Giving Machines Memory and Focus: Evaluating Attention-Based and Memory-Based Neural Network Models on Large Q&A Datasets

Dan Strawser

December 14, 2015

Abstract

One of the most basic tasks in natural language processing are question/answering tasks. Recent advancements in deep neural networks have shown promise in extracting information over long sequences of data as well as integrating a semantic knowledge base. In this work, I look at recently introduced neural network models including the Memory Network and Dynamic Memory Network. These models seek to apply a semantic memory as well as an attention mechanism to answering questions about articles. While they have seen success on smaller corpuses (such as the Babi tasks), it remains to be seen whether they can be successful on larger datasets. I evaluate the models on such datasets (Wiki QA, Google-CNN) and observe a considerable improvement over a baseline GRU model. The size of the dataset affects the result, with the best results achieved by the Memory Network on the large Google-CNN dataset.

1 Introduction

Question answering tasks are some of the most general in natural language processing. They are broad because many natural language processing tasks from machine translation to recommendation systems can be stated as question answering tasks. Additionally, QA algorithms provide potential for widespread applications : from winning game shows [1], to helping humans plan their evening, to providing medical care in the developing world. Developing effective and general QA approaches would greatly benefit field of natural language processing as well as the people using its technology.

However, developing effective QA systems is difficult because of the massive amounts of data and wide variety of questions that they must handle. Given a question and an information source, it is key to know where (for example, which sentences) to look for an answer. Furthermore, certain QA tasks may require inference. For example, to answer the question posed in Fig. 1, the algorithm must be able to find the relevant phrases and infer Sally's motivation.

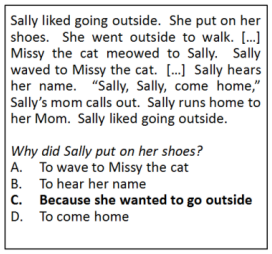

Figure 1: Example QA task from MC Test dataset. Taken from [3].

One relatively new approach to QA problems are deep neural networks. These neural networks have recently seen wide success in vision and speech recognition; however, it is still unclear whether they offer such dramatic improvements to natural language processing [2]. Advancements in deep neural networks over the past year, such as adding memory and attention mechanisms, could show significant improvements. However, these methods have been tested primarily on toy datasets. This projects seeks to evaluate these methods on more realistic corpora.

2 Prior Work

Many approaches have been proposed to solve QA tasks, among them, feature-based methods and parsing-based. One of the earlier approaches presents a pipeline approach where questions are classified into types, keywords are extracted, and a database is searched for possible answers, which are then ranked [4]. An approach described in [5] uses a semantic parsing approach to align natural language phrases with logical forms that can then be used to query a knowledge base. The method is effective because of its ability to generate logical predicates when the question is unclear or a predicate is not in the lexicon. Another approach works with the MC Test dataset and uses hidden variables to model which sentences are important to answering a question [3].

Neural networks provide another approach to solving this problem. Ideally, they have the major advantage of avoiding tedious feature engineering or requiring alignment of sentences with logical forms. This means that one could train on simple (*text*, *question*, *answer*) tuples, which can, in theory, be generated from raw text. Recurrent neural networks seem especially well-suited for this problem because of their ability to use sequential data as input. More recent models such as LSTM (Long Short Term Memory) units and GRU (Gated Recurrent Units) make it possible to train on longer sequences of data by avoiding the problem of exploding gradients [8]. However, some question and answer tasks require analyzing very large sequences of data and even these models have difficulty [13].

Advancements made in the past year in deep neural networks provide more

promise. These networks combine the sequential encoding found in RNNs and add memory and attention. Attention mechanisms are important because, given a large amount of information and a question, finding the relevant information is a challenge. While these have shown promise on smaller corpora, it remains to be seen their advantages on larger datasets. The goal of this project is to evaluate these neural network architectures and compare them against one another. Specifically, I investigate the Memory Network described in [6] and the Dynamic Memory Network from [7].

3 Neural Network Architectures

In this project, I consider three neural network architectures: a simple GRU encoder (baseline), a Memory Network, and a Dynamic Memory Network.

