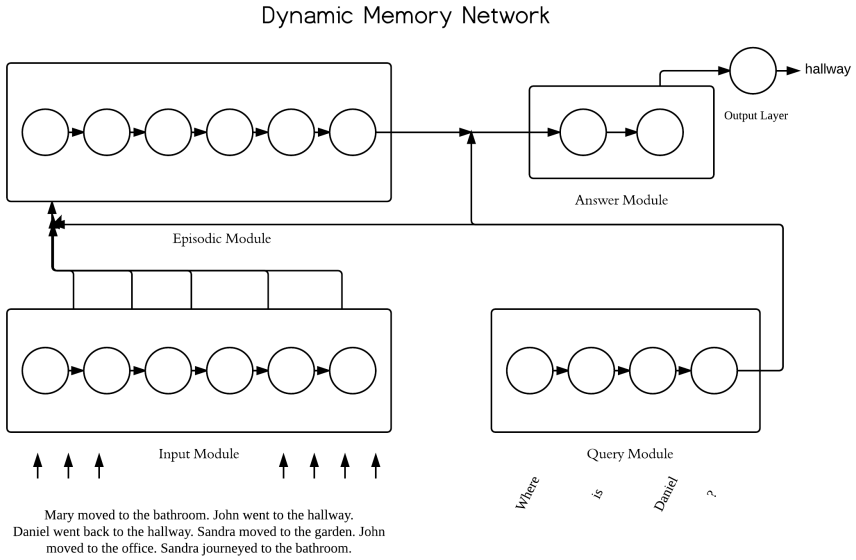


Figure 1: Schematic of the Dynamic Memory Network built

A Dynamic Memory Network can be described by 4 main modules:

1) Question Module
2) Input Module
3) Episodic Memory Module
4) Answer Module

At a high level, the Question Module takes the text of a question and encodes it into a distributed vector representation. The Input Module takes a series of text facts and also encodes them into, for each fact, into a distributed vector representation. The Episodic Memory Module, given the question and input fact representations, tries to determine which parts of the input to focus on using its attention mechanism. Finally, the Answer Module uses the output of the Episodic Memory Module along with the query to determine the answer.

More specifically, for the Question and Input Modules, we will use a gated recurrent network (GRU) that takes the word vectors for the text associated with each module and outputs another vector. In the case of the Input Module, as there are multiple facts/sentences, we concatenate all the words from all the facts togethers and add end-of-sentence tags. Then, for the Episodic Module, we extract the output vectors at the end-of-sentence states, giving us essentially a vector for each fact.

The Episodic Memory Module then, with the outputs of both the Input and Question Modules, uses a GRU to determine the relevant facts. The first input for the Episodic Module is the query vector itself, and then followed by the fact vectors in order. Finally, the Answer Module takes the final hidden state of the Episodic Memory Module and the query vector, and uses them both with another GRU to generate a vector. A softmax layer is then applied to this vector, resulting in an output vector with dimension of the vocabulary size. Hence, the entries of this final vector are treated as the probability of being the possible answer for each word in the vocabulary. To get the final answer, we take the word with highest probability.

It is important to note that the Episodic Memory Module we ended up using to report results is simpler than the one found in the Dynamic Memory Networks paper (just the GRU described above). After implementing a faithful, iterative version of the model found in the Kumar et al. paper (but without l2 loss on all the layers or dropout on the embedding layer) we were unable to achieve good results with the model, potentially due to differences in hyperparameter optimization. After running the "Single Supporting Fact" task from the bAbi dataset on our implementation of the original Dynamic Memory Network model, after 10 epochs the model was unable to achieve more than 20% accuracy, seemingly getting stuck around 18%. Since our model with a simpler version of episodic memory performed far better and converged far quicker, we use it to report results.

For our initial distributed vector representations of words, we are using word vectors from GloVe (Global Vectors For Word Representation), released by Stanford [3]. We experiment with word vector sizes of 50, 100, 200, and 300.

For our loss function, we use standard cross entropy loss and trained with the Adam optimizer in TensorFlow. [4]

5.1 Gated Recurrent Network

Gated Recurrent Networks form the basis for many of the modules, and so we describe them in detail.

