Question Answering System

Jagmeet Singh Sidhu
Department of Computer Science
Lakehead University
Student id. 1102275
jsidhu10@lakeheadu.ca

Manva Trivedi
Department of Computer Science
Lakehead University
Student id. 1095816
trivedim@lakeheadu.ca

Taresh Dewan
Department of Computer Science
Lakehead University
Student id. 1106447
tdewan@lakeheadu.ca

***Abstract*—This paper proposes a question answering system which gives user an answer based on the story on which the model has been trained. We have build a model which comprises of multiple sequential layers with a combination of various layers such as dropout layers, Long short-term memory(LSTM) layers, dense layers and applied activation function (Softmax). The dataset which we have used is available here. The dataset contains short stories with each story limited to 68 words, questions and answers. For implementation we have used python libraries like keras and tensorflow along with packages like functools, tarfile, numpy, IPython, matplotlib and pandas. We have performed hyper-parameter tuning to achieve highest accuracy. The accuracy which have achieved with the proposed model is 95.56%.**

***Index Terms*—Question Answering System, Neural Network, NLP, LSTM, Vectorization, Keras, Tensorflow**

I. Introduction

Question Answering (QA) system is an information retrieval system where a direct response to a submitted question is provided, rather than a collection of references that may contain the answers [2]. The fundamental concept behind the QA method is to assist communication between man and machine.

QA systems are useful to access valuable web information and to provide insights [1]. The QA process can be split into two parts:

1) Information Retrieval:
   Finding a document that contains the answer to the question [1].
2) Reading Comprehension:
   Seeking proper response to the question given the document [1].

Our proposed work falls in the category of Reading Comprehension. Neural networks are a series of algorithms that are loosely modelled after the human brain and intended to identify patterns [3]. They perceive sensory data through some form of machine perception, marking, or raw input clustering [3].

The patterns they know are numerical, stored in vectors, into which all real-world data must be converted, be it images, sound, text or time series [3]. In this paper we have proposed a neural network model which aims to give accurate answers to the questions raised from the story.

II. Literature Review

Neural network has wide applications in areas such as Text Classification and Categorization, Named Entity Recognition (NER), Part-of-Speech (POS) Tagging, Paraphrase Detection, etc. The main reason of using neural networks in our implementation is because it allows the development of high performance question answering systems. Neural Network contains layers which has neurons with learnable weights and biases.

Each neuron receives several inputs, takes a weighted sum over them, passes it through an activation function and responds with an output [4]. Each Layer takes the input from the previous layer and preserves the relationship between the attributes. Then, the output is either fed to activation function or is passed on to next layers such as dense layers, max pooling layers, flattening layers, LSTM, etc.

Long Short Term Memory Networks – typically referred to as ”LSTMs” – are a special form of Recurrent Neural Networks (RNN) that can learn long-term dependencies. LSTMs are explicitly designed to avoid the issue of long-term dependency. RNNs have the form of a chain of repeating modules of neural network. As structure, LSTMs do have this chain but the repeating module has a different structure. Instead than having a single layer of neural network, there are four, which communicate in a very unique way.

We have used LSTM in our proposed model along with dense layers and dropout layers. A dense layer is used to change the dimensions of the vector whereas a dropout layer is used for regularization where we randomly set the dimensions of our input vector to be zero with probability.

III. Proposed CNN Model

A. Dataset

The dataset used for the implementation is taken from amzon web services and can be downloaded by clicking here. It contains first set of 20 tasks in the bAbI project to test text comprehension and reasoning. The goal is to test a specific feature of text and reasoning for each task, and thus test different learning models capabilities. For each task, there are 1000 questions for training, and 1000 for testing.

There are several directories in the datasest for the following reasons:

1) en/- English-language activities, accessible by humans.
2) Hn/- Tasks in Hindi, human readable.
3) Shuffled/- the same functions with shuffled letters so that they are not readable by humans and can not be used in a straightforward manner for current parsers and taggers to exploit extra resources.
4) en-10k/ shuffled-10k/ and hn-10k/ — the same tasks in the three formats but with 10,000 examples of training rather than 1000 examples.

The file format for each task is as follows:
ID text
ID text
ID text
ID question[tab]answer[tab]supporting fact IDS.

The IDs start at 1 for a given ”story,” and increase. The following sentences can be viewed as a new ”story” if the IDs in a file reset back to 1 . Supporting fact identifications only ever references the sentences in a ”story”.

