Question Answering using Dynamic Memory Networks

Deepak Menghani
Stanford University
deemeng@stanford.edu

Abstract

This work investigates the effectiveness of Dynamic Memory Networks for Question Answering (QA) tasks. One of the distinguishing features of the Dynamic memory networks is the attention mechanism and this work investigates how these models and their attention mechanisms perform in a weakly supervised setting with only answer labels, and under strong supervision (with information on the relevant context for question). The results indicate that strongly supervised training leads to better accuracies pointing to the effectiveness of the attention mechanism.

1 Introduction

Solving the task of Question Answering (QA) is widely considered as a necessary step towards developing a general purpose AI agent that can hold a natural language dialog with a human participant. Fundamentally, a QA system can be thought of as two different tasks: retrieval and inference. The QA system must use the input (*"context"*) presented to it in some convenient internal representation (*"knowledge"*) and it must search through this knowledge bank and retrieve the right pieces of information (*retrieval*) and reason over it (*inference*) to answer the question posed to the system. Both tasks, retrieval and inference together, are critical to the success of a QA system.
One of the major reasons for studying QA is that it is extremely broad, and many NLP tasks can be reformulated in the QA setup. For example, a machine translation (MT) task, can be formulated as a QA problem by using the input sentence as *context* and question asking it to translate it to the desired language (eg. "What is the translation of this sentence in Hindi?"). This implies that devising better models for improving performance on QA tasks may be generally helpful for other NLP tasks too and serve as a general baseline model for all NLP tasks.
In this project, the goal is to explore Dynamic Memory Networks framework for QA as presented in [1]. The next section gives an literature overview. Section 3 describes the problem statement and the dataset used. Section 4 describes the approach with technical details. Section 5 provides results from the experiments conducted, which is followed by the conclusion.

2 Background

[7] provides a good discussion and comparison of new and traditional methods for machine comprehension of text(MCT), which can be thought of as general QA systems for a given text inputs and a question. Early AI methods used heuristic rules and built ontologies, but these were non-scalable across different datasets. Over the last decade, significant progress has been made on hard NLP tasks such as QA using Deep Learning methods trained with large amounts of input data.

Deep Learning approaches, such as RNNs and LSTMs, do not require hand-engineered features and have proven to be quite effective at handling large ordered sequences of inputs for tasks such as language modeling [11] and machine translation [12]. However, all of these approaches lack an effective mechanism to combine long-term memory with inference, which is critical for QA tasks. Where RNNs and LSTMs fall short, is that their memory encoded by hidden states is typically too small, and is not structured to only identify relevant pieces of information to simplify the inference task.

Recently, there has been new research on Deep-learning models with attention mechanisms [1, 5, 6, 8, 9] to solve such an issue. Memory Networks, recently introduced in [5], implement the attention mechanism with explicit inference and long-term memory components as part of the model. The memory component serves as a knowledge base to recall facts from the past. Similar approaches for combining deep networks with a memory component for other tasks have also been published recently [6]. The model implements the attention mechanism by learning a scoring function to rank relevant memories. At prediction time, the model finds the relevant memories according to the scoring function and conditions its output based on these. However, this approach processes sentences independently and not via a sequence model. It requires separate features to capture the sequence information.

Dynamic Memory Networks, introduced in [1], are a more general class of Deep Learning model, with explicit attention mechanism, that can be used for a large variety of NLP tasks such as QA, POS and sentiment analysis. In this work, we explore the application of this model to the QA task in the subsequent sections.

3 Problem Statement

3.1 Dataset

In [2], the authors introduced a dataset ("bAbI") which consists of a set of toy QA tasks as a step towards AI-complete question answering. The dataset consists of 20 different tasks in English and each task is intended to check one skill that a reasoning system must have. The list of the skills tested in each task is displayed in the Appendix Table A.1. These tasks are generated through simulation and the dataset is available at [3].
For each task, we have a list of questions available. For each question we have the following information-

1. Context – Information needed to answer the question.
2. Answer- The correct answer to the question. The answers to the questions in the dataset are single word answers (eg. Yes/No, smaller/bigger, etc.)
3. Supporting context- The relevant context that is required to the answer the particular question

Given that the dataset contains information about the supporting context, it can be used to train the model in selecting the correct context information in addition to using the answer labels.

3.2 Data Split

Since, the authors of the dataset only provide train and test datasets, we split the given training data into train (80%) and validation splits (20%) randomly for each task. The generalization performance of the model is estimated from the union of validation set for all the tasks. The hyper-parameters of the model are selected based on the model performance on the validation set.

