Question-Answering System for SQuAD

Weiquan Mao
Department of Electrical Engineering
Stanford University
mwq@stanford.edu

Abstract

In this paper, we produce a question answering system that works well on SQuAD. BiDAF model is used as our baseline model, which pushes Dev F1 score to 60 and Dev EM score to 57. And SRU architecture is applied to accelerate the training process as well as improve the performance. Then we applied a language representation model called BERT (Bidirectional Encoder Representations from Transformers) on SQuAD dataset. With one additional output layer, we experiment with different hyper-parameters in fine-tuning pre-trained BERT representations. Aiming to improve upon a standard BERT implementation, we have tried adding additional layers after BERT, applying L1 regularization. After ensembling all models, we now pushes SQuAD 2.0 question answering Dev F1 score to 79.944, Dev EM score to 73.643, Test F1 score to 78.841 and Test EM score to 76.010.

1 Introduction

Question-Answering System is one of the most popular natural language process tasks due to the creation of large question answer datasets. This can be used in many practical applications such as virtual assistants and automated customer service. The release of the Stanford Question Answering Dataset [3] has facilitated rapid progress in this field. The input to our model is a paragraph and a question about that paragraph. Our model uses BiDAF as baseline, Simple Recurrent Unit-BiDAF as a method to speed up the training, BERT as the core, L1 as regularization. The goal is to answer the question correctly - select the span of text or N/A if there is no answer in the paragraph. Another method to improve the performance of our model is ensemble, where we tried multiple models with different hyperparameters and mechanisms.

2 Related work

In the past few years, reading comprehension with neural networks has been studied thoroughly. Most of the high-performing models uses neural attention mechanism to combine the representations for the context and the question. Bi-Directional Attention Flow network[4] is one among them, which represents the context at different levels of granularity and uses a bi-directional attention flow mechanism to achieve a query-aware context representation without early summarization. Besides BiDAF, there are also other attention mechanism such as self-attention[5] and coattention[7]. However, since last year, Bidirectional Encoder Representations from Transformers [1] (BERT) has achieved state-of-the-art performance for eleven NLP tasks, like Question Answering[3] and Question Natural Language Inference[6].
3 Dataset and Features

We use SQuAD 2.0 as the reading comprehension data set. The paragraphs in SQuAD are from
Wikipedia. The questions and answers are using labeling from Amazon Mechanical Turk. There
are around 150k questions in total, and roughly half of the questions cannot be answered using the
provided paragraph. However, if the question is answerable, the answer is a chunk of text taken
directly from the paragraph. This means that SQuAD systems don’t have to generate the answer text
– they just have to select the span of text in the paragraph that answers the question.

The SQuAD dataset has been split into three sets:

- Train set with 129,941 examples, all taken from the official SQuAD 2.0 training set.
- Dev set with 6078 examples, randomly selected from the official dev set. For the milestone,
  we are only evaluating on the dev set, and has not tested anything on the test set yet.
- Test set with 5921 examples, the remaining examples from the official dev set along with
  some hand-labeled examples.

Here is an example of data:

- **Question**: What does not depend on the immune system’s ability to distinguish between the
  self and others?
- **Context**: Both innate and adaptive immunity depend on the ability of the immune system to
  distinguish between self and non-self molecules. In immunology, self molecules are those
  components of an organism’s body that can be distinguished from foreign substances by the
  immune system. Conversely, non-self molecules are those recognized as foreign molecules.
  One class of non-self molecules are called antigens (short for antibody generators) and are
  defined as substances that bind to specific immune receptors and elicit an immune response.
- **Answer**: N/A

4 Methods

4.1 Baseline: BiDAF

Our baseline model based on BiDAF [4]. It is composed of Embedding Layer, Encoder Layer,
Attention Layer, Modeling Layer and Output layer.

