Natural Language Processing - Home Work 4

CSE 538 Fall '19

Prof. Niranjan Balasubramaniam

Submission By:

Narayan Acharya

112734365

Contents

1 Base Model: GRU Implementation

1. **Model Architecture:**

   Our model is a neural network for relationship extraction. We try to emulate the model as mentioned in the paper [1]. In accordance with the model architecture in the paper we build a 3 layer architecture - Bi-directional GRU layer, an attention layer, and an output layer. [The architecture in the paper has an additional embedding layer mentioned separately as well.]

   For the first layer, we have a Bi-directional GRU. The input to the GRU is a concatenated tensor for our word embeddings and Part of Speecg (PoS) embeddings. The shape of the input to this layer is [*batch_size*, *sequence_length*, *embeddings_size*]. In our case, we used GloVe embeddings of 100 dimension for both words and PoS. Therefore on concatenating we received a 200 dimensional embedding for every token in our input, i.e. *embeddings_size* is 200. *sequence_length* is the maximum number of token in the current batch. For sequences with lesser than the max number of tokens, we have padding (zeros) appended to the input. In order to work with this, we use a mask along with our input to the GRU to indicate to the GRU of what values it needs to be working on and which ones to ignore.

   The second layer is our attention layer. We calculate the attention based on the equations 9, 10, 11 and 12 as suggested in the paper [1]. The input to the attention layer is the output from the GRU layer of shape [*batch_size*, *sequence_length*, $2 \times$ *hidden_size*]. The output of the attention layer is a tensor of shape [*batch_size*, $2 \times$ *hidden_size*] which is essentially a sentence representation for each sentence in our batch. The important deviation from the equation is our calculation of r in equation 11 from the paper is where in the calculation of $H\alpha^T$ we take a weighted sum of the output vectors, something not clearly mentioned as part of the equations but is mentioned in the text.

Experiment	**F1 Score (5 epochs)**
Baseline - Word + PoS + Shortest Path	**0.5683**
Experiment 1 - Only Word Embeddings	0.5503
Experiment 2 - Only Word + PoS Embeddings	0.5115
Experiment 3 - Only Word + Shortest Path	0.5886

Table 1: Bi-GRU Experiments Comparison

The final layer, is the output layer which takes in the output of the attention layer and outputs a tensor of shape [*batch_size*, *num_classes*]. This is essentially a fully connected/dense layer that acts as our classification layer to assign probability of the input in each of the possible 19 classes. We have 9 possible relationship classes, but since they can be different in each direction, we get 18 different possibilities from them and we have one class which we use the 'Other' class.

Another task was to calculate the regularization loss, using a regularization factor of 10^{-5}, as mentioned in the paper while back-propagating the gradients back through our network. The regularization loss consists of L2 loss for all our trainable variables in the model.

2. **Observations of experiments:**

   As part of the base model, we look to carry out certain experiments as outlines in table 1.

   Some inferences one can draw from the results of the experiments:

   (a) From the results for the baseline and experiment 3, one can infer that calculating the shortest path is critical for better performance. The shortest path possibly focuses our model to only use the most important/relevant information for the task at hand.

   (b) Using just the word embeddings, like in experiment 1, is not sufficient information for our model to confidently classify the relationship into the possible classes.

   (c) Extra information, like that of the PoS tags, like in experiment 2, without the shortest path did not help but actually decreased the F1 score. We gave our model too much information for it to reliably choose from the possible classes. The PoS tags of only the words around our entities seem to have a say in the class of the relation rather than all the tokens in the entire context.

2 Advanced Model: CNN with Multiple Filter Convolution & Max Pooling

1. Model Inputs: The advanced model I chose was loosely based on the paper [2]. The model performed better than the baseline GRU model and uses only word embeddings and positional embeddings(as defined below) for the task. PoS embeddings and shortest path were *not* used.

   Positional Embeddings: We assign numerical values to tokens based on how close to our entities they are. Consider the following example:

*The **ship** is in the **dock**.*

ship and **dock** here are our two entities and lets say the relation between the two is *Contains(**dock**, **ship**)*. We assign positional embeddings based on both entity 1 and 2, which will look something like this for our case as shown below.

$$position\_1 = [1, 0, 1, 2, 3, 4]$$
$$position\_2 = [5, 4, 3, 2, 1, 0]$$

Notice that entities are at positions 1 and 5 (indexing based on 0) in our sentence, where *position_1* and *position_2* mark them as 0 and as you move away from them, the positional values increases by 1.

