Memory Network Analysis for Question-Answering

Saharsh Samir Oza
University of Texas, Austin
oza.saharsh@utexas.edu

Chandana Amanchi
University of Texas, Austin
chandana@utexas.edu

Abstract

Question-Answering(QnA) is a challenging problem in Natural Language Processing. Memory Networks have been shown to perform well on this task [11]. In this work, Memory networks are studied using bAbi [1] Question-Answering dataset. The focus of the project is to analyze the errors on bAbi tasks and interpret the learnt embeddings of the model. Various experiments are performed to study the sensitivity of the model to hyperparameters. The trained model is able to achieve the results presented in [11], for the 20 tasks. Studying the trained embeddings reveal that for the same accuracy, some models may be more interpretable than others.

1 Introduction

Question-Answering is a popular field of NLP with its applications ranging from Information Retrieval to Chatbots. There are two major types of QnA problems, namely, Closed-domain and Open-domain QnA. In closed-domain problems answers are predicted based on given information where as open-domain problems require world-knowledge to answer the questions. The reasoning involved in QnA makes the task complex.

A number of QnA datasets have been released in recent times to foster research in the field. bAbi is a synthetic QnA dataset released by *Facebook*. It consists of 20 diverse tasks with the goal of developing learning algorithms for text understanding and reasoning. Each of these tasks test one aspect of the intended behavior of a good QnA system. Each training/testing example in this dataset consists of few sentences that form a story and a question with a single or multiple words answer.

Earlier models on QnA problem used sequential recurrent neural networks such as LSTM and GRU [12] [2]. However, these models cannot capture dependencies for very long distances. To overcome this problem, end-to-end Memory Networks were developed [11]. They work by capturing correct attention through memory hops. A variant of this architecture, Dynamic Memory Networks [8] have been introduced to solve QnA problems using episodic memory. Recently, Convolutional Match Networks [9] have also been developed that show improved performance on this task.

In this work, Memory Networks are used for the QnA problem. The goal is three fold. First, to achieve SOTA results as presented in [11]. This required significant hyperparameter tuning. Second, to analyze the results and third to be able to interpret the trained model.

In order to achieve good performance, various hyperparameters had to be tweaked. These include the number of layers in the network, the word embedding size used for the words in the data, regularization value and the non-linearity used to predict the output at each layer. The sensitivity of the network on each of these across the 20 tasks is studied.

Results and analysis involved computing the confusion matrix and F1 score of specific tasks. The results reveal that good accuracy correlates well with a good F1 score.

For the final goal of model interpretation, the attention flow property of memory networks was studied based upon the hypothesis that a trained memory network should enable good attention flow across memory hops of the network. In order to measure this property the intersection over union (IoU) metric was used. The impact of IoU on test accuracy and learnt embeddings was studied. This was backed with qualitative results.

The rest of the paper is organized as follows. Section 2 presents details on the bAbi dataset. Section 3 discusses different models for QnA problem. Section 4 gives the experimental results on tuning hyperparameters of the memory network across all tasks. Section 5 analyses the results obtained and section 6 discusses the interpretation of the trained model and specifically the learnt embeddings. Finally, Section 8 summarizes the work and Section 9 discusses future work.

2 Dataset

In this section, details of the *bAbi* dataset are discussed. *bAbi* is a synthetic QnA dataset from Facebook with 20 QnA categories (tasks). Each task has 1000 training and 1000 testing examples. Each example consists of a story, a question, an answer and supporting facts. Every story consists of a few sentences. Answer is a single word or a list of words. Supporting facts give the relevant sentences from the story that are needed to answer the question. Few examples from the tasks are presented in Figure 1.

Task 3: Three Supporting Facts
John picked up the apple.
John went to the office.
John went to the kitchen.
John dropped the apple.
Where was the apple before the kitchen? A:office

Task 6: Yes/No Questions
John moved to the playground.
Daniel went to the bathroom.
John went back to the hallway.
Is John in the playground? A:no
Is Daniel in the bathroom? A:yes

Task 7: Counting
Daniel picked up the football.
Daniel dropped the football.
Daniel got the milk.
Daniel took the apple.
How many objects is Daniel holding? A: two

Task 19: Path Finding
The kitchen is north of the hallway.
The bathroom is west of the bedroom.
The den is east of the hallway.
The office is south of the bedroom.
How do you go from den to kitchen? A: west, north
How do you go from office to bathroom? A: north, west

