# Question Answering Using Deep Learning

Durga Harish
Oregon State University
dayapuld@oregonstate.edu

Duy Nguyen
Oregon State University
nguyend6@oregonstate.edu

Mario Gomez
Oregon State University
gomezfem@oregonstate.edu

## 1. Abstract

This paper is submitted as a requirement for deep learning course work. In this project, we deal with Question answering using Deep Networks. We explored different deep architecture available in the literature. We focused mainly on LSTM, GRU and state-of-the-art Memory Networks ( Soft Attention, Hard Attention). We introduced two novels ideas in the paper which are not present in original papers[3][6].(i) selecting story sentences using Jaccard Similarity (ii) Adding Gaussian noise to the gradients which were motivated by (Neelakantan.A et.al[4]). We evaluated these networks using bAbI dataset, which is a synthetic dataset. By using Jaccard similarity, we observed the performance for some tasks improved significantly . Whereas, adding Gaussian noise to the gradient didn't have a positive impact on accuracy. At last, we tunned different hyperparameters and choose the one having highest test accuracy.

## 2. Introduction

In the recent year, there has been an increasing interest in the application of deep learning methods in information retrieval, information extraction, and natural language processing.

One of the many challenges in natural language processing is to map raw text to a well-known entity, known as trivia, where four to six sentences are provided and are associated with a particular answer, typically a fact. This is also known as Factoidal Question and Answering. Each of the sentences is guaranteed to have clues that identify its answer, even without the context of other sentences. The typical approach to Question and Answering based on NLP techniques follow the following general architecture[5]:

- Question analysis: the question is analyzed to produce a set of keywords for information retrieval.
- Document or passage retrieval: the keywords are used to perform a search for information retrieval.
- Answer extraction: given the top passages or documents, the answer extraction produces a list of candidates and ranks them according to some function.

Previous work on memory networks used recency approach to select sentences from the story. Our hypothesis is that when the story size is greater than memory size, using recency approach may lose important information to answer the query. Instead, one has to smartly choose the sentences to fit in the memory module. In this project, we applied Jaccard similarity approach to filter sentences and select them for training. This report covers the performance using different memory sizes using Jaccard and without Jaccard. This process can also be thought as a pre-processing layer for Memory networks.

The paper is structured as follow. Section 2 introduces the three different recurrent units that are used in this paper. They include Long Short Term Memory, Gated Recurrent Unit, and End-to-End Memory Networks. In Section 3, we talk about the Data that are used in our project. Next, we discuss our result in Section 4. Section 5 is our conclusion of the paper. Finally, Section 6 describes our limitation of this project.

### 2.1. Model Description

#### 2.1.1 Long Short-Term Memory[2]

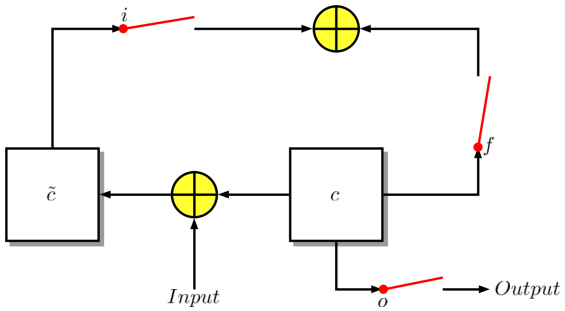


Figure 1. Long Short-Term Memory Unit

The idea behind the Long Short-Term Memory is that given an input and a previous state, we can update the memory cell ($c$) by calculating a candidate state ($\tilde{c}$):

$$\tilde{c}_t^j = \sigma_{tn}(W_c x_t + U_c h_{t-1})^j$$

where $\sigma_{tn} = tanh()$, $j$ is layer, and $t$ is time stepThe memory cell update is then moderated by the input ($i$) and forget

($f$) gates, i.e. content being added or forgotten in the unit, as

$$c_t^j = f_t^j c_{t-1}^j + i_t^j \tilde{c}_t^j$$

and,

$$f_t^j = \sigma_{sg}(W_f x_t + U_f h_{t-1} + V_f c_{t-1})^j$$
$$i_t^j = \sigma_{sg}(W_i x_t + U_i h_{t-1} + V_i c_{t-1})^j$$

where $\sigma_{sg}$ is the sigmoid function. Lastly, once memory cell has been updated, the output gate and unit's output is computed as

$$o_t^j = \sigma_{sg}(W_o x_t + U_o h_{t-1} + V_o c_t)^j)$$
$$h_t^j = o_t^j \sigma_{tn}(c_t^j)$$

