Question answering using soft-attention Neural Network model

Arun Verma, Vishwajeet Kumar, Abhiram Singh, Ashish Jatav

November 20, 2016

1 Introduction

Question answering (QA) is the problem of predicting answer to a question after learning from a set of sentences or from paragraphs(evidence). Question answering is a challenging task in real world scenarios as this involves understanding knowledge base along with its embedded context as well as complete understanding of the question.
We are using neural attention model over a possibly large external memory. The architecture is a form of Memory Network but unlike the model in that work, it is trained end to end, and hence requires significantly less supervision during training, making it more generally applicable in realistic settings. Even though LSTM is considered to be good model for these tasks but due exploding gradients problem. Also LSTM have problems with long sentences and even worse for question based on context. The attention mechanism can solve this problem by "attend to" some portion of the contexts while paying less attention to others. Intuitively this puts a lighter burden on the vector to represent all the semantic information.

The model can be considered a continuous form of the Memory Network. The model in the work was not easy to train via backpropagation, and required supervision at each layer of the network. The continuity of the model we present here means that it can be trained end-to-end from input-output pairs, and so is applicable to more tasks, i.e. tasks where such supervision is not available, such as in language modelling or realistically supervised question answering tasks. This model can also be seen as a version of RNNsearch with multiple computational steps (which we term "hops") per output symbol. Neural attention model is unique in the sense that in addition to the hidden states, it relies on an external memory representation that we can analyze during the learning process.

2 Problem Statement

Given a set of stories, questions based on these stories, which also have some temporal sequence, the problem is to learn a model to correctly predict the answer given the story(set of facts) and the question.

3 Background

Question answering model can be learned using various technique such as MLP, RNN, LSTM etc.
RNN: The idea behind RNNs is to make use of sequential information. In a traditional neural network we assume that all inputs (and outputs) are independent of each other. But for many tasks that's a very bad idea. If you want to predict the next word in a sentence you better know which words came before it. RNNs are called recurrent because they perform the same task for every element of a sequence, with the output being depended on the previous computations. Another way to think about RNNs is that they have a "memory" which captures information about what has been calculated so far. In theory RNNs can make use of information in arbitrarily long sequences, but in practice they are limited to looking back only a few steps (more on this later). Here is what a typical RNN looks like:

The above diagram shows a RNN being unrolled (or unfolded) into a full network. By unrolling we simply mean that we write out the network for the complete sequence. For example, if the sequence we care about is a sentence of 5 words, the network would be unrolled into a 5-layer neural network, one layer for each word. The formulas that govern the computation happening in a RNN are as follows:

- $x_t$is the input at time step t. For example, x_1 could be a one-hot vector corresponding to the second word of a sentence.
- $s_t$is the hidden state at time step t. It's the "memory" of the network.s_t is calculated based on the previous hidden state and the input at the current step: $s_t = f(Ux_t + Ws_{t-1})$. The function f usually is a nonlinearity such as tanh or ReLU. s_{-1}, which is required to calculate the first hidden state, is typically initialized to all zeroes.
- $o_t$is the output at step t. For example, if we wanted to predict the next word in a sentence it would be a vector of probabilities across our vocabulary. $o_t = softmax(V_{s_t}$

LSTM:
LSTM networks are quite popular these days. LSTMs don't have a fundamentally different architecture from RNNs, but they use a different function to compute the hidden state. The memory in LSTMs are called cells and you can think of them as black boxes that take as input the previous state h_{t-1}what to

erase from) memory. They then combine the previous state h_{t-1}, the current memory x_t, and the input. It turns out that these types of units are very efficient at capturing long-term dependencies.

In a traditional recurrent neural network, during the gradient back-propagation phase, the gradient signal can end up being multiplied a large number of times (as many as the number of time steps) by the weight matrix associated with the connections between the neurons of the recurrent hidden layer. This means that, the magnitude of weights in the transition matrix can have a strong impact on the learning process.