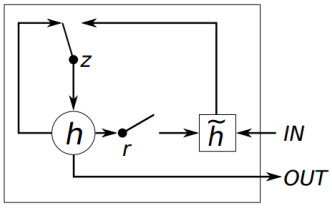

Figure 2: Gated Recurrent Layer Architecture. Taken from [8].

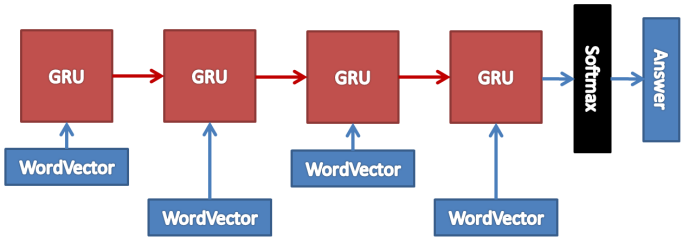

Figure 3: Simple GRU Encoder.

3.1 GRU Encoder

The baseline algorithm is simply a Gated Recurrent Unit (GRU) encoder, pictured in Fig. 3. This is simply a recurrent neural network that reads in each word of a sequence and, through a softmax layer, produces an output answer. To form the sequence, a vector representing the question is concatenated with a vector containing the article (or information from which an answer is gener-

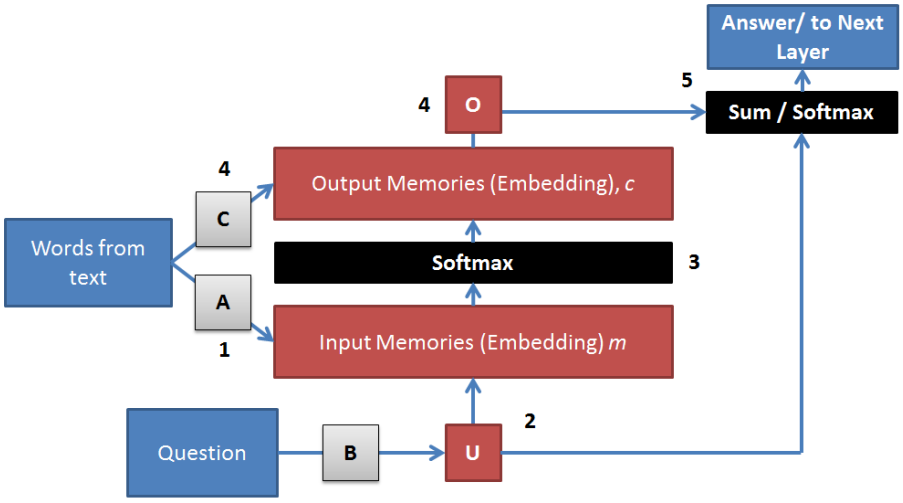

Figure 4: Memory Network Overview.

ated). While the inputs can be generated by Word2Vec, for these experiments they were either one-hot vectors or word indices.

The base unit for the encoder (and also the subsequent Dynamic Memory Network) is the Gated Recurrent Unit. The unit consists of two gates, a reset gate and an update gate, which determine whether or not to update the unit's hidden state with an input or simply propagate the previous hidden state. A detailed description of the Gated Recurrent Unit is out of the scope of this report, but one is referred to [8] for more information. The GRU was chosen over an LSTM because it is reportedly quicker to train while maintaining the LSTM's ability to deal with long sequences of information.

3.2 Memory Network

During the previous year, researchers at Facebook have developed the Memory Network as an end-to-end neural network architecture that combines a memory into which facts can be encoded as well as an attention mechanism that can make inference over these facts [6]. The fact that it is end-to-end is important because it means that the algorithm only requires (*text*, *question*, *answer*) tuples instead of relying on fact annotations, which requires less effort to produce in a dataset.