Figure 1 shows an example of the circuit logic of a LSTM

#### 2.1.2 Gated Recurrent Unit[1]

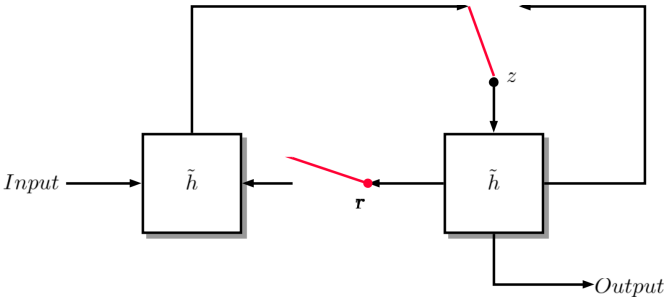


Figure 2. Gated Recurrent Unit

The idea behind the Gated Recurrent Unit is quite simple, Figure 2 shows an example of the circuit logic of a GRU. Given an input value, a candidate ($\tilde{h}$) is generated as:

$$\tilde{h}_t^j = \sigma_{tn}(W x_t + r_t \odot (U h_{t-1}))^j$$

where $\sigma_{tn} = tanh()$, $j$ is layer,$\odot$ is the hadamard product, and $t$ is time step. Then the update gate ($z$) and the reset gate ($r$) are calculated as follow:

$$z_t^j = \sigma_{sg}(W_z x_t + U_z h_{t-1})^j$$
$$r_t^j = \sigma_{sg}(W_r x_t + U_r h_{t-1})^j$$

where $\sigma_{sg}$ is the sigmoid function. The update function then decides how much the activation gate ($h$) is updated:

$$h_t^j = (1 - z_t^j) h_{t-1}^j + z_t^j \tilde{h}_t^j$$

#### 2.1.3 End-to-End Memory Networks

The End-to- End memory network has two modules, Memory module and controller module. In the case of QA Tasks, the memory module $\{m_1, m_2, m_3, ....m_n\}$ stores the information of the story. In more general, one can store images or any other relevant information to answer the query q. The memory module stores a fixed amount of memory n. The controller module sends input to the memory module and generates the answers. There are different variants of memory networks. In this work, we considered weakly supervised and soft attention mem networks. Figure 3 depicts the blue print of end-to-end memory networks. The input vector which is a query sentence $\vec{u}$ is inputted to the memory module. Then, the vector is taken a dot product with memory vectors $\{m_1, m_2, m_3, ....m_n\}$, after that it is applied to at softmax layer. The weighted sum of the memory vector is added to the previous state of the query vector. This process is executed for $h$ hops. This is the basic architecture of the End-end Memory network. The network is trained from output to memory and input layer using backpropagation mechanism (SGD), which makes the name end-to-end. Previously, Weston et.al (2014)[3] used Hard Attention instead of soft attention, which made the network difficult to train using backpropagation. As most of the stories have a temporal component in it, In order to consider time factor, we add a time tag for every sentence in the story based on the position of the sentence in the story.

## 3. Data

We use bAbI dataset[6] from Facebook for our project. The dataset consists of 20 prerequisite question and answer tasks. The tasks are range from simple problems such as answering the question using one, two, or three supporting facts to more complex problems such as induction and deduction reasoning. The complete list of the task is listed in the table below 1. The dataset is in two languages(Hindi,English), we considered English for the analysis. The answers for each task in babI data are single-worded. We chose to use bAbi as our starting point because it is well-known and used dataset.

Table 1. bAbI Data Task List

| | |
|---|---|
| T1: Single Supporting Fact | T11: Basic Coreference |
| T2: Two Supporting Facts | T12: Conjunction |
| T3: Three Supporting Facts | T13: Compound Coreference |
| T4: Two Argument Relations | T14: Time Reasoning |
| T5: Three Argument Relations | T15: Basic Deduction |
| T6: Yes/No Questions | T16: Basic Induction |
| T7: Counting | T17: Positional Reasoning |
| T8: Lists/Sets | T18: Size Reasoning |
| T9: Simple Negation | T19: Path Finding |
| T10: Indefinite Knowledge | T20: Agents Motivations |

