

QA Bot

Md. Zahid Fesabelilla, Md. Rokibul Alam

Computer Science and Engineering, Ahsanullah University of Science and Technology

Ahsanullah University of Science and Technology

Dhaka, Bangladesh

160204082@aust.edu, 130204082@aust.edu

I. INTRODUCTION

We will be implementing a chat bot that can answer questions based on "story" given to the bot.

We have been working with the Babi Data Set from Facebook Research. So the type of data we're going to be working with has three main components. It has a story component. So for example we can have a simple story of "Jane went to the store . Mike ran to the bedroom ." So we have a story and then we'll have a yes or no question. And the question here is "Is Mike in the store ?" and our network is going to be able to understand that since Mike ran to the bedroom. He is not in the store. So we have a correct answer of no.

So again three main components the data a story a question or answer so a story or sentences question a query and then an answer so we're going to be following along with a paper called "End-To-End Memory Networks".

Story: Jane went to the store . Mike ran to the bedroom .

Question : Is Mike in the store ?

Ans : no.

When we want to find any occurrence or event but need to read a long story. It's really boring !! So we think why not make a system where you give your document or story and ask the system about this event and the system give a reply about this occurrence happens or not. We think its help school students to find true/false type question answer. It can be used in messaging bot to give reply on a yes-no type answer in a specific corpus.

The task is challenging because, we present RNN and LSTM architecture where the recurrence reads from a possibly large external memory multiple times before outputting a symbol. The model in that work was not easy to train via back propagation, and required supervision at each layer of the network.

For this reason we take help from "End-To-End Memory Networks" this paper . Use this architecture can easily compute gradients and back-propagate through it.

II. RELATED WORKS

Not exactly the same but likes the same type of work done by the Facebook AI Research team. They

have created a End-To-End Memory Networks system [<https://arxiv.org/pdf/1503.08895.pdf>] based on baBI data set.

We basically used their End-To-End Memory Networks. They use Recurrent neural networks (RNN) and we use one more thing to improve our model which is Long short-term memory (LSTM).

III. PROJECT OBJECTIVE

Subtask Of our project :

- 1) Load The dataset
- 2) Exploring the format of the data
- 3) Split train and test data into 10:1 ratio
- 4) Setting up vocabulary of all words
- 5) Vectorizing the data
- 6) Vectorize Stories
 - Functionalize Vectorization
- 7) Creating the model
 - Placeholders for inputs
 - Building the End-To-End Memory Networks
 - a) Create Input Encoder m
 - b) Create Input Encoder C
 - c) Create Question Encoder
 - d) Encode the sequence
 - e) Use dot product to compute the match between first input vector seq and the query
 - f) Add this match matrix with the second input vector sequence
 - g) Concatenate the match matrix with the question vector sequence
 - h) Reduce with RNN (LSTM) [LSTM(32)]
 - i) Regularization with Dropout
 - j) Our output a probability distribution over the vocabulary
 - k) Build the final model
- 8) Train the model
- 9) Test the model
- 10) Evaluating the Model

Dummy input and output of our system:

Input : John left the kitchen . Sandra dropped the football in the garden . Is the football in the garden ?

Output : yes

Input : Sandra went to the garden . John took the football .
Mary left the kitchen . Is Mary took football ?

Output: no

Input : Sandra went to the garden . John took the football .
Mary left the kitchen . Is Mary left the kitchen ?

Output : yes

Input : Is Sandra left the kitchen ? Output : no

Input : Mary took the apple there . Sandra journeyed to the
garden . Is Mary in the bathroom ?

Output : no

Input : Daniel grabbed the milk there . Mary went to the
office . Is Mary in the bedroom ?

Output : no

IV. METHODOLOGIES / MODEL

We are getting our architecture of the network from so the overall idea of the model is that it's going to takes a discrete set of inputs $x_1, x_2 \dots x_n$ etc that are to be stored in the memory, a query q , and outputs an answer a .

Each of the x , q , and a contains symbols coming from a dictionary with V words. With the amount of words so we'll be calling V a vocabulary. So have a set or dictionary that contains the entire vocabulary across the entire datasets.