If the weights in this matrix are small (or, more formally, if the leading eigenvalue of the weight matrix is smaller than 1.0), it can lead to a situation called vanishing gradients where the gradient signal gets so small that learning either becomes very slow or stops working altogether. It can also make more difficult the task of learning long-term dependencies in the data. Conversely, if the weights in this matrix are large (or, again, more formally, if the leading eigenvalue of the weight matrix is larger than 1.0), it can lead to a situation where the gradient signal is so large that it can cause learning to diverge. This is often referred to as exploding gradients.
These issues are the main motivation behind the LSTM model which introduces a new structure called a memory cell (see Figure 1 below). A memory cell is composed of four main elements: an input gate, a neuron with a self-recurrent connection (a connection to itself), a forget gate and an output gate. The self-recurrent connection has a weight of 1.0 and ensures that, barring any outside interference, the state of a memory cell can remain constant from one time step to another. The gates serve to modulate the interactions between the memory cell itself and its environment. The input gate can allow incoming signal to alter the state of the memory cell or block it. On the other hand, the output gate can allow the state of the memory cell to have an effect on other neurons or prevent it. Finally, the forget gate can modulate the memory cell's self-recurrent connection, allowing the cell to remember or forget its previous state, as needed.

4 Approach

we implemented MLP, LSTM and neural attention model and tested the accuracy. The main motivation behind using neural attention model is due to the fact that the problem require us to learn long term dependencies, out-of order access and unordered set. Figure 1 depicts an example of out of order in example. Existing model still struggles with these type of dependencies. NAM model takes a discrete set of inputs $x_1, x_2...x_n$ that are to be stored in the memory, a query q, and outputs an answer a. Each of the x_i, q, and a contains symbols coming from a dictionary with V words. The model writes all x to the memory up to a fixed buffer size, and then finds a continuous representation for the x and q. The continuous representation is then processed via multiple hops to

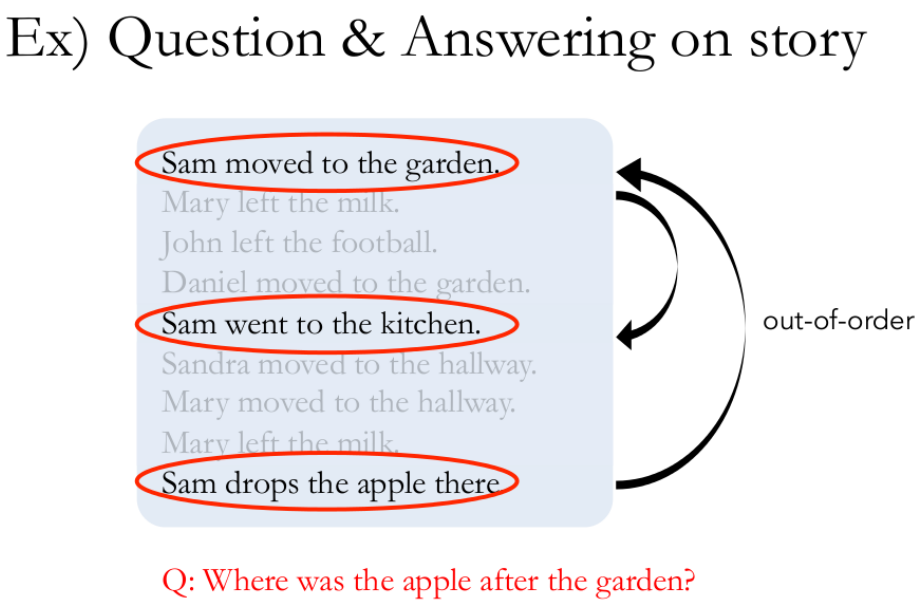

Figure 1: Out of order Example.

output a. The Figure 2 represents overall architecture of the system.

5 Single Layer

We will describe the single layer case then we wil go ahead and extend it to multilayer setting. Input representation: Suppose we are given an input set $x_1, x_2, x_3....x_n$ to be stored in memory. The entire set of x_i are converted into memory vectors m_i of dimension d computed by embedding each x_i in a continuous space, in the simplest case, using an embedding matrix A (of size $(d \times V)$).

The query q is also embedded (again, in the simplest case via another embedding matrix B with the same dimensions as A) to obtain an internal state u. Also this model captures the temporal aspects using a special vector see Figure 4 below. One example of temporal aspect is "Sam was in kitchen before he was in bedroom". In the embedding space, we compute the match between u and each memory mi by taking the inner product followed by a softmax:

$$P = softmax(u^T m_i) \tag{1}$$

where $Softmax(z_i) = e_{zi} = P_j e_{zj}$. Defined in this way p is a probability vector over the inputs.

