Improving SQuAD Baseline Using BiDAF Refinements and Experimenting with Semi-Supervised Learning

Allison Koenecke
ICME
koenecke@stanford.edu

Varun Vasudevan
ICME
devan@stanford.edu

Abstract

In this project, we improve the SQuAD baseline model by reimplementing the attention layer and the modeling layer from the Bidirectional Attention Flow model (BiDAF). We also tune the hyperparameters and make finer improvements by studying the dataset and through experimentation. Eventually, our best model is able to get a F1 score of 75.754 and EM score of 65.719 on the test leaderboard. In addition, we also observe the performance of our model(s) in the semi-supervised learning context.

1 Introduction

Natural Language Processing researchers have long studied the problem of answering a question, given a paragraph of context. As a way to improve the ability of machines to learn reading comprehension, the Stanford Question Answering Dataset (SQuAD) was created. SQuAD includes Wikipedia articles as context, with over 100,000 corresponding question and answer pairs relevant to the articles. The answer to each question is always a span in the text. Using this dataset, researchers are able to train models to predict answers to questions as a first step to reading comprehension. To-date, state-of-the-art strategies have nearly outperformed humans [1].

Our goal is two-fold. Firstly, we explore ways to improve the SQuAD baseline model via different incremental additions, including a partial re-implementation of the Bi-Directional Attention Flow (BiDAF) model. Secondly, we observe the ability of our models to perform with drastically lower shares of training data; this allows us to form some intuition on semi-supervised learning with respect to SQuAD. Overall, we are able to attain an F1 score of 75.754 and EM score of 65.719 on the test leaderboard using our model. Further, our semi-supervised methods allow us to lessen overfitting when training sizes are small.

2 Background

Extensive work has been done over the past several years to implement models to answer SQuAD test set questions with accuracy rivaling that of humans. Several models in particular have been highly successful. Firstly, the Bi-Directional Attention Flow (BiDAF) model [2] uses a specialized attention mechanism to summarize small bits of context in a fixed-size vector, both forwards and backwards. The model uses a multi-stage hierarchical process wherein context is represented both with token- and character-level Convolutional Neural Networks (CNNs), additional LSTM layers are inserted between attention and output, and the attention flow mechanism avoids early summarization. Secondly, the Dynamic Coattention Network uses a different attention mechanism, wherein second-level attention computation is performed [3]. We explore the BiDAF attention mechanism in our work.

Stanford CS224n, Winter 2018.

Meanwhile, in the context of semi-supervised learning, generative techniques have been used to improve SQuAD results in instances where some subset of training data is unlabeled [4]. Specifically, in order to generate questions based on the given answers, the authors propose a generative model (analogous to a GAN) that learns the conditional probability of generating a question given the context and answer, using a sequence-to-sequence model and a GRU. We take inspiration from the baseline model of creating a window around the answers to simulate the question, and then combining these generated data with the existing labeled training data.

3 Our Approach – Architecture

The baseline model has three components: a RNN encoder layer, an attention layer, and an output layer. Both context and question that are fed as input to the encoder layer are fixed, pre-trained, d-dimensional GloVe embeddings. The embedding size d can take one of the four values – 50, 100, 200, 300. Let $\mathbf{x}_1, \ldots \mathbf{x}_N \in \mathbb{R}^d$ and $\mathbf{y}_1, \ldots \mathbf{y}_M \in \mathbb{R}^d$ denote the context and question embeddings. The encoder contains a 1-layer bidirectional GRU that operates on the embeddings to form the context states $\mathbf{c}_i \in \mathbb{R}^{2h}$, for all i from 1 through N, and question hidden states $\mathbf{q}_j \in \mathbb{R}^{2h}$, for all j from 1 through M. Here h denotes the number of hidden states. The attention layer combines the context and question representations to form the blended representations. The output layer applies a fully connected layer and then two separate softmax layers, one each for the start and end location of the answer span [5].

Our neural network includes the following improvements from the baseline model: a restriction on span length, the use of LSTMs in the encoder, the use of Bi-Directional attention, and an additional modeling layer between the attention and the output layer. The modeling layer consists of two bidirectional LSTMs with an output size of h in each direction as described in the Bidirectional Attention Flow model [2]. The modeling layer captures the interaction among the context words conditioned on the query.