Table 2. Shows the accuracies for LSTM and GRU for baseline purpose

| Task# | LSTM | GRU | Task# | LSTM | GRU |
|---|---|---|---|---|---|
| T1 | 50 | 47.9 | T11 | 72 | 67.6 |
| T2 | 20 | 29.8 | T12 | 74 | 63.9 |
| T3 | 20 | 20.0 | T13 | 94 | 91.9 |
| T4 | 61 | 69.8 | T14 | 27 | 36.8 |
| T5 | 70 | 56.4 | T15 | 21 | 51.4 |
| T6 | 48 | 49.1 | T16 | 23 | 50.1 |
| T7 | 49 | 76.5 | T17 | 51 | 49.0 |
| T8 | 45 | 68.9 | T18 | 52 | 90.5 |
| T9 | 64 | 62.8 | T19 | 8 | 9.0 |
| T10 | 44 | 45.3 | T20 | 91 | 95.6 |

Table 3. This table shows the pass/fail statistics for Simple GRU and LSTM model.

| Statistics | Pass(>.9) | Fail |
|---|---|---|
| LSTM | 2 | 18 |
| GRU | 3 | 17 |

## 4. Results and Discussion

As mentioned in the data section, the answers for the babI tasks are single-worded. In order to calculate the accuracy of the model, we matched the predicted output of the model with the actual answer for the query q. As there are too many tasks, we classified tasks using Pass and Fail. Any task having accuracy $> 90\%$ on testing data is assigned to Pass class otherwise to Fail class.

### 4.1. Base-line Accuracy using GRU and LSTM

To show the advantage of Memory networks kind of architecture, we worked on simple GRU and LSTM model. We used different sets of learning rate and fixed 80 epochs for training the model. The accuracies are shown in the Table 2, and pass/fail statistics are shown in Table 3. The results using LSTM and GRU are poor, this shows the basic LSTM and GRU kind of architecture are not good at handling these kinds of task. The behavior can be attributed to the unordered facts in the story, long term relationship with query and sentences. So, adding a memory unit can significantly improve the performance. In the next section, we dedicate analysis on memory networks.

### 4.2. End-to-End Memory Network

In order to check the potential of the memory network using hard Attention mechanism without any further enhancements. We trained memory network using two different base units LSTM, GRU. The results of pass/fail statistics of the trained network shown in Table 4. From next section, we show the results by adding Jaccard Similarity and Gaussian Noise to the Memory Network.

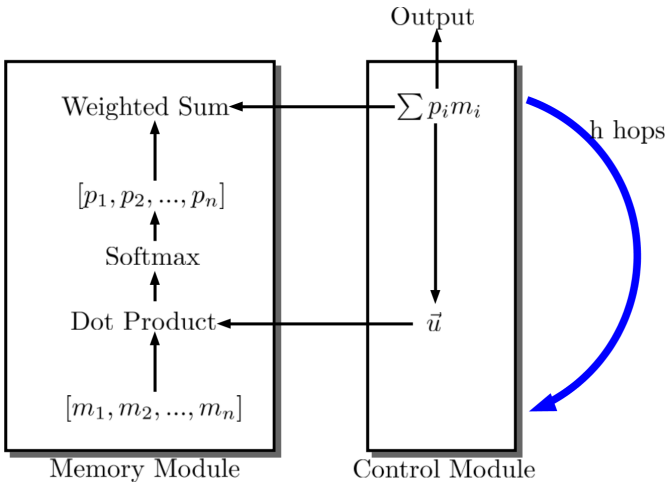


Figure 3. End-to-End Memory Module

Table 4. Results for Hard Attention Memory Networks

| Test | Pass (>.9) | Fail |
|---|---|---|
| Memory net with GRU | 10 | 10 |
| Memory net with LSTM | 9 | 11 |

#### 4.2.1 Jaccard Similarity

The Jaccard similarity metric (JSM) is defines as the ratio of the intersection over the union as shown in Equation 1. The JSM has been included in our model as it strengths the relation between the question and the supporting sentences to improve the answer selection as it only sorts the highest ranked sentences.

$$J = \frac{A \cap B}{A \cup B} \tag{1}$$

This metric has been implemented to the complete data set; however, a focus on was given to tasks whose story length is greater than the memory size. Four different memory sizes(20,50,70,100) were tested on tasks having story length greater than memory size.