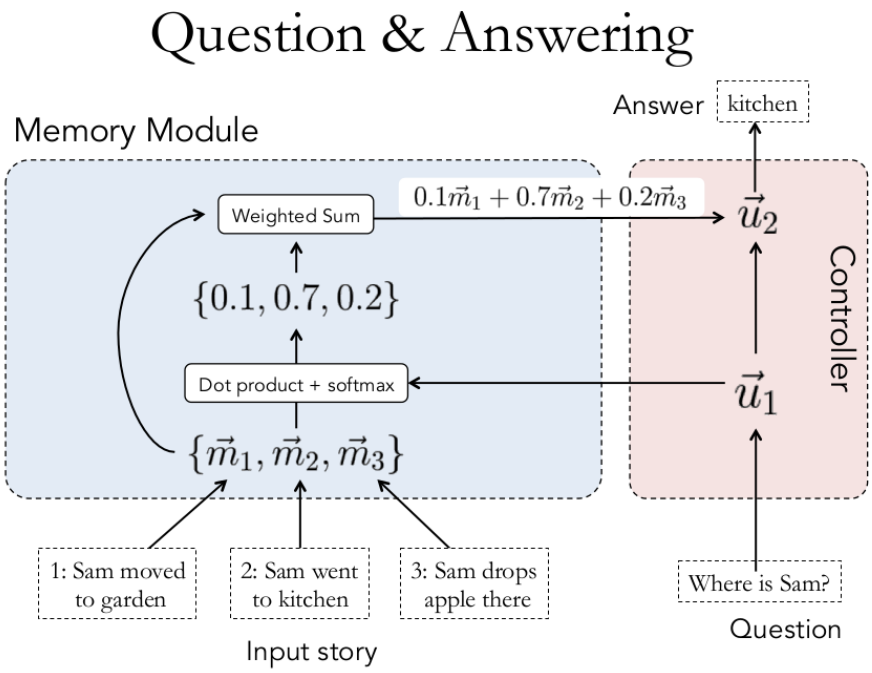

Figure 2: Overall Architecture .

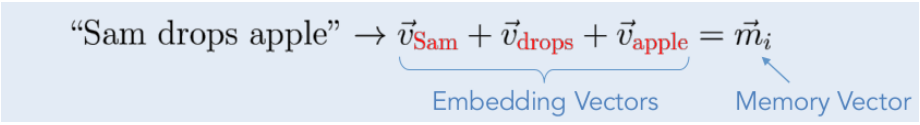

Figure 3: One example memory vector.

E.g.) **temporal structure:** special words for time and include them in BoW

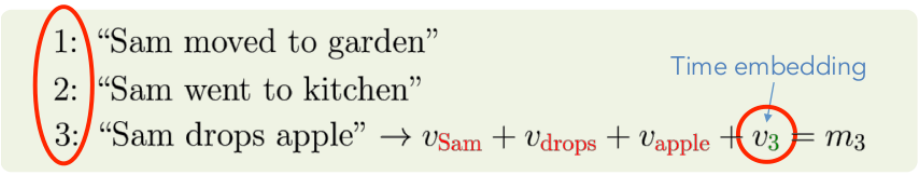

Figure 4: capturing temporal aspect.

5.1 Output Memory Representation

Each x_i has a corresponding output vector c_i (given in the simplest case by another embedding matrix C). The response vector from the memory o is then

a sum over the transformed inputs c_i, weighted by the probability vector from the input:

$$o = \sum_i [p_i c_i] \tag{2}$$

Because the function from input to output is smooth, we can easily compute gradients and backpropagate through it.

5.2 Generating the final prediction

In the single layer case, the sum of the output vector o and the input embedding u is then passed through a final weight matrix W (of size $V \times d$) and a softmax to produce the predicted label:

$$a = softmax(W(o + u)) \tag{3}$$

Following Figure 5 give and example of how the answer is predicted.