The baseline attention layer is unidirectional – context hidden states attending to question hidden states. Whereas in the BiDAF attention is computed in two directions – from context-to-query and query-to-context. Context-to-query (C2Q) attention signifies which query words are most relevant to each context word. Query-to-context (Q2C) attention signifies which context words have the closest similarity to one of the query words and are hence critical for answering the query. Both the attentions are computed from a single similarity matrix $\boldsymbol{S} \in \mathbb{R}^{N \times M}$ which is calculated using the context and question hidden states as

$$\boldsymbol{S}_{ij} = \begin{bmatrix} \mathbf{c}_i^T & \mathbf{q}_j^T & (\mathbf{c}_i \circ \mathbf{q}_j)^T \end{bmatrix} \mathbf{w}_s \in \mathbb{R},$$

where $\mathbf{c}_i \circ \mathbf{q}_j$ is an elementwise product and $\mathbf{w}_S \in \mathbb{R}^{6h}$ is a weight vector. The C2Q attention outputs $\mathbf{a}_i$ is computed as

$$\alpha^i = softmax(\boldsymbol{S}[i, :]) \in \mathbb{R}^M,$$
$$\mathbf{a}_i = \sum_{j=1}^{M} \alpha_j^i \mathbf{q}_j \in \mathbb{R}^{2h},$$

and the Q2C attention output $\mathbf{c}'$ is computed as follows

$$\mathbf{n}_i = \max_j \boldsymbol{S}_{ij} \in \mathbb{R}$$
$$\beta = softmax(\mathbf{n}) \in \mathbb{R}^N$$
$$\mathbf{c}' = \sum_{i=1}^{N} \beta_i \mathbf{c}_i \in \mathbb{R}^{2h},$$

where i is from 1 to N and j is from 1 to M. The blended representation $\mathbf{b}_i$ for each context location is then calculated as

$$\mathbf{b}_i = \begin{bmatrix} \mathbf{c}_i^T & \mathbf{a}_i^T & (\mathbf{c}_i \circ \mathbf{a}_i)^T & (\mathbf{c}_i \circ \mathbf{c}')^T \end{bmatrix}^T \in \mathbb{R}^{8h}.$$

Further, our semi-supervised results show that we are able to improve upon the problem of having little training data by artificially generating missing data. In unlabeled data, if we are given answer

$a = (c_j, c_{j+1}, ..., c_k)$ from context $c = (c_1, ..., c_n)$, we define window size W and generate the question (which is training data that we do not have) as follows:

$$q = (c_{j-W}, c_{j-W+1}, ..., c_{j-1}, c_{k+1}, c_{k+2}, ..., c_{k+W}) \tag{1}$$

That is, we artificially assume that the question is the concatenation of the W tokens before the first token of the answer, and the W tokens after the last token of the answer. We can then combine the unlabeled data with labeled data as our improved training set.

4 Experiments

In the following subsections, we first describe the dataset and outcome measures used for our study. Then, we describe the models and results for our improvements to the SQuAD baseline. Next, we describe the models and results for our semi-supervised exploration.

4.1 Datasets and Evaluation Metrics Used

We use the SQuAD data [6] in its original form, wherein each sentence is tokenized using NLTK [7], and word embeddings are done using GloVe [8]. From these data, we find that the training data contexts tend to be slightly right-skewed, where the majority of contexts are between 100 and 200 tokens long. Since nearly all contexts have length fewer than 400 tokens, we set this as the maximum vector length of contexts to decrease training time as our baseline. Further, training questions tend to be distributed similarly between lengths of 0 and 30 tokens (with spikes at lengths of 9 and 12 tokens), so we set the vector length for questions to be 30. Lastly, the training answers are heavily right-skewed wherein the vast majority of answers are 2 tokens long, and nearly all answers are fewer than 20 tokens long.

The primary evaluation metrics we use are F1 and EM. However, we also observe the loss in each epoch over time for general trends, and also plot the means and standard deviations of the outputs of each layer to ensure that they are within reasonable bounds and in a realistic ordering relative to the different versions of models run.

Further, we use qualitative evaluation by manually observing training examples to determine whether certain issues in the baseline model were ameliorated.

