SQuAD Challenge using BiLSTM and Bidirectional Attention Flow

Mojtaba Sharifzadeh[*]
Stanford University
Stanford, CA 94305
mojtabas@stanford.edu

Abstract

Stanford Question Answering Dataset [1] is a rich dataset with 100k short questions and their corresponding answers tagged in a paragraph. To be more precise, the dataset contains pairs of questions and paragraphs, answers are tagged in the paragraph by start and end positions. Our goal is to train a neural network on this train/dev dataset, optimize the model architecture and the hyper-parameters, and finally evaluate its performance on the hidden test dataset. Given the time limits, we explore some known neural network models in the literature. The presented model achieved F1 score of 70.8% and EM score of 60.6% on test set that is significant improvement against the base model. We finally give a few of our ideas as a good path to be explored in the future.

1 Introduction

Suppose it is 2050 and you are using future internet and search for who is Donald Trump. The future Google, let's call it Foogle, opens a Fixipedia and shows you one sentence from Fixipedia. "Donald Trump was a president of United States between 2017 and ...". Isn't that great!

This is a problem where the machine is given a query about a given context and is required to predict the answer.This task that is helpful and we work on special case of it where we do not need to make up the answer from the context, instead we assume the answer can be inferred from short part of the context. The training and testing dataset comes from SQuAD [1].

SQuAD consists of 100K question-answer pairs, along with a context paragraph for each pair. There is also a public leader board [2] available. The state of the art is already very competitive, as there are many methods that are passing human level performance.

We train a variety of models such as biGRU, BiLSTM with two different attention layers. One is a basic concatenation attention layer in the base model and the second is BiDAF attention layer intorduced in [3]. The discussion on these layers will follow in later sections.

2 Technical Details

2.1 Data Analysis

Before looking at the model or anything, it is good to investigate the dataset look. It can help us to make correct choices when it is needed. We looked at the data, we have around $100k$ samples. Most of the questions/contexts length are less than 20/300. It might be a good idea to filter our all others from training to have a more consistent batch of the data to train. We also observe that the answers

[*]For any questions about this project, alternative contact info: *msharifzadeh@ieee.org*, collaborator: Ashkan Jafarpour, *ashkan@google.com*

are very short, ≤ 7 most of the time. It is good to filter out the outliers, keeping only 99% of the data and ignoring the 1% outliers during the training time. These 1% may only bring headache.

2.2 Problem Definition

Given context with length N, $c = \{c_1, c_2, ..., c_N\}$ and question with length M, $q = \{q_1, q_2, ..., q_M\}$, the model needs to find start and end location of the answer to q in c, i.e., a function of $(c, q) \rightarrow \{s, e\}$ where s and e are pair of scalar indices pointing to the start and end location.

2.3 Evaluation metrics

We evaluate performance of the model based on Exact Match, EM, and F1. EM is a binary value, one if answer matches exactly, zero otherwise. F1 is a harmonic mean of precision and recall between model and true answer. We might have multiple acceptable answers for each question and we will take best of metric values for F1 and EM in those cases.

2.4 Model Size

The model size has a great effect on the training time. It also has effect on over-fitting or under-fitting of the model. Beside having a good architecture, it is important to have near-correct size in each of the layers to have a good performance. For lack of training time, we sacrificed the model size for our final model with BiDAF architecture.

2.5 Overfitting

There has been overfitting in the base model and the dev performance was decreasing with more iterations. This is not acceptable and we should optimize our layer sizes or using the dropout trick to overcome to this issue. Given the lack of time, these are postponed to future improvements.

3 Model

3.1 Model Description

Base model

- **Embeding:** Pretrained GloVe embedding. It outputs

$$(x_1, \ldots, x_N), (y_1, \ldots, y_M)$$

  for context and question respectively.

- **Context and question input layer:** 1-layer bidirectional GRU input layer for each. it outputs

$$\{\overrightarrow{c}_1, \overleftarrow{c}_1, ..., \overrightarrow{c}_N, \overleftarrow{c}_N\}, \{\overrightarrow{q}_1, \overleftarrow{q}_1, ..., \overrightarrow{q}_M, \overleftarrow{q}_M\}$$

  each c_i and q_i comes from concatenation in $2h$ dimension.

- **Attention layer:** Simple attention layer by using dot-product attention and putting the query inside each context. At the end we have context hidden states c_i attended with weighted sum of query hidden states, a_i. context hidden layer with query attended inside the context together are b_i which is $4h$ dimensional vector.

- **Output layer:** $\{b_1, \ldots, b_N\}$ are fed through a fully connected layer followed by a ReLU non-linearity: and we get $\{b'_1, \ldots, b'_N\}$

- **Output layer** Applying same weight and bias vectors to each of the b'_i and then taking softmax over them. We have two separate output layers for start and end locations.

- **Loss:** Sum of the cross entropy loss for each of the start and end locations.

- **Predication for evaluation:** Taking arg max over both start and end and then calculating EM/F1 for performance measures on dev dataset.

Performance of base model
In order to have a base for our comparisons we simulated the performance of the base model in phase 1 milestone of the project. It achieved performance of 28% and 39% for EM and F1 respectively.