Story (1: 1 supporting fact)	Support	Hop 1	Hop 2	Hop 3
Daniel went to the bathroom.		0.00	0.00	0.03
Mary travelled to the hallway.		0.00	0.00	0.00
John went to the bedroom.		0.37	0.02	0.00
John travelled to the bathroom.	yes	0.60	0.98	0.96
Mary went to the office.		0.01	0.00	0.00
Where is John? Answer: bathroom Prediction: bathroom				

Story (2: 2 supporting facts)	Support	Hop 1	Hop 2	Hop 3
John dropped the milk.		0.06	0.00	0.00
John took the milk there.	yes	0.88	1.00	0.00
Sandra went back to the bathroom.		0.00	0.00	0.00
John moved to the hallway.	yes	0.00	0.00	1.00
Mary went back to the bedroom.		0.00	0.00	0.00
Where is the milk? Answer: hallway Prediction: hallway				

Story (16: basic induction)	Support	Hop 1	Hop 2	Hop 3
Brian is a frog.	yes	0.00	0.98	0.00
Lily is gray.		0.07	0.00	0.00
Brian is yellow.	yes	0.07	0.00	1.00
Julius is green.		0.06	0.00	0.00
Greg is a frog.	yes	0.76	0.02	0.00
What color is Greg? Answer: yellow Prediction: yellow				

Story (18: size reasoning)	Support	Hop 1	Hop 2	Hop 3
The suitcase is bigger than the chest.	yes	0.00	0.88	0.00
The box is bigger than the chocolate.		0.04	0.05	0.10
The chest is bigger than the chocolate.	yes	0.17	0.07	0.90
The chest fits inside the container.		0.00	0.00	0.00
The chest fits inside the box.		0.00	0.00	0.00
Does the suitcase fit in the chocolate? Answer: no Prediction: no				

Figure 5: One example of prediction

6 Detailed Explanation

Consider few example sentences from story we will go over each parameter and try to get intuition: consider the sentences below:
mary moved to the bathroom.
John went to the hallway.
Question: Where is mary?
Answer: bathroom

```
[['mary', 'moved', 'to', 'the', 'bathroom'],
['john', 'went', 'to', 'the', 'hallway']],
['where', 'is', 'mary'], ['bathroom']

Following is vectorized representation of story.
```

```
array([[11, 12, 19, 18,  2,  0],
       [ 8, 21, 19, 18,  6,  0],
       [ 0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0]])

Below we have vector for question and one hot encoding of Answer.

[22,  7, 11,  0,  0,  0]

[ 0.,  0.,  1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
        0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.]

This vectorized representation is generated using following vocabulary of words from questio
```

The parameters(A,B,C,W) generated for above examples will be following sizes:
A : 20×20, B: 20×20, C:20×20 and W: 20×20 The memory size is equal to the maximum number of evidences for a question in this case it is 10. And spatial encoding is of size 6×20.

A basic algorithm for training the network:

```
Algorithm for training the soft attention Neural Network:
1. Read train and test data.
2. Make a vocabulary of words from whole data.
3. convert into stories and question into vectors using vocabulary.
3. For each story question and answers:

a) Using embedding A(dxV), project each sentences of story in V dimension space.
        b)  Using embedding B(dxV), project each question in V dimension space which is u.
        c)Take dot product of question with each sentence in order to find similarity.

        d) Using softmax find their respective probabilities.
        (step c and d will tell us which sentence is similar to question
        and assign higher probability according to similarity)

        e) using embedding C(dxV) which may be similar to A, take weighted sum of each
        sentence using probabilites from previous step.
        (it will return a vector o of dimension V which tells us which position
        of sentence is more important to question eg. if 'sam' is in question then it
```

```
will give higher probability to the position in sentences where 'sam' is occuring)

f) Now add u and o to give more weight to position which is more important
according to question eg. like if 'sam' is more important then in each iteration
weight of position containg sam 'sam' is increased so that in next iteration
we can use this information.
g) In final hop, calculate sum of position weight with question weights followed
by projecting this sum using embedding weight matrix w(Vxd), after that apply
softmax to get high probability postion in a sentence which will be the answer.

4) Repeat step 3 untill convergence.
```

7 Architecture

The overall model is shown in Figure below During training, all three embedding matrices A, B and C, as well as W are jointly learned by minimizing a standard cross-entropy loss between predicted label p_a and the true label a. Training is performed using Ada optimizer.

