# Question Answering system using Siamese networks

Shashank Kumar
*Email: sk128@iu.edu*

Sumanth Gopalkrishna
*Email: sgopalk@iu.edu*

Srikanth Velpuri
*Email:svelpuri@iu.edu*

***Abstract*—We are surrounded by vast volumes of data in the forms of papers, blogs, websites and other forms of media and getting information from all these resources would take a lot of time .We want a direct response in most of these circumstances instead of looking into a whole document and Question Answering Systems are used for this purpose. These systems are an automated method of retrieving correct answers to the questions posed in natural language by humans as they scan through a whole document and provide the paragraph or relevant answer. However, in many applications, it's hard to find large amount of labelled data and we need to come up with prediction with whatever amount of data we have. In this study, we use a segment of Google Natural Questions dataset to train a Siamese network to get the best similarity from the question and answers embedding. In the Siamese network ,we used two different twin architectures, Bidirectional LSTM and BERT. We compare our results of both the architecture and evaluate it with state of art model.**

## 1. Introduction

Question Answering Systems targets the people who are looking for pieces of information from a large pool of documents. The requirement to query information from many formats has grown in importance and hence Question Answering Systems are increasingly growing in demand and are essential to meet this demand for query information content. In this study, we are looking at a version of QAS wherein we will be giving the long answer related to the given query from a pool of paragraphs from Wikipedia documents.

We are using the Natural Questions Dataset provided by Google which contains questions from real users. The questions are made out of real aggregated and anonymized inquiries made from google search engine and annotations are made out of top Wikipedia Search results. A single question is a rendered Wikipedia page which is tokenized and annotated by the annotator in each json record. The NQ represent a substantially harder research challenge than the previous question answering task like SQuAD 2.0 because the documents in which the answer is to found are much longer than the documents used in the existing question answering challenge [1].

## 2. Data Description

We are using the Natural Questions Dataset which contains questions from real users.It requires the QA systems to understand the entire wikipedia page which may contain the answer to the question and this is more challenging and realistic because of the inclusion of genuine user queries and the requirement that the machine had to read the complete page to locate the answer.The questions are made out of real aggregated inquiries made from google search engine. An annotator is given a query and Wikipedia Search results and it annotates a set of long answers. Some insight about the columns and data The data for this task comes in a rather complex form. The raw data is in a json file with each line of json containing the following:

- question_text: A string representing a question asked on Google.
- document_text: A string containing the HTML of the Wikipedia page relevant to the question.
- annotations/long answer: A json object containing the start and end index of the correct answer sub string (sub string of the document_text)
- long_answer_candidates: An array of the possible long answers to the questions. These are basically the paragraphs within the Wikipedia page.

## 3. Approach

The NQ dataset had more than 300k question from the actual google queries. Processing this large dataset is computational quite heavy and given the resource we had it was very hard. We want to see if we can achieve a good performance using only a segment of data. So to predict on a smaller dataset, we used the concept of information retrieval and semantic similarity to achieve a good prediction. We compared our results of long answer with the state of the art model.

After pre-processing , our dataset was very imbalanced, the proportion of right answers to wrong for a question is 1:120. Additionally, as we decreased our training data, we required a network which can extract information and give us sematic similarity between the question and answer pairs. For this reason we used a Siamese network to build embedding of the question and answers pairs and match the answer to the question which have the highest similarity.

Below are the key approaches we used.

1) Break down the document (Wikipedia search space) into parts and extract the text from it with the help of "long answer candidates" which have the information of various indices of a paragraph in Wikipedia. Label the true indices (answers) as one and others as 0. Convert the json into pandas data frame. Each row would represent a question text, answer text and its corresponding label.

2) Training Data: We load a subset of answers with a sampling rate of S such that there is only 1/S of negative labelled data. For eg: If there are 120 false answer for a questions, and sampling rate is 10 then we keep only 12 wrong answer for a question. Validation data: We don't do any sampling, we keep this as representative of real world scenario.

