Abstract

Project Progress Report

Archie: Building a Question Answering model

**Noul Singla Vipul Sharma**

Northeastern University, Northeastern University,

Boston, Massachusetts Boston, Massachusetts

Singla.n@husky.neu.edu sharma.vip@husky.neu.edu

This project will be based on building a machine learning model which can read an article and then based on the article, answer a few questions. Dataset used in this project is taken from the Stanford: Stanford Question Answering Dataset (SQuAD).

1 Problem Description

We all have interacted with Wikipedia articles for our research or just for casual reading about a topic, and many times thought if there would be a better option to skim through it to get what we want. Even in our daily life reading an article, a newsfeed or some social media post, we want to just quickly scroll through all the stuff and get our curiosities answered.

The main problem we face is that we have questions in our heads and simple search through the article doesn’t yield any useful information we are searching. Search results on a page can help us search a word but there is no way that we can ask a question which is answered automatically. With the recent progress in Machine learning and deep learning methods in the Natural Language Processing (NLP), there is a belief that things that needed a manual intervention can be now automated using these advanced the concepts. NLP with these machine learning models gives the machine the ability to understand human language and semantically parsing the language into questions for machine to understand and respond in natural language. In this project an exploration of such a functionality is being considered with usage of Neural network or deep learning framework to carry out this complex task. Using this technology will not only save time but can enhance the capability of the digital assistants.

2 Reference/Related work

The SQUAD dataset is downloaded and placed in the data/squad folder. The input data is a json format with the context or paragraph at first, followed by an answer and the question that should be asked from the paragraph. The main JSON file containing data is read and is tokenized by the spacy tokenizer tool and distributed into four files including context, question, answer and answer span. Lines in the files are aligned to each other. Each aligned line in the answer span file contains two numbers: the first number refers to the index of the first word in the answer in the context paragraph. The second number is the index of the last word of the answer in the context paragraph.

3 Methodology

Pre-Processing:

The SQUAD dataset is downloaded and placed in the data/squad folder. The input data is a json format with the context or paragraph at first, followed by an answer and the question that should be asked from the paragraph. The main JSON file containing data is read and is tokenized by the spacy tokenizer tool and distributed into four files including context, question, answer and answer span. Lines in the files are aligned to each other. Each aligned line in the answer span file contains two numbers: the first number refers to the index of the first word in the answer in the context paragraph. The second number is the index of the last word of the answer in the context paragraph.

The data is then vectorized using GloVe [2] word embeddings. GloVe is an unsupervised learning algorithm for obtaining vector representation of words. Training is performed on aggregated global word-word co-occurrence statistics for a corpus, and resulting representation showcase interesting linear substructure of the word vector space. GloVe word embeddings of dimensionality d = 100that have been pretrained on Wikipedia 2014 and Gigaword 5 are downloaded and stored in the data/squad subfolder. We are using the embedding dimensionality of 100 throughout the initial phase.

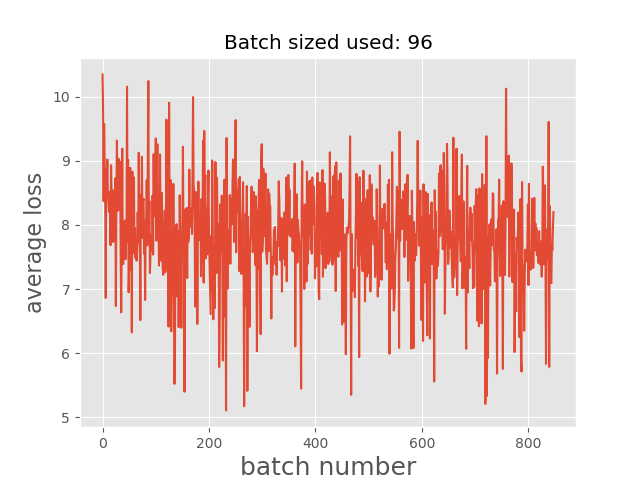
For SQuAD dataset, the train dataset is split into two parts a train or a development data and a val or validation data. We are training the data and validating the result on the validation data throughout the process. The final test data is held by the SQuAD developed where we can submit the final model to get the result.