Assume we have an input x_t and hidden state h_t at each time step t. Then, the GRU can be described with the following equations:

$$
\begin{aligned}
z_t &= \sigma(W^{(z)}x_t + U^{(z)}h_{t-1} + b^{(z)}) \\
r_t &= \sigma(W^{(r)}x_t + U^{(r)}h_{t-1} + b^{(r)}) \\
\tilde{h}_t &= \tanh(Wx_t + r_t \circ Uh_{t-1} + b^{(h)}) \\
h_t &= z_t \circ h_{t-1} + (1 - z_t) \circ \tilde{h}_t
\end{aligned}
$$

where $W^{(z)}, W^{(r)}, W \in \mathbb{R}^{n_H \times n_I}$ and $U^{(z)}, U^{(r)}, U \in \mathbb{R}^{n_H \times n_H}$. The n dimensions are hyperparameters (n_H being hidden size and n_I being input size) [5].

6 Results

We will discuss some of the experiments we conducted that helped us better solve the task at hand and additionally figure out our model's strengths and weaknesses. The evaluation metric we are using is accuracy; the number of questions our model can answer correctly on a given task. We consider tuning in the context of Task 1 on the bAbi dataset, Single Supporting Fact.

First, we calculate the loss history curves on the Training and Validation set when running our model on the Single Supporting Fact task with 1000 examples to understand how well the model fits. The result is as follows:

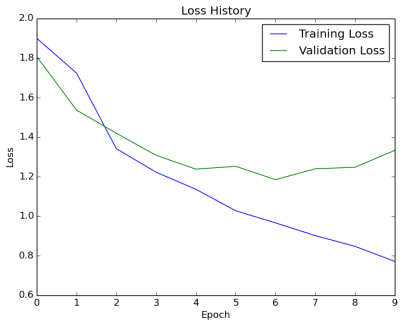


Figure 2: Loss history for Single Supporting Fact with 1000 examples.

These curves seem to indicate that with more epochs, the training error could indeed go to 0, but that the validation loss has reached its peak possible performance (and hence our training schedule subsequently early stops).

We made sure that the model was robust to effects of stochasticity by, with the same hyperparameters, rerunning the model a few times. This results in the following random restart curves:

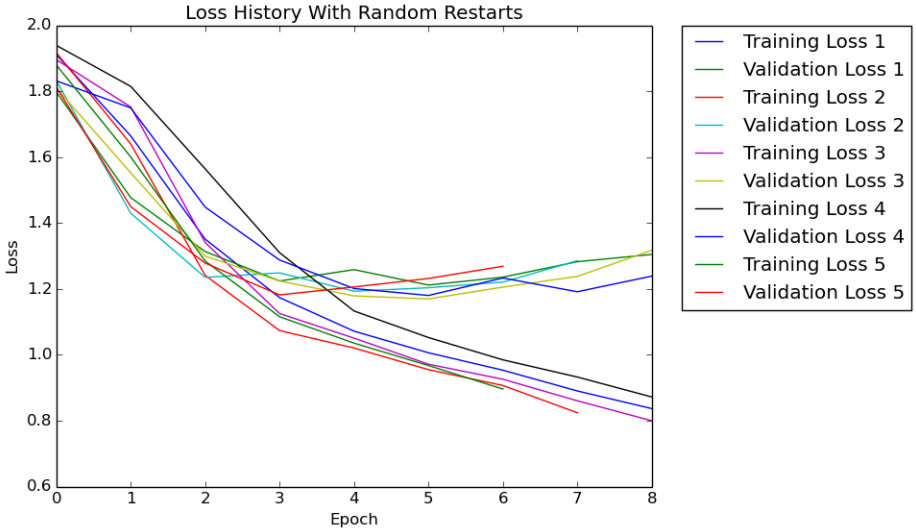
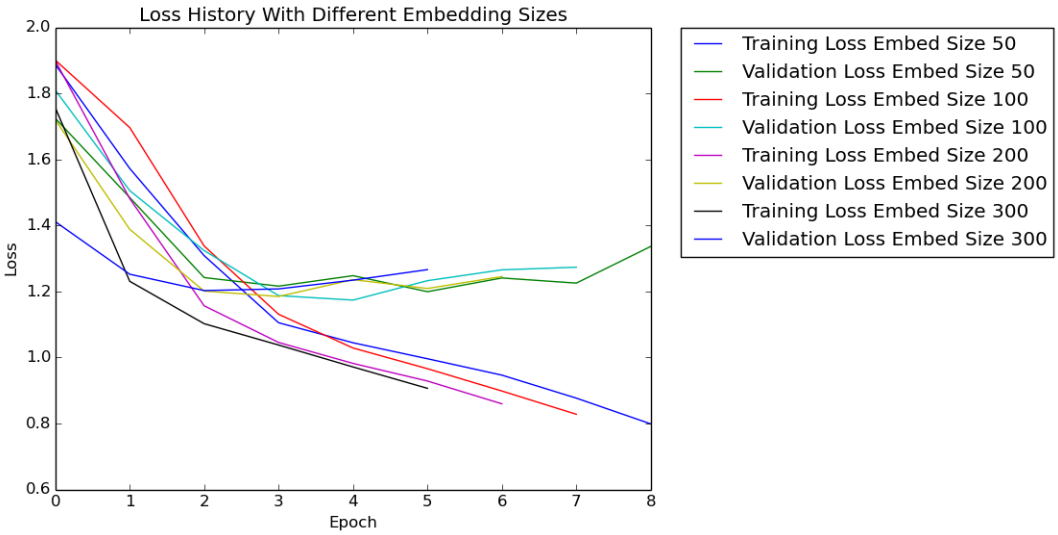