Figure 1: bAbi tasks examples

It is instructive to look at a few examples in Figure 1 to understand the complexity of the problem. To answer the question in Task 3, the network needs to attend to the first three sentences and also understand the timeline. In Task 7, the network needs to maintain a map of each object and their corresponding counts and a mapping of verbs like *dropping* to *subtract*, *picking* to *addition*. Reasoning in Task 19 is further complicated as the network not only needs to attend to the corresponding sentences, but also needs to make a deduction that if A is east of B, B is west of A. From these examples it is clear that each of these tasks tests different aspects of the reasoning of the network. Many of these tasks require the network to attend to multiple sentences. The distance between the sentences to be attended could be longer. The order in which the sentences should be attended differs in different questions. Some of these tasks are even challenging to humans. The fact that each task's goal is clearly specified makes the bAbi dataset ideal to perform interesting analysis on Memory Networks.

Training on this dataset can be done separately for each task or jointly for all the tasks. In joint training, the model should typically perform better with better generalization due to more data. Single training allows better analysis of the trained embeddings and hence helps in model interpretation. Another choice of training is weakly supervised and strong supervised training. In weakly supervised, the supporting facts are not used in the training process where as its used in strong supervised. Here, weakly supervised training is used in all the experiments since the goal is to test the ability of memory networks to attend to relevant sentences in the story.

Figure 2 shows the story length distribution of bAbi dataset. It is observed that most of the stories have less than 20 sentences. But while training the model, stories are padded to have the size of maximum story length for all the stories to be of the same length.

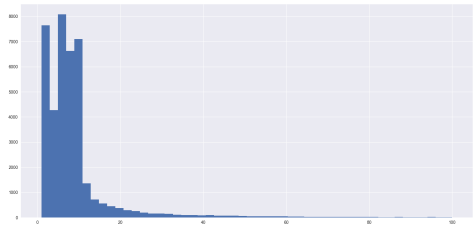


Figure 2: Story length distribution

3 Models

In this section two models are discussed for the QnA problem. The first is a seq2seq model and the second is a memory network.

3.1 Seq2seq model

Seq2seq models consist of a RNN to encode the story into a single vector and the query into a single vector. The two vectors are then concatenated as shown in Figure 3. This encoded vector is then used as input to a decoder RNN which predicts each word of the correct answer. For the bAbi tasks, the answers are typically a single word. Hence a softmax operation on the encoded vector would be able to predict the correct answer from the vocabulary. LSTM or GRU cells are typically used in the RNN network to improve performance. This model has been used to study the baseline results on bAbi in [13].

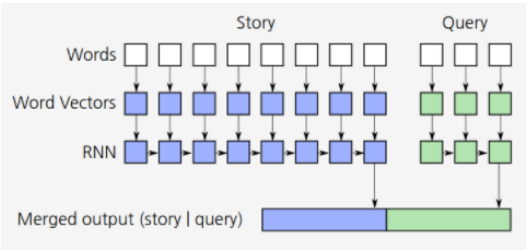


Figure 3: Seq2seq model for QnA

3.2 End-to-end Memory Networks

While seq2seq models are ideal to learn sequences, they have the following drawbacks. First, for long sentences, LSTM cells are unable to capture relationships across the sequence. Since the stories and questions in open domain QnA problems use large story and question lengths, this drawback results in errors. Second, standard seq2seq architectures lack attention.

Even when attention is added to the model, the model does not provide a clear framework of how the attention must be trained as each cell represents a single word. To counter these, memory networks were first introduced in [13]. The model discussed in this work is based on a variant of the architecture as presented in [11]

Memory Networks are trained to attend to relevant sentences of the story at each hop based on the question it must answer. This is achieved by weighting each sentence embedding by the attention of the question on it. This avoids the need to setup long recurrent networks for the story. Figure 4 from [11] illustrates the memory network architecture. The network is split into a variable number of steps or *memory hops*. At each hop, a *memory embedding* of the story is multiplied with a *query embedding*. The resulting dot products, after a softmax operation, produce probabilities that represent the attention value of a sentence in that hop. A second embedding, called the *output embedding*, is weighted with these attention values and reduced to get an embedding vector. This is used as a query in the next hop. These steps are repeated till a final softmax produces a one-hot encoding of the predicted word in the vocabulary.