The Real Work:

We start with preprocessing the data to get the multiple files which can be used for the machine learning activity. After processing the dataset, we need to work on the contextual binding of these vectors. We have used the glove vector to get the words vector format which can be used to train a model based on loss and error. The currently implemented model has a Recurrent Neural Network implement in python using the Tensorflow architecture. In this layer the paragraph or context is used to learn the model based on the vector position and then based on the question and answers vectors. This is based on the sequence to Sequence or sq2seq generation based on encoder and decoder functionality using the RNN as the base layer. To prevent the one to word mapping and getting more context to the data, a windows size can be configured which will work in symmetry with the previously used words of the given window size. Currently used window size is 3 or 4. The encoder tries to encode the vectors into new features and decoder tries to use those features to decode the reduced information into the original information.

The optimization function used is the adam optimizer which is an adaptive gradient descent method already present in the Tensorflow architecture. Learning rate is set at .0048 so that the process grows slow and reach a global minimum for error and does not overshoot minima. Loss function which has been defined has to be reduced to the minimum and it is defined based on the two separate losses: loss1 and loss2. Both losses are defined based on the cross entropy and softmax function. while loss1 is trying to reduce error of not starting the answer at the right position, loss2 tries to reduce the error that the answer is not ended at the right position.

To implement the window and improve the performance an implementation of Bi directional long short-term memory with RNN is used rather than a plain RNN layer. Here the BiLSTM layer takes care of the RNN and window functionality before the encoding and decoding of the data.

The model has many other configurable parameters which include the batch size i.e. how many contexts to be fed into the model at once and validation size that defines after each batch, on what size should the model be tested. These variables are changed as per empirical methods to improve end results. The list of configuration parameter changed, and the results is provided in the next section.

4 Result and Evaluation

The evaluation of the SQuAD dataset models is done conventionally using the F1 and EM score and we plan on using the same evaluation metrics. The dataset has is already divided into learning and dev test dataset which would be used for model learning and model evaluations at a local level. There is also a provision to utilize an online test set to submit the model which can be explored if needed for further validation.

F1 score is based on the confusion matrix while EM means Exact Match which is used with the SQuAD dataset frequently to specify how much of the answers were exactly correctly predicted.

We have trained 2 models with considerable results and the configuration. Results are as follows: -

**Parameters**

batch\_size: 96

eval\_num: 250

window\_size: 4

samples\_used\_for\_evaluation: 500

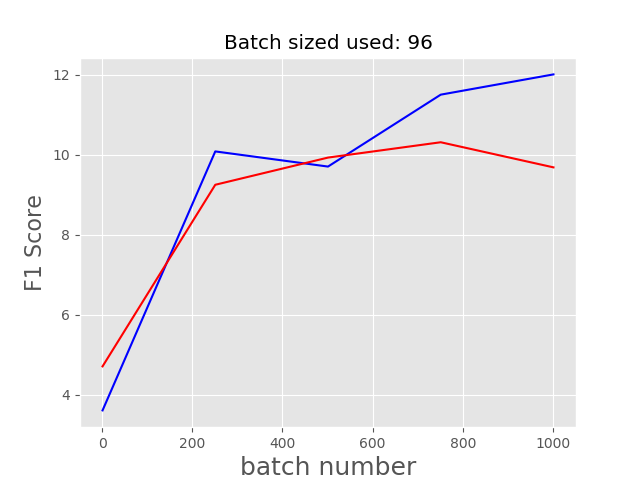
num\_epochs: 1

learning\_rate: 0.048

F1 score in this case maxed out at 12 on the train set while at 10 on the validation dataset. The batch size 96 and epoch very considered low and we implemented another model to improve this. Exact Match was 7 on the train while 5 on the validation and the loss function did not change much even though it did decrease.

**Parameters**

A close up of a map

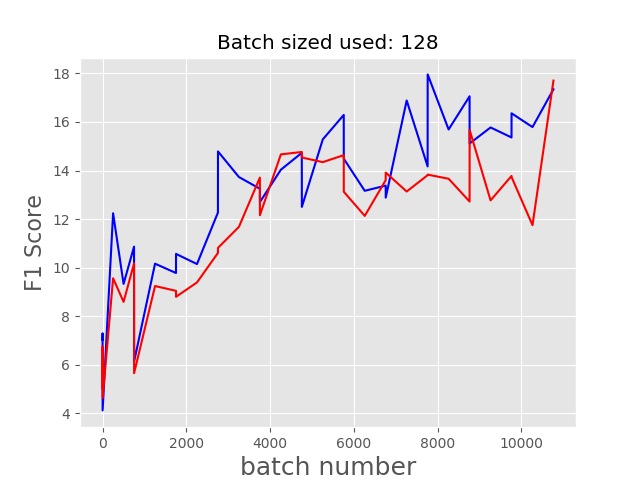
Description generated with high confidencebatch\_size: 128