Specifically, the embedding layer performs an embedding lookup to convert the indices into word
embedding, which is done for both the context and the question. A Highway Network is also used to
refine the embedded representation. The encoder layer uses a bidirectional LSTM to allow the model
to incorporate temporal dependencies between timesteps of the embedding layer’s output. The main
idea of attention layer is that attention flows both ways - from the context to the question and from
the question to the context. The modeling layer is tasked with refining the sequence of vectors after
the attention layer. It integrates temporal information between context representations conditioned on
the question. The output layer is tasked with producing a vector of probabilities corresponding to each
position in the context.

Our loss function for the baseline model is the cross-entropy loss for the start and end locations. We
average across the batch and use Adadelta optimizer to minimize the loss.

4.2 SRU-BiDAF

The recurrent architectures like LSTM we used in the baseline use gating to control the information
flow to alleviate vanishing and exploding gradient problems. However, the computation of the
feed-forward network, especially the matrix multiplication is the most expensive operation in the
process. We applied a Simple Recurrent Unit(SRU)[2] architecture. The core idea is making the gate
computation dependent only on the current input of the recurrence. This leaves only the point-wise
multiplication computation as dependent on previous steps, which is relatively lightweight.
4.3 Bidirectional Encoder Representations from Transformers

BERT achieves state-of-the-art performance for eleven NLP tasks, like Question Answering and Question Natural Language Inference, through only fine-tuning the last layer. BERT has such noteworthy achievement because it learns a more powerful bi-directional representation than most of the previous approaches. BERT’s architecture is mainly multi-layer bidirectional Transformer encoder with bidirectional self-attention mechanism. The encoder of BERT is pre-trained with two tasks, “masked language model” (MLM) and Next Sentence Prediction. These two objectives help encoder to learn both left and right contextual of a word in the sentence and provides significant support for downstream tasks like question answering.

4.4 BERT-additional Layer

With just one additional output layer, the pre-trained BERT representations can create state-of-the-art models. It is natural to try a “deeper” neural network after the BERT instead of a single output layer. It is widely believed that deep models are able to extract better features than shallow models and hence, extra layers help in learning features. In the experiment, we have added one extra layer before the output layer and feed it to the ensembling experiment.

4.5 Ensembling

In modern Machine Learning, Ensembling Methods are extensively used to combine multiple learning algorithms, preferably from different model classes, into an aggregate model with better performance than any single model. Each of the BERT and BIDAF-based model we built make different predictions on probabilities of start and end positions, and therefore we use Ensembling Methods in hope of utilizing the information extracted from all models. Guided Random Search for Weighted Average Ensembling

Assume that we have a total of n models to ensemble, and for each inputting (Question, Paragraph) pair, model k outputs \( k_{\text{start}} \), \( k_{\text{end}} \), \( 0 \leq i \leq \text{len(Paragraph)} \) where \( k_{\text{start}} \) is the probability that the i-th position is the start position, and \( k_{\text{end}} \) is the probability that the i-th position is the end position. Following this notation, \( k_{ij} := P(\text{start_pos} = i, \text{end_pos} = j) = p_{\text{start}_i} \times p_{\text{end}_i} \) predicted by the k-th model \( p_{ij} := P(\text{start_pos} = i, \text{end_pos} = j) = \sum_k w_k \times p^k_{ij} \), \( w \in \mathbb{R}^n \) predicted by the weighted average ensembling model \( (\text{start_pos, end_pos}) = \text{argmax}_{(i,j)} p_{ij} \) So our goal is
now reduced to finding the best $w \in \mathbb{R}^n$. With this motivation, we develop a pipeline that learns the weights $w$, and make predictions on the prediction set. The details are described in Algorithm 1. In high level, our algorithm randomly assign weights to each model, with the only restriction that a better model should never be assigned a lower weight than a model not as good. With the weights learned, we re-do the predictions by taking the weighted average of the probabilities predicted by each model.

