Research of LSTM Additions on Top of SQuAD BERT Hidden Transform Layers

Adam Thorne, Zac Farnsworth, Oscar Matus
Stanford University
Stanford, CA 94305
athorne@stanford.edu, zacharyf@stanford.edu, omatus@stanford.edu

Abstract

The Stanford Question Answering Dataset (SQuAD) is a reading comprehension dataset, consisting of questions posed by crowdworkers on a set of Wikipedia articles, where the answer to every question is either a segment of text from the corresponding reading passage, or unanswerable [1]. Current NLP tasks aim at answering these questions at as close to a human level of accuracy as possible. Improving reading comprehension NLP models is an extremely pervasive subject, as online search machines could be improved significantly with increased robustness in these models. One NLT model currently ranked extremely high on the SQUAD leaderboards is Bidirectional Encoder Representations from Transformers (BERT). Our aim in this project was to improve on F1 and EM scores from a baseline BERT implementation by modifying the architecture to use Long-Short-Term-Memory (LSTM). We trained a baseline BERT model [2] on the SQUAD 2.0 dataset (a series of over 150,000 reading comprehension questions, where over 50,000 questions do not have an answer found in the associated text) which yielded satisfactory F1 and EM scores. From there, we modified the architecture to use an LSTM to attempt to improve on the scores acheived by the baseline BERT model. We successfully trained a modified BERT model with an LSTM, but were unable to tune the model to achieve better results than the baseline.

1 Introduction

Since its inception, the SQuAD challenge has seen various techniques come to be known as the "state of the art", only to soon be surpassed by some more advanced method. When the original ELMo (Embeddings from Language Models) paper [3] was published in early 2018, it revolutionized the challenge and easily became the top performer. However, just a few months later, the original BERT (Bidirectional Encoder Representations from Transformers) [2] paper surpassed ELMo to become the new state of the art. Since the publishing of the original BERT paper, many people have experimented with modifications to or fine-tuning of the baseline BERT model to achieve even better results. Clearly, the standard of excellence is constantly changing, and there is always room for a little improvement to any model. Discovering exactly what can be done to improve a model is the true challenge that those wishing to tackle the SQuAD challenge face.

The problem we wish to solve for the SQuAD challenge is that, given a segment of text (the "context") and a corresponding question, we wish to return the answer to that question (which will be a certain phrase from the context). The challenge is further complicated by the fact that some questions are unanswerable given the context. Current models have nearly matched human performance at this task, but there is still some room for improvement. This is important because if a model can truly excel at this task, it opens up new possibilities for asking questions on search engines, asking questions about an article, or other AI related tasks that can make everyday tasks more convenient and efficient.

Most of the best performing current methods use BERT in some way. There are many different ensembled models and fine-tuned models currently in top positions on the leaderboard. However, we are not aware of any attempt to modify the BERT architecture by including an LSTM in the network, and we believe that this may be an effective way to improve on the baseline BERT model.

2 Related Work

Many different techniques have been employed to attempt to solve the SQuAD challenge. Some of these techniques use pretrained contextual embeddings (PCE models). We chose to focus on this type of model for our project.

We were intially inspired by the paper, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding" [2]. In this paper, the authors introduce the BERT (Bidirectional Encoder Representations from Transformers) language representational model. BERT works by jointly conditioning on the left and right contexts to create a set of pre-trained deep bidirectional representations. This paper was the new state of the art when it was published, and many attempts have been made to fine tune this baseline BERT algorithm to achieve marginally better results. Nearly all of the top entries on the SQuAD leaderboard use BERT in some way.

Another paper that sparked our interest was "A Long Short-Term Memory Model for Answer Sentence Selection in Question Answering" [4]. The authors implemented a bidirectional long-short-term-memory (LSTM) to attempt to answer questions. The interesting thing about this paper is that they did not perform any syntactic parsing, rather they only relied on relevance scores and keywords. Their model successfully outperformed previous similar models. The authors of the paper, "LSTM-BASED DEEP LEARNING MODELS FOR NONFACTOID ANSWER SELECTION" [5] explore a similar technique, although they also explore combining bidirectional LSTM's with convolutional neural networks. The authors of this paper also achieved strong results. These papers suggest that LSTM's can successfully be used in question answering tasks.