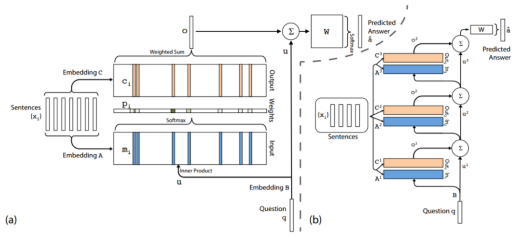


Figure 4: End to end memory network model for QnA

There are key architectural choices that can cause a significant difference in the results achieved. The first is the use of positional encoding. In order to preserve sentence structure each position in the sentences has to be encoded differently. The exact steps to achieve this are detailed in [11]. Next, the choice of sharing embeddings is critical for good performance. Using independent embeddings for each memory and output step at each hop, or a common memory and output embedding for all hops gives poor performance. Good performance is achieved by using shared embeddings across hops. Specifically, the output embedding of the previous layer and memory embedding of the next layer use the same embedding. The remainder of the report will use this adjacent weight sharing strategy as the baseline architecture.

4 Experiments

In order to achieve the best performance for bAbi using memory networks, various hyperparameter combinations had to be evaluated. Training in bAbi can be conducted in a joint or single setting as discussed. Here, both were run for each combination. This section details the experiments that were conducted and the rationale behind each of them. Unless otherwise specified, the following hyperparameters are used in the experiments

- Optimizer: Adam with decayed learning rate=0.1
- No regularization
- Embedding Size: 20 for Single and 40 for Joint
- Number of hops: 3

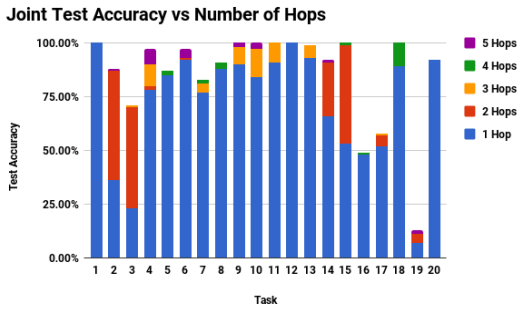
4.1 Number of Hops

Memory networks learn different word embeddings in each memory hop when adjacent weight sharing is employed. These embeddings dictate the sentences that will be attended to in the next hop. Hence, the rationale behind the experiment is that more hops would enable a more fine tuned attention flow across the hops and thereby result in better test accuracy. The number of hops were varied from 1 to 5 for both single and joint training. Figures 5a and 5b display the results across the tasks. The X-axis represents each task and the Y-axis shows the incremental improvement in test accuracy as hops are added. Following are some observations made from the results.

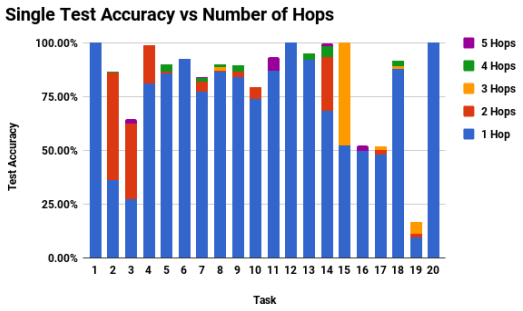
- Tasks 9, 10, 11, 12 improve with joint training as the number of hops are added. However, the number of hops have little effect on the same task during single training.
- Task 15, with single training, is able to achieve close to a 100% accuracy in 3 hops. However, joint training helps achieve this accuracy in 2 hops.
- Memory networks fail at task 19 as it requires reasoning that the memory network is unable to gain with joint or single training. An example of this is deducing that if A is to the west of B, B is to the east of A.
- The hypothesis that tasks with more supporting facts necessarily need more hops to get good accuracy is shown not to always hold true by this experiment. Although the trend is observed in tasks 1-3, task 13 with 2 supporting facts is able to achieve very high accuracy in a single hop.

4.2 Regularization

The experiments on varying the number of hops revealed that barring a few tasks, the training accuracy always reaches 100% although the validation and test accuracy stay behind. To identify if regularization can be used to improve performance, a sweep over the regularization constant from 0.001 to 1 was performed in multiples of 10. The experiment was performed on both joint and single training. Figures 6a and 6b display the results of the sweep. The X-axis is the test



(a) Joint training

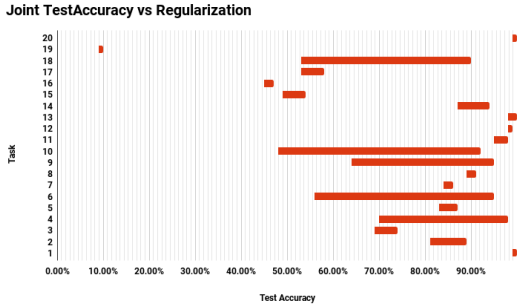


(b) Single training

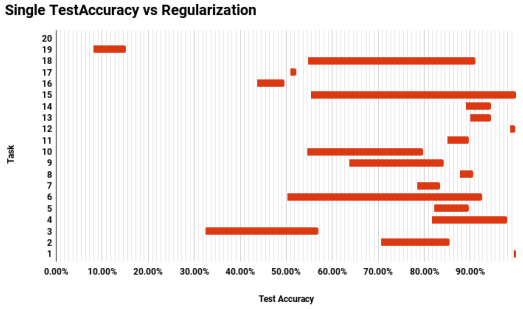
Figure 5: Incremental test accuracy improvement by varying number of hops