3.3 Evaluation Metrics

The model will be evaluated on answers for all the questions in the test sets for all the 20 tasks. A single model is used in order to avoid task-specific feature engineering. Since, we have the correct answers, which are single word tokens, we calculate the accuracy of the model as the number of correct answers divided by the total number of questions. An answer is considered correct, if the output answer token exactly matches the answer from the dataset.
The authors of the bAbI, stipulate a 95% threshold accuracy to "pass" the task. So, additionally, the number of tasks passed (out of 20) is also an important metric.

4 Technical Approach & Implementation Details

4.1 Model Details

The Dynamic memory network model as described in [1] is used. The network architecture, shown in Fig 1, consists of 4 modules-

1. Input Module: This module takes in the input context and converts them into a list of hidden state encodings, $s_1, s_2, \ldots s_{Tc}$, one for each sentence. Given words $w_1^i, w_2^i \ldots, w_n^i$ corresponding to sentence i. We obtain the vector representation of the sentence as the last hidden state $s_i = h_n^i$. It is computed using a RNN with GRU cells as,
$$h_t^i = GRU(L(w_t^i), h_{t-1}^i)$$
Where, L is the word-embedding matrix.
All the GRU cells in the input module share the same weights. The dimension of h_t^i, n_{hidden}, is a hyper parameter of the model

2. Question Module: This module takes the input question and transforms it into a vector representation q using RNN with GRUs. All the GRU units in the question module share the same weights.
$$q_t = GRU(L(w_t^Q),\ q_{t-1})$$
$$q = q_T$$
Where, w_t^Q is the word at position t in the question
T is the total number of words in the question
The dimension of vector q is taken to be n_{hidden}

3. Episodic Memory Module: This module takes in the inputs $s_1, s_2, \ldots s_{Tc}$ and transforms it into the final memory state vector m^{Tm}.The module iterates over the context list multiple times to "select" particular sentences in the input context by computing scalar gate values for each s_i.
$$h_t^i = g_t^i\, GRU(c_t, h_{t-1}^i) + (1 - g_t^i)\, h_{t-1}^i$$
$$e^i = h_{Tc}^i$$
$$m_i = \text{GRU}\ (e_i,\ m_{i-1})$$
Where, i is the iteration number over the context in the module. We set a max number of iterations Tm for which the episodic module runs. The initial state m^0 is set as q. Hence, m has the dimension, n_{hidden}, same as q.

4. Answering Module: This module simply takes in the final memory and question as input and uses a projection layer to compute the final answer probabilities y
$$y = softmax(W^a m^{Tm})$$
where,
W^a has the shape label size X n_{hidden}

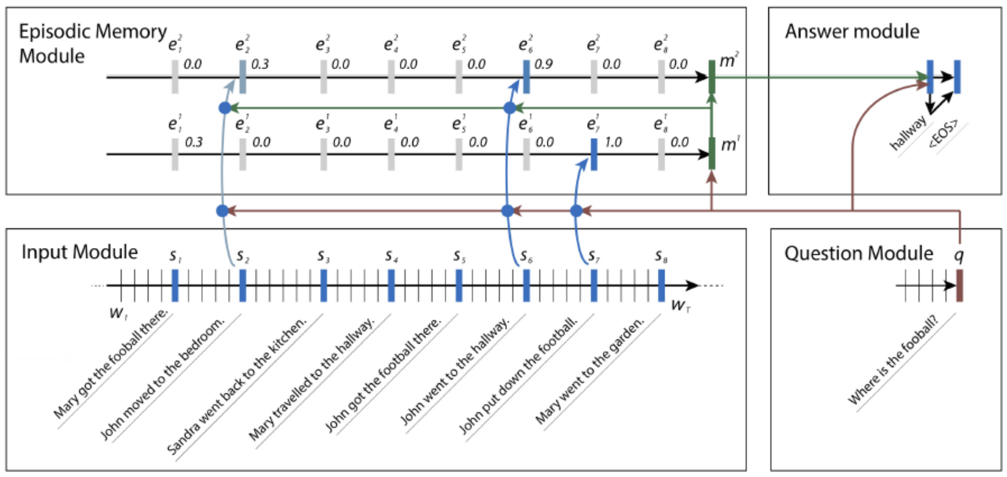

Episodic Memory Module
Answer module
Input Module
Question Module
hallway
<EOS>
Mary got the fooball there.
John moved to the bedroom.
Sandra went back to the kitchen.
Mary travelled to the hallway.
John got the football there.
John went to the hallway.
John put down the football.
Mary went to the garden.
Where is the fooball?