Another paper that was significantly different from our task but nonetheless relevant and interesting was "Stacked Attention Networks for Image Question Answering" [6]. This paper takes on the task of answering questions based on provided images. Although the task is quite distinct, the authors of this paper still used an LSTM to encode the text of the question, which gives credence to the idea that LSTM's can be useful, in general, for question answering.

3 Approach

Our approach to this project can be divided into two subsections - setting up and understanding a baseline model to answer the SQuAD 2.0 questions, and implementing our own advancements to the existing model. These two phases served the functions of allowing us to become acquainted with the models, functions, and datasets in a natural manner, and afterwards look for places to provide improvements to the existing framework.

3.1 BiDAF

We began this problem by training the SQuAD 2.0 dataset on a baseline Bi-Directional Attention Flow (BiDAF)[7] provided to us from the Chris Chute [8] repository. The BiDAF model creates character embedding and word embeddings for the words in the context and query vocabularies. These character and word embeddings are combined and passed into a contextual embeddings layer (a bi-directional long short-term memory network (LSTM)) in order to model temporary interactions between words. The outputs from the contextual embeddings are passed through an attention flow layer, which is then passed into a modelling layer and finally a softmax output layer to yield predictions.

3.2 BERT

After training the BiDAF baseline, we attempted to get the HuggingFace [9] implementation of BERT to integrate with our custom datasets. BERT is a multi-layer bidirectional transform encoder. Transforms are multi-head attention mechanisms (encoders and decoders) that are used in sequence dependency NLP tasks [10] and have been proven to be more effective than Recurrent Neural Networks (RNN) and LSTMS. In short, Tranformers attend to the problem in ways that RNNs cannot, due to their parallel nature. They look at all words in the window instantaneously, and learn importance and position in parallel. This meaning that words can "see" each other in ways impossible with an RNN, transformers are an especially useful tool when position and context for a word vector are so important.

We began training the baseline BERT implementation by cloning the HuggingFace repo, adding a folder containing our datasets, and running the training with the default hyperparameters. However, this process proved to be far more challenging than we initially thought.

After trying many different configurations and after numerous unsuccessful attempts that ran out of memory, we arrived at a working configuration of hyperparameters. These (and the attempted hyperparameters) are described in the "Experiments" section.

3.3 Adding LSTM

With the goal of extending an already impressive model into a more successful question answering architecture, we explored the ideas of adding either a neural network or an LSTM to the final fully connected layer of the transformers within Bert. This idea was presented in lecture on 2/19.

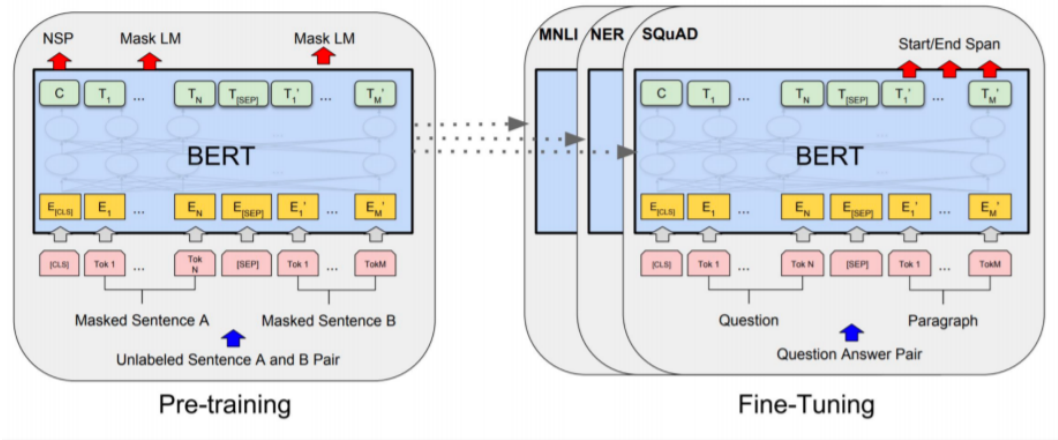


Figure 1: HuggingFace BERT architecture

For our specific task, we spent some time debating what architecture we'd like to append to the BERT embeddings and settled on a Bidirectional LSTM. We recognized the advantages that a bidirectional LSTM would have over an NN or an RNN such as resource efficiency and bidirectional word awareness, as well as the added benefit of our team being somewhat familiar with their implementation (based off the previous homeworks).