Task	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
No of supporting facts	1	2	3	1	1	1	5	6	1	1	2	1	2	2	2	3	2	2	2	1

Table 1: Tasks - Supporting facts



(a) Joint training



(b) Single training

Figure 6: Test Accuracy sensitivity to varying regularization

accuracy from the least to the highest value achieved for each task in the Y-axis. The key observation is that joint models are more robust to variation in regularization than single models across all tasks.

4.3 Word Embedding

The core of the memory network training is learning word embeddings at each hop for the memory and output steps. A series of experiments with different choices of embeddings were performed including using fine tuning Glove and varying the embedding architecture. The results were poor and hence not presented here. Once the adjacent weight sharing strategy with random initialization was fixed, the next experiment was to change the embedding size and hence the dimensionality of the word embeddings at each hop. These were performed on single and joint training. This however revealed that there was very little impact of dimensionality on test accuracy. This result can be explained by the small vocabulary of the tasks. The variation of test accuracy across tasks for different embedding sizes in single training is presented in Figure 7.

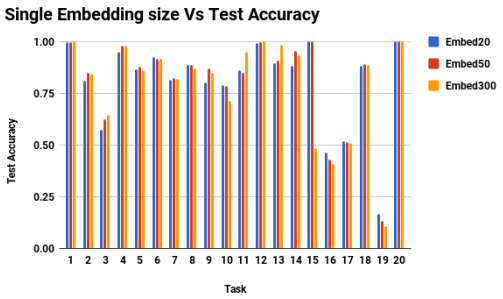


Figure 7: Single Test Accuracy versus different embedding size across tasks

4.4 Non Linearity

The effect of using an extra non linearity between consecutive hops in the memory network was also studied. The rationale behind using an additional non linearity is to further decouple the embeddings learnt across hops. Relu and Leaky Relu were used for each task in single training. While the runs reveal that adding non linearity helps perform better for some tasks, no clear trend is observed. Figure 8 displays the results for the two

non linearities in the memory networks. The Y-axis represents the positive or negative change in test accuracy versus using no non linearity in the network while the X-axis represents each of the 20 tasks. Relu shows better results on most of the tasks.

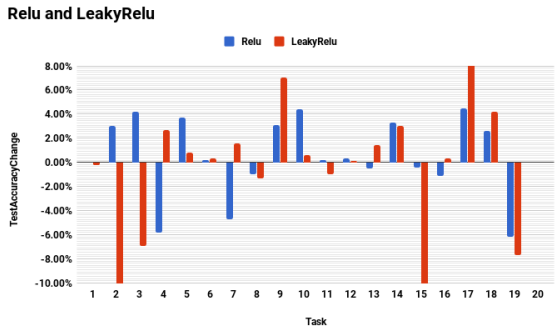


Figure 8: Test Accuracy Vs Non-linearity

5 Results and Analysis

This section summarizes the test accuracies obtained through the experiments in section 4. Further, the results are analyzed qualitatively for specific tasks.

Task	Single				Joint		
	Train	Val	Test	F1	Train	Val	Test
1	1	1	1	0.5	1	1	1
2	1	0.81	0.86	0.43	1	0.85	0.89
3	1	0.73	0.64	0.32	0.99	0.83	0.74
4	1	0.99	0.99	0.49	1	0.98	0.98
5	1	0.93	0.9	0.46	1	0.87	0.88
6	1	0.98	0.93	0.46	1	0.93	0.97
7	0.98	0.87	0.84	0.37	0.89	0.91	0.86
8	0.94	0.93	0.91	0.34	0.99	0.92	0.91
9	1	0.9	0.87	0.43	1	0.99	0.95
10	0.98	0.88	0.83	0.41	1	0.9	0.92
11	1	0.97	0.95	0.48	1	0.99	0.99
12	1	1	1	0.5	1	0.99	1
13	1	0.99	0.99	0.49	1	1	1
14	1	0.97	0.95	0.48	1	0.97	0.94
15	1	1	1	0.5	1	1	1
16	0.49	0.54	0.5	0.25	0.51	0.37	0.48
17	0.8	0.65	0.6	0.3	0.74	0.64	0.58
18	0.97	0.93	0.93	0.46	0.97	0.94	0.92
19	0.43	0.2	0.17	0.08	0.41	0.07	0.12
20	1	1	1	0.5	1	1	1

Table 2: Best results on bAbi tasks

5.1 Results

Table 5.1 summarizes the best test accuracies obtained for each of the 20 tasks. The F1 scores for single training were also computed. This was done by computing the confusion matrix across all questions in the task. The precision and recall was computed for each answer label and the resulting per-label-F1's were averaged. Based on the results, the following observations can be made.

