Project Progress Report

Archie: Building a Question Answering model

Noul Singla
Northeastern University,
Boston, Massachusetts
singla.n@husky.neu.edu

Vipul Sharma
Northeastern University,
Boston, Massachusetts
sharma.vip@husky.neu.edu

Abstract

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

Broadly speaking, a large section of Machine Comprehension technique utilize Attention/memory which helps the model develop an understanding of which parts of one piece of text are relevant to another piece to text. A common pattern that I found in works for this problem was:

- Encode both the question and answer.
- Combine both encodings using attention/similarity mechanisms.
- Combine the results of the previous step.
- Produce a classification over the combined representation.

The finer points of models within each step differ quite a bit. For the encoding step, a bidirectional RNN was popular choice, with some variation in the cell used within the RNN. LSTM's are a popular choice as the cell, e.g. in "Multi-perspective Context Matching for machine Comprehension" by Wang et al.[1], in "DYNAMIC COATTENTION NETWORKS FOR QUESTION ANSWERING" by Xiong et al[2]. On the other hand, Chen et al[3]. propose using GRU in "A Thorough Examination of the CNN/Daly Mail Reading Comprehension Task", citing that it sped up their implementation as compared to an LSTM, without significant losses in performance.

The attention/similarity mechanism has a wide range of variation proposed in different works, with this step being the bulk of innovation. Other implementations range from using simple Attention to BiDAF.

For our project we are using RNNs with the LSTM cells and applying Adam Optimizer on top using the seq2seq model.

The official website for SQuAD maintains a leaderboard with submissions from various organizations like Alibaba, Google and Microsoft and individuals; who can build a model that works on the expectations of the dev set and submit it to get official scores on the dev and a hidden test set.

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.

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.

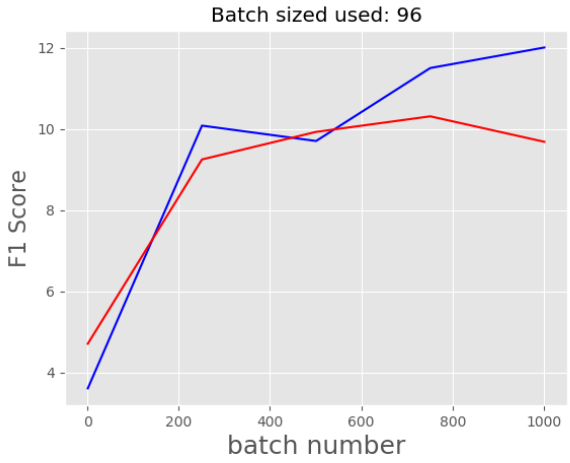
The model has many other configurable parameters which include the batch size i.e. how many contexts to be fed into the model at once and validation size that defines after each batch, on what size should the model be tested. These variables are changed as per empirical methods to improve end results.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

