Introduction

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 a segment of text, or span, from the corresponding reading passage, or the question might be unanswerable.

The current landscape of neural reading comprehension models are at training to predict single tokens or entities but fail to take into account the start and ends of the candidate answer spans into the probability distribution. The goal of this project is to explore this avenue and develop one such model, the Dynamic Chunk Reader (DCR) proposed by Yu et al. [3]. We use BiDAF as a baseline, improve the BiDAF model by extending the embedding layer to incorporate character-level embedding, then compare the performance of the implemented DCR model against the baseline.

Dataset/Task

The SQuAD 2.0 dataset is used as the reading comprehension dataset. Dataset is split into 129,941 samples in train set, 6,678 samples in dev set, and 5,291 samples in test set.

Background: Dynamic Chunk Reader

Dynamic Chunk Reader (DCR) explores the idea of modeling the question answering problem as a probability distribution over all possible answer chunks in the context paragraph, as opposed to modeling start and end indexes of the answer separately. DCR works in four steps (Fig. 2):

Encoder Layer: The encoder layer makes use of two bi-directional RNN encoders with gated recurrent units (GRU) to encode each passage (P) and question (Q) for each example i to retrieve a hidden state for each word position, and . For each position i, a GRU computes with input and previous state .

Attention Layer: The attention layer introduces a novel attention mechanism based on word-by-word style attention methods.

Chank-Representation Layer: This layer dynamically generates the answer chunks candidates, and produces their representation. For an answer chunk candidate ..., spanning from position to , the chunk representation is given by concatenating the hidden state of the first word in the chunk in the forward RNN and that of the last word in the backward RNN:

Encoder Layer: Finally, we rank the generated answer chunks by their similarity score to the question representation. As for answer spans, a question with RNN encoder outputs and for backward and forward passes respectively. At stage 4, it has a representation .

Then, we model the probability of chunk \( a \in A \) as

\[
P(a; \theta) = \text{softmax} \left( \sum_{i=1}^{L_a} \left[ R_a[i] \cdot \mathbf{r} \right] \right)
\]

The chunk with the highest probability is taken as the answer, and the negative log-likelihood is minimized for training.

Approach

Our first contribution is to present an implementation of the Dynamic Chunk Reader model in PyTorch. As a second contribution, we implemented character-level word embeddings in our embedding layers. We used this layer for both BiDAF model, as well as our newly implemented DCR model.

Finally, our third contribution is what we call the Static Chunk Reader (SCR). The SCR model generates candidate answer chunks in the 3rd layer in Figure 2, and does so in a rather arbitrary, we propose and implement method of candidate answer generation based on the dependency structure of the context paragraph.

Experiments

We present the results of 3 experiments: 3 variations of BiDAF, and 2 variations of DCR.

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
<th>EM</th>
</tr>
</thead>
<tbody>
<tr>
<td>BiDAF (baseline)</td>
<td>62.49</td>
<td>59.30</td>
</tr>
<tr>
<td>BiDAF w/ char embed 500</td>
<td>64.99</td>
<td>61.30</td>
</tr>
<tr>
<td>DCR w/ char embed 200</td>
<td>65.85</td>
<td>62.29</td>
</tr>
<tr>
<td>SCR</td>
<td>52.2</td>
<td>52.1</td>
</tr>
</tbody>
</table>

Notice that the BiDAF implementation with character-level embeddings performs better than the baseline BiDAF model, and character-level embeddings do not have too big of an impact on the model accuracy. We also observe that DCR and SCR models performs poorly on the SQuAD 2.0 dataset. The results of the DCR model are lower than what we expected, as it was shown in [1] to perform well on the SQuAD 1.0 dataset. It could be that this model does not generalize similarly well on the SQuAD 2.0 dataset. Similarly, the performance of SCR was less than expected. What’s reassuring is that this model has lots of room for experimentation with different dependency-based chunk generation, which we expect will improve the performance.

Conclusions

We conclude that the Dynamic Chunk Reader, at least in the form presented in [3], is not fit for the SQuAD 2.0 challenge. Static Chunk Reader did not perform better, but we are hopeful it can be adapted or combined with a different architecture to deliver more promising results.

References