- Task 16 and 19 have a poor train accuracy. This implies that memory networks in their current architecture cannot deduce inductive relationships across words.
- Good test accuracies for single training correlate with good F1 scores as well.
- With the exception of Task 3, the test accuracies are similar for both single and joint training. This implies that the word embeddings learnt at each hop are not strongly tied to the task.

5.2 Analysis

In this subsection, the confusion matrix corresponding to 2 tasks are discussed.

- Task 2: Figure 9 shows the confusion matrix per question and for all the questions. It is clear that the errors are equally spread out and not biased towards any single answer.
- Task 3: The confusion matrix for this task in Figure 10 illustrates that the model is able to understand time constructs. The question posed is *Where was the milk before the garden?* The model learns that *garden* can never be the answer for this question.

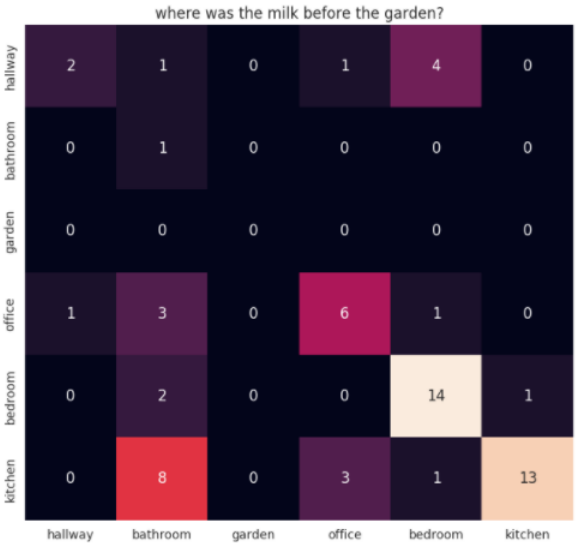
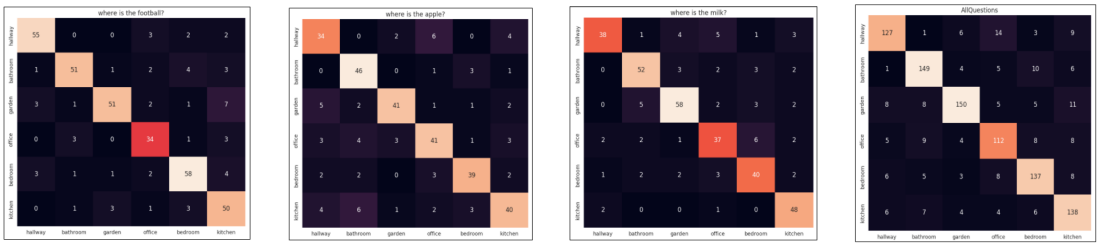


Figure 10: Confusion Matrix for Task 3

6 Model Interpretation

The experiments and results presented in the sections above provide metrics to evaluate the ability of the model to predict the answer for a given task. The key difference of memory networks over seq2seq is its ability to attend to meaningful sentences across hops. Figure 11 illustrates this. Each column represents a memory hop and every row corresponds to a sentence. The value in each cell is the attention probability assigned to the sentence in that hop.



(a) Where is the football? (b) Where is the apple? (c) Where is the milk? (d) All questions

Figure 9: Confusion matrices for Task 2 questions

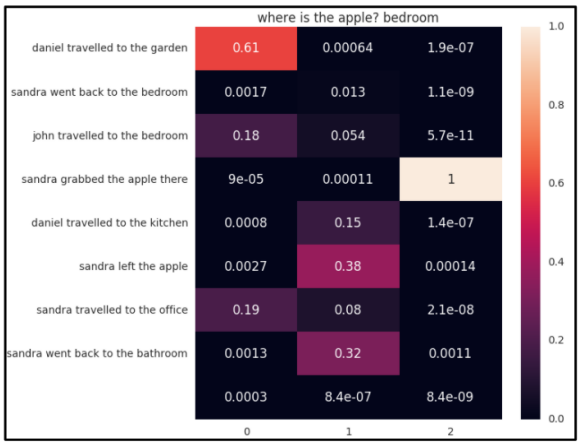


Figure 11: Attention visualization: In the first hop, the network must learn to attend to the sentences that have the word *apple* in them. The next hop must attend to sentences with *Sandra* in them. The last sentence must attend to the sentence that has the correct answer, *bedroom*, in it.