Figure 3: Determing if random restarts affect model performance.

The random restarts all seem to be consistent, indicating that our model results are generally invariant to stochasticity. We next experimented with the word vector size:



It seemed that the dimension of the word vector did not have an effect on the accuracy. However, it does seem that the larger the word vector size, the less epochs were required before early stopping. This seems to potentially indicate larger word vectors help by providing more information initially such that the model does not have to spend more time extracting this information.

We then shifted our focus to seeing if potentially varying the learning rate would help us find a better optimal accuracy that the learning rate of 0.001 could not (either because the current learning rate was too large and misses the optimal value, or it was too small and it does not get to the optimal value). The results of varying the learning rate are below:

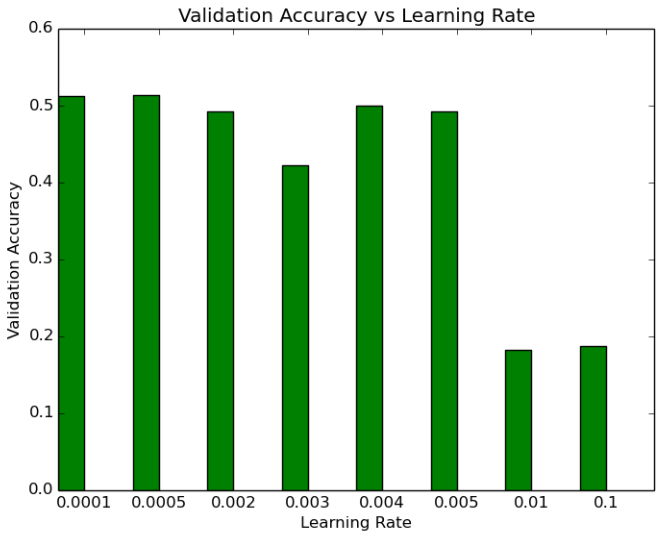


Figure 4: Learning rate effect on performance on Single Supporting Fact task.

We found that changes in the learning rate did not help – very small learning rates had comparable performance to the original learning rate of 0.001. Slightly larger learning rates also performed as well. Learning rates a magnitude greater than the current learning rate did not perform as well, however. These larger values may potentially vary the parameters too quickly to find the optimal

loss.