The Memory Network proceeds as follows: an input sequence of words W and an input question Q are both transformed into embeddings through embedding matrices A and B respectively. Input embeddings are transformed into *memory* embeddings m (Numbered **1** in Fig. 4) and questions into *state* embeddings u (Numbered **2** in Fig. 4). At this point a softmax layer is applied between the

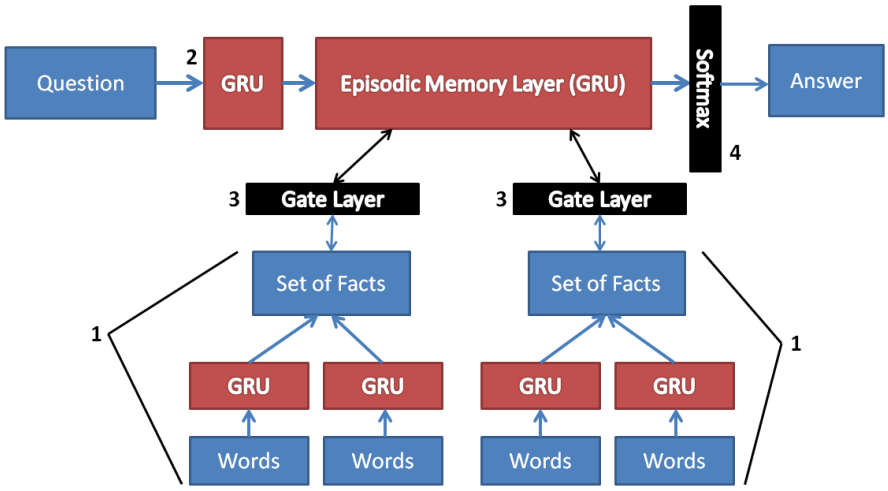

Figure 5: Dynamic Memory Network Overview. The DMN shown represents a DMN that reads a document twice.

input and question embeddings (Numbered **3** in Fig. 4):

$$p_i = \text{Softmax}\left(u^T m_i\right) \tag{1}$$

Intuitively, this softmax determines which memories are relevant to answering the question. In addition to the memory embeddings m, the input stream is also embed into an output embedding, C. The result of the softmax layer, Eq. 1, is multiplied by this output embedding, mumbered **4** in the figure. This result passes through a summation with the question state u, numbered **5**. Finally, this can either be passed to a softmax for a final answer or passed to another memory network layer. That is, the layers are stacked and the result of one is passed to another as an input state u. In the implementation, the layers are stacked 2-3 times before an answer is output.

3.3 Dynamic Memory Network

The final model investigated is the Dynamic Memory Network as presented in [7]. The dynamic memory network is somewhat similar to the Memory Network; however, it relies more explicitly on GRU encodings and adds the potential of "re-reading" sentences.

First, each sentence is encoded into a fact through a GRU-based RNN. This RNN reads each word in the sentence and updates its hidden state. The final output, a "fact" encoding, is passed to the next layer. This process is repeated for the entire document to generate a set of fact encodings, which is shown as **1** in Fig. 5.

Next, given a set of facts from the entire document, it must be determined which of them is relevant to answering the given question. For this, a gating mechanism is deployed, numbered **3** in the figure. The functional form of this gate is:

$$z\left(c_t^i, m^i, q\right) = \left[c_t^i, m^i, q, c_t^i \circ q, c_t^i \circ m^i, |c_t^i - q|, |c_t^i - m^i|, {c_t^i}^T W^b q, {c_t^i}^T W^b m^i\right] \tag{2}$$

$$g_t^i\left(c_t^i, m^i, q\right) = \sigma\left(W^a \tanh\left(W^c z\left(c_t^i, m^i, q\right)\right)\right) \tag{3}$$

where c_t^i is the t^{th} fact encoding at the i^{th} reading, m^i is the hidden state of the *Episodic Layer* at reading i (described below), and q is the question encoding. The magnitude of gate g_t^i - a scalar - determines whether or not fact encoding c_t^i will be propagated forwards. This is done through a update equation in the form of a modified GRU-RNN:

$$h_t^i = g_t^i \text{GRU}\left(c_t^i, h_{t-1}^i\right) + \left(1 - g_t^i\right) h_{t-1}^i \tag{4}$$

Finally, the final state of this modified-GRU-RNN (that is, when variable t is equal to the total number of facts) is passed to another GRU-RNN representing the *Episodic Layer*. This layer provides the ability for the Dynamic Memory Network to read the document multiple times (where the current reading is indexed by i in the notation above). This may be advantageous when the ordering of facts matters but when the second fact in the text occurred *before* the first fact in the text, i.e. *John went to the store on Saturday. On the Wednesday prior, John drove to California.* Therefore, the example shown in Fig. 5 is shown reading a document twice, producing two sets of facts (one fact for each sentence), and choosing episodes for each of these. Finally, the result of the *Episodic Layer* is fed through a decoder to produce an answer.