**Algorithm 1** Guided Random search + Weighted Average Ensembling

**Input:**
1. set of $k$ models: $M \in \mathbb{R}^k$
2. dev set: $\{(X_i, Y_i = (\text{start pos}, \text{end pos}))\}_{i=1}^{\text{n dev}}$
3. prediction set: $\{X_m\}_{m=1}^{\text{n pred}}$
4. max_num_iter

**Output:** Predictions on the prediction set

$\text{best_weight} = \text{Guided Random Search on Weight}(M, \text{max_num_iter}, \text{dev set})$

$\text{Return Make Predictions}(M, \text{best_weight}, \text{prediction set})$

5 Experiments/Results/Discussion

5.1 Evaluation Method

We mainly use two types of evaluation metrics, Exact Match and F1 score.

Exact Match (EM) is a binary measure (i.e., true/false) of whether the system output matches the ground truth answer exactly. In our evaluation, EM stands for the percentage of outputs that match exactly with the ground truth.

F1 is the harmonic mean of precision and recall, more specifically:

$$F_1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

$$\text{precision} = \frac{\text{truepositives}}{\text{truepositives} + \text{falsepositives}} \quad \text{; recall} = \frac{\text{truepositives}}{\text{truepositives} + \text{falsenegatives}}$$

For questions that do have answers, we take the maximum F1 and EM scores across the three human-provided answers for that question. And for those without answers, both the F1 and EM score are 1 if the model predicts no-answer, and 0 otherwise.

5.2 Baseline

First, we trained the baseline model and compared the loss, AvNA (Answer vs. No Answer), EM, and F1 (official SQuAD evaluation metrics) for both train and dev sets. Over 3 million iterations we find that:

- The train loss continues to improve throughout
- The dev loss begins to rise around 2M iterations (overfitting)
- The dev AvNA reaches about 68, the dev F1 reachess about 60 and the dev EM score reaches around 57.
- Although the dev NLL improves throughout the training period, the dev EM and F1 scores initially get worse at the start of training, before then improving.

5.3 BERT

1. **Fine-tuning:** BERT is the first fine-tuning based representation model that achieves state-of-the-art performance on a large suite of sentence-level and token-level tasks. For fine-tuning, most model hyperparameters are the same as in pre-training, with the exception of the batch
size, learning rate, and number of training epochs. Due to the issue of out of memory, when we change the batch size, we need to change the maximum sequence length accordingly. The dropout probability was always kept at 0.1. We visualize the loss curves of all BERT fine-tuning experiments below:

![Learning curve of all BERT fine-tuning experiments](image)

Figure 3: Learning curve of all BERT fine-tuning experiments

We can find that that with big learning rates in the scale of $e^{-4}$ ($5e^{-4}$ and $3e^{-4}$), the learning curves spike after around 5k iterations, and the losses fail to converge. Moreover, when learning rate = $5e^{-5}$ or $3e^{-5}$, the losses converge the fastest. It turns out that when max_seq_length = 245, and batch_size = 12, learning_rate = $3e^{-5}$ the performance is the best, whose Dev F1 score achieved 77.166.

2. **L1 regularization:** In order to experiment the effect of L1 regularization, we have fixed the maximum sequence length to 140, batch size to 24, learning rate to $3e^{-5}$ and epoch to 4. By changing the L1 regularization parameter from $1e^{-4}$, $1e^{-3}$ to $1e^{-2}$

<table>
<thead>
<tr>
<th>L1 regularization parameter</th>
<th>Dev F1</th>
<th>Dev EM</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda = 0$</td>
<td>74.679</td>
<td>71.915</td>
</tr>
<tr>
<td>$\lambda = 1e^{-4}$</td>
<td>75.705</td>
<td>73.001</td>
</tr>
<tr>
<td>$\lambda = 1e^{-3}$</td>
<td>76.666</td>
<td>73.824</td>
</tr>
<tr>
<td>$\lambda = 1e^{-2}$</td>
<td>76.76</td>
<td>73.955</td>
</tr>
</tbody>
</table>

As we can see from the table, when we applied L1 regularization and increasing the L1 regularization parameter, the performance becomes better.