The next focus of our tuning was experimenting with the output size of the Input and Query modules. This would help understand what amount of information was needed from the Input and Query modules. For Single Supporting Fact, we have the following results:

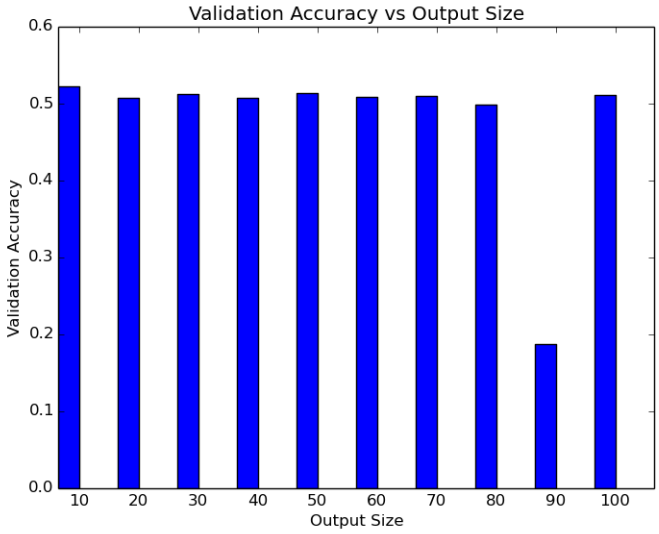


Figure 5: Size of GRU output effect on performance on Single Supporting Fact task.

It seems that the model is invariant to differences in the output size of the Input and Query modules. We hypothesize this is the case because Single Supporting Fact is not too complicated of a task, and the model does not need all of the extra information encoded by larger output vectors. To confirm this hypothesis, we varied GRU output size for the more complicated Two Supporting Facts task:

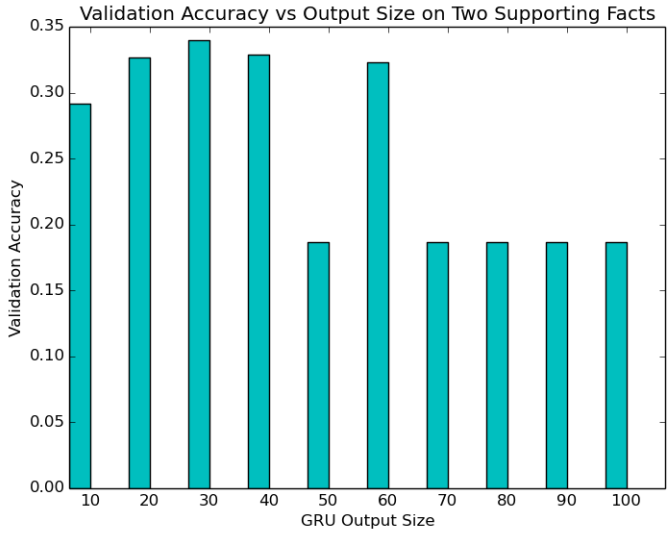


Figure 6: Size of GRU output effect on performance on Two Supporting Facts task.

In fact, we see here that output size does matter. Specifically, there seems to be an optimal output size of around 30 for this task.

Next, the bAbi dataset has two types of subdatasets – one in which there are 1000 training and testing examples for a given task, and another in which there are 10,000 training and testing examples for a given task. As our previous testing consisted of only testing with 1000 examples, we proceeded to test the model with 10,000 examples and see if our model can perform better. We tested this in the context of the Single Supporting Fact and Two Supporting Facts tasks:

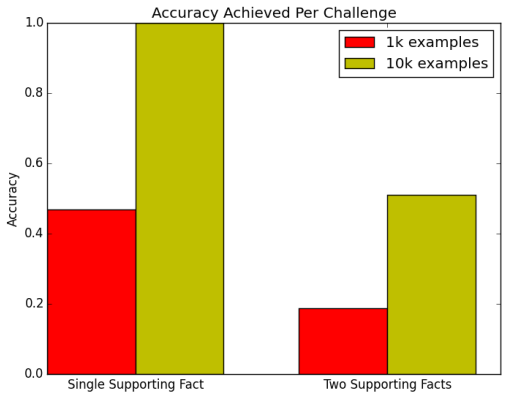


Figure 7: Model performance comparison with number of examples.