4.2 SQuAD Baseline Improvement: Model Configurations

1. For all models, we keep the learning rate at 0.001, and the context length is restricted to 400.
2. We first improved upon the baseline model by restricting the end position of the prediction. We tested two versions: firstly, we restrict the end position probability distribution to occur strictly after the start position with highest probability. Secondly, we restrict the end position to be within 18 tokens of the start position. Specifically, we check to see whether the start or end probability maximum for each batch is highest, and either restrict the end position to be within 18 tokens after the start position, or in the other case, restrict the start position to be within 18 tokens before the end position. The number 18 was chosen based on a histogram analysis of the distribution of answer lengths (see Section 4.1, wherein there were nearly no instances of answer lengths greater than 18. The latter constraint was found to be effective, increasing development set F1 by up to 6 percentage points. Using these truncations, the training time decreases.
3. Next, we experiment by using LSTMs as opposed to GRUs in the RNN encoder layer. As expected, the training time takes longer with LSTMs, but our accuracy increases as well, due to the ability for longer historical memory of context.
4. Then, we replace the baseline attention module with Bi-Directional attention. We refer to the model that includes this and all of the aforementioned changes as Model 1.
5. On top of Model 1, we then add a modeling layer between the Bi-directional attention and the output layer. We also change the batch size from 100 to 50, and refer to this as Model 2.

Table 1: Score Increases by Incremental Improvement to Baseline SQuAD Model (15,000 iterations)

Improvement	Train EM	Train F1	Dev EM	Dev F1	Dev Loss
Baseline (Orange)	0.47	0.57	0.29	0.40	4.7
End after Start Position	0.49	0.62	0.32	0.44	4.7
End within 18 tokens after Start Position	0.52	0.64	0.33	0.46	4.7
LSTM instead of GRU	0.58	0.69	0.35	0.48	4.6
Model 1 – BiDAF Attention (Gray)	0.65	0.78	0.38	0.52	4.3
Model 2 – Additional BiLSTM Layer (Red)	0.76	0.87	0.54	0.69	3.1

6. On top of Model 2, we experiment with changing the batch size to 75. While this lessens runtime to some extent, the results are minimally different from Model 2, so we exclude this experiment from resulting tables and figures.

4.3 SQuAD Baseline Improvement: Results

4.3.1 Quantitative Results

Quantitatively, we point to Table 1 as evidence that while Model 1 yields fair improvement over the baseline (at the expense of some overfitting, as seen by the widening gap between train and dev EM and F1 scores), the additional BiLSTM layer added between the attention layer and output layer increases our performance the most. Specifically, this change in Model 2 increases our EM and F1 scores by more than all of the incremental changes done in Model 1. More importantly, this fixes much of the overfitting from Model 1; the difference between train and dev scores is lessened by nearly 10 percentage points.

The progress made among incremental changes is shown in Table 1, which includes in parentheticals the colors corresponding to selected plots provided in Figures 1 and 2 below.

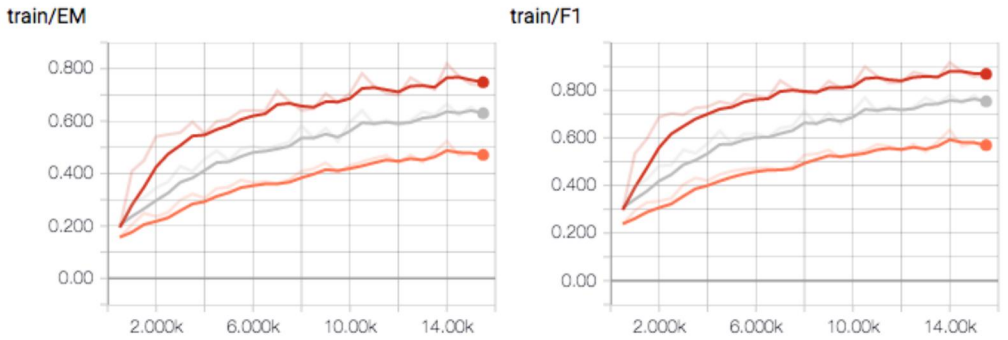

Figure 1: Train set EM and F1 scores (plotted against number of iterations) on the SQuAD baseline, Model 1, and Model 2. Color legend in Table 1.

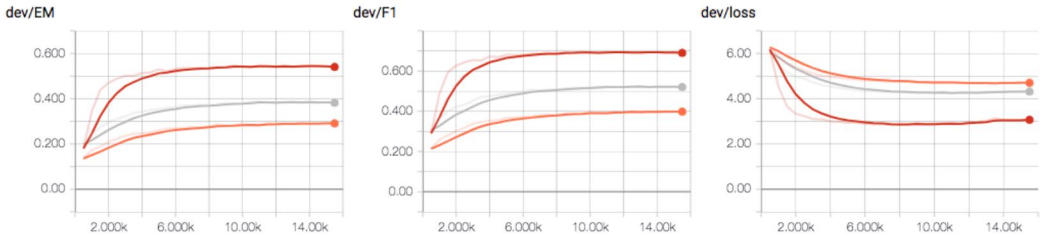

Figure 2: Development set EM, F1 scores and loss (plotted against number of iterations) on the SQuAD baseline, Model 1, and Model 2. Color legend in Table 1.