5.4 Ensembling

We run the ensembling algorithm on all 26 models we have (BiDAF, SRU-BiDAF, fine-tuning BERT, BERT-additional layer, BERT-L1 regularization) and the performance of the best model is listed in Table 2.

<table>
<thead>
<tr>
<th>Ensembling Method</th>
<th>Dev F1</th>
<th>Dev EM</th>
<th>Test F1</th>
<th>Test EM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Guided Random Search for Weighted Average</td>
<td>79.944</td>
<td>77.081</td>
<td>78.841</td>
<td>76.010</td>
</tr>
</tbody>
</table>

6 Conclusion/Future Work

As we can see from SQuAD leaderboard, almost every leading submission uses BERT. In our report, a lot of methods we have tried are based on BERT, which can outperform even the best non pretrained contextual embeddings models. After training about 26 BiDAF-based and BERT-based models, and ensemble them with guided random search for weighted average algorithm, we can rank 30 with a relatively small dataset. For future work, we would combine the BERT and BiDAF together, which means that we replace BiDAF’s GloVe word embedding with BERT last layer’s output as as contextual word embedding. Hopefully we can improve our performance more with this idea.
References

Appendix 1: Details of all models

<table>
<thead>
<tr>
<th>ID</th>
<th>Experiment Name</th>
<th>Pre-trained model</th>
<th>Number of Epochs</th>
<th>Learning Rate</th>
<th>Batch Size</th>
<th>Max Sequence Length</th>
<th>Note</th>
<th>Dev F1</th>
<th>Dev EM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>out_maxseqlen245_bs12_lr3e-5_ep6l1</td>
<td>BERT-Base, Cased</td>
<td>6</td>
<td>3e-5</td>
<td>12</td>
<td>245</td>
<td>λ = 1e-4</td>
<td>77.206</td>
<td>73.478</td>
</tr>
<tr>
<td>2</td>
<td>out_maxseqlen245_bs12_lr3e-5_ep4</td>
<td>BERT-Base, Cased</td>
<td>4</td>
<td>3e-5</td>
<td>12</td>
<td>245</td>
<td>77.166</td>
<td>73.643</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>out_maxseqlen140_bs24_lr3e-5_ep4_11e-2</td>
<td>BERT-Base, Cased</td>
<td>4</td>
<td>3e-5</td>
<td>24</td>
<td>140</td>
<td>λ = 1e-2</td>
<td>76.76</td>
<td>73.955</td>
</tr>
<tr>
<td>4</td>
<td>out_maxseqlen140_bs24_lr3e-5_ep4_11e-3</td>
<td>BERT-Base, Cased</td>
<td>4</td>
<td>3e-5</td>
<td>24</td>
<td>140</td>
<td>λ = 1e-3</td>
<td>76.666</td>
<td>73.824</td>
</tr>
<tr>
<td>5</td>
<td>out_maxseqlen245_bs12_lr3e-5_ep6</td>
<td>BERT-Base, Cased</td>
<td>6</td>
<td>3e-5</td>
<td>12</td>
<td>245</td>
<td>75.925</td>
<td>72.343</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>out_maxseqlen140_bs24_lr3e-5_ep4_11e-4_uuncased</td>
<td>BERT-Base, Uncased</td>
<td>4</td>
<td>3e-5</td>
<td>24</td>
<td>140</td>
<td>λ = 1e-4</td>
<td>75.899</td>
<td>72.902</td>
</tr>
<tr>
<td>7</td>
<td>out_maxseqlen140_bs24_lr3e-5_ep4_11e-4</td>
<td>BERT-Base, Cased</td>
<td>4</td>
<td>3e-5</td>
<td>24</td>
<td>140</td>
<td>λ = 1e-4</td>
<td>75.705</td>
<td>73.001</td>
</tr>
<tr>
<td>8</td>
<td>out_maxseqlen245_bs12_lr3e-5_ep4_11e-4</td>
<td>BERT-Base, Cased</td>