4 Datasets

One of the challenges with QA tasks is designing datasets that are nontrivial and require inference to answer questions but are also large enough that the neural network will learn well. For this project, I investigated the Babi Tasks, a manually annotated dataset from Wikpedia, and a large Google-CNN dataset.

4.1 Babi Tasks

The Babi Tasks are a set of 20 tasks created by researchers at Facebook and are described in [15]. The motivation is to create a set of tasks to test a general question answering algorithm. The algorithm should be general in that it should be able to perform well on all of them. The set contains a wide variety of tasks. For example, understanding orientation: *The hallway is east of the bathroom. The bedroom is west of the bathroom. What is the bathroom east of?* or understanding counting objects: *Mary moved to the bathroom. John went to the kitchen. Mary took the football there. How many object is Mary carrying?*

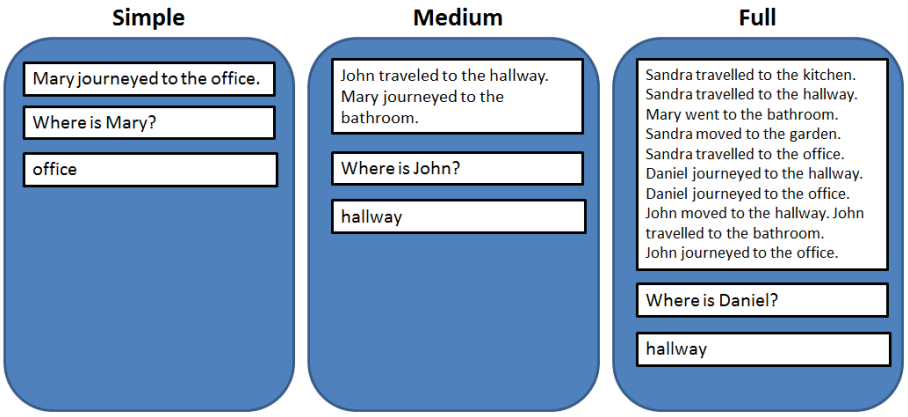

Figure 6: Examples of Babi tasks used for this project.

For this project, I focused on Babi Task 1, which involves determining where people went. I split the task into simple, medium, and full versions as pictured in Fig. 6. The simple and medium were primarily for debugging - even the simple GRU encoder can easily obtain 100% correct responses on the simple version.

An advantage of the Babi set is that it tests the neural networks on attention and inference. However, their corpora is limited in size (Babi Task 1 only contains about 20 different words) and does not represent actual corpora that one may see in the real world.

4.2 WikiQA

The second dataset that I used was one suggested by researchers at Carnegie Mellon University involving Wikipedia articles and questions that can be answered upon reading the articles [9]. One example is shown in 7. An advantage of this dataset is that it represents a more realistic corpus size than the Babi Tasks. However, a major disadvantage of the set is that it is very small. It consists of approximately 200 articles and 5,000 questions concerning those articles. As seen in the results, the neural networks have difficulty learning anything but basic yes/no questions on this dataset.