Figure 1: Dyanmic Memory Network Architechure as presented in [1]

4.2 Word Vectors

The current implementation of the model uses pre-trained Glove word-vectors [4] for the initialization of the word-embedding matrix. The word vectors are also updated during the training of the Dynamic memory network. The dimensionality of the word vectors are taken to be a hyper-parameter of the model

4.3 Batch-training

To implement batch training on tensor flow, we require fixed size inputs to use the same computational graph for the model. We achieve this by building the graph using a pre-defined maximum number of input sentences to the Input module and maximum number of words per sentence. Given an input of any size, we prepend with 0`s to prepare it to be fed into the model. Since, the maximum number of sentences for a particular set in the dataset is large (>250), building such a large fixed size model is quite memory and computationally intensive.

4.4 Loss Function

Weakly-supervised: Since, all datasets do not contain the information on the specific sentences in context that are required to answer the question, the weakly supervised loss function only uses the final answer labels to compute the total loss. We use a cross-entropy loss function

$$J_{WS} = \sum_i CE\ (y_i, \hat{y}_i) + J_{reg}$$

where, y_i = one-hot vector corresponding to the answer

$\hat{y}_i$ = the predicted probabilities by the model

J_{reg}= Regularization component of loss

Strongly Supervised: When we have the information about the supporting facts required by model (eg. bAbI dataset), we can use the information to train the model by incorporating it in the loss function.

$$J_s = \alpha \sum_i CE\ (y_i, \hat{y}_i) + \beta\ [\sum_t (CE(g_i^t,\ \hat{g}_i^t) +\ CE(\,1 - g_i^t, 1 - \hat{g}_i^t))] + J_{reg}$$

Where, $g_i^t\ = 1\ if\ t^{th}$ fact is required to answer question i, 0 otherwise

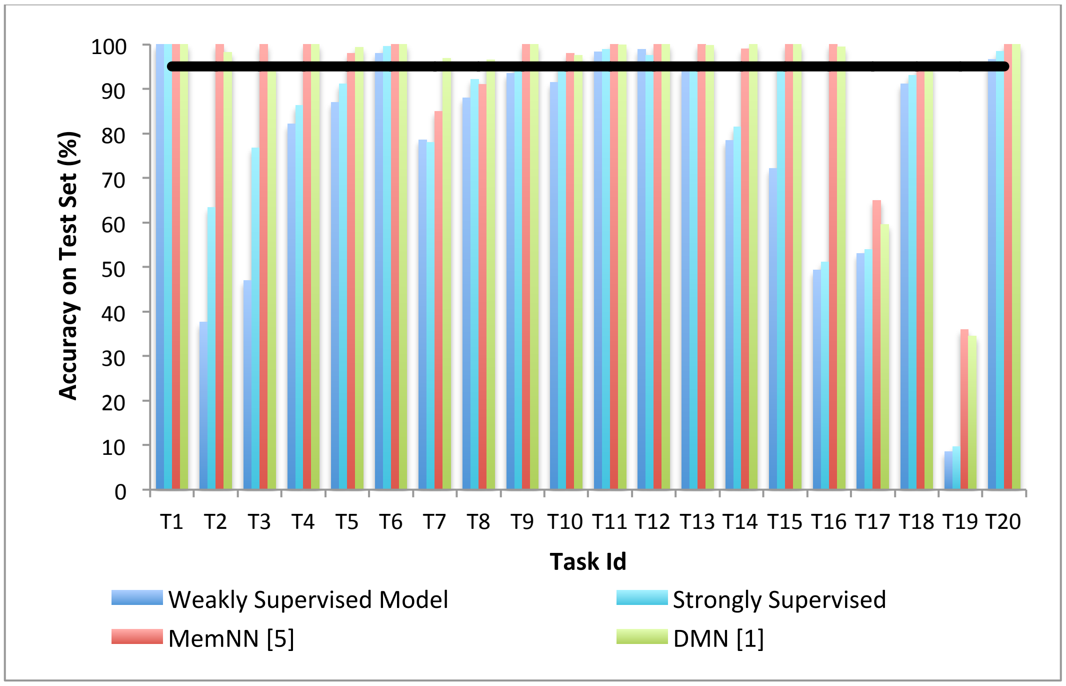
$\hat{g}_i^t$ = Value of gate in last time step in episodic memory module for t^{th} fact for ith question

5 Results

5.1 Accuracy

The model was trained using both the weakly and strongly supervised loss functions. The strongly supervised model was trained using 2-step training- First, $\alpha = 0$ and $\beta = 1$, was used to train the episodic module to identify the right gates. After convergence, for these weights, we set $\alpha = 1$ and $\beta = 1$ to include the loss due to answers. This method helps train the episodic module to learn the right gates first and then trains the weights in the answer module to output the correct answer. The number of iterations in the episodic memory was set to 5 due to memory constraints on the machine. The word vector dimension, hidden dimensions of the input module, question module, episodic memory module and answer module were set to 50. Dropout probability was set to 0.1 (keep probability=0.9). The results for both the weakly supervised and strongly supervised models on all the 20 different tasks are presented in Figure [] below. Also, shown are the results from [1] and [5] as the state of the art benchmark for the tasks.