So the model model is going to writes all x to the memory up to a fixed buffer size, and then finds a continuous representation for the x and q .

And there's three main components to the end to end network. There's going to be the input memory representation basically how do we actually take in these stories and the questions then we have an output memory representation and then we're going to be able to generate a final prediction.

And once we have that sub component of the network we're essentially going to create a full model using recurrent neural networks with multiple layers.

- End to End Network:

- 1) Input Memory Representation
- 2) Output Memory Representation
- 3) Generate Final Prediction

- Create a full model with RNN and Multiple Layers

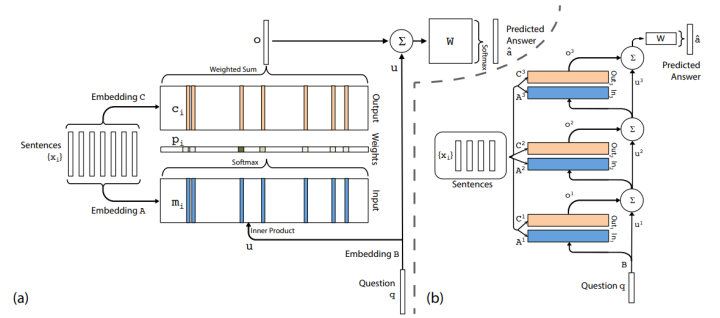


Fig. 1. (a): A single layer version of the model. (b): A three layer version of the model.

(a) A single layer version of the model: key components

- Input Memory Representation of stories
- Use Keras for Embedding to convert sentences X .
- Encoders C and M
- Question Encoder

So there's a single layer case which implements a single memory hop operation. So we first start of with the first key component of the single layer which is the input memory representation.

So what we are going to receive is an input set of x_1, x_2, x_3 etc. Those are the sentences or stories to be stored in memory. And we're going to convert that entire set of X s into memory vectors. So those memory vectors we are going to call m_i .

Here we essentially have two types of encoders and the bottom one is the m_i . That's one of input there. We use Keras for Embedding to convert sentences X .

Here we can see that the bottom of the diagram we have Question Q . It is going through an embedding process and then we have the result which is just going to be an internal state of this embedding inside the single layer called u .

In the embedding space within this single layer we are going to compute the match between u and memory m_i by taking the inner product followed by a soft Max operation so that will look something like this :

Input Memory Representation of stories :

$$p_i = \text{Softmax}(u^T m_i)$$

$$\text{where } \text{Softmax}(z_i) = e^{z_i} / \sum_j e^{z_j}$$

We take that question embedded through embedding B then we have u and then we are going to take u transpose take the product of that with m_i that's the embedding of the sentences and take the soft max of that and then we get p_i . So that is the input memory representation.

For output memory representation each x_i has a corresponding Output vector C_i and this is given in the simplest case by another embedding matrix which is called C. So that's kind of that top embedding embedding C then the response vector from the memory O is then a sum over the transform inputs

Ci waited by the probability vector for the input and here's the equation :

$$o = \sum_i p_i C_i$$

Because the function from input output is smooth we can then compute gradients and back propagate through this then the final third step is generating a single final prediction.

So 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 and a softmax to produce the predicted label:

$$a = \text{Softmax}(W(o+u))$$

So we have that final weight matrix W and then we have $o + u$ and we pass it in through a softmax and that's essentially going to give us probabilities of the predicted answer.

This network is actually going to produce a probability for every single word in the vocabulary however we should only expect some relatively high probability on either the yes or no.

(b) Multiple Layers :

Now expand this single layer into multiple layers using recurrent neural networks. We are simply going to take the output of one of the single layers and have that be the input for the next layer. We are just repeating it over and over again taking the inputs from one layer and then setting that to be the output of the next layer.

V. EXPERIMENTS

(a) Dataset

We have been working with the Babi Data Set from Facebook Research. For reading the data from the train set and test set, we importing pickle because we have a specialized file format for the data which is going to be a pickle file essentially a compressed way of saving our data.

Here total number of data is 11000. We split the total data into a 10:01 ratio. For this reason, we separate 10000 data for the training dataset and 1000 data for the testing dataset.