In order to measure this property of "good attention", we used the intersection over union metric (IoU). IoU can be in the range [0-1] with 1 representing correct attention at each hop and 0 representing wrong attention at each hop. The algorithm to compute the metric is shown in Algorithm 1. At each hop, 2 sets are computed. The first set holds sentences that have common words with the question. Let this be of size k. The second set holds the first k sentences sorted by their attention. If the intersection over union of these sets is close to 1, it would mean that the network learns to attend to the sentences that have high overlap with the query at each hop. As an example, in Figure 11, the query set will contain *apple*. The first set will include sentence indices *3,5*. This set is of size 2. Hence, the second set will contain the first 2 sentences sorted based on attention, the set of *3,4*. The IoU for these sets is 0.333. This will be the IoU for the first hop. While calculating the IoU, all the sets are created by removing the stop words that commonly occur in the sentences.

This metric was computed for each experiment detailed in the previous sections. It was observed that for a given task, different configurations with very similar test accuracies had very different IoU metrics for each

function IoU ($query$, $story$);
Output: iou: List holding IoU for each hop
$Qset$ = Set of words in the query
$AnswerSet$ = Set of words in the answer
for *every hop* **do**
  $CorrSet$ = sentence with non null intersection in $QSet$
  $k = size(CorrSet)$
  $AttnSet$ = Sentence with top k attention probabilities
  $iou[hop] = \frac{CorrSet \cap AttnSet}{CorrSet \cup AttnSet}$
  $QSet$ = Set of words in attended sentences not in $Qset$
end

Algorithm 1: IoU Metric

Hops	Test Accuracy	IoU First Hop	IoU Final Hop
1	0.36	0.26	0.26
2	0.86	0.88	0.49
3	0.81	0.10	0.46
4	0.82	0.007	0.47
5	0.81	0.93	0.51

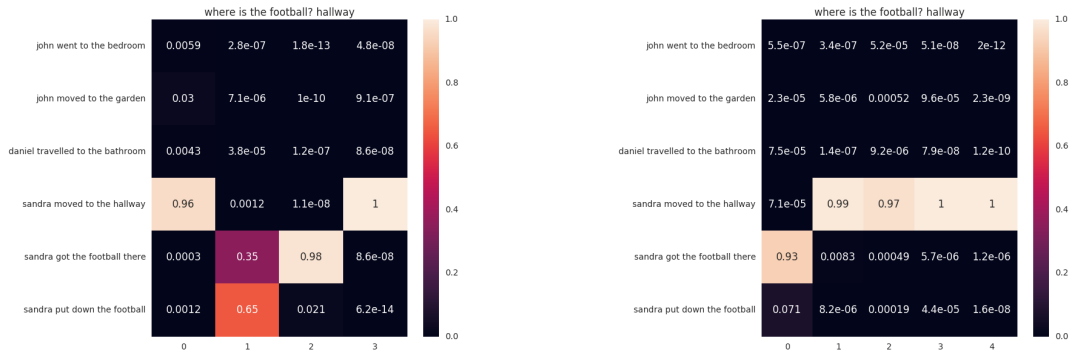
Table 3: Test accuracy and IoU for Task 2

hop. One such example was for task 2 with varying number of hops. Data for the experiment is shown in Table 3. It can be seen that although the test accuracy for 4 and 5 hops are roughly the same, they have very different IoU numbers for the first hop. This would imply that the attention flow for a trained model with 4 hops would be substantially less.

In order to validate this claim qualitatively, the attention flow of the same example in both models is visualized in Figures 12a and 12b. It can be seen that the 5 hops model attends to *football* first and then the sentence with *hallway*. The 4 hop model on the other hand has very scattered attention that is hard to interpret.