3) Built a Siamese network using two network architecture. First, using Bidirectional LSTM with Glove embedding followed by dropout and maxpooling layer and second, using uncased BERT embedding

# 4. Data Preprocessing

Data Pre-processing can be divided into three sections: Prepossessing for Question Answering System, Bidirectional LSTM and BERT. To convert the problem statement into a binary classification we Break down the document (Wikipedia search space) into parts and extract the text from it with the help of "long answer candidates" column, which have the information of various indices of a paragraph in Wikipedia. Label the true indices (answers) as one and others as 0. Convert the json into pandas data frame. Each row would represent a question text, answer text and its corresponding label. We sampled our training example based on the approach described above. Validation data was left untouched. We then remove the html tags present in the long answer candidates.

## 4.1. Data pre-processing for Bidirectional LSTM

For any NLP problem, we need to convert the textual data into a vector space so that the model can interpret the text. We first create tokens of word for the long answer and questions. We pad the sequence having the max length of 300 so that each sentences are of same length. We used a pretrained Glove Embedding with 100 dimension size to represent the tokens into vector space.

## 4.2. Data pre-processing for BERT

We are using uncased BERT present in the tfhub. We need to transform the text input to numeric token id. To convert into token id, we add "[CLS]" and "[SEP]" at the start and end of the input sequence. The input features to the BERT model are input word id, input masks and input type ids. Input word id represent word tokens from tokenizer, Input mask represent mask for the padding sequence. The mask has 1 for real tokens and 0 for padding tokens, so that attention can be given to real tokens. Input type ids only have one value (0) because this is a single sentence input. We create these three input features for question and answers for both training and validation sets.

# 5. Modelling

## 5.1. Siamese network for Question Answering

Siamese nets were first introduced in the early 1990s by Bromley and LeCun to solve signature verification as an image matching problem (Bromley et al., 1993). Siamese neural network is a class of neural network architecture that contains two or more identical subnetworks. The parameters between the two subnetworks networks are tied. Weight tying guarantees that the question and answer which are very similar could not possibly be mapped by their respective networks to very different locations in feature space because each network computes the same function. Intuitively, instead of trying to classify inputs, a Siamese network learns to differentiate between inputs, learning their similarity. Usually neural networks require a large amount of data to train, however, the Siamese networks works well even though the data is less to train. It is also robust to class imbalance dataset. Once the Siamese network has been tuned, we can capitalize the similarity and discriminative features to generalize the predictive power of network to the new data.

In the twin network of Siamese, we are using two different architecture to compare our results. First, using a combination Bidirectional LSTM, dropout and maxpooling layers and second is uncased BERT.

## 5.2. Bidirectional LSTM Model

Bidirectional LSTM train two LSTM's on the input sequence. The first LSTM runs on the forward direction and second on a reverse direction of the input sequence. Unidirectional LSTM only preserves the information of the past because it has only seen the inputs from the past. Bidirectional LSTM will run the input from both the direction. The LSTM that runs backwards preserve the information from the future and combining these two hidden states the information of both past and future is preserved.

## 5.3. Architecture of Twin network

The first layer consist of Glove embedding vectors, we add a layer of spatial dropout followed by a bidirectional LSTM layer and max pooling layer. We added 2D spatial dropout of with dropout rate of 0.1 to regularize our neural networks. Spatial dropout performs same function as dropout, however, it drops entire feature maps instead of

individual elements. We added maxpooling layer to reduce the dimension of the feature maps. Thus, reducing the number of parameter to learn and amount of computation performed by the network

TABLE 1. HYPERPARAMETERS USED FOR OUR BILSTM MODEL

| **Hyperparameters** | Values |
|---|---|
| Batch size: | 64 |
| Optimizer: | Adam |
| Loss | Cross entropy |
| Learning rate | 0.001 |
| epochs | 40 |

### 5.4. BERT Model

BERT stands for Bidirectional Encoder Representation from Transformers. It is basically a Encoder stack of transformer architecture. A transformer architecture is an encoder decoder network that uses self-attention on the encoder side and attention on the decoder side. We are using BERT uncased model which belong to BERT base model. It used 12 transformer blocks, with a hidden size of 768 and 12 attention heads. The model has been trained for English on the Wikipedia and BooksCorpus . The uncased means the inputs has been lower cased before tokenization into word pieces, and any accent markers has been stripped off . After doing the pre-processing ,we would have three inputs for BERT; word id, input mask and input type id. Using pretrained BERT embeddings we fine tune our model. The outputs of BERT is a dense vector representation which we get from [CLS] token in the sequence output of the BERT.