The total number of questions in the training dataset return 5012 'yes' answer and 4988 'no' answer. The total number of questions in the testing dataset return 497 'yes' answer and 503 'no' answer.

```
[ ] ### For story
    '.join(train_data[0][0])

    'Mary moved to the bathroom . Sandra journeyed to the bedroom .'

### For Q
    '.join(train_data[0][1])

    'Is Sandra in the hallway ?'

### for ans
    train_data[0][2]

    'no'
```

Fig. 2. Sample Data


(b) Evaluation Metric

Evaluation metrics are used to measure the quality of the statistical or machine learning model. Evaluating machine learning models or algorithms is essential for any project. There are many different types of evaluation metrics available to test a model. These include classification accuracy, logarithmic loss, confusion matrix, and others.

Accuracy : The accuracy of a machine learning classification algorithm is one way to measure how often the algorithm classifies a data point correctly. Accuracy is the number of correctly predicted data points out of all the data points.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Categorical Cross-Entropy Loss : Also called Softmax Loss. It is a Softmax activation plus a Cross-Entropy loss.



$$f(s)_i = \frac{e^{s_i}}{\sum_j e^{s_j}} \quad CE = - \sum_i^C t_i \log(f(s)_i)$$

(c) Results

Settings 1	Settings 2	Settings 3
LSTM(1)	LSTM(32)	LSTM(32)
Batch Size = 100	Batch Size = 100	Batch Size = 70
Epochs = 50	Epochs = 100	Epochs = 100
7ms/step	8ms/step	6ms/step
Train Loss: 0.6933	Train loss: 0.3053	Train loss: 0.2455
TrainAccuracy:0.4980	TrainAccuracy:0.8724	TrainAccuracy:0.8921
Test Loss:0.6931	Test Loss: 0.3569	Test Loss: 0.4100
TestAccuracy:0.5030	TestAccuracy:0.8280	TestAccuracy:0.8330

Settings 4	Settings 5
LSTM(32)	LSTM(50)
Batch Size = 32	Batch Size = 70
Epochs = 100	Epochs = 100
7ms/step	7ms/step
Train Loss: 0.2055	Train Loss: 0.2355
Train Accuracy: 0.9157	Train Accuracy: 0.9018
Test Loss: 0.4404	Test Loss: 0.4135
Test Accuracy: 0.8430	Test Accuracy: 0.8340

Here setting4 gives us the best training accuracy, best testing accuracy, need less time per step, give less loss for both training and testing.

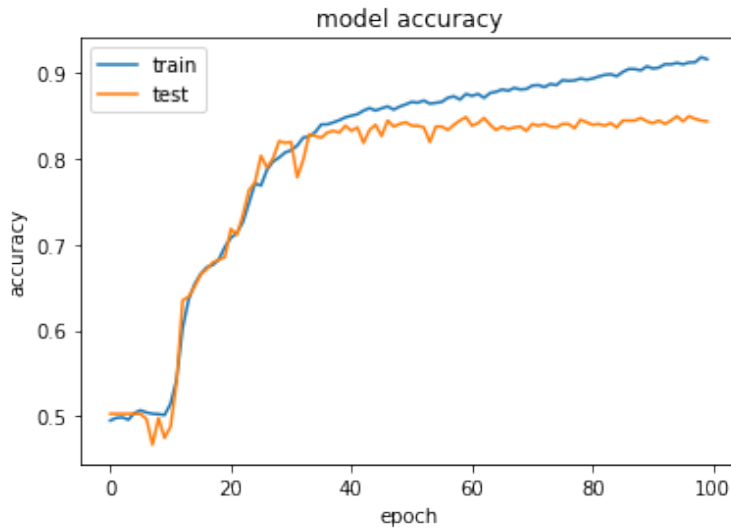


Fig. 3. Setting4

VI. CONCLUSION

Finally, we successfully build the system with the help of "End-To-End Memory Networks". We experimented with different hyperparameter settings. And setting4 gives us much more accurate answers in a selective corpus. We can use this system in a lot of things. We can use the model in the Bengali Language, FAQ segment, true false checker, etc.