BiLSTM/BiDAF Architecture
Figure 1 shows the architecture in [3] which includes BiLSTM encoders at input with both word and character embedded layers, bidirectional attention flow layer, BiDAF, for both query2context and context2query and a final modeling layer that includes 2-layer BiLSTM and output layer, that is softmax. Due to complexity of this model, we implemented only word embedding and BiGRU instead of BiLSTM at input layer. We experimented two Basic-Attention and BiDAF layers followed by 2-layer BiLSTMs and softmax output layer in this project.

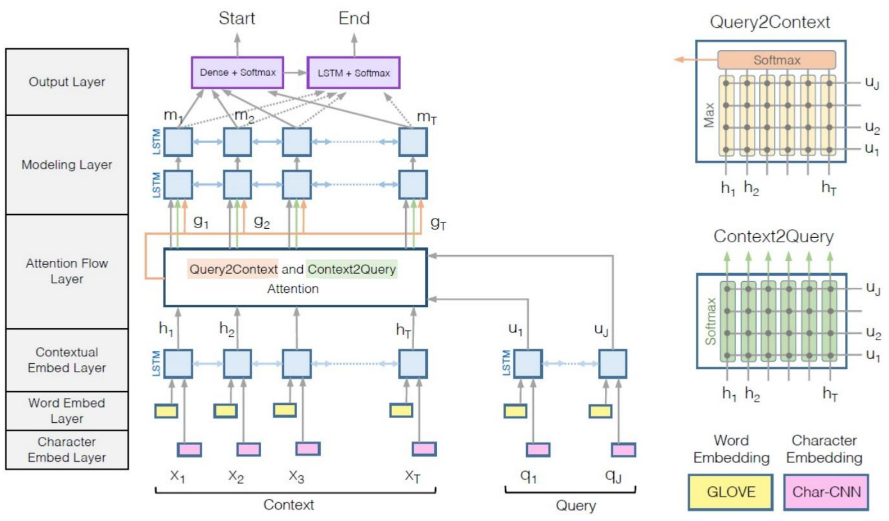

Figure 1: BiDAF model Architecture used in [3]

4 Results and Analysis

The 2-layer BiLSTM decoder was added to the base model. Figure 2 and 3 show its F1 and EM performance against the baseline on train and dev sets.

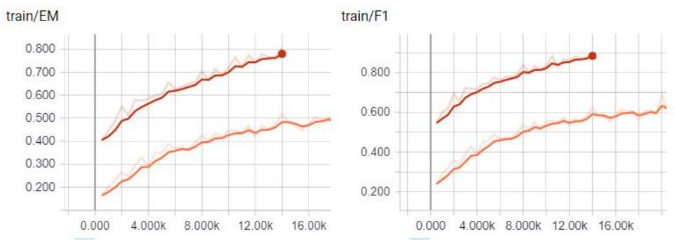

Figure 2: Performance of BiLSTM (red) vs. Base model (orange) on train set

Train F1/EM gets better over more iterations as expected but as seen in dev results, overfitting to dev set occurs above $10k$ iterations and best F1 score of 66% is obtained which is significant improvement over baseline F1 score of 39%.

Next, more complex BiDAF attention layers was added to the model that has larger output size $8h$ in dimension compared with $2h$ in the original Basic-Attention layer. Therefore, the default hyper parameter settings of batch-size$=100$, learning-rate$=0.001$, hidden-size$=200$, and embedding-size$=100$ resulted in very large model size giving memory errors on 1 GPU. To maintain a minimum

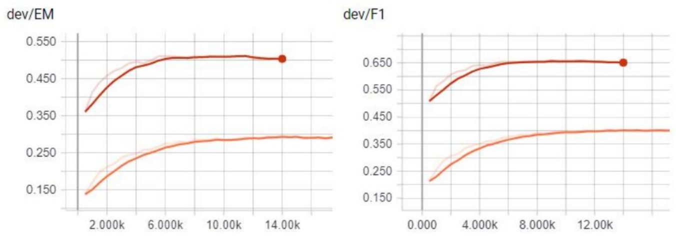

Figure 3: Performance of BiLSTM (red) vs. Base model (orange) on dev set

batch size of 50 we set the learning-rate$= 0.002$ and compromised embedding-size$= 50$ and hidden-size$= 50$. There is another solution by ignoring a small amount of contexts that are large in size > 300 to eliminate the memory error or run the code on multiple GPUs.

Figure 4 shows the performance of the final model with added BiDAF attention layer. Although, we would expect better performance after adding BiDAF layer the results shown in the figure reaches lower performance after running the training for one day. However, while we observed overfitting for initial BiLSTM model on the right plot around $10k$ iterations, the dev-loss for BiLSTM-BiDAF model is still improving and should surpass the BiLSTM model after running the training for a few days. Unfortunately, we did not have enough time to report the final result by the submission of this report.