TABLE 2. HYPERPARAMETERS USED FOR OUR BERT MODEL

| **Hyperparameters** | Values |
|---|---|
| Batch size: | 8 |
| Optimizer: | Adam |
| Loss | Cross entropy |
| Learning rate | 0.001 |
| epochs | 3 |

## 6. Experiments

We trained our model for 20,000 question. In the training data, each questions had about 10-12 long answer candidates. So the size of the data frame was about 200k rows. We made prediction on different sizes of validation data ranging from 3k, 5k and 10k questions and collected the results. In validation data, the proportion of true labels were approx 1:120

For Bidirectional LSTM there were about 23M parameter to train. It took about 5 hours to train in Google colab pro. For BERT it had 109M parameter to train. It took 8 hours to run the model for 3 epochs.

We took batch size of 8 in BERT, as it kept throwing resource allocation error if we increased the batch size. The batch size of Bidirectional LSTM was 64. Both the architecture was optimized by Adam optimizer and the loss function used was binary cross entropy.

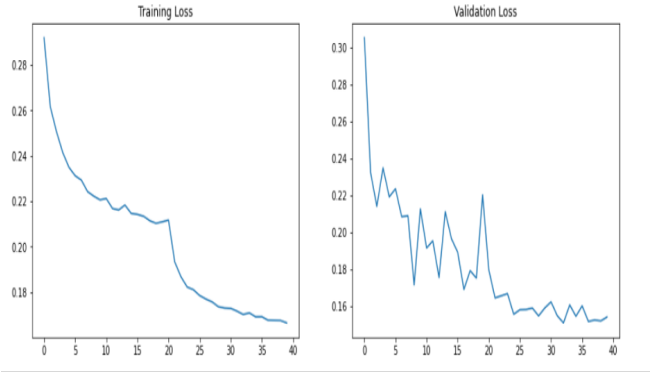


TABLE 3. MODEL PERFORMANCE COMPARISON

| **Models** | **F1 Score** | **Recall** | **Precision** |
|---|---|---|---|
| Siamese biLSTM | 57.6 | 45.5 | 80.4 |
| Siamese BERT | 39.4 | 32.2 | 50.5 |
| Google BERT | 66.2 | 64.1 | 68.3 |

We wanted to check the performance of F1 score by varying the validation data. Table 4: give those results

TABLE 4. BILSTM COMPARISON WITH DIFFERENT VALIDATION SETS

| **Validation Size** | **3k** | **5k** | **10k** |
|---|---|---|---|
| F1-Score | 63 | 60 | 57.6 |

Note: The testing data for Google's BERT joint model and then model we developed are different. They have tested in a very large corpus of data compared to us. We kept their results to bench mark our performance.

The recall for the bidirectional LSTM is high and precision is low suggesting that False positive is high in the prediction and the model are predicting more number of True than required. Surprisingly, the BERT model under formed the Bidirectional LSTM. We might need decode this issue.

## 7. Application in real world

In many fields, real life scenarios are often left untouched by the newest advances in research. They usually required solution for specific task applied to a specific domain, all the while providing the small amount of data to begin with. Our model can be used in multiple scenarios where the amount collected is not very huge. One such scenario where it would be really helpful is understanding the tax bills. Whenever the government introduces some legislation or changes in tax structure, the document is quite big to go through all the pages. A large share of professional

people has some legit doubts and it's hard to go to all the pages. These types of Question Answering system can be trained and used in the production to answer those queries

## 8. Conclusions and Next steps

We have demonstrated that it is possible to create a question answering system with a smaller dataset. As of now, the Bidirectional Siamese network is giving results which is not too far from state of the art model. However, the model is predicting lot many True values than required. We might need to fine tune our model further. One of the thing which we can try is to change is the class weights and see the variation in performance. In respect to BERT Siamese model, the model performed far below results than expected. We might to debug the model further.

The authors would like to thank...

## References

1 https://arxiv.org/pdf/1901.08634.pdf

2 https://arxiv.org/pdf/2004.14448.pdf

3 https://www.geeksforgeeks.org/explanation-of-bert-model-nlp

4 https://tfhub.dev/tensorflow/bert_en_uncased

5 https://www.cs.cmu.edu/ rsalakhu/papers/oneshot1.pdf

6 https://keras.io/examples/vision/siamese_network

7 https://medium.com/@raghavaggarwal0089/bilstmbc3d68da8bd0