Figure 4 shows the result on the application of the JSM and the results without the JSM along with the number of hops used in a network of size 20. For these particular tasks, the addition of the JSM provides significant performance improvement to task 6 and some improvements to task 7 and 8. Similarly, applying the JSM to a network of size 50, the performance of task 2, 3, and 5 is significantly increased as shown Figure 5. Moreover, the best performance is achieved when the number of hops is 2, and adding additional hops seems bring no benefit. Furthermore, the addition of more memory units seems only to have little to no improvement on answer selection as shown in Figure 6 and Figure 7

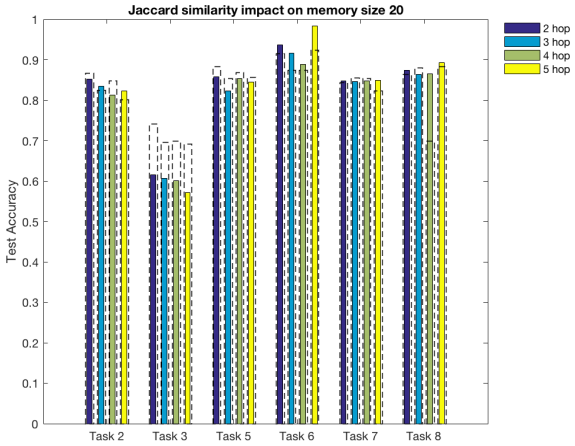


Figure 4. Results of the end-to-end network of size 20 with different number of hops. The dashed line represents the results without the use of the Jaccard similarity metric and the color bars represent the result obtained using a Jaccard similarity metric

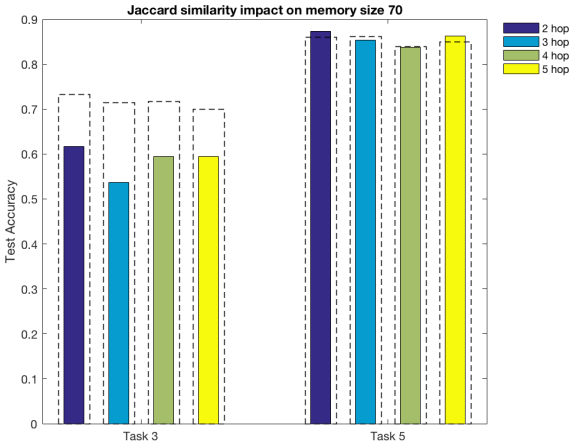


Figure 6. Results of the end-to-end network of size 70 with different number of hops. The dashed line represents the results without the use of the Jaccard similarity metric and the color bars represent the result obtained using a Jaccard similarity metric

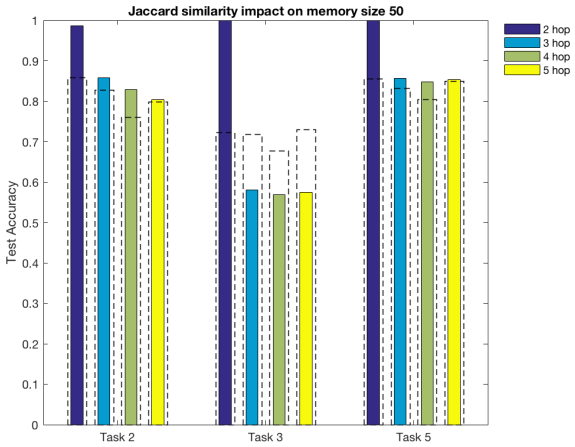


Figure 5. Results of the end-to-end network of size 50 with different number of hops. The dashed line represents the results without the use of the Jaccard similarity metric and the color bars represent the result obtained using a Jaccard similarity metric

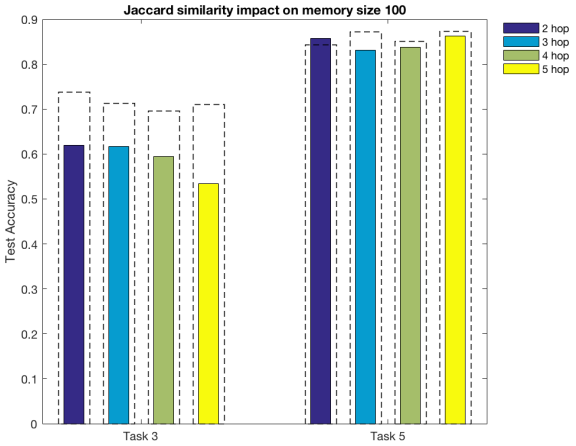


Figure 7. Results of the end-to-end network of size 100 with different number of hops. The dashed line represents the results without the use of the Jaccard similarity metric and the color bars represent the result obtained using a Jaccard similarity metric

#### 4.2.2 Gaussian Noise

As indicated by Arvind Neelakantan [4]'s paper, adding gradient noise helps to avoid overfitting and can result in lower training loss for deep and complex architectures. Thus, the application of Gaussian Noise seem was considered. However, unlike our expectation, adding the Gaussian Noise do not have any significant effect on the performance of our model. Table 5 shows the results of 3 different Gaussian Noises (0.1, 0.01, 0.001) added to the model. In order to summarize the results of our 20 task, we divided the tasks into two categories: pass and fail using their testing accuracy, a task is passed if its testing accuracy is greater than 0.9 else it is said to be failed. As we increased the noise from 0.001 to 0.1, we did not see any positive impact on our network.