<td>4</td>
<td>3e-5</td>
<td>12</td>
<td>245</td>
<td>λ = 1e-4</td>
<td>75.671</td>
<td>72.606</td>
</tr>
<tr>
<td>9</td>
<td>out_maxseqlen245_bs12_lr3e-5_ep4_11e-4</td>
<td>BERT-Base, Cased</td>
<td>5</td>
<td>3e-5</td>
<td>12</td>
<td>245</td>
<td>λ = 1e-4, add one layer</td>
<td>75.354</td>
<td>71.685</td>
</tr>
<tr>
<td>10</td>
<td>out_maxseqlen245_bs12_lr3e-5_ep4_uuncased</td>
<td>BERT-Base, Cased_uncased</td>
<td>4</td>
<td>3e-5</td>
<td>12</td>
<td>245</td>
<td>75.071</td>
<td>71.372</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>out_maxseqlen140_bs24_lr3e-5_ep6l4</td>
<td>BERT-Base, Cased</td>
<td>4</td>
<td>3e-5</td>
<td>24</td>
<td>140</td>
<td>74.679</td>
<td>71.915</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>out_maxseqlen290_bs10_lr3e-5_ep6l4</td>
<td>BERT-Base, Cased</td>
<td>4</td>
<td>3e-5</td>
<td>10</td>
<td>290</td>
<td>74.633</td>
<td>71.372</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>out_maxseqlen245_bs12_lr3e-5_ep4</td>
<td>BERT-Base, Cased</td>
<td>4</td>
<td>3e-5</td>
<td>12</td>
<td>245</td>
<td>74.546</td>
<td>71.092</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>out_maxseqlen200_bs12_lr3e-5_ep4</td>
<td>BERT-Base, Cased</td>
<td>4</td>
<td>3e-5</td>
<td>12</td>
<td>200</td>
<td>74.356</td>
<td>71.241</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>out_maxseqlen140_bs24_lr3e-5_ep5</td>
<td>BERT-Base, Cased</td>
<td>5</td>
<td>1e-5</td>
<td>12</td>
<td>245</td>
<td>73.885</td>
<td>70.829</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>out_maxseqlen240_bs6_lr3e-5_ep6l4</td>
<td>BERT-Base, Cased</td>
<td>4</td>
<td>3e-5</td>
<td>6</td>
<td>425</td>
<td>73.725</td>
<td>70.5</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>out_maxseqlen128_bs12_lr3e-5_ep4</td>
<td>BERT-Base, Cased</td>
<td>4</td>
<td>3e-5</td>
<td>12</td>
<td>128</td>
<td>73.638</td>
<td>71.142</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>out_maxseqlen245_bs12_lr3e-5_ep4+</td>
<td>BERT-Base, Cased</td>
<td>4</td>
<td>3e-5</td>
<td>12</td>
<td>245</td>
<td>add one layer</td>
<td>73.292</td>
<td>69.908</td>
</tr>
<tr>
<td>19</td>
<td>out_maxseqlen90_bs48_lr3e-5_ep4</td>
<td>BERT-Base, Cased</td>
<td>4</td>
<td>3e-5</td>
<td>48</td>
<td>90</td>
<td>72.854</td>
<td>70.813</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Experiment Name</th>
<th>Pre-trained model</th>
<th>Number of Epochs</th>
<th>Learning Rate</th>
<th>Encoder</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>baseline_sru</td>
<td>GloVe</td>
<td>30</td>
<td>0.5</td>
<td>SRU</td>
<td>64.08</td>
</tr>
<tr>
<td>21</td>
<td>baseline</td>
<td>GloVe</td>
<td>30</td>
<td>0.5</td>
<td>LSTM</td>
<td>Baseline</td>
</tr>
</tbody>
</table>

| ID | Experiment Name | Pre-trained model | Number of Epochs | Learning Rate | Note | |
|----|-----------------|--------------------|------------------|---------------|------| |
| 21 | Guided Random Search for Weighted Average | 79.944 | 77.081 |