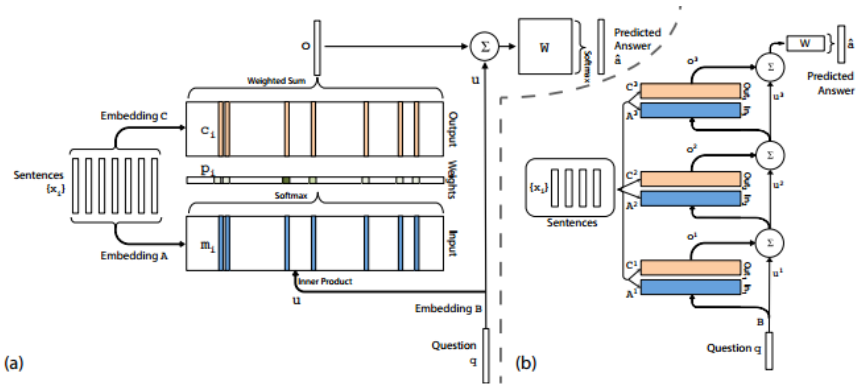

Figure 6: High level view of model.

Few important points :

- Neural Network is trained End to End from stories to question to answer.
- Trained model takes a set of inputs $x_1, x_2, ...x_n$to be stored in the memory, a query q, and outputs an answer a.
- Each of x_i, q, a contains symbols coming from a dictionary with V words.

- All x is written to memory up to a fixed buffer size, then find a continuous representation for the x and q.
- The continuous representation is then processed via multiple hop to output a.
- This allows back-propagation of the error signal through multiple memory accesses to input during training.
- A,B,C are embedding matrices (of size $d \times V$) used to convert the input to the d-dimensional vectors m_i.
- A match is computed between u and each memory by taking the inner product followed by a softmax.
- There are a total of 20 different types of tasks that test different forms of reasoning and deduction.
- Note that for each question, only some subset of the statement contain information needed for the answer, and the others are essentially irrelevant distractors.

8 Network Design

We used 3 hops neural network design as proposed by arthur et.al .Adjacent weight sharing was used to ease training and reduce the number of parameters. Figure 6 right side represents the three layers in the model.

9 Implementaion

We implemented the system using tensor flow [1] in python. We also implemented a demo of the system in node js, where you can enter a question for a story and the system will give you the answer. We referred to this [2] and [3] for some of the implementation ideas.

10 Experimental Details

10.1 Data Analysis

We trained our model using facebook BABI task dataset (20 QA tasks). A task is a set of example problems. A problem is a set of I sentences x_i where $I =< 320$, a question q and an answer a. The vocabulary is of size $V = 177$! Two versions of the data are used, one that has 1000 training problems per task, and one with 10,000 per task.

[1] https://www.tensorflow.org/
[2] https://github.com/farizrahman4u/seq2seq
[3] https://github.com/facebook/Memnn

10.2 Results

Overall soft attention neural network model performs better than other systems.

Model	**Accuracy**
MLP	25
LSTM	35
Neural Attention model	99

11 Conclusion

We applied soft attention neural model on question answering task on BABI dataset released by facebook, we found out that Neural attention model performs better than other models such as MLP, LSTM. LSTM has two main problem, 1. It can not store the memory given long time ago. 2. it does not take question representation into account when it encodes context sentence. Attention mechanism addresses both the problems. In attention mechanism is all about input focus, sequence data, context and weights. The attention mechanism can solve this problem by "attend to" some portion of the contexts while paying less attention to others. Intuitively this puts a lighter burden on the vector to represent all the semantic information.

References

[1] J. Weston, S. Chopra, and A. Bordes, "Memory networks," *arXiv preprint arXiv:1410.3916*, 2014.

[2] J. L. Elman, "Distributed representations, simple recurrent networks, and grammatical structure," *Machine learning*, vol. 7, no. 2-3, pp. 195–225, 1991.

[3] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[4] S. Sukhbaatar, J. Weston, R. Fergus, *et al.*, "End-to-end memory networks," in *Advances in neural information processing systems*, pp. 2440–2448, 2015.

[5] 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.

[6] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[7] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *Advances in Neural Information Processing Systems*, pp. 577–585, 2015.