Based off the lecture slides, and the sequential notion in the embeddings output from the BERT transformers, we decided to place the LSTM at the output of the sequential transformers. The LSTM was initialized as 1-layer and bidirectional, with a dropout of .1.

4 Experiments

4.1 Data

For our dataset, we used the CS224n - modified SQuAD 2.0 dataset [8]. This currently consists of 129,941 training examples, 6078 dev examples, and 5921 test examples. All of these examples consist of a question and its corresponding answer (or non-answer).

4.2 Evaluation method

For evaluations we will compare the F1 and EM scores from our architecture and (when we have added additional features on top of the baseline BERT model) compare these scores to the F1 and EM scores achieved by previous baseline BERT models. We hope that our model can eventually perform slightly better than the baseline BERT model.

EM stands for exact match and is the stricter of the two evaluation methods. EM is a binary measurement (true/false) of whether the answer is a true exact match. If not, no credit is given to the answer. An example is if the answer is 'Albert Einstein', but the model predicts 'Einstein', no credit will be assigned. F1 is the looser evaluation method. It is a function of truth and recall, embodied in the equation

$$score = \frac{2(prediction)(recall)}{precision + recall}$$

Where precision and recall are percentages of how many words from the predicted answer are in the actual answer.

4.3 Experimental Details

4.3.1 Parameter Summary

Table 1: Initial BERT Experiments

Model	BERT Baseline	BERT + Added Hidden Layers	BERT + Unidirectional LSTM Layer
Num Hidden Layers	12	15	12
Max Sequence Length	384	384	384
Batch Size	6	6	6
Learning Rate	3.00E-05	3.00E-05	3.00E-05
Num epochs	2	2	2
Type of LSTM	None	None	Unidirectional LSTM
Dropout Prob	0.1	0.2	0.1
Training Time	10 hrs	12 hrs	12 hrs

Table 2: Final BERT Experiments

Model	BERT + Bidirectional LSTM Layer	BERT + Bidirectional LSTM Layer, dropout
Num Hidden Layers	12	12
Max Sequence Length	384	384
Batch Size	6	6
Learning Rate	3.00E-05	3.00E-05
Num epochs	2	2
Type of LSTM	Bidirectional LSTM	Bidirectional LSTM
Dropout Prob	0.2	0.2
Training Time	14 hrs	14 hrs

4.3.2 BiDAF

We ran the train.py and test.py files in the Chute squad repository, with the default test and train parameters provided in the args.py file.

4.3.3 BERT Baseline

We uploaded the SQuAD v2.0 data into the HuggingFace repo and (eventually) found that the hyperparameters described in 4.3.1 lead to successful training. We initially thought that decreasing the max sequence length to 64 would help prevent memory issues on the VM, but ultimately found that it was still able to run with the longer length. We did decrease the batch size to 6 (the original parameter was 12), and this did help with the memory issues. The main obstacle in being able to train the baseline, however, was the fact that we weren't deleting a cache file (train-v2.0.json_bert-base-uncased_384_128_64) in the data folder of the huggingface repo when trying to tune hyper-parameters and get the baseline training to work. This led to memory errors every time we tried to re-run the model, as the cache grew to be too large.

4.3.4 BERT + Added Hidden Layers

From our BERT baseline, we hypothesized ways to improve upon F1 and EM scores by modifying/adding on top of the frozen layers in the BERT model. In the baseline BERT model there are 12 transformer/self attention layers that output into the linear classifier layer. We changed this to be 15 layers in the code and also increased the dropout probability to be 0.2 from 0.1. By increasing dropout, we hoped to prevent from over-fitting our more complex model to the training data.