For Example:
1 Mary moved to the bathroom.
2 John went to the hallway.
3 Where is Mary? bathroom 1
4 Daniel went back to the hallway.
5 Sandra moved to the garden.
6 Where is Daniel? hallway 2
7 John moved to the office.
8 Sandra journeyed to the bathroom.
9 Where is Daniel? hallway 3

It contains three attributes - short stories, questions and answers which are shown in table 1.

TABLE I
ATTRIBUTE DESCRIPTION

Attribute	Data type	Description
Text	String	Passage which can be referred for question-answering
Question	String	Queries raised from the passage
Answer	String	one word answer for respective question
Supporting fact IDs	integer	ID of the supporting fact

B. Libraries

We have used the pandas library to load the data and perform pre-processing. In order to implement our question answering system, we have used libraries which are listed below:

- tensorflow
- keras
- functools
- tarfile
- matplotlib
- numpy
- IPython
- pandas

We have used keras, as it helps in fast experimentation with deep neural networks and it acts as a wrapper for the low-level library like tensorflow. Pandas and numpy are used to manipulate data and convert it into tabular form. Matplotlib is used to visualize accuracy and loss of each optimizer.

C. Coding

We have used google colab to for implementation as it provides free GPU support, all the major python libraries such as TensorFlow, Scikit-learn, Matplotlib,etc. are already installed, supports bash commands, etc.

To load the data into pandas data frame we have used get_file method. We have used this get_file method because by using this we don't have to upload the data set every time to the Colab directory. The snippet of the code is show in figure 1.

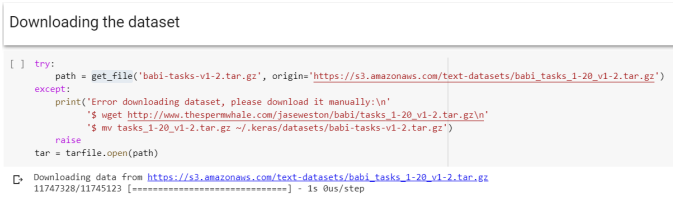


Fig. 1. Load Dataset using get_file method

In order to train and test stories, we have used two different text files which can be seen in the figure 2.

Fig. 2. Getting train and test stories

D. Hyper Parameter Tuning

Hyper-parameters are the variables which our algorithm uses to adjust to the data. In our model following are the hyper-parameters:

- LSTM size- 64
- Epochs- 50
- Batch Size- 32

- Optimizer- Adam
- Dropout value- 0.3

After analysing the accuracy for different combination of the hyper-parameters, we have received highest accuracy when LSTM size is 64, optimizer is Adam, epochs are 50, batch size is 32 and dropout value is 0.3. We received accuracy of 95.56% for the training dataset. For testing, we are comparing ground truth to the predicted value of the answer obtained from the trained model.

E. Data Pre-Processing

We have used a dataset from amazon web services which contains first set of 20 tasks in the bAbI project to test text comprehension and reasoning. For pre-processing of data we have merged stories, queries and answers to create the vocabulary. Then, we have performed vectorization and tokenization on the vocabulary for further implementation.

Snippet **??** shows vocabulary creation and snippet 4 shows Vector formation for the created list of unique words.

```
[ ]  vocab = set()
     for story, q, answer in train_stories + test_stories:
         vocab |= set(story + q + [answer])
     vocab = sorted(vocab)

     # Reserve 0 for masking via pad_sequences
     vocab_size = len(vocab) + 1
     story_maxlen = max(map(len, (x for x, _, _ in train_stories + test_stories)))
     query_maxlen = max(map(len, (x for _, x, _ in train_stories + test_stories)))
```

Fig. 3. Vocabulary Creation

```
-
Vocab size: 22 unique words
Story max length: 68 words
Query max length: 4 words
Number of training stories: 10000
Number of test stories: 1000
-
Here's what a "story" tuple looks like (input, query, answer):
(['Mary', 'moved', 'to', 'the', 'bathroom', '.', 'John', 'went', 'to', 'the', 'hallway', '.'], ['Where', 'is', 'Mary', '?'], 'bathroom')
-
Vectorizing the word sequences...
```

Fig. 4. Vector formation

F. Proposed Model

To build a neural network we have two input layers: one is used to input story and the other one is used to input question. Then, we have added two sequential layers one for each input layer.