4.3.2 Qualitative Results

Qualitatively, we point to several examples wherein we are able to pinpoint where a question answered incorrectly by the baseline is improved upon by our model.

Example 1 **Question**. "a block of west end avenue that houses an abc news building was renamed for what abc anchor ?"
True answer. Peter Jennings
Baseline model. Yielded a blank answer because the start position was found to occur after the end position.
Observation. By the time we implement the 18-token rule of end position occurring after start position, our answer is augmented to "Peter Jennings Way".

Example 2 **Context excerpt**. "...the prime number theorem , proven at the end of the 19th century , which says that the probability that a given , randomly chosen number n is prime is inversely proportional to its number of digits , or to the logarithm of n ."
Question. "what theorem states that the probability that a number n is prime is inversely proportional to its logarithm ?"
Baseline model. The predicted answer was "distribution of primes , that is to say , the statistical behaviour of primes in the large , can be modelled . the first result in that direction is the prime number theorem".
Observation. The true answer (simply "the prime number theorem") is achieved when switching from GRUs to LSTMs (along with truncating the answer width to 18 tokens long). This makes sense because the true answer is significantly shorter, and also because LSTMs are better at remembering longer sequences than GRUs due to its additional gate. Because the word "logarithm" is so far away from "theorem" in the context, this historical memory is necessary.

Example 3 **Question**. "which architect , famous for designing london 's st. paul cathedral , is represented in the riba collection ?"
Baseline model. Yielded a long list of architects, rather than a single name.
Observation. Switching to the BiDAF attention mechanism in Model 1 allows us to come closer to the answer format of a single name (though the name itself is incorrect).

4.3.3 SQuAD Semi-Supervised Learning: Model Configurations

We first split the training data into 6 datasets: one pair containing a random split of 30% and 70% of the rows, one pair containing a random split of 20% and 80% of the rows, and one pair containing a random split of 10% and 90% of the rows.

The first set of comparisons we run are on Model 1 (as described above), on the 30%, 20%, and 10% training datasets as compared to the full original SQuAD dataset.

The next set of comparisons we run are among all of the above dataset subsets, against the data subsets concatenated with the unlabeled data. That is, we operate under the assumption that, e.g., the 30% subset of data is fully labeled, but the separate 70% dataset subset is unlabeled, i.e. containing the relevant context and answer pairs, but missing the question. For this purpose, we construct questions for each of the rows of the 70% dataset according to Equation 1. We test results using window sizes W of 5 as this seems reasonable based on the lengths of questions and answers as determined in Section 4.1.

4.3.4 SQuAD Semi-Supervised Learning: Results

We find that our simple model to generate questions based on answers yields minor gains to accuracy, as shown in Table 2. While results are still significantly worse than what could be achieved with full training data, semi-supervised training still appears to be a viable option on SQuAD data. In particular, we note that the difference between train and dev EM and F1 scores is vast with only subsetted training data, signifying vast overfitting (which could be ameliorated with the inclusion of more data, obviously, or a regularization term). However, with the semi-supervised datasets, the train EM and F1 scores are much more reasonable, signaling that though it takes longer to reach comparable dev set scores, the loss is progressing more steadily with semi-supervised learning, and there is less overfitting because there is more data (see comparison of Figures 3 and 4). In particular,

Table 2: Low Sample Size vs. Semi-Supervised Training on Model 1 After 15,000 iterations

Labeled Share	Unlabeled Share	Train EM	Train F1	Dev EM	Dev F1	Dev Loss
100% (Gray)	No	0.65	0.78	0.38	0.52	4.3
30% (Orange)	No	0.96	0.98	0.27	0.40	7.6
20% (Dark Blue)	No	0.98	0.99	0.23	0.35	10.2
10% (Red)	No	0.99	0.99	0.18	0.30	14.7
30% (Light blue)	Yes	0.58	0.70	0.29	0.41	5.5
20% (Pink)	Yes	0.56	0.68	0.25	0.36	6.1
10% (Green)	Yes	0.82	0.88	0.20	0.31	6.9

we see that the semi-supervised results are much more similar in shape to our original baseline of Model 1.