4.3.5 BERT + Unidirectional LSTM Layer, dropout = 0.1

In our literature search we found multiple examples of using LSTMs for question answering and therefore decided to add one on top of the frozen BERT hidden layers. Our first attempt was a unidirectional and single layer LSTM added on top of the frozen layers. We trained this model with the initial parameters from our baseline BERT model.

4.3.6 BERT + Bidirectional LSTM Layer, dropout = 0.1

Our baseline LSTM implementation didn't create very promising results, and we realized that it was likely due to the fact that it was initialized to be unidirectional. Thus, it was learning the start and end indexes for question answering from a forward pass of the hidden embeddings instead of both forward and backwards passes. Thus, we implemented a bidirectional LSTM on top of the frozen BERT layers with the initial parameters from our baseline BERT model.

4.3.7 BERT + Bidirectional LSTM Layer, dropout = 0.2

Our last experiment was increasing the dropout probability of our bidirectional LSTM model from 0.1 to 0.2. The goal of this was once again to prevent our model from overfitting the training data and ultimately increase F1 and EM scores.

5 Results

5.1 Results Data

Table 3: Dev Set Scores

Model	EM Score	F1 Score
BERT Baseline	73.001	76.069
BERT + Added Hidden Layers	72.524	75.608
BERT + Unidirectional LSTM	0.148	4.014
BERT + Bidirectional LSTM	72.524	75.608
BERT + Bidirectional LSTM, dropout = 0.2	72.46 (TEST SET)	75.79 (TEST SET)

5.2 BiDAF

Our baseline BiDAF model yielded an F1 score of 61.28 and EM score of 57.72.

5.3 BERT

Our baseline BERT model yielded an F1 score of 76.069 and an EM score of 73.001 on the Dev set. This is comparable to scores from other groups who used baseline BERT implementations on SQUAD. The scores are also higher than the BiDAF baseline that we established, which is what we expected.

5.4 BERT + Added Hidden Layers

We expected that adding extra hidden transformer layers on top of our BERT baseline model would increase F1 and EM scores, but this unfortunately did not occur. We hoped that adding the extra transformer layers would only increase the ability of our model to understand sequence contexts and better predict the start and end indices of answers. However, it appears that overfitting occured and the model was too finely tuned to the training data.

5.5 BERT + Unidirectional LSTM

This experiment was our first attempt at training an LSTM. While we were excited that we had debugged our code to the point where successful training occurred, the results were abysmal. We think that we somehow wrote to our predictions.json file incorrectly. We ran our model with only the training flag and proceeded to run the model with the prediction flag separately instead of running both the training and predicting flags at the same time. Either way, we quickly realized that a unidirectional LSTM would not give us very robust predictions, as it wasn't learning contexts from both directions (start and end) in the sequence.

5.6 BERT + Bidirectional LSTM, dropout = 0.1

Our bidirectional LSTM implementation (a one layer LSTM on top of the frozen BERT layers with the same hyperparameters as the original BERT model) gave us F1 and EM scores extremely close to the baseline, which was promising. We were disappointed, however, that this model did not improve upon our baseline score. This was likely due to the fact that the LSTM overfit the training data and thus didn't improve performance on the dev set. Also, it should be noted that our BERT + Added Hidden Layers and BERT + Bidirectional LSTM had the exact same F1 and EM scores. We think that we somehow overwrote one of our predictions in one of these models, as the chances of having exact same evaluation scores with different models is very small.

5.7 BERT + Bidirectional LSTM, dropout = 0.2

We decided to submit a Bidirectional LSTM with an increased dropout probability (0.2) to the test set. The hope was that the increased dropout probability would prevent from overfitting, and allow our added LSTM layer to improve upon the BERT baseline. We unfortunately ran out of time to run this model on both the dev and test sets (seeing as our predictions were always extremely bad when we trained separately from predicting), so it is hard to tell whether this implementation is better than our BERT baseline, since they were both evaluated on different sets. Our final submission to the test set had EM and F1 scores of **72.46** and **75.79**

5.8 Results Summary

We feel that our experimental procedures had a logical flow to them, but in hindsight we would have changed our strategy at attacking this problem. Dedicating more time to hyperparameter fine tuning of the BERT baseline was likely a better use of our time than continuing to add different models on top of the baseline. We also should have been more aware of potential over fitting pitfalls when adding layers to the baseline before we added them - and focused more of our time and energy on hypothesizing ways to overcome this issue before implementing models and then trying to fix the overfitting issue. The analysis section attempts to explain why we saw the data that we did and provide further conclusions.

