NLP ASSIGNMENT 4
SAKSHI GUPTA-112552239

Section 1: GRU+ Attention Implementation

As mentioned in the paper, this is the basic architecture of the system:
(1) Input layer: input sentence to this model;
(2) Embedding layer: map each word into a low dimension vector;
(3) LSTM layer: utilize BLSTM to get high level features from step (2);
(4) Attention layer: produce a weight vector, and merge word-level features from each time step into a sentence-level feature vector, by multiplying the weight vector;
(5) Output layer: the sentence-level feature vector is finally used for relation classification.

In our architecture we have a bidirectional GRU which takes as input the concatenated input of the word embeddings and POS tag embeddings provided to us. The hidden size for our GRU layer is kept as 128 and then we use keras.layers.Bidirectional to construct our bidirectional GRU network.
We pass our input to the network above and its output is then passed to the attention layer for further processing.

In the attention layer, we perform the following steps as mentioned in the paper:
(1) Apply tanh non linear function to the outputs from the BiGRU or BiLSTM
(2) Then we multiply the above output with omegas given to us to get a confidence score for each word
(3) Then we apply softmax to the above output to normalize the scores
(4) Then we multiply the original output of the BiLSTM with the above normalized score
(5) Apply tanh to the above output

The answer is returned back and it is passed to the decoder which is basically a keras.layers.Dense function to produce the final output’s shape to be compatible with the number of classes that we have.

Section 2: Observations of GRU

- All (Basic):
  Epoch 0: Val loss for epoch: 2.0974 Val F1 score: 0.3258
  Epoch 1: Val loss for epoch: 1.8976 Val F1 score: 0.3865
  Epoch 2: Val loss for epoch: 1.7051 Val F1 score: 0.5019
  Epoch 3: Val loss for epoch: 1.7319 Val F1 score: 0.5476
  Epoch 4: Val loss for epoch: 1.7969 Val F1 score: 0.5483

These are the results of the basic configuration

- Only Word Embeddings:
  Epoch 0: Val loss for epoch: 2.0443 Val F1 score: 0.3873
  Epoch 1: Val loss for epoch: 1.7445 Val F1 score: 0.4349
  Epoch 2: Val loss for epoch: 1.5558 Val F1 score: 0.5273
  Epoch 3: Val loss for epoch: 1.6281 Val F1 score: 0.5525
  Epoch 4: Val loss for epoch: 1.808 Val F1 score: 0.5211

When we only use the word embeddings the model performs its worst.

- Word +pos:
  Epoch 0: Val loss for epoch: 2.1949 Val F1 score: 0.3679
  Epoch 1: Val loss for epoch: 1.8076 Val F1 score: 0.4267
  Epoch 2: Val loss for epoch: 1.5824 Val F1 score: 0.5186
  Epoch 3: Val loss for epoch: 1.6294 Val F1 score: 0.5342
  Epoch 4: Val loss for epoch: 1.8358 Val F1 score: 0.5439

When we include both the word embeddings and POS tags information the model still performs better than the previous configuration and almost at par with the basic configuration.

- Word + dep:
  Epoch 0: Val loss for epoch: 1.92 Val F1 score: 0.427
  Epoch 1: Val loss for epoch: 1.6591 Val F1 score: 0.4677
  Epoch 2: Val loss for epoch: 1.5374 Val F1 score: 0.5047
  Epoch 3: Val loss for epoch: 1.5884 Val F1 score: 0.5693
  Epoch 4: Val loss for epoch: 1.7551 Val F1 score: 0.5868

When we use the word embeddings without concatenating it with POS embeddings and then use the dependency parser then we get the best performance which is even better than the basic setup.

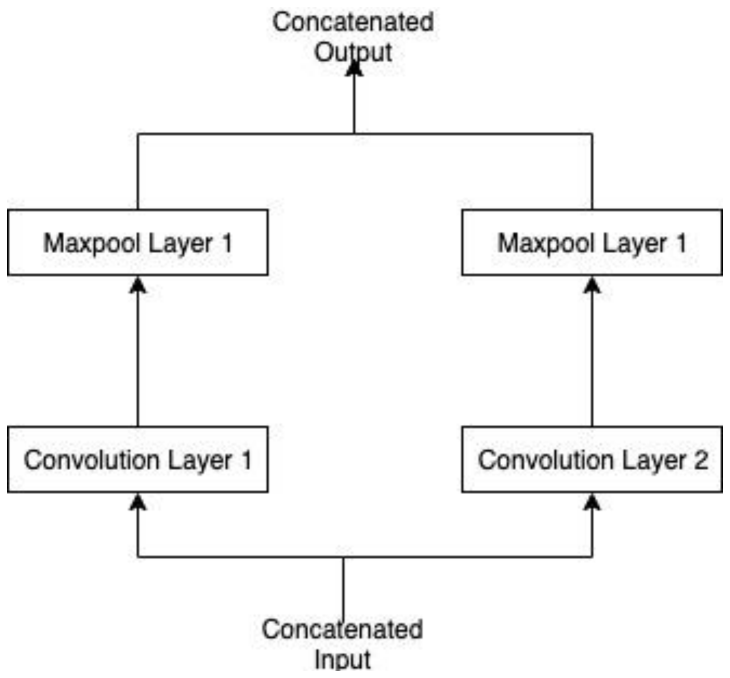
SECTION 3: ADVANCED MODEL IMPLEMENTATION

For my advanced model, I referred the second paper given on piazza
https://arxiv.org/pdf/1605.07333.pdf

In this paper they have described the CNN structure as:
CNNs perform a discrete convolution on an input matrix with a set of different filters. For NLP tasks, the input matrix represents a sentence: Each column of the matrix stores the word embedding of the corresponding word. By applying a filter with a width of, e.g., three columns, three neighboring words (trigram) are convolved. Afterwards, the results of the convolution are pooled. Following Collobert et al. (2011), we perform max-pooling which extracts the maximum value for each filter and, thus, the most informative n-gram for the following steps. Finally, the resulting values are concatenated and used for classifying the relation expressed in the sentence.

We have 3 basic layers in this system:
(1)2sets of 1D Convolution layer with activation function as ReLU
(2)2 sets of Maxpool1D layer to perform max-pooling limited by the length of the maximum sentence.

The basic structure of the model is:
(1)We pass the concatenated word embeddings and POS tags embeddings as the input to the first convolution layer
(2)We pass the same input to the second convolution layer
(3)We pass the output of the first step to the first maxpooling layer
(4)We pass the output of the second step to the second maxpooling layer
(5)We then concatenate the two outputs from above and send them to the decoder as in the basic model.



The CNN model in the paper was the baseline model which got an accuracy of 73.0. Our model here only considers word embeddings and POS tags information and gives an F1 score of 0.5866 after running it for 10 epochs.

3 variations of advanced model:
(1)Filters in convolution layer as 32
Val loss for epoch: 1.9439 Val F1 score: 0.5866
(2)Filters in convolution layer as 64
Val loss for epoch: 2.1391 Val F1 score: 0.58
(3)With Dropout at rate=0.5
Val loss for epoch: 1.6871 Val F1 score: 0.5838