Question Answering for SQUAD 2.0

Stanford CS224N Default Project

Sreya Guha Department of Computer Science
Stanford University
sguha22@stanford.edu

Abstract

Reading comprehension is one of the central goals of natural language processing. Therefore, exploring question answering on SQuAD 2.0, a dataset with both answerable and unanswerable questions is highly important. In this paper, I describe my implementation of a baseline testing as well as attempted implementation of QANet. QANet is a model that has been shown to have great performance on reading comprehension tasks [3]. Though my attempt at implementing this model failed, I describe my approach and limitations.

1 Key Information to include

- Mentor: N/A
- External Collaborators (if you have any): N/A
- Sharing project: N/A

2 Introduction

Question answering and reading comprehension is a central task in language processing. In the given task, a model will be given a paragraph, and a question about the content of that paragraph as input. These systems are important for understanding information from large sources of text and can prove to be an interesting research question. Question Answering has a wide range of applicability from information retrieval to simulating human dialogue to parsing. Given the relevance of the research question, it is important to develop complex and strong models to address the need for accurate machine reading comprehension. Developing high-performance models on SQuAD 2.0 is particularly challenging as this requires a lot of computational power and access to data. In addition, the introduction of unanswerable questions complicates the models.

3 Related Work

In 2017, Seo et al. introduced the Bi-Directional Attention Flow (BIDAF) network[2]. BIDAF makes use of a bidirectional attention flow system to gain a query-aware context representation. The system also makes use of word-level, character-level and contextual embeddings. Seo et al. made three noteworthy contributions to the attention mechanism: first, reducing information loss on the attention layer; second, an attention mechanism that is memory-less and lastly, bidirectional attention flows (query-to-context and context-to-query). Through his first contribution, Seo choose to compute attention at every time step instead constraining the attention layer to be of a fixed size allowing for the attention layer to reduce information loss. In his second, Seo allows attention from previous time steps to be independent from attention at the current time step, meaning that attention at any given point is not affected by previous errors and can instead focus on the query interaction[2].

In 2018, Yu et al. proposed a new QA system named QANet [3]. The QANet encoder exclusively includes separable convolutions and self attention and does not require recurrent networks. Their

model consists of five main components: an embedding layer, an encoder layer, a context-query attention layer, a model encoder layer and an output layer. The end-to-end model proposed was both fast and accurate and they were able to achieve a F1 score of 84.6.

4 Approach

In the default model, a baseline was provided for use. This default model was based on Bidirectional Attention Flow (BiDAF)[2]. I also attempted to implement the QANet implementation described in the Yu et al paper. This model consists of five main components: an input embedding layer, an embedding encoding layer, a model encoding layer and an output layer. This model is based on the architecture described in the Yu Paper[3]. I was unable to eventually implement this model but the text that follows is my thought process and attempt at creating the model.

In the input embedding layer, the model concatenates character and word vectors to construct a singular representation for each word, using word vectors pretrained from Glove. In implementing this, I had difficulty with my code as I was not able to access the pretrained embeddings for word and character vectors. I tried to calculate the embeddings at the character-level by using a one-dimensional convolutional neural network with a rectified linear activation function (ReLU) and a maximum pooling layer. To obtain the resulting embedding, we simply concatenate both the word-level embedding and the character-level embedding. However, here, I ran into difficulty and was unable to concatenate them.

Next, I attempted the embedding encoding layer. Before feeding the input into the encoder block, I created positional encodings added to the embeddings. Additionally, there are three convolutional layers with depthwise separable convolutions and a feedforward layer[1].

Next, I pass the context to query attention matrix and the query to context attention and the encoded context into the model encoding layer. This layer consists of seven encoder blocks which each consist of two convolution layers. From this, we get three output matrices.

Lastly, we pass the three output matrices from the previous layer through a linear layer and then through a softmax function to construct a probability distribution for the starting position of the answer span. We also construct a probability distribution for the ending position of the answer span.

5 Experiments

5.1 Data

Since this was the default project, I used the SQUAD 2.0 dataset. This dataset combines 100,000 questions from the original SQuAD with more than 50,000 unanswerable questions. Each datapoint is as follows: a question and answer from crowdsourced using Amazon Mechanical Turk with a paragraph in SQuAD are from Wikipedia. The output is the answer found as a substring of the context paragraph or 'No Answer' if the question provided is unanswerable.

5.2 Evaluation method

Quantitatively, I evaluated our models on both the dev and test leaderboard, where the model was assigned Exact Match (EM) and F1 scores. The Exact Match score is a binary measure of whether the system is able to output the ground truth answer exactly. The F1 score is less strict, as it takes the harmonic mean of precision and recall. When a model is evaluated, the maximum EM and F1 scores across three human-provided answers is taken for the specific question, making evaluation more forgiving.

Given that I could not get my complete model working, I submitted the baseline results but have included my code and details for consideration in grading.

5.3 Experimental details

5.4 Results

For the baseline, the EM was 49.521 while the F1 was 52.873 for the dev set. For the test set, the EM was 47.591 while the F1 was 50.959.

Given that I could not get my final model working, my results were worse than intended. I expected that the QANet model I created would reproduce better results but could not get it working on my virtual machine.

6 Conclusion

In this paper, we attempted to implement QANet. QANet in particular has been shown to perform well on reading comprehension tasks given its feed-forward nature. Moreover, QANet is an end-to-end model that avoids recurrent networks and uses self attention and convolution to encode query and context. In the future, we would be interested in training different models such as Match-LSTM and transformers to achieve a better question-answering and a better position in the leaderboard.

References

[1] Lukasz Kaiser, Aidan N. Gomez, and François Chollet. Depthwise separable convolutions for neural machine translation. *CoRR*, abs/1706.03059, 2017.

[2] Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603, 2016.

[3] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. Qanet: Combining local convolution with global self-attention for reading comprehension. *CoRR*, abs/1804.09541, 2018.