Exploring Question Answering on SQuAD 2.0
Using Character Embeddings, Self-Attention, and QANet

Constance Hong  Daniel Ma  Jialin Zhuo
Department of Computer Science, Stanford University

Introduction

We explored two end-to-end models to optimize performance for the Question Answering task on the SQuAD dataset: Biaffine Bidirectional Attention Flow (BiDAF) and QANet. First, we added character embeddings and self-attention to the baseline BiDAF model, and we found that character embeddings improved the EM and F1 scores by a considerable amount. Then, we implemented QANet, which leverages local convolution and self-attention.

BiDAF Model

The original BiDAF model contains an embedding layer, encoder layer, bidirectional attention layer, modeling layer, and output layer. The provided baseline model only included word embeddings, so we added character embeddings as outlined in the 2016 paper to improve performance. We chose to implement a 2D convolutional neural network for this model.

Next, we implemented self-attention as outlined in the "R-Net: Machine Reading Comprehension With Self-Matching Networks" paper, which uses the current passage word and its matching question information to extract evidence from the whole passage. The resulting passage representation $d^p$ is as follows [1]:

$$d^p = BiRNN(s^p; [c^p, a^p])$$

where $a^p$ is the attention-pooling vector of the whole passage. The output attention-pooling vector of BiRNN is $d^p$:

$$a^p = \sum_{i=1}^{T} e^{x_i}$$

Character Embeddings Tensorboard

We were able to optimize the most using the BiDAF model with character embeddings, as shown in the below Tensorboard.

QANet Model

1. Embedding Layer

We broke down the question strings into sequences of word and character tokens to prepare our input. The first 16 characters of the word were concatenated, then each embedding was expanded to 128 characters through convolution with a 1D kernel of size 4. Then, to get a vector representation of each word, we took the maximum across the character dimension. Finally, we applied a two-layer highway network on the embedding vector. We performed this process for every word in the context and question.

2. Encoder Block

This is the main component of QANet. First, we have the positional encoding, which consists of both sinusoidal and cosinusoidal encodings. Then, we have several convolution layers, a multi-head attention layer with 8 heads, and a feed-forward layer. All of these layers are followed by a normalization layer to stabilize the hidden state dynamics. We use the encoder block to process the input embeddings and bi-directional attention outputs (Figure 1).

![Figure 1: Encoder Block](image)

3. Context-Query Attention

We applied the BiDAF context-query attention mechanism to QANet. [2]

4. Output

Our output layer computes the softmax using two different concatenations of encoder outputs. Let $M_b, M_l, M_c$ denote the output of the encoder block stack. The loss function is computed as the negative sum of log probabilities of the predicted distributions averaged over all training samples. Here, we predict the probability of each position in the context being the start end of an answer span:

$$P_{Span} = softmax(W_2[M_b, M_l, M_c])$$

where $W_1$ and $W_2$ are learnable parameters.

![Figure 2: A visualization of the QANet architecture](image)

Experiments

Data

We used the SQuAD dataset, which has already been split into the training, dev, and test sets. The training set is 40 MB, and the dev set is 4 MB. We used the provided preprocessing code to retrieve the word tokens and populate our counters, then added code to also retrieve character tokens.

Model Evaluation

To evaluate the models we explored, we looked at Exact Match (EM) and F1 scores. EM is a strict binary measure indicating whether the output matches ground truth, whereas F1 is the harmonic mean of precision and recall.

Results

<table>
<thead>
<tr>
<th>Model</th>
<th>EM</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>61.72</td>
<td>56.37</td>
</tr>
<tr>
<td>BiDAF + Character Embeddings</td>
<td>63.64</td>
<td>60.29</td>
</tr>
<tr>
<td>BiDAF + Character Embeddings + GRU</td>
<td>61.10</td>
<td>57.57</td>
</tr>
<tr>
<td>BiDAF + Self Attention</td>
<td>61.17</td>
<td>58.74</td>
</tr>
<tr>
<td>BiDAF + Character Embeddings + Self-Attention</td>
<td>62.63</td>
<td>58.42</td>
</tr>
<tr>
<td>QANet</td>
<td>52.19</td>
<td>52.19</td>
</tr>
</tbody>
</table>

Table 1. EM and F1 scores for all models we experimented with on the dev set.

Analysis

After our explorations with character embeddings, self-attention, and QANet, we found that our BiDAF model with added character embeddings optimized performance the most. It achieved EM and F1 scores of 63.64 and 60.29 respectively on the dev set, and 63.24 and 59.81 on the test set, which marked a notable increase from the given baseline model.

Unfortunately, adding self-attention did not improve performance beyond the baseline. In the future, we would like to explore other attention mechanisms, like co-attention (as outlined in the R-Net paper) to hopefully improve EM and F1 scores.

Our QANet implementation also ran into some challenges as our EM and F1 scores plateaued to a linear value of 52.19. We experimented with leakyReLUs, lower learning rates, and dropout rate reductions, but were unable to achieve higher scores.

References