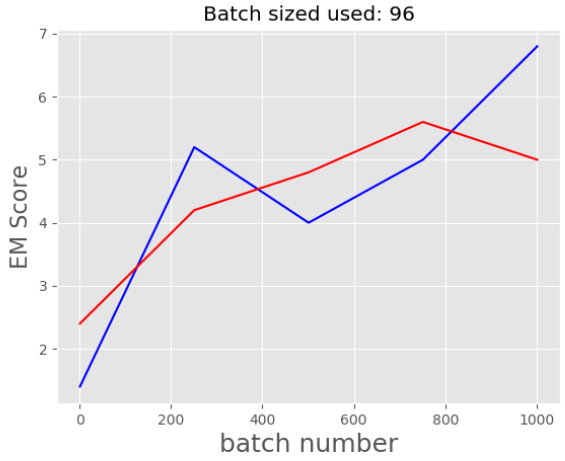
4 Result and Evaluation

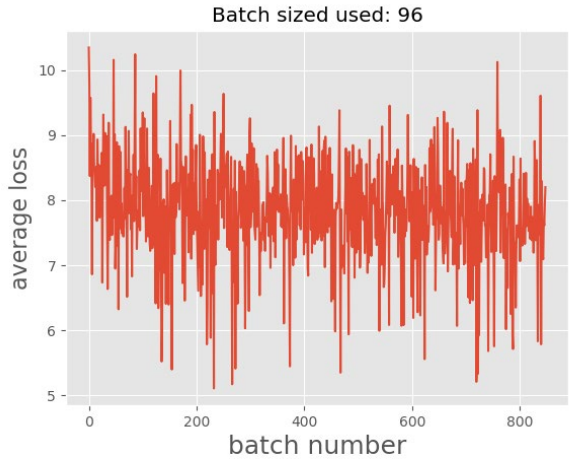
The evaluation of the SQuAD dataset models is done conventionally using the F1 and EM score and we plan on using the same evaluation metrics. The dataset has is already divided into learning and dev test dataset which would be used for model learning and model evaluations at a local level. There is also 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. The Baseline of the model is already specified on the Squad Website and is set at 20 for F1 score which will be considered as a standard baseline. We also consider our implementation of a basic Seq2seq model with 1 epoch as the baseline for our model and try to improve this to outscore the standard baseline of 20 and also the baseline of basic Seq2Seq without any modifications and enhacements.

We have trained multiple models with learning from the results and making changes on the configuration. Results are as shared henceforth.
NOTE: In line charts, blue is for training and red for test set.
Parameters
batch_size: 96
eval_num: 250
window_size: 4
samples_used_for_evaluation: 500
num_epochs: 1
learning_rate: 0.048

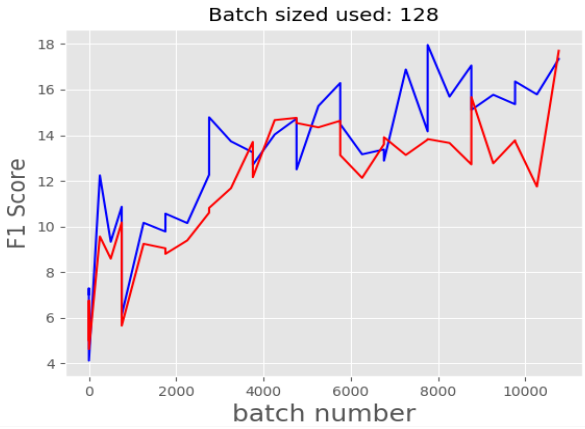


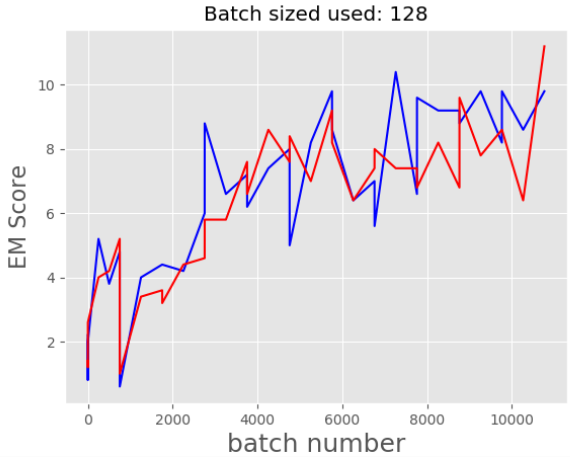




F1 score in this case maxed out at 12 on the train set while at 10 on the validation dataset. The batch size 96 and epoch very considered low and we implemented another model to improve this. Exact Match was 7 on the train while 5 on the validation and the loss function did not change much even though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

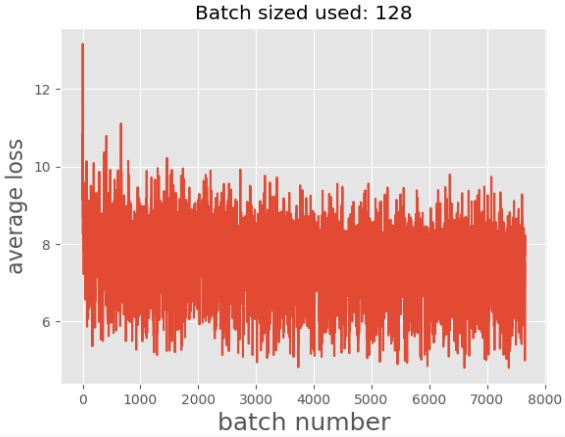
Next step was to tune the model with increasing train size.





