CS224N: Assignment 4

Saghar Hosseini
Microsoft
One Microsoft Way
Redmond, WA
Saghar.hosseini@microsoft.com

Yun Li
Microsoft
One Microsoft Way
Redmond, WA

Abstract

In this assignment, the goal is to implement a deep learning neural networks for Reading Comprehension using the recently published Stanford Question Answering Dataset (SQuAD)[1].

1 Introduction

The Machine Reading Comprehension task in this scope refers to the task of finding the span of answer in a context. The dataset for this task is called SQuAD [2] which is comprised of around 100K question-answer pairs, along with a context paragraph. The context paragraphs were extracted from a set of articles from Wikipedia. Humans generated questions using that paragraph as a context, and selected a span from the same paragraph as the target answer.

In our approach, we will implement the baseline which is very similar to [1]. The question is encoded using a BiLSTM and then the context is encoded conditioned on the question encoding. For this step we used attention mechanism and BiLSTM. Then, using bidirectional Match-LSTM [1], we can mix the encoding representation of question and context in a single tensor and call it knowledge representation. Finally, LSTM will be performed over the knowledge representation to provide two classifiers. One classifier determines the index of the start of the answer's span, and the other classifier predicts the end index of the answer's span.

The majority of our time and effort were invested on implementation of this method. In the process, we have faced many challenges including designing the architecture, debugging this complex neural network, and constraints on memory availability and computational power. Therefore, we were not able to perform as many experiments as we hoped for this assignment, and it will be performed as our future work. However, we enjoyed performing this assignment through which we dived deeper into TensorFlow functionalities and documentations, learned how to efficiently debug a neural network. In addition, we have become more familiar with different Recurrent Neural Network cells and their implementations.

2 Background & Related Work

3 Approach

We used a similar approach to what was suggested in handouts to implement the baseline. The data preprocessing include creating masks and mapping word indexes to their embedding representation.

3.1 Encoder

a. We first ran a BiLSTM over the question sequences using *tf.nn.bidirectional_dynamic_rnn(…)* and concatenated the two final hidden states from forward and backward passes. This will give us an encoding of the question represented by $H^q \in \mathbb{R}^{2l \times Q}$ where l is the size of a hidden state in LSTM cell and Q is the number of tokens in questions. Note that we padded all the questions where the maximum length is 100.

b. We ran a BiLSTM over context conditioned on the H^q by using an attention LSTM cell. In other words, we have:

$$H^p = LSTM(Q), \qquad Atten_p = \sum_{i=1}^{2l} \left(H^q \odot h_t^p\right)_i, \qquad z_t = \left[h_t^p, H^q Atten_p\right],$$
$$h_{t+1}^c = LSTM(z_t, h_t^c)$$

Where $h_t^c \in \mathbb{R}^l$ and we perform the above operation for the backward sequence and concatenate the hidden states of forward and backward operations. Therefore, thee context representation is $H^c \in \mathbb{R}^{2l\times P}$ where P is the number of tokens in context and note that we padded all the paragraphs where the maximum length is 766.

3.2 Decoder

c. Then we used Match-LSTM approach in [1] to calculate a knowledge representation based on H^p and H^q. The match-LSTM sequentially goes through the paragraph. At token t it uses the word-by-word attention mechanism:

$$G_t = \tanh\left(W^q H^q + \left(W^p h_t^p + W^r h_{t-1}^r + b^p\right) \otimes e_Q\right) \in \mathbb{R}^{2l\times Q}$$

$$\alpha_t = softmax(w^T G_t, b \otimes e_Q)$$

$$z_t = \left[h_t^p, H^q \alpha_t\right], \qquad h_{t+1}^r = LSTM(z_t, h_t^r)$$

Where $W^q, W^p, W^r \in \mathbb{R}^{l\times l}, b^p, w \in \mathbb{R}^l, b \in \mathbb{R}$, $h_{t+1}^r \in \mathbb{R}^l$ and we perform the above operation for the backward sequence and concatenate the hidden states of forward and backward operations. This produce a knowledge representation $H^r \in \mathbb{R}^{2l\times P}$.

3.3 Answer Pointer Layer

d. In this layer we use the Boundary Model in [1]. This layer uses the knowledge representation H^r and predicts two probability distribution over the tokens in the context. One probability distribution is for the start index and the other one is for the end index. The all the tokens between these two indices are considered to be the answer.

The attention mechanism is used to produce logits over the tokens in context:

$$F_t = \tanh(VH^r + (W^a h_t^a + b^a) \otimes e_P) \in \mathbb{R}^{l\times P}$$

$$\beta_t = softmax(v^T F_t, c \otimes e_P)$$

$$z_t = \left[h_t^p, H^q \alpha_t\right], \qquad h_{t+1}^r = LSTM(z_t, h_t^r)$$

Where $W^a \in \mathbb{R}^{l\times l}, V \in \mathbb{R}^{l\times 2l}, b^a, v \in \mathbb{R}^l, c \in \mathbb{R}$, $h_{t+1}^a \in \mathbb{R}^l$.

We can model the probability distribution of the start index as:

$$p(a_s|H^r) = \prod_t p(a_t = 1|a_1, a_2, \ldots, a_{t-1}, H^r)$$

Where : $p(a_t = 1|a_1, a_2, \ldots, a_{t-1}, H^r) = \beta_t$.

Similarly, we perform another answer pointer layer for the end index of the answer and to train the model, we minimize the summation of the cross-entropy loss functions over these two multi-classification problems with P classes.

3.4 Evaluation Metrics

Finally the evaluation metrics are F1 score and Exact-Match (EM). F1 score is calculated for each token with binary classes, part of answer or not part of the answer. Then the micro average of the classification results will be presented as F1 score where $F1 = \frac{2TP}{2TP+FN+FP}$, $EM = TP$

4 Experiments

We tested our model on the SQuAD [1] dataset. It is comprised of around 100K question-answer pairs, along with a context paragraph. The context paragraphs are extracted from Wikipedia with answers as human labeled span within the context paragraphs. With the limitation of computational resource, we manager to run our model with reduced parameters. The batch size is 10, state size is 10 and output size is 40. And we trained out model on 500 samples with embedding size 100. We were able to run the code with a small F1 score.

Challenges:

To get familiar with a new model in a very short time. Also good architecture of the model in the code is a new challenge for us.

What we can do to improve:

Running efficiency. We did the implementation in a rush, there is a lot of room to improve in term of efficiency. Eg. The tf.nn.rnn_cell.BasicLSTMCell we use tensor as input, if using tuple instead, we can avoid array_ops.split and improve our speed.

Also we did not optimize the memory usage. During the training, we were experiencing a lot of out of memory exceptions. As a results of that, we have had to reduce the parameters to accommodate this limitation.

5 Conclusion

In this assignment, we implemented a deep learning model to answer questions for SQuAD. We use BiLSTM to encode the questions and context, bidirectional Match-LSTM [1] for mixing them (knowledge representation). Finally, LSTM will be performed over the knowledge representation to determine the start and end index. We learnt a lot from this project, including implementing deep learning model from scratch, Dealing with practical constraints and tones of trouble shooting experience.

6 References

[1] ShuohangWang and Jing Jiang. Machine comprehension using match-lstm and answer pointer. arXiv preprint arXiv:1608.07905, 2016.

[2] https://stanford-qa.com