Dynamic and Static Chunk Reader (DCR/SCR) on SQuAD 2.0 Task

Rubens Lacouture [1] Sho Ko [1] Mahammad Shirinov [2]

[1]Department of Electrical Engineering, Stanford University [2]Department of Management Science & Engineering, Stanford University

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 aim at learning 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 [1]. 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 examples in train set, 6078 examples in dev set, and 5291 examples 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 end end indices 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_i) and question (Q_i) for each example i to retrieve a hidden state for each word position p_{ij} and q_{ik}. For each position t, GRU computes h_t with input x_t and previous state h_{t-1}

$$
\begin{aligned}
r_t &= \sigma\left(W_r x_t + U_r h_{t-1}\right) \\
u_t &= \sigma\left(W_u x_t + U_u h_{t-1}\right) \\
\tilde{h}_t &= \tanh\left(W x_t + U\left(r_t \odot h_{t-1}\right)\right) \\
h_t &= (1-u_t)\cdot h_{t-1} + u_t \cdot \tilde{h}_t
\end{aligned}
$$

  For a word t, the bi-directional contextual encoding of input x_t is represented as $h_t = \left[\overrightarrow{h_t}; \overleftarrow{h_t}\right]$

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

$$
\begin{aligned}
\alpha_{jk} &= h_j^p \cdot h_k^q \\
\beta_j &= \sum_{k=1}^{|Q|} \alpha_{jk} h_k^q \\
v_j &= \left[h_j^p; \beta_j\right]
\end{aligned}
$$

- **Chunk Representation Layer** This layer dynamically generates the answer chunk candidates, and produces their representation. For an answer chunk candidate $c^{m,n}$ spanning from position m to n, the chunk representation $\tilde{\gamma}_{m,n}$ 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:

$$
\tilde{\gamma}_{m,n} = g\left(\gamma_m, \ldots, \gamma_n\right) = \left[\overrightarrow{\gamma_m}; \overleftarrow{\gamma_n}\right]
$$

- **Ranker Layer** Finally, we rank the generated answer chunks by their similarity score to the question representation. As for answer spans, a question q with RNN encoder outputs $\overrightarrow{h_k^q}$ and $\overleftarrow{h_k^q}$ for backward and forward passes respectively, at stage k, has a representation $\left[\overrightarrow{h_{|q|}^q}; \overleftarrow{h_1^q}\right]$. Then, we model the probability of chunk $c^{m,n}$ as

$$
\mathbb{P}\left(c^{m,n} \mid q\right) = \operatorname{softmax}\left(\tilde{\gamma}_{m,n} \cdot \left[\overrightarrow{h_{|q|}^q}; \overleftarrow{h_1^q}\right]\right)
$$

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 that we 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 DCR model generates candidate answer chunks in the 3rd layer in Figure 2, and does so rather arbitrarily. We propose and implement a method of candidate answer generation based on the dependency structure of the context paragraph.

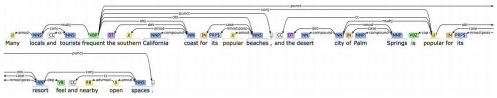

Figure 1. An example of a SQuAD context parsed via Stanford CoreNLP tool.

In a dependency graph, words of a sentence are connected by dependency relationships, in form of $head \rightarrow child$. Let us call the words that do not have any children *leaves*. In Figure 1, some leaves are "southern", "Palm", and "nearby". As one of the variants, we propose candidate answer chunks that are either (1) leaves, or (2) spans of text between a leaf and its parent. Examples of (2) in Figure 1 would be "southern California coast", "Palm Springs" and "nearby open spaces", all of which are, in fact, answers to SQuAD questions! Our preliminary research shows that >95% of answers in the SQuAD dataset fit this primitive "leaf" or "leaf-parent" description. Note also that such chunks are much less in number than the chunks generated by [1], and have other favorable properties.

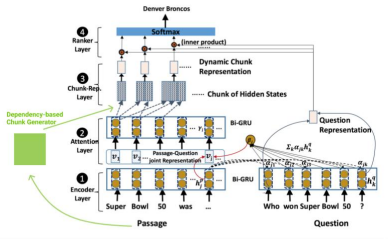

Figure 2. Dynamic Chunk Reader architecture from [1]. The green box and arrows show the dependency-based chunk generator that is used in SCR.

The Static Chunk Reader (SCR), then, is a variation of DCR where the candidate answer chunks are *static* – generated from text (e.g. at preprocessing time or runtime). The green box and arrows on the left side of Figure 2 demonstrate this addition.

Experiments

We present the results of 5 experiments: 3 variations of BiDAF, and 2 variations of DCR.

Model	F1	EM
BiDAF (Baseline)	62.649	59.301
BiDAF w. char embed-100	64.894	61.267
BiDAF w. char embed-200	63.85	60.29
DCR w. char embed-100	52.2	52.2
SCR	52.1	52.1

Table 1. Summary of model results.

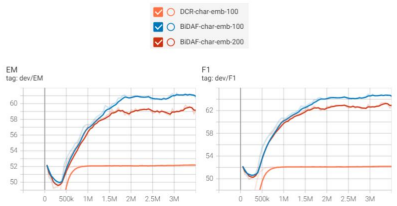

Figure 3. Tensorboard plots of model performance.

Analysis

The results show that the BiDAF implementation with character-level embeddings perform better than the baseline BiDAF model, and character-level embeddings size doesn't have too big of an impact on the model accuracy. We also observe that DCR and SCR models perform poorly on the SQuAD 2.0 dataset. The results of the DCR model are lower than what we expected, as it was shows 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 generators, which we expect will increse the performance.

Conclusions

We conclude that the Dynamic Chunk Reader, at least in the form presented in [1] is not fit for the SQuAD 2.0 challenge. Static Chunk Reader did not perform better, but we are hopeful that it can be adapted or combined with a different architecture to deliver more promising results.

References

[1] Yang Yu, Wei Zhang, Kazi Hasan, Mo Yu, Bing Xiang, and Bowen Zhou. End-to-end answer chunk extraction and ranking for reading comprehension. *arXiv preprint arXiv:1610.09996*, 2016.