Parameters
batch_size: 128
eval_num: 500
window_size: 3
num_epochs: 10
learning_rate: 0.048

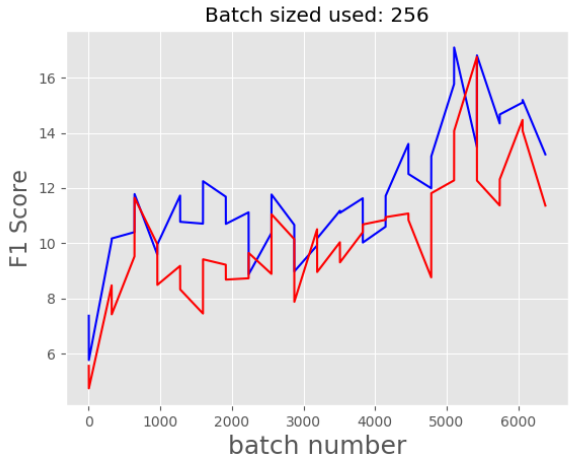
F1 score in this case reached a maximum of 18 on the train set while at 17 on the validation dataset. The batch size 128 and epoch were able to give a lot more data to the neural net-



work 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.

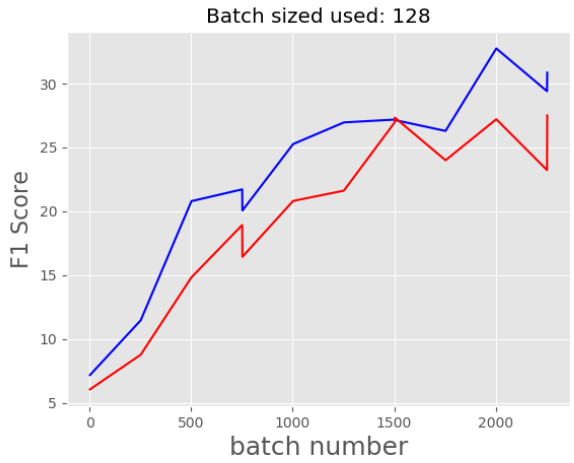
We then tweaked the parameters to increase the epoch and batch size to see if there is significant improvement.

Parameters
batch_size: 256
eval_num: 500
window_size: 3
num_epochs: 20
learning_rate: 0.048



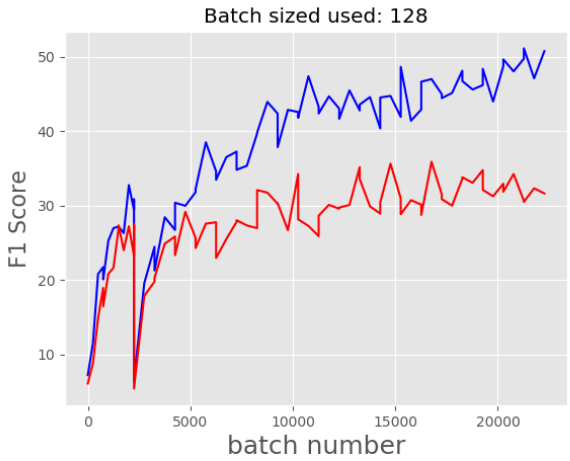
There was a need for better learning and we further needed to train a better model and the first step was to increase the epochs and batch size. Here we tried to use the important feature of LSTM to increase the window size and also common to all ML model of reducing the learning rate so that optimization functions find a better global minimum.

Parameters
batch_size: 128
eval_num: 250
window_size: 8
num_epochs: 15
learning_rate: 0.0048
hidden_size: 200



The model showed a significant improvement over the basic model and outperformed the standard baseline. To continue better performance, both epoch and window size were further improved and at same time evaluation at each batch was increased to 500 sample for prevent any over-fitting.

Parameters
batch_size: 128
eval_num: 500
window_size: 10
num_epochs: 20
learning_rate: 0.0048
hidden_size: 200