Table 5. Table shows the pass/fail statistics for different gaussian noises

| Gaussian Noise(Std) | Pass(>.9) | Fail |
|---|---|---|
| 0.001 | 9 | 11 |
| **0.01** | **10** | **10** |
| 0.1 | 7 | 13 |

#### 4.2.3 Learning rate and number of hops

As we did not see any significant improvement by adding Gaussian Noise, we tuned different parameters learning rate and number of hops. For this experiment, We chose four different hop number 2,3,4, and 5 and tested it on five different learning rate: 0.001, 0.005, 0.01, 0.05, and 0.1. The result is showed in Table 6 , and we used the same sorting system (pass and fail) as the Gaussian Noise result to showcase our data. Our result indicated that 0.01 is the best learning rate for our network as it performed the best across four different hops. Furthermore, we have the best result is when the hop is equal to 4.

Table 6. Variation of the pass/fail statistics for different learning rates, hops. Bold one shows the best performance

| Hop | Learning Rate | Fail | Pass |
|---|---|---|---|
| 3 | 0.001 | 15 | 5 |
| 3 | 0.005 | 13 | 7 |
| **3** | **0.01** | **11** | **9** |
| 3 | 0.05 | 15 | 5 |
| 3 | 0.1 | 16 | 4 |
| 4 | 0.001 | 15 | 5 |
| 4 | 0.005 | 13 | 7 |
| **4** | **0.01** | **8** | **12** |
| 4 | 0.05 | 16 | 4 |
| 4 | 0.1 | 18 | 2 |
| 5 | 0.001 | 15 | 5 |
| 5 | 0.005 | 12 | 8 |
| **5** | **0.01** | **10** | **10** |
| 5 | 0.05 | 16 | 4 |
| 5 | 0.1 | 20 | 0 |
| 6 | 0.001 | 15 | 5 |
| 6 | 0.005 | 14 | 6 |
| **6** | **0.01** | **10** | **10** |
| 6 | 0.05 | 18 | 2 |
| 6 | 0.1 | 19 | 1 |

## 5. Conclusion

This report consists of the application of an end-to-end memory network, in which multiple parameters and additional metrics were implemented to improve the performance on specific tasks of the bAbI data set. The addition of the Jaccard similarity metric showed to have a positive impact on the performance of some tasks. The addition of a small Gaussian noise had very little impact in our model. The selection of the number of hops and the learning rate of the Adams optimizer does have an impact on the model's performance. Overall, we were able to fine tune many parameters and added other metrics and noise to the memory network to increase the performance on specific tasks.

## 6. Limitations

Some of the limitations for us were at the beginning of the project as we tried to run an LSTM model, however, it would have taken multiple days to get similar benchmarks as the authors in [6] as it took roughly 700 seconds per iteration (couldn't be run in pelican for compatibility issues). The structure was modified to include an extra drop out layer between RNN layer and the repeated vector layer.

Resources limitation also affects another aspect of our study which is the number of trails we can run for our data. For instance, in multiple machine learning projects, the researchers run their code multiple times and take the average of those results to validate their findings. However, we can only run each of our results once at a time in Pelican due to the long running time.

Another limitation about our reports can be the method we use to present our result. We do not know what is the best way to represent the accuracy of all 20 tasks, Hence, we decide to use two categories pass and fail to summary the successful rate of the network. However, by discretized them, we do not know precisely the performance of each task between different models. For example, we see that our vanilla memory network4 perform as good as our end-to-end memory network6 (both of them has 10 passes and 10 fails). But, if we look at the accuracy of each task, we might see an improvement in the accuracy of the end-to-end network.

## References


[1] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014.

[2] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[3] A. B. Jason Weston, Sumit Chopra. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014.

[4] A. Neelakantan, L. Vilnis, Q. V. Le, I. Sutskever, L. Kaiser, K. Kurach, and J. Martens. Adding gradient noise improves learning for very deep networks. *arXiv preprint arXiv:1511.06807*, 2015.

[5] M. Wang. A survey of answer extraction techniques in factoid question answering. In *Proceedings of the Human Language Technology Conference and North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, 2006.

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