A dot layer is added to combine the output of previous sequential layers and application function ids applied to this layer. A sequential layer is applied to the output of the activation function.

The output of this layer and activation layer is combined in the add layer and LSTM layer is added which is basically used to prevent itself from the problem of vanishing gradient.

In practice, RNN need to hold up the information for very long, but usually this does not happen so we use LSTM for that purpose.

Then, output of LSTM layer is fed to dropout layer for regularization followed by dense layer which has activation function softmax which can be understood by the figure 5.

Layer (type)	Output Shape	Param #	Connected to
input_7 (InputLayer)	(None, 68)	0	
input_8 (InputLayer)	(None, 4)	0	
sequential_10 (Sequential)	multiple	1408	input_7[0][0]
sequential_12 (Sequential)	(None, 4, 64)	1408	input_8[0][0]
dot_4 (Dot)	(None, 68, 4)	0	sequential_10[1][0] sequential_12[1][0]
activation_7 (Activation)	(None, 68, 4)	0	dot_4[0][0]
sequential_11 (Sequential)	multiple	88	input_7[0][0]
add_4 (Add)	(None, 68, 4)	0	activation_7[0][0] sequential_11[1][0]
permute_4 (Permute)	(None, 4, 68)	0	add_4[0][0]
concatenate_4 (Concatenate)	(None, 4, 132)	0	permute_4[0][0] sequential_12[1][0]
lstm_4 (LSTM)	(None, 64)	50432	concatenate_4[0][0]
dropout_16 (Dropout)	(None, 64)	0	lstm_4[0][0]
dense_4 (Dense)	(None, 22)	1430	dropout_16[0][0]
activation_8 (Activation)	(None, 22)	0	dense_4[0][0]

Fig. 5. Model architecture

After our model is ready, we need to build training model. In order to make training model, we need to find the best and optimum hyper parameters which gives the best results for our proposed model. After performing Hyper-Parameter tuning, we came up with the optimum hyper-parameters which are shown in table 2.

TABLE II
OPTIMUM HYPER-PARAMETERS FOR PROPOSED MODEL

Hyper-Parameter	Value
LSTM size	64
Optimizer	Adam
Epoch	50
Batch Size	32
Dropout value	0.3

From the table 3 it can observed that the adam optimizer works best and gives the highest accuracy and lowest L1 Loss.

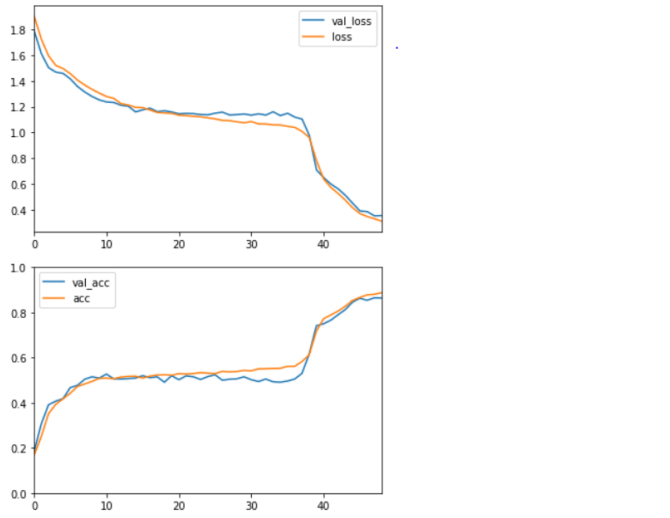
TABLE III
COMPARISON OF OPTIMIZER FOR L1 LOSS AND ACCURACY

Optimizer	L1 Loss	Accuracy
adadelta	0.2944	0.8934
adagrad	1.2328	0.50
Rmsprop	0.2546	0.9043
Adam	0.1266	0.9556

Visualization of performances of different optimizers can be depicted from figure 6, 7, 8 and 9.