eval\_num: 500

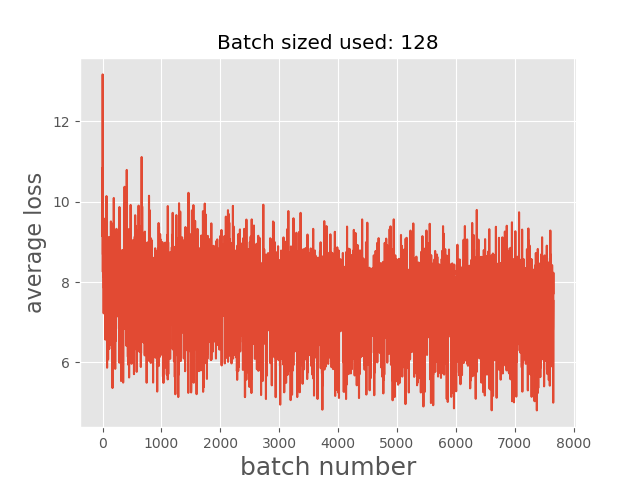
window\_size: 3

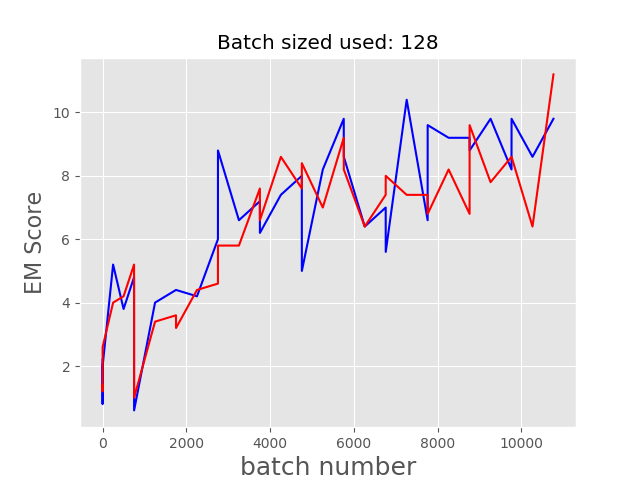
num\_epochs: 10

learning\_rate: 0.048



F1 score in this case reached a maximum of 18 on the train set while at 17 on the validation dataset. The batch size 128 and epoch were able to give a lot more data to the neural network to learn and hence it performed quite well on the train as well as test data. EM value reach to a maximum of 11 on the validation which is quite high from previous run. The loss function has a subtle drop but did not change drastically with each run.



5 What is working and What is wrong

The RNN model with BiLSTM is working and giving acceptable results and there is a need for further scope of improvement by tuning the hyperparameters. Even though there is scope of improvement, expecting this model to be significantly important in the long run is not worthy to consider. Performance is not good enough as we are unable to configure Tensorflow on the GPU/CUDA because of the version issues and working to resolve that issue to utilize the computation to vary more parameters and improve the performance.

Model is not good enough to be considered as at a mature level to get the most of it from the current stage.

6 Future Work

The major drawback in the current implementation and what the BiDAF model overcomes is using the attention to get a better context from the data. Here using window of a fixed size have 2 issues, firstly we can only use a fixed number while the context can be in varying length on the text and secondly, we are not able to keep only the important context even if we increase the window size. We will try to use an attention layer to top of the present work to implement the working of the BiDAF model and see how much it will help the case. So, enhancing current model to include to improve the output and accuracy are the steps we are considering taking forward.

References

[1] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. [pdf] [bib]

[2] [Stanford. 2014]. Jeffrey Pennington, Richard Socher, Christopher D. Manning. *GloVe: Global Vectors for Word Representation*

*[3]* [*https://google.github.io/seq2seq/*](https://google.github.io/seq2seq/)

*[4]* [*https://github.com/google/seq2seq*](https://github.com/google/seq2seq)

*[5]* [*https://www.cs.cmu.edu/~rsalakhu/10707/Lectures/Lecture\_Seq2Seq1.pdf*](https://www.cs.cmu.edu/~rsalakhu/10707/Lectures/Lecture_Seq2Seq1.pdf)

*[6]* [*https://machinelearningmastery.com/encoder-decoder-recurrent-neural-network-models-neural-machine-translation/*](https://machinelearningmastery.com/encoder-decoder-recurrent-neural-network-models-neural-machine-translation/)

*[7] Stanford CS224N lecture available at: https://www.youtube.com/playlist?list=PLqdrfNEc5QnuV9RwUAhoJcoQvu4Q46Lja*