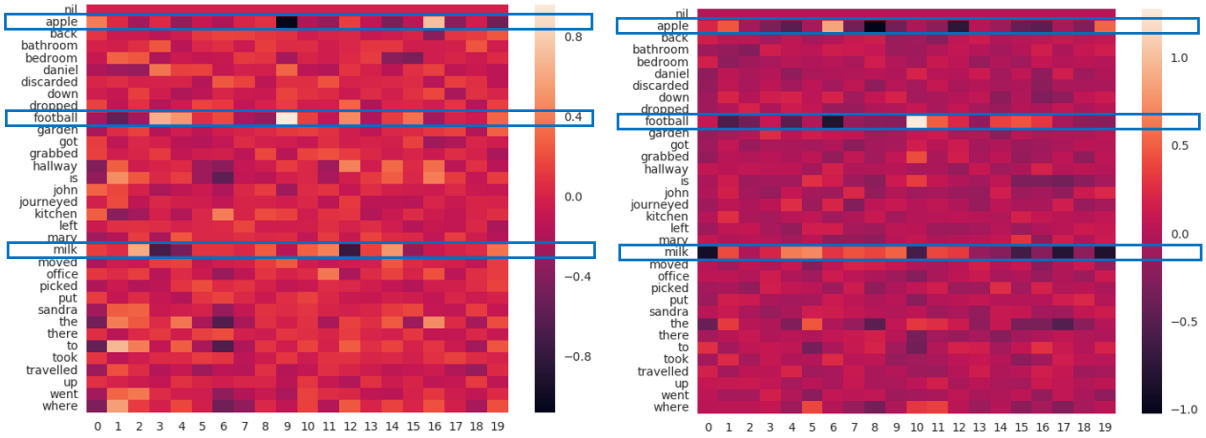
To further differentiate what the 2 models learn, we analyse the word embeddings at the first hop for both models. Figures 13a and 13b illustrate this. The embedding in Figure 13b focuses only on words such as *football, apple, milk*. As a result, the embeddings learn to attend to the correct sentences in a structured manner across hops.

Visualizing the embeddings across hops can shed



(a) 4 hops (b) 5 hops

Figure 12: Attention Flow for Task 2



(a) 4 hops (b) 5 hops

Figure 13: Embedding Matrix for Task 2

light onto the words that the model learns to attend to across epochs. In order to analyze this, a model trained on task 8 was used. Task 8 asks questions on lists and sets. The model must learn how to add and remove items from memory. An example story, question and answer is as follows.

Story
Daniel grabbed the milk there.
Sandra went back to the bathroom.
Q1: What is Daniel carrying? **Ans:** milk
Daniel went back to the garden.
Daniel dropped the milk.
Q2: What is Daniel carrying? **Ans:** nothing

In order to answer the second question correctly, the model must learn to remove *milk* from its set when it sees the word *dropped*. All questions in this task are of the form *What is ⟨person⟩ carrying?*

Hence, the first hop must learn to attend to the names, the intermediate hops must attend to objects and words that define append or delete operations and the final hop must attend to objects. Figure 14 illustrates this. The words are presented sorted in decreasing order by their embedding norm in the final hop. The Y-axis has the embedding norm for each word across all hops clustered together. The following observations can be made.

- Objects: The final hop learns large embedding norms for the objects alone. With the exception of some operation words like *left* for *subtraction* or *got* for *addition*, the large magnitude word embeddings are only objects.
- Names: Early in the network, the model must learn to attend to names in the story. But in subsequent hops, this information becomes less important. For example, *John, Mary, Daniel, Sandra* have the highest norms in the first hop. But this value decreases significantly by the last hop.
- Locations: Since locations such as *office, bathroom, bedroom* do not affect the prediction, their embeddings are very small.

7 Code

The code for this network was forked from [3]. The modified code with the experiments and visualization scripts is available at https://github.com/saharshoza/QuestionAnswering.