Figure 1: Results for Weakly and Strongly Supervised Models



As we can see from the results, the strongly supervised model performs better in terms of accuracy than the weakly supervised model on all of the 20 tasks. The strongly supervised model is able to pass on 8 out of the 20 tasks compared with 16 out of 20 for the SOTA models. The final trained model under the strongly supervised setting, results in a high accuracy of correct gate selection for all the 20 tasks on the test set which indicates that the attention mechanism for the model architecture is effective.

6. Conclusions & Future Work

Dynamic Memory Networks are an effective framework for handling QA tasks. Their extensibility to other NLP tasks by modifying the question and answer modules also makes them a good candidate for a general benchmark model. The results from the strongly supervised and weakly supervised models, show the effectiveness of the attention mechanism which is the key differentiator of the model compared to other Deep Learning architectures like RNNs and LSTMs.

The implementation of the model was not able to achieve the same state of the art benchmarks as in [1]. There might be a few potential reasons for this. Due to the complexity and time to train the model, it was not possible to carry out an exhaustive hyper parameter search. Different weights for the gate loss, word vector and hidden dimensions can be tried to better tune the model. The number of iterations in the episodic module can be increased to achieve better results. A bi-directional RNN or LSTM can be tried in the input module instead of a vanilla RNN to better encode the input information and bring the model performance closer to the benchmarks. Since, the gate accuracy of the model is good, the problem for low test set answer accuracies could be caused by bad hyper-parameter selection or insufficient training of weights in the answer module.

Finally, investigation into effectiveness of Dynamic Memory networks for more real world datasets compared to the synthetically generated bAbI test needs to be carried out to help understand their real-time performance.

References

[1] Kumar, A., Irsoy, O., Su, J., Bradbury, J., English, R., Pierce, B., Ondruska, P., Iyyer, M., Gulrajani, I., & Socher, R. (2015). Ask me anything: Dynamic memory networks for natural language processing. arXiv:1506.07285

[2] Weston, J., Bordes, A., Chopra, S., Rush, A.M., Merriënboer, B.v., Joulin, A., & Mikolov, T. (2015). "Towards AI-Complete Question Answering: A Set of Prerequisite Toy Tasks", *arXiv:1502.05698*

[3] BAbI. (n.d.). Retrieved May 17, 2016, from https://research.facebook.com/research/-babi/

[4] Pennington, J., Socher, R. & Christopher D. Manning(2014). GloVe: Global Vectors for Word Representation.

[5] Weston, J., Chopra, S., & Bordes, A. (2015). Memory Networks. arXiv:1410.3916v8gr

[6] Sukhbaatar, S., Szlam, A., Weston, J. & Fergus, R. End-to-End Memory Networks(2015). arXiv:1503.08895

[7] Burges, C. (2013). Towards the Machine Comprehension of Text: An Essay. Microsoft Research Technical Report MSR-TR-2013-125, 2013

[8] Bahdanau, D., Cho, K. & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. International Conference on Learning Representations (ICLR).

[9] Graves, A., Wayne, G. & Danihelka, I. Neural turing machines(2014). arXiv preprint:1410.5401

[10] Fader, A., Luke & Z., Etzioni, O. (2013). Paraphrase-Driven Learning for Open Question Answering. ACL (1)

[11] Mikolov, T., Zweig, G. Context dependent recurrent neural network language model (2012). In SLT, pp. 234–239. IEEE. ISBN 978-1-4673-5125-6.

[12] Sutskever, I., Vinyals, O. & Le, Q.V. (2014). Sequence to sequence learning with neural networks. NIPS, 2014

Appendix

Table A.1: Task Descriptions of the bAbI dataset [2]

Task Id	Task Id
T1	Single Supporting Fact
T2	Two Supporting Facts
T3	Three Supporting Facts
T4	Two arguments relations
T5	Three arguments relations
T6	Yes/no questions
T7	Counting
T8	Sets
T9	Simple Negation
T10	Indefinite Knowledge
T11	Basic Coreference
T12	Conjunction
T13	Compound Coreference
T14	Time reasoning
T15	Basic deduction
T16	Basic induction
T17	Positional reasoning
T18	Size reasoning
T19	Path finding
T20	Agent`s motivation