4.3 Google - CNN

The third dataset that I considered was provided by researchers at Google and described in [10]. This dataset consists of articles from CNN (another similar dataset from Daily Mail is also described) with machine-generated question statements and answers. The question statements are generated through the simple, but effective, observation that all CNN/Daily Mail articles have accompanying human-written summaries. Words can be replaced from these sum-

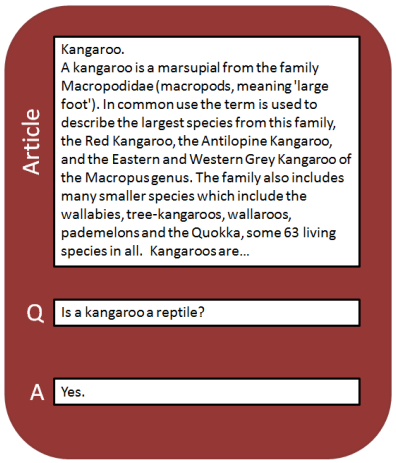

Figure 7: Example of article, question, and answer tuple from WikiQA dataset.

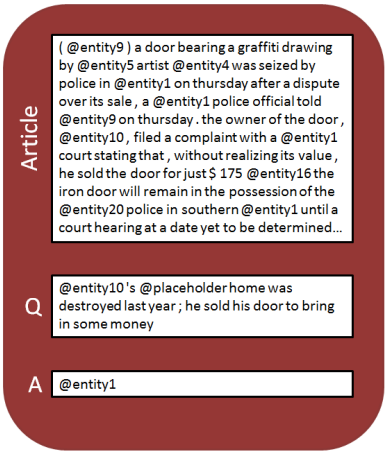

Figure 8: Example of Google-CNN article, question statement, and answer tuple.

maries to create question statements where an algorithm attempts to predict the word that was removed. The advantage with creating training examples from a source like CNN is that a huge number of examples can be produced (the full dataset is approximately one million articles).

5 Implementation

The three above-described networks were implemented and tested on the datasets, when possible. All of the networks relied on the Theano neural network library [11]. The simple GRU encoder and Memory Network were implemented in Lasagne, an easy-to-use neural network library that runs on top of Theano [12]. Furthermore, the Memory Network implementation was greatly aided by an implementation that the paper's authors provide on Github, which only needed slight modifications for this work. The Dymanic Memory Network's gating feature was difficult to implement using Lasagne and, therefore, pure Theano was used to construct this network.

Stochastic gradient descent was used to learn the models with a learning rate of 0.01. The objective was the negative log-likelihood of the training data. Gradient clippings were used on the models. Training was done using batches to improve convergence rates, with batch sizes of 30 - 100 tested. All tests were done either on my local desktop machine with a 3.4 GHz Intel Core i7 CPU and 8 GB of RAM or my laptop with a 2.7 GHz Intel Core i7 CPU and 12 GB of RAM. A GPU was not used for these experiments but would definitely be an improvement for future work as the large datasets (Google-CNN) take a full day for processing 5-6 epochs.

Dataset	GRU Encoder	MemNet	DynamMemNet
Babi Task 1 (Simple)	100%	100%	100%
Babi Task 1(Medium)	100%	100%	100%
Babi Task 1(Full)	18%	100%	100%
Wiki QA (Yes/No)	-	76%	-
Wiki QA (Full)	-	14%	-
Google-CNN, 3,000 Examples	-	51.2%	-
Google-CNN, 30,000 Examples	-	73.5%	-

Table 1: Accuracy of neural networks on various datasets.

6 Results

Results for the experiments are shown in Table. 1. All algorithms perform well on Babi Task 1, the simple and medium versions (where examples of the Simple, Medium, and Full Babi task are provided in Fig. 6). The Memory Network and Dynamic Memory network are both able to achieve 100% on the full Babi Task 1 as well. Regarding tests on larger datasets, the Memory Network was the only network to achieve convergence. This is most likely due to the considerably longer sequences of the WikiQA dataset and Google-CNN dataset that the GRUs of the GRU encoder and Dynamic Memory Network must encode.

Concerning the Memory Network's performance, one can see the effect of increased dataset size. The Wikipedia QA dataset does relatively well when only Yes/No answers are used for testing (an approach suggested in [13]). However, when all possible answers are used, the result is merely 14% accuracy. The Google-CNN dataset performs much better - accuracy is improved by more than 20% moving from 3,000 training examples to 30,000.