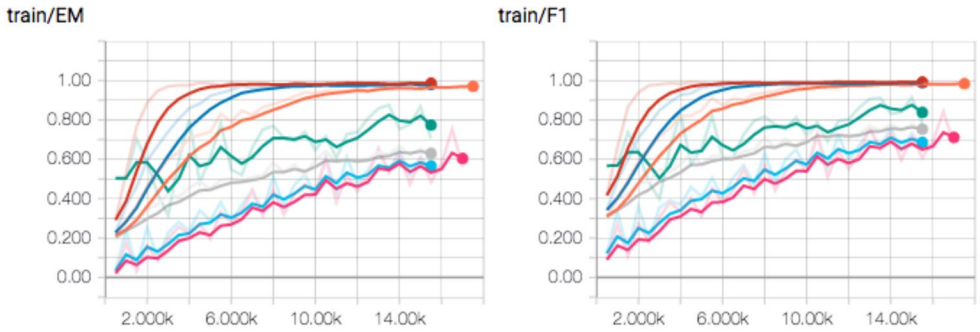

Figure 3: Training set EM and F1 scores (plotted against number of iterations) on Model 1, Model 1 training sample subsets, and concatenation of Model 1 subsets with window-generated questions. Color legend in Table 2.

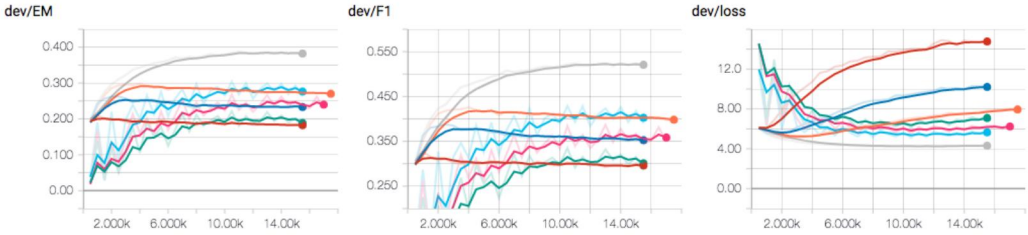

Figure 4: Development set EM, F1 scores and loss (plotted against number of iterations) on Model 1, Model 1 training sample subsets, and concatenation of Model 1 subsets with window-generated questions. Color legend in Table 2.

As a qualitative comparison, the 30% model excluding the unlabeled share incorrectly answers the question: "in which etude of neumes rythmiques do the primes 41 , 43 , 47 and 53 appear in ?" The corresponding context includes: "the primes 41 , 43 , 47 and 53 appear in the third etude , neumes rythmiques ." While the 30% model predicts "43", the semi-supervised model (30% labeled, including the other 70% with a generated question for each answer) correctly predicts "the third etude". This could likely be due to the size-5 window that we used to train the 70% of the data, wherein the generated question would include two of the numbers appearing in the actual question, and the words "neumes rythmiques," perhaps yielding more accuracy surrounding the location of the answer.

5 Conclusion

We find that implementing BiDAF attention layer and the modeling layer consisting of two bidirectional LSTMs give the most value-add to the baseline model, while simple one-line tweaks together can provide modest gains in EM and F1 accuracy as well. Further, we find that semi-supervised training is a viable method to reduce overfitting by using a basic window-based model for generating questions.

Future work on our SQuAD project could include using a voting panel of multiple models. Running many models on the same data simultaneously, and then taking the most-often-generated answer among them, seems to increase accuracy to a great extent in previous work. On semi-supervised training, it would be interesting to see a more rigorous probabilistic model to generate windows. One way to do this could be to take into account the span differences between existing questions and existing answers, and using a separate model to forecast optimal window size.

References

[1] "The stanford question answering dataset." https://rajpurkar.github.io/SQuAD-explorer/. Accessed: 2018-03-14.

[2] M. J. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi, "Bidirectional attention flow for machine comprehension," *CoRR*, vol. abs/1611.01603, 2016.

[3] C. Xiong, V. Zhong, and R. Socher, "Dynamic coattention networks for question answering," *CoRR*, vol. abs/1611.01604, 2016.

[4] Z. Yang, J. Hu, R. Salakhutdinov, and W. W. Cohen, "Semi-supervised QA with generative domain-adaptive nets," *CoRR*, vol. abs/1702.02206, 2017.

[5] C. T. Staff, "Cs224n default project handout," 2018.

[6] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100, 000+ questions for machine comprehension of text," *CoRR*, vol. abs/1606.05250, 2016.

[7] E. Loper and S. Bird, "NLTK: the natural language toolkit," *CoRR*, vol. cs.CL/0205028, 2002.

[8] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014.