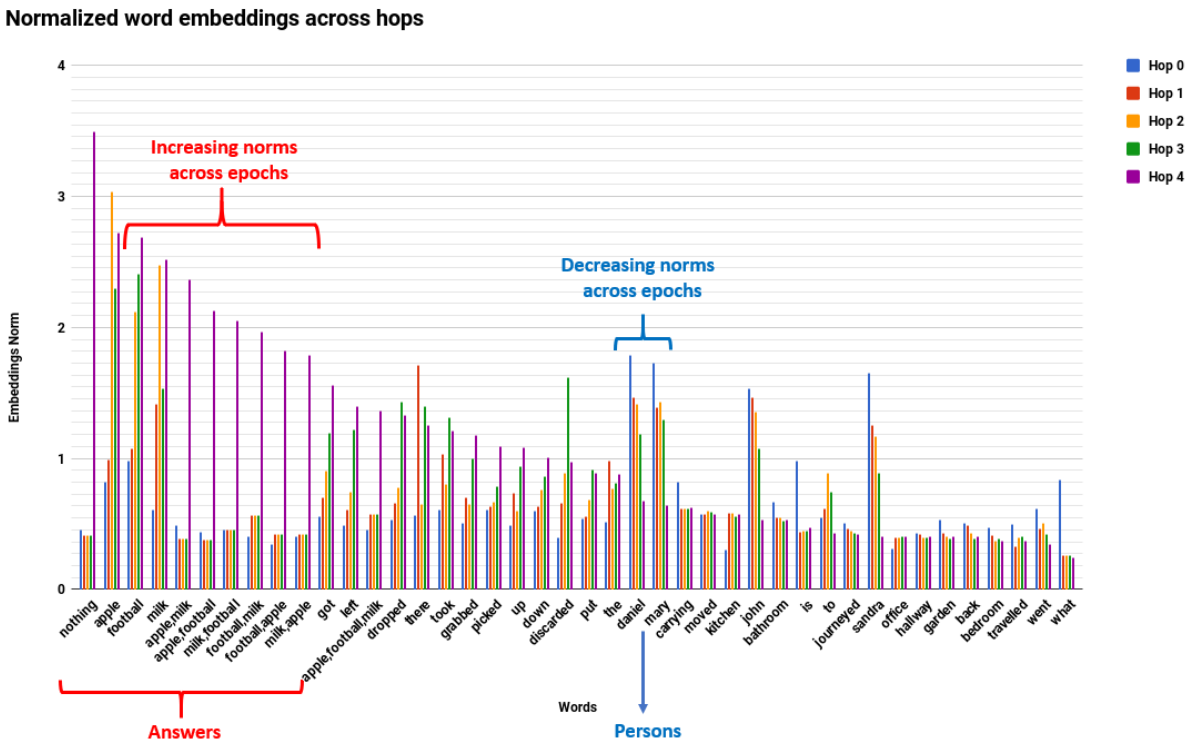


Figure 14: Normalized word embeddings across hops for Task 8

8 Conclusion

In this report, memory networks are analyzed for Question and Answering using bAbi dataset. The model is tuned with various hyperparameters to achieve the best test accuracy across tasks. The tradeoffs in changing number of memory hops, regularization constant, embedding size and non linearity types are studied. It is observed that training in a single and joint setting gives similar results barring a few tasks. Further, the F1 scores for single training are computed and confusion matrices visualized to understand the errors the model makes.

The attention flow property of memory networks is quantified using the IoU metric. Evaluating the IoU metric reveals that models with the same test accuracy may learn very different embeddings and that a higher IoU metric means a more interpretable model. The role of different word embeddings across hops is studied by taking an example of task 8 with 4 hops.

9 Future Work

Memory networks provide an intuitive model to solve the QnA problem in NLP. Yet, there remain experiments that would be interesting to perform.

- bAbi: Tasks 16 and 19 perform very poorly even on the training set with a memory network. A similar analysis of the embeddings in those tasks may help identify the reason for the poor performance of these networks on them.
- Other QnA datasets: It would be interesting to use this model to train datasets like Squad and MCtest to analyze its performance on less structured tasks.
- Convolution Kernel: Convolution Match Networks [9] have shown significant improvement over traditional memory networks on bAbi tasks. Incorporating a convolutional kernel between hops may improve the attention flow and thereby the accuracy of the task.

Acknowledgement

The authors acknowledge the Texas Advanced Computing Center (TACC) at The University of Texas at Austin for providing HPC, GPGPU resources that have contributed to the research results reported in this paper. URL: http://www.tacc.utexas.edu

References

[1] https://research.fb.com/downloads/babi.

[2] https://github.com/keras-team/keras/blob/master/examples/babi_rnn.py.

[3] https://github.com/domluna/memn2n.

[4] https://github.com/facebook/bAbI-tasks.

[5] https://github.com/saharshoza/QuestionAnswering.

[6] http://cs224d.stanford.edu/reports/qian.pdf.

[7] https://cs224d.stanford.edu/reports/StrohMathur.pdf.

[8] Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. Ask me anything: Dynamic memory networks for natural language processing. In *International Conference on Machine Learning*, pages 1378–1387, 2016.

[9] Michael Fairbank Maria Fasli Spyridon Samothrakis, Tom Vodopivec. Convolutional-match networks for question answering. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 2686–2692, 2017.

[10] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. Weakly supervised memory networks. *CoRR*, abs/1503.08895, 2015.

[11] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448, 2015.

[12] Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*, 2015.

[13] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014.

[14] Wenpeng Yin, Mo Yu, Bing Xiang, Bowen Zhou, and Hinrich Schütze. Simple question answering by attentive convolutional neural network. *CoRR*, abs/1606.03391, 2016.