Note that the training and test examples for Google-CNN that generated the above results were simpler than full examples in that the articles only contained sentences that contained the desired answer. This was done because of memory limitations of my computer and rendered the task slightly simpler though still non-trivial because of the articles' complexity. An improved implementation could get around this by only loading batches of articles into memory at one time.

7 Conclusion & Future Steps

The proposed Memory Network and Dynamic Memory Network neural architectures definitely show an advantage over the baseline model. As the results show on the Babi Dataset, these models are able to deal with QA tasks well over relatively long streams of characters. The Dynamic Memory Network and Memory Network are able to correctly answer all the statements of Babi Task 1.

More difficulty was experienced getting results on the larger datasets. This difficulty involved both processing the larger datasets on an ordinary CPU and

achieving neural network convergence on larger sequences: the Google-CNN dataset contained articles of a few hundred sentences in length and sentences of a few hundred words. This was problematic for the Dynamic Memory Network, which needed to encode these sentences into facts. Training required setting very small learning rates and extreme clipping of gradients, which caused it to proceed very slowly.

Despite the difficulties of working with larger datasets, there are still clear advantages to training the deep neural models on them - as the results of moving from 3,000 to 30,000 CNN examples on the Memory Network show. These results indicate that further study of these models on larger datasets is definitely warranted, albeit with improved learning mechanisms (one possibility could be using adaptive gradient methods) and improved hardware (such as using GPUs). Another valuable comparison would be of neural and non-neural models on similar datasets. This would provide greater insight on whether or under what circumstances deep neural networks improve natural language tasks, as they have improved other fields of machine learning.

References

[1] D. A. Ferrucci, E. W. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. M. Prager, N. Schlaefer, and C. A. Welty, "Building watson: An overview of the deepqa project," *AI Magazine*, vol. 31, no. 3, pp. 59–79, 2010. [Online]. Available: http://www.aaai.org/ojs/index.php/aimagazine/article/view/2303

[2] R. Barzilay, "6.864: Advanced natural language processing, lecture 1," 2015.

[3] K. Narasimhan and R. Barzilay, "Machine comprehension with discourse relations," *ACL 2015*, 2015.

[4] R. Cooper and S. Rueger, "A simple question answering system," *Proceedings of TREC-9*, 2000.

[5] J. Berant, A. Chou, R. Frostig, and P. Liang, "Semantic parsing on Freebase from question-answer pairs," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2013.

[6] S. Sukhbaatar, A. Szlam, J. Weston, and R. Fergus, "End-to-end memory networks," *NIPS 2015*, 2015.

[7] A. Kumar, O. Irsoy, P. Ondruska, M. Iyyer, J. Bradbury, I. Gulrajani, and R. Socher, "Ask me anything: Dynamic memory networks for natural language processing," *CoRR*, 2015.

[8] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *NIPS 2014 Deep Learning and Representation Learning Workshop*, 2014.

[9] N. A. Smith, M. Heilman, and R. Hwa, "Question generation as a competitive undergraduate course project," *Proceedings of the NSF Workshop on the Question Generation Shared Task and Evaluation Challenge*, 2008.

[10] K. Hermann, T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom, "Teaching machines to read and comprehend," *NIPS 2015*, 2015.

[11] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio, "Theano: new features and speed improvements," Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[12] S. Dieleman, J. Schlter, C. Raffel, E. Olson, S. K. Snderby, D. Nouri, D. Maturana, M. Thoma, E. Battenberg, J. Kelly, J. D. Fauw, M. Heilman, diogo149, B. McFee, H. Weideman, takacsg84, peterderivaz, Jon, instagibbs, D. K. Rasul, CongLiu, Britefury, and J. Degrave, "Lasagne: First release." Aug. 2015. [Online]. Available: http://dx.doi.org/10.5281/zenodo.27878

[13] P. Kapashi, Darshan; Shah, "Answering reading comprehension using memory networks," *Stanford 224d Course Project*, 2015.

[14] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *NIPS 2014*, 2015.

[15] J. Weston, A. Bordes, S. Chopra, T. Mikolov, A. M. Rush, and B. van Merrienboer, "Towards ai-complete question answering: A set of prerequisite toy tasks," *arXiv:1502.05698*, 2015.