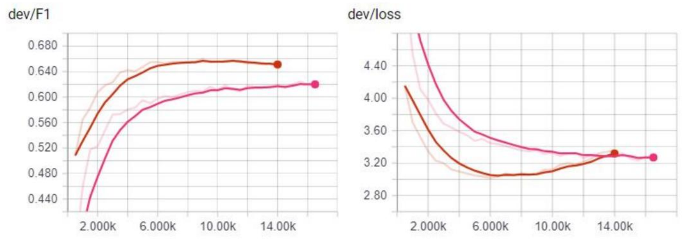

Figure 4: Performance of BiLSTM (red) vs. BiLSTM-BiDAF (purple) on dev set

In order to not compromise hidden and embedding size and at least keeping them above 100, for better training accuracy, reducing the batch size extremely to 25/12 and increasing the learning rate proportionally, was also tried. Due to the very small batch-size updates are noisy and improvement did not happen during training as shown in Figure 5. Therefore, we need to keep the batch-size and control the maximum context length away from 600.

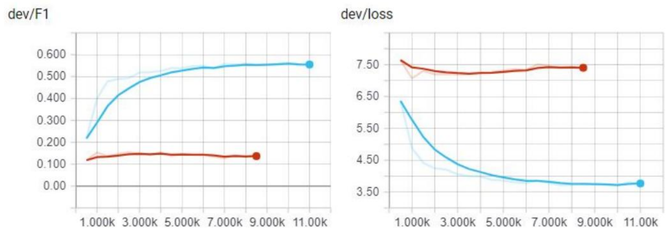

Figure 5: Performance of BiLSTM-BiDAF model with batch-size $= 25$ (blue) vs. batch-size $= 12$ (red)

Table 1: Performance summary with different methods

Model	**F1 Dev-set**	**EM Dev-set**	**F1 Test-set**	**EM Test-set**
Base Model	39	28	–	–
BiLSTM	66	51	70.8	60.6
BiLSTM-BiDAF (1-day training)	62	47	–	–
BiDAF [3]	77.3	67.7	77.3	68

5 Future Ideas

5.1 Output Layer Improvement

Problem
The loss for the output s based on the cross entropy loss for the probabilities that the model assign to start and end locations. But the evaluation metric is based on F1 or EM scores. Sometime the answer to the question is not in a uniques location. For example:

Question: what is 224n
Context: deep learning course deep learning course deep learning course
Answer: deep learning course

Clearly we have repeated answer and the model should not be penalized for finding one of the repeated answers during the training time. Using a single start and end location for the training is an issue here and there is an inconsistently between minimizing the start and end locations loss and the final EM/F1 scores.

Another problem independency of start and end locations. In fact, end can be smaller than start. We are going to address these issues in the next subsection without implementation because of lack of time.

Possible Solution
There are only $K = \frac{N(N+1)}{2}$ possible solutions while a model with independent start and end can output N^2 solutions. The first step is to change the output layer so that it outputs one of these K logical solutions instead of 2 independent start and end locations. In addition, since start and end locations in the solution are close by usually, $\approx< 7$, we only need to output one of the $\approx 7 \times N$ solutions. This way we do not explode the output layer size and we can solve the problem of having wrong ordered start and end locations

For the cross entropy loss layer with repeated good start and end solutions, it might be a better idea to have the true output y to be distributed (normalized) over all the possible start and ends where it is exact match. This way we can solve the repeated solution in the context and have a more consistent loss layer.

5.2 Optimizing Hidden Layer Sizes

Currently there is a single parameter h the captures size of the output layers. We believe the layers in different depth should have possibly different sizes. In each layer, the amount of information to flow is different, therefore, different layer size might be needed but currently they are all multiple of h.

6 Conclusion and Results Summary

Our final results were obtained from using BiGRU at input layer and BiDAF attention layer followed by 2-layer BiLSTM and softmax at output. We achieved the performance of 60.6% for EM and 70.8% for F1 on test set. Compared to the base model, this is a significant improvement. We believe the reason behind this improvement compared to simple linear down projection followed by ReLU in base model is utilizing two layers of bidirectional LSTMs.

We implemented and used BiDAF[2] as a new attention layer obligating us to reduce the batch and hidden layer size to 50 to compensate the speed and memory issues. The new model gave descent performance similar to using BiLSTM with base attention but taking more time to train. As mentioned, adding BiDAF attention layer, compared to basic attention should improve the performance, however, due to lack of time after one day of training on one GPU, we are approaching to the performance of BiLSTM (F1$= 66\%$) at F1 score of 62%. Future work can be done on improving attention layers to allow a better query and context interactions.

Acknowledgments

The author would like to thank the TAs and Dr. Richard Socher for insightful discussions and timely feedback about improving the models. The author also appreciates fruitful discussions with Dr. Ashkan Jafarpour at Google about the project.

References

[1] Rajpurkar P, Zhang J, Lopyrev K, et al. Squad: 100,000+ questions for machine comprehension of text[J]. *arXiv preprint* arXiv:1606.05250, 2016.

[2] `https://rajpurkar.github.io/SQuAD-explorer`

[3] Seo, Minjoon, et al. Bidirectional Attention Flow for Machine Comprehension. *arXiv preprint* arXiv:1611.01603 (2016).