6 Analysis

Upon training multiple models on top of our BERT baseline we analyzed our results to glean insights as to why we were unable to improve upon our baseline score on the dev set.

6.1 Blank v Non Blank Answer Predictions

We first decided to compare how many times the BERT baseline model predicted a blank response (empty string, "") to a question versus our other models. The differences in proportions of our baseline

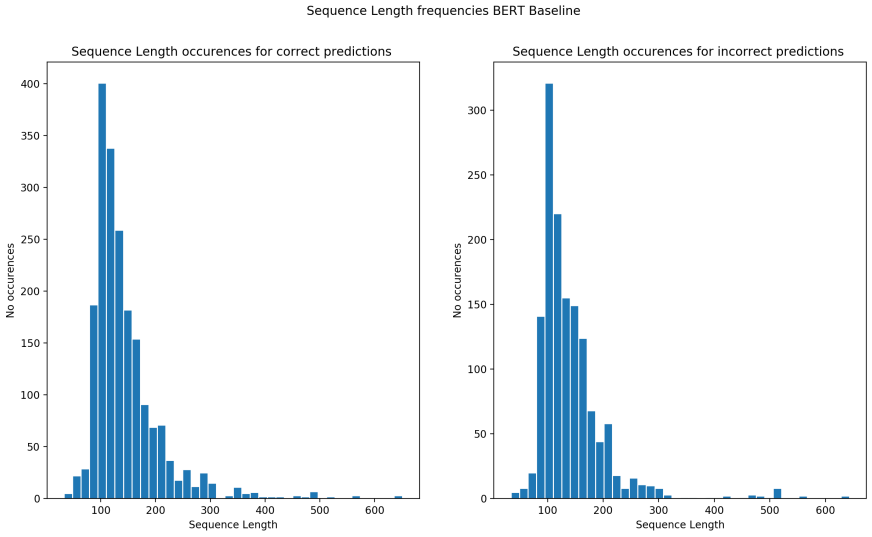
Table 4: Proportion of Empty and Non Empty Answers for our model

Model	% No answer predictions	% Answer Predictions
BERT Baseline	44	56
BERT + Bidirectional LSTM 0.1 dropout	45	55
BERT + Bidirectional LSTM 0.2 dropout	45	55

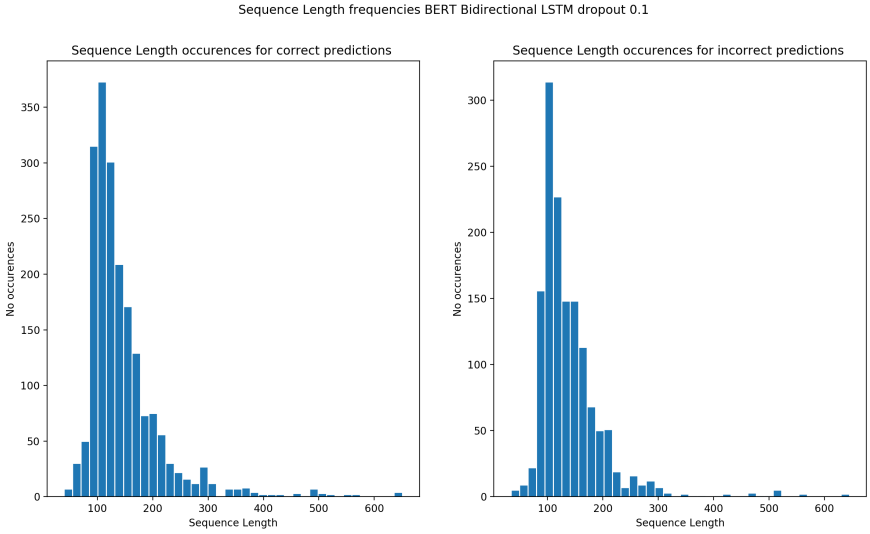
model to our LSTM models are minimal. However, the differences in F1 and EM scores between these three models are also minimal (differences of **0.477** and **0.461** for EM and F1 scores in LSTM v Baseline models, see 5.1). The 44 percent of no answer predictions in the Baseline model compared to 45 percent in the other models corresponds to roughly 40 less predicted blank answers in the Baseline Model compared to LSTM models. Though this is a stretch, perhaps overtraining of our models caused us to predict blank answers more frequently, as during training it became hyper aware of questions that had no answer to them and thus predicted a blank answer for any question in the dev set that resembled a blank answer question in the training set, even when the answer was not blank.