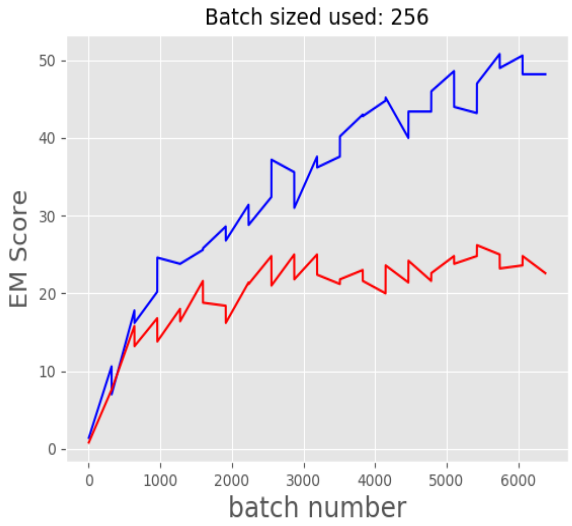
This model even though not many changes were made, outperformed the previous model and our expectation. For the first time, an F1 score of 50 was breached. We noticed that there is a strange drop in one of the batches, this is surely a sign that the model is still overfitting someplace and the results were too volatile and not a smooth.

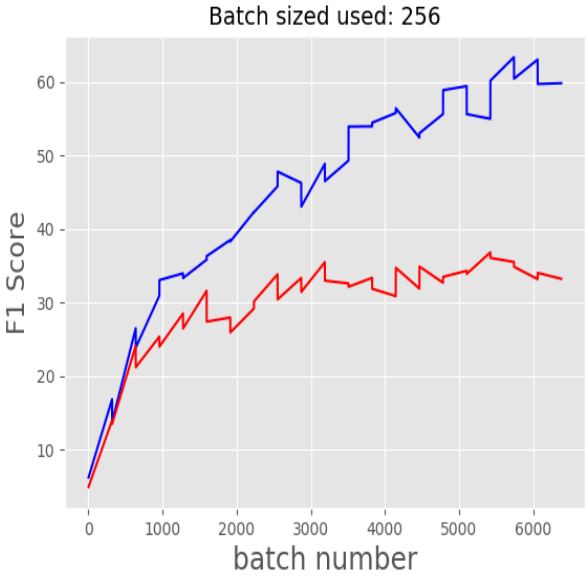
This prompted us to use even a smaller learning rate, and we ran our final model with a learning rate of .003. At the same time, we increased the batch size to make sure that learning is over a considerable batch and chances of overfitting are reduced.

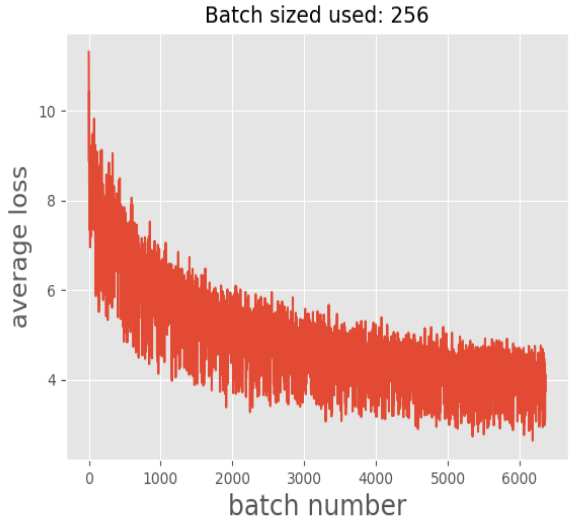
This was done with aim to not just improve the model but to also prevent overfitting and also achieve a smooth fit on the data.

Parameters
batch_size: 256
eval_num: 500
window_size: 10
num_epochs: 20
learning_rate: 0.003

hidden_size: 200







Finally, the F1 score in this case reached a maximum of 64 on the train set while at 37 on the validation dataset. The batch size 256 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 51 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.

We have tried to use a less resource intensive model on the entire dataset and achieve a good predictive ability. Considering the improvement and the final score, the results look promising.

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] Zhiguo Wang, Wael Hamza, and Radu Florian. Bilateral multi-perspective matching for natural language sentences. arXiv preprint arXiv:1702.03814, 2017

[2] Caiming Xiong, Victor Zhong, and Richard Socher. Dynamic coattention networks for question answering. arXiv preprint arXiv:1611.01604, 2016.

[3] Danqi Chen, Jason Bolton, and Christopher D Manning. A thorough examination of the cnn/daily mail reading comprehension task. arXiv preprint arXiv:1606.02858, 2016.

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

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

[6] *https://google.github.io/seq2seq/*

[7] *https://github.com/google/seq2seq*

[8] *https://www.cs.cmu.edu/~rsalakhu/10707/Lectures/Lecture_Seq2Seq1.pdf*

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

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