Stony Brook University
Department Of Computer Science

CSE 538 Natural Language Processing - Fall 2019
Instructor: Niranjan Balasubramanian

Assignment 4

Madhusmita Dash (SBU ID: **112715430**)

Basic Model:

a. GRU implementation:

The Bi-GRU attention model is implemented in three steps:

init(): The model is initialized with a bidirectional GRU layer to capture the future and past contexts in the representation. The number of hidden layers is passed as a parameter.

call(): We lookup the word embeddings of the entity annotated sentence. We also get the POS embeddings of the words. The shortest dependency path is chosen in the data.py if we consider the dependency path between the two entities. We concat the word embeddings and POS embeddings to form [batch_size,sequence length, 2*embedding_size] vector and pass it to the model with the input mask. The input mask is applied to the input to consider only the non-zero tokens.

attention(): To compute the attention, we first pass the Bi-GRU output through a tanh non-linearity. We then take the tensordot of the weight matrix [hidden_size*2,1] and the output of tanh, take the softmax to get the attention score of the tokens. Finally we take the attention-weighted average of the tokens to get the sentence representation. The sentence representation is returned after applying the tanh function.

Decoder(): Pass the sentence representation through a dense layer and return the logits.

We add the L2 regularization to avoid overfitting.

Experiments:

Configuration	F1 score
Word only	0.5211
Word+POS	0.5136
Word+Dep	0.5868

b. Observations of GRU experiment 1 (Word only):

The observed F1 score is better than using random initialization of word embeddings because the pre-trained word embeddings already capture some features of the words.

b. Observations of GRU experiment 1 (Word + POS):

POS tags of the words gave lower accuracy than the word only experiment. This could be because the POS tags of each word in the sentence (instead of a few keywords which actually determine the relation) is considered which might not be contributing anything to the relation between the two entities. Considering the POS tags of all the tokens and concatenating them thus does not add any relevant information required for the relation extraction.

b. Observations of GRU experiment 1 (Word + Dep):

The shortest dependency path between two entities in a sentence captures the most important context required to get the relation between the two entities in a condensed representation. Also, the inverse relation between two local substructures is also learnt during the backward pass in a bidirectional GRU. Thus this combination gives the best F1 score.

Motivation of advanced model choice:

For the advanced model, I tried out several combinations like:
a. BiLSTM (1 layer)
b. BiLSTM (2-3 layers)
c. BiGRU (2-3 layers) with max-pooling

As mentioned in this paper, the authors have proposed that using max-pooling technique over the learned representation of a bidirectional sequential network is useful. This is because in a LSTM/GRU the long term information is lost due to accumulation of the representation. Instead a max-pooling idea (like in the filters of CNN) will capture the most important features in each embedding dimension. I also tried mean-pooling but the max-pooling performed better because even though it did not capture the whole sentence, only a few keywords generally trigger the relation between two entities which is captured well by the max-pooling. Also, using multiple GRU layers gave better results than LSTMs. The multiple layers gave better results than single layer because multiple layers learn different features from the sentence.

Experiments:

Configuration	F1 score
Word+Dep+BiGRU (2)+Maxpool	0.6036

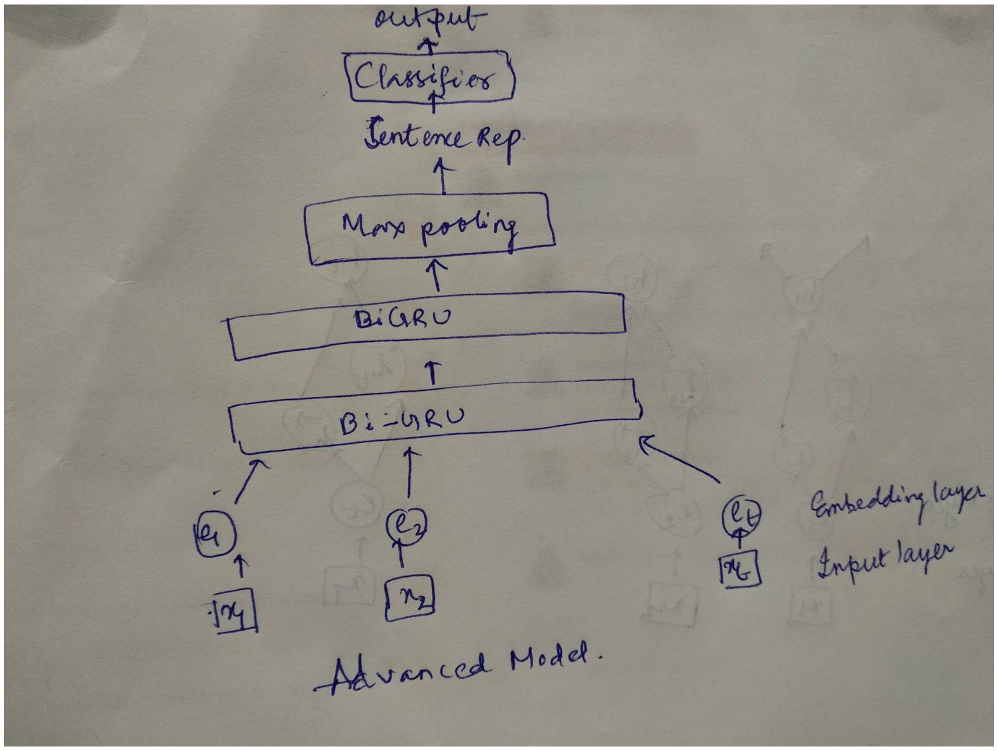

Fig. Advanced model

References:

1. Relation Classification via Recurrent Neural Network