6.2 Sequence Length v Incorrect Predictions

We also wanted to investigate whether sequence length had any effect on incorrect predictions, and whether the patterns in sequence lengths between incorrect and correct predictions changed between models.



BERT Baseline Histogram - Sequence Length v Incorrect Predictions



BERT Baseline Histogram - Sequence Length v Incorrect Predictions

One thing we found intriguing in these histograms was the fact that our bidirectional LSTM had a much higher proportion of correct predictions with sequence lengths of around 80-100 than the baseline BERT model, along with larger amounts of correct answers for sequence lengths even less than 80 words. This leads us to believe that our LSTM model is less effective at predicting correct answers for longer sequence lengths. This is supported in literature [11], and is something we should have been aware of beforehand.

7 Conclusion

Bidirectional LSTMS and Transformers that incorporate multihead attention and feedforward networks have experienced wild success in industry and tackling specific NLP problems. We thought that incorporating a bidirectional LSTM at the end of the attention layers in BERT would have much stronger effectiveness than we saw in our implementations. We saw worse performance and scores of roughly half a point lower than the baseline. Our team still believes in this strategy here, but perhaps there may be further optimizations in our hyperparameters or architecture. We believe the theory and merging of the two strategies do hold some value.

We believe that a large part of the lack of improvement in our model can be explained by over-fitting. We hypothesize that our model was tuned too close to the training set to the point where random fluctuations in the training data led to incorrect predictions in the dev and test sets. Another source of error in our implementations could be the fact that LSTMs struggle in classification questions with small numbers of outputs associated with longer sequence lengths.

If we were given more time with our implementation, further ideas we'd like to explore are: changing the placement of the LSTM, attempting to mitigate LSTM weaknesses in the context of question answering, further tuning the hyperparameters of the BERT baseline model and added LSTM layer, and increasing the number of epochs for models we attempt to add to the BERT baseline.

References

[1] "SQuAD2.0." SQuAD - the Stanford Question Answering Dataset, rajpurkar.github.io/SQuAD-explorer/

[2] Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).

[3] Peters, Matthew E., et al. "Deep contextualized word representations." arXiv preprint arXiv:1802.05365 (2018).

[4] Wang, Di, and Eric Nyberg. "A long short-term memory model for answer sentence selection in question answering." Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers). Vol. 2. 2015.

[5] Tan, Ming, et al. "LSTM-based deep learning models for non-factoid answer selection." arXiv preprint arXiv:1511.04108 (2015).

[6] Yang, Zichao, et al. "Stacked attention networks for image question answering." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[7] Seo, Minjoon, et al. "Bidirectional attention flow for machine comprehension." arXiv preprint arXiv:1611.01603 (2016).

[8] https://github.com/chrischute/squad

[9] https://github.com/huggingface/pytorch-pretrained-BERT

[10] Vaswani, Ashish, et al. "Attention is all you need." Advances in Neural Information Processing Systems. 2017.

[11] “The Fall of RNN / LSTM.” Towards Data Science, Towards Data Science, 13 Apr. 2018, towardsdatascience.com/the-fall-of-rnn-lstm-2d1594c74ce0.