2. Model Architecture:

   High level architecture of the advanced model can be seen in figure 1

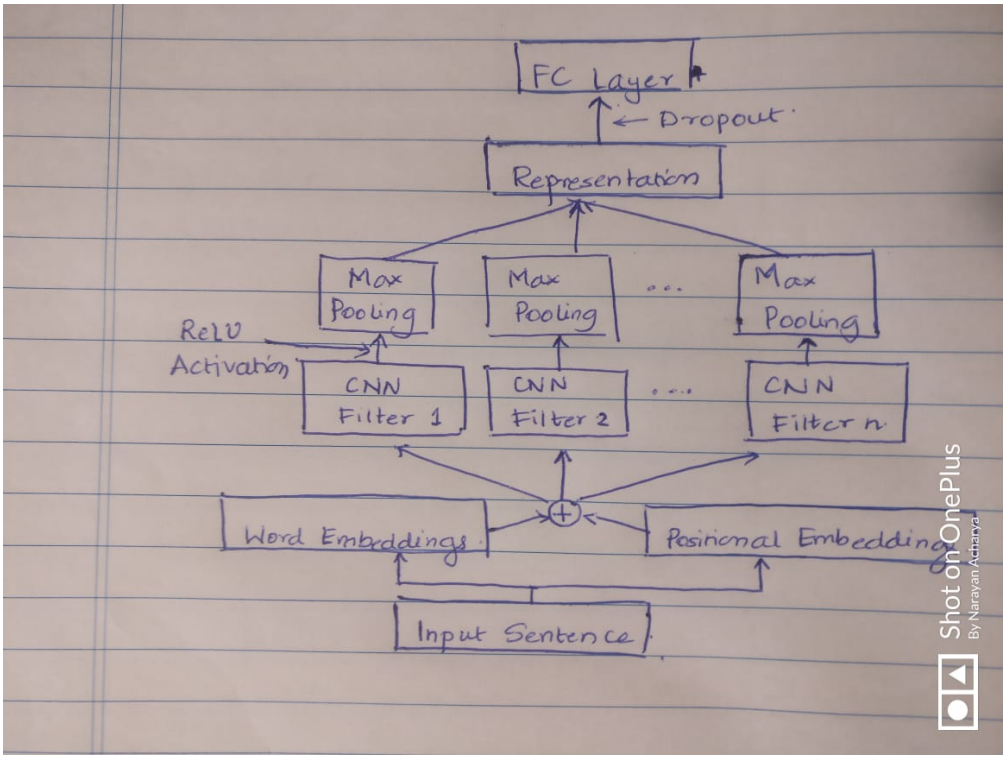

Figure 1: Advanced Model: High Level Architecture

We perform similar steps to tokenize our sentence like we did in the base model and get the word token, which we then use our GloVe embeddings for finding the representation of the word. We also calculation position values as shown above. We also maintain an embedding matrix for these positional values, which we learn as we train our model. We then concatenate the word embedding, position according to entity 1 embedding and position according to entity 2 and use this as our input to the next layer.

In the next layer, we convolve filters of multiple sizes across the input from the previous layers, the embeddings in this case. We use ReLU activation with each CNN filter and use Max Pooling on top of each.

After each of the filters are applied, we concatenate the output from each into a single representation and flatten it out to pass to our Fully Connected Layer for acting as the final classification layer. We use dropout before we send it to Fully Connected layer to address for over-fitting.

NOTE: For position embeddings, another approach one can try out (not done as part of this assignment) is assigning higher numbers to the entities that decrease as you move from the entity token indices. This intuitively fits better with how we convolve and use Max-Pooling on the the convolved outputs.

3. Model Parameters & Performance:

   We use the following parameters for training our model which varies slightly from the numbers as suggested in the paper.

   (a) We use GloVe embeddings of dimension 300. We define the positional embeddings to be of size 100. GloVe embeddings are *not* pre-trained while positional embeddings are something we learn as we train.

   (b) We use 4 filter sizes of value 2, 3, 4, 5 and each filter size having 128 filters. We use ReLU activation and Max-Pooling with each filter size.

   (c) We use a dropout value of 0.3 and a mini-batch size 0f 20 for training across 6 epochs.

   With the above parameters the model got an F1-score of 0.6289 which is better than the baseline Bi-GRU model which got an F1 score of 0.5683 with word, PoS and shortest path features.

   Other experiments were run as:

   (a) Tweaking the batch size to 30 and training for 5 epochs resulted in an F1 score of 0.6191.

   (b) Tweaking the embedding dimension used to be 200 over training 5 epochs resulted in an F1 score of 0.58.

References

[1] Attention-Based Bidirectional Long Short-Term Memory Networks for Relation Classification, `https://www.aclweb.org/anthology/P16-2034.pdf`

[2] Relation Extraction: Perspective from Convolutional Neural Networks, `https://cs.nyu.edu/~thien/pubs/vector15.pdf`