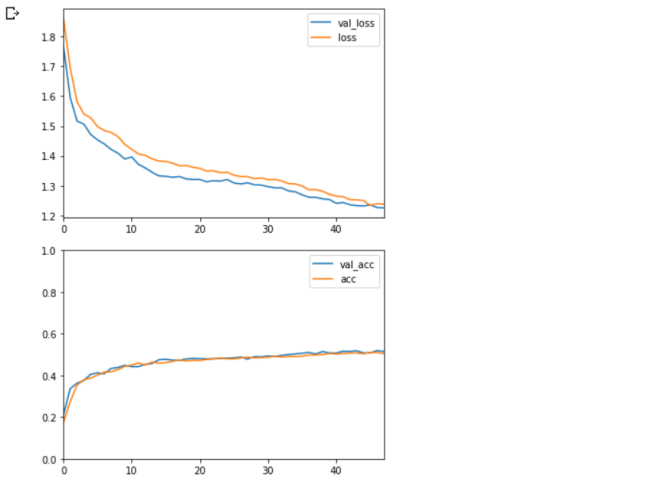
G. Model Storing

After the model is created, model is saved using the model.save function, so that anyone can use such model for the classification purpose just by loading the model on their



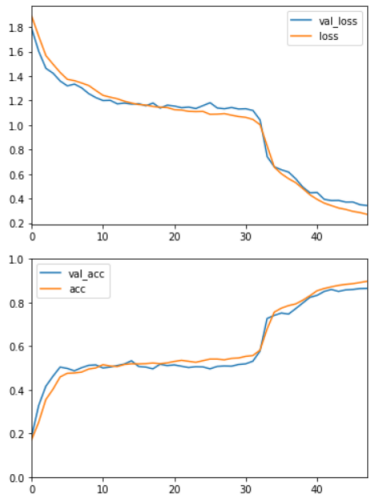

```
Epoch 50/50
 9600/10000 [===========================>..] - ETA: 0s - loss: 0.2944 - acc: 0.8934
```

Fig. 6. Adadelta



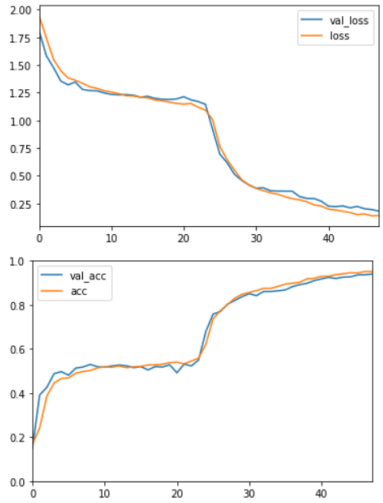

```
Epoch 49/50
 7936/10000 [======================>.......] - ETA: 0s - loss: 1.2328 - acc: 0.5095
```

Fig. 7. Adagrad




```
Epoch 49/50
 9280/10000 [==========================>...] - ETA: 0s - loss: 0.2546 - acc: 0.9043
```

Fig. 8. RMSprop




```
Epoch 49/50
 3488/10000 [=========>.....................] - ETA: 2s - loss: 0.1226 - acc: 0.9556
```

Fig. 9. Adam

own dataset. We have saved our model too, and loaded it again to try it on the testing dataset, which can be seen from Figure 10.

TABLE IV
CONTRIBUTION OF GROUP MEMBERS

Name	Percentage
Jagmeet Singh Sidhu	33.33%
Manva Trivedi	33.33%
Taresh Dewan	33.33%

ACKNOWLEDGMENT

We would like to thank our professor Dr. Thangarajah Akilan who made us learn the concepts of NLP during the lab sessions of our class. And along with him we would also thank both the TAs, Andrew and Punardeep who took keen interest in explaining us the way to do modular coding and telling us the back-end of the code being implemented.

CONCLUSION

In this project, we have used babi-tasks dataset from amazonaws.com which contains a larger number of text files for both training and testing dataset and each text file contains the short story, query and its corresponding answer, which is learned by our model, with Adamax as our optimizer,lstm size as 64, and batch size as 32. With such model we were able to learn and predict the correct answers for the asked questions as per the stories referred, with a high training accuracy of 95.56% which was way better than RMSprop, adadelta and

```
[19] model.fit([inputs_train, queries_train], answers_train, batch_size, train_epochs,callbacks=[TrainingVisualizer()],
                validation_data=([inputs_test, queries_test], answers_test))

     model.save('7_project_2_TT.h5')
```

Fig. 10. Model Storing

adagrad which was quite bad at it as can be seen from table 3.

REFERENCES

[1] https://medium.com/@akshaynavalakha/nlp-question-answering-system-f05825ef35c8

[2] An Overview of Question Answering System *R.Mervin* International Journal Of Research In Advance Technology In Engineering (IJRATE) Volume 1, Special Issue, October 2013

[3] https://pathmind.com/wiki/neural-network

[4] https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148