With 10,000 examples, the performance of both tasks at least doubled – with Single Supporting Fact achieving 100 percent accuracy and Two Supporting Facts achieving around 50 percent accuracy on 10,000 test cases. Clearly, the model was able to be far more effective with more data to consider.

With this knowledge, we conclude with final results for various tasks with 10,000 training and testing examples:

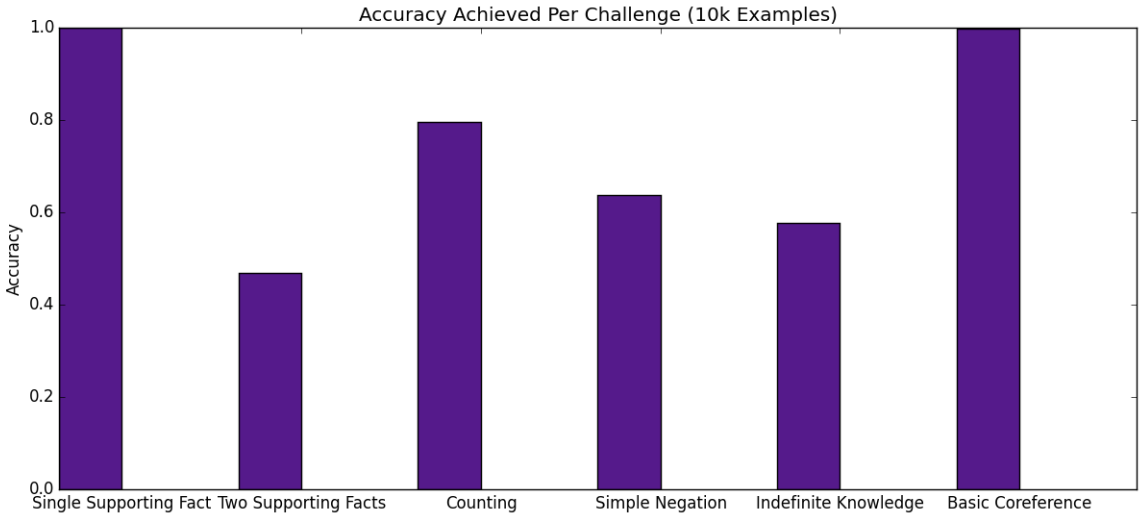


Figure 8: Model performance on multiple tasks with 10,000 examples.

7 Conclusion

We were able to achieve strong results on easier tasks with many examples available.

For more complicated tasks, and with less data, our model seems to underfit. In particular, the model is not able to capture the facts well enough and focus on the important facts/parts of facts. More experimentation with the Episodic Memory Module is required to be able to handle these tasks then.

Additionally, there have been some recent modifications to the Dynamic Memory Network model, including using bi-directional GRUs for the input module. The current, one directional GRU can only get the context from sentences before, and not after. Also, the GRU may have trouble capturing interactions between distant supporting facts spatially. Hence, the bi-directional GRU helps better utilize the information. [6]

A strength of model, moving forward to continue to improve it, is that it is highly modularized. Hence, with the necessary infrastructure set up (as we have done), we now can simply focus on model building for specific modules. We look forward to increasing the power of our attention mechanisms and being able to encode information to use it in a more effective and efficient manner.

8 Acknowledgements

We adapted some of the data processing work from an Open Source project by GitHub user YerevaNN. This included a script to help with collecting and loading the GloVe dataset in a convenient fashion and another script to help with collecting and loading the bAbi dataset in a convenient fashion. Finally, we utilized some code and ideas from previous CS 224D PSETs, and we would like to acknowledge the effort behind this.

References

[1] Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. Ask me anything: Dynamic memory networks for natural language processing. *arXiv preprint arXiv:1506.07285*, 2015.

[2] Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*, 2015.

[3] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.

[4] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[5] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[6] Caiming Xiong, Stephen Merity, and Richard Socher. Dynamic memory networks for visual and textual question answering. *arXiv preprint arXiv:1603.01417*, 2016.