NLP Assignment 4: Relation Extraction

Utkarsh Garg

1

1 Bidirectional GRU with Attention

1.1 GRU Implementation:

The GRU is like a long short-term memory (LSTM) with forget gate but has fewer parameters than LSTM, as it lacks an output gate. In this assignment, we aim to extract relations from the given sentence corpus drawn from English texts(the data released for SemEval's relation extraction task from 2010.). We employ BiGRU to access both past and future context information and learn patterns of relations from raw text data.

This model will leverage an interesting feature called shortest dependency path(SDP) between two nominals in order to aid Relation-Extraction task.

Architecture of BiGRU with Attention model:

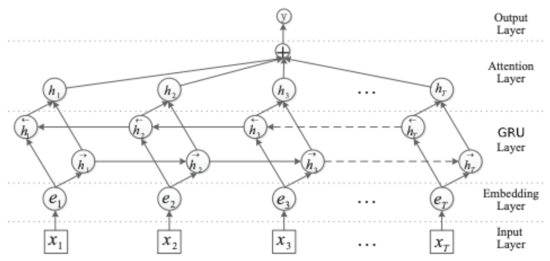

Fig:1.1: BiDirectional GRU Model with Attention.

The model contains following components:

- It takes as input the input sentence ,plus the embedding layer ,mapping each word into a low-dimensional vector.
- This is followed by our custom-defined BiGRU layer to get high-level features from step (1).
- The output from GRU is passed to custom-defined Attention layer, that merges word-level features from each time step into a sentence-level vector.
- Finally the attention layer output is fed to Dense(Feedforward softmax layer), that generates the probability of each sentence belonging to a Relation class.

1.2 BIGRU

We initialize our model by defining the __init__ constructor. We initialise several parameters like vocab_size , embedding_dimensions, hidden_size, and finally our GRU cell with dimenstions of hidden_size using tf.keras.layers.GRU . Since ours is a Bidirectional implementation, we also define a bidirectional layer using the inbuilt keras.layers library.

For our call method, we lookup our word_embeddings and pos_embeddings , inside embeddings dictionary already provided to us. The 3rd feature with Shortest dependency path is enabled by default , as defined in data.py .

With the embeddings of Word and POS features(3-Dimensional) in hand, we then concat them along the axis=2, resulting in extended word_pos tensor of shape (10,5,200).

Further we define a mask to allow for only real word tokens(ignore padding tokens) in each sentence from the current batch.

Thats it for preprocessing, we further call our biGRU with the masked_tokens for inputs , in training mode (set to true by default). This step generates the RNN_outputs.

output from GRU is fed to our custom-attention layer, declared as follows:

1.3 Attention

The rnn_outputs from GRU layer are stored in the form of a matrix consisting of output vectors of total length T (sentence length) ,from each time step. This is then passed through tanh() to generate a new matrix M.

Omegas is trained parameter vector predefined in our init call. We take a dot product of M with omegas to create (10,5,1) shaped tensor.

This is followed by softmax() over the computed dot-product.

Finally, the representation 'r' of the sentence is formed by a weighted sum of these output vectors.

Below are the set of equations, for computing Attention layer outputs:

$$M = \tanh(H)$$

$$\alpha = softmax\left(w^T M\right)$$

$$r = H\alpha^T$$

$$h^* = \tanh(r)$$

1.4 Regularization:

We define a L2 regularization constant lambda = 10^{-5} . Further , we cumulatively add the l2_loss across each trainable-parameter defined in θ (model.trainable_variables) to our loss function,multiplied by lambda ,defined earlier. This gives us the required Regularization factor .

1.5 Experiments with Basic Att-BiGRU Model:

Model	Epochs 5	Epoch 10	Epochs 15	
Basic(All 3)	0.5483			
Word embeddings only	0.5211			
Word+POS	0.5439			
Word+Dependency Path	0.5868			
Advanced Model (All 3)	0.5796	0.6063	0.6264	

- **Experiment with Word Embedding Features only:** We witness the accuracy(F1) score of model increases linearly with increase in batch_size. However, the epochs seem to have a positive slope from 5 to 10, which then reduces as we increase epochs from 10 to 15.
- **Experiment with Word + POS Features:** Similar results observed with respect to increasing batch-size and epochs. However, the F1 score shows a lil boost over training BiGRU just on word_embeddings.
- **Experiment with Word + Dependency Structure:** F1-score increase is proportional to the increase in epochs and batch_size . This is evident as a larger batch and more epochs can help train the model in a better way, thereby eliminating chances of overfitting. The score of model when ran with dependency path is seemingly the **highest among the 4 models**. This is evident from the fact that the dependency path provides the shortest path spanning the 2 nominals ,provided in a given sentence, and hence helps reduce noise in the training data.

2 Advanced Model:

2.1 Justification/Motivation

For my Advanced Model, I have implemented an ensemble of Convolutional Neural Network with BiLSTM as part of learning sentence embeddings for training and applying it on SemEval Relation Extraction task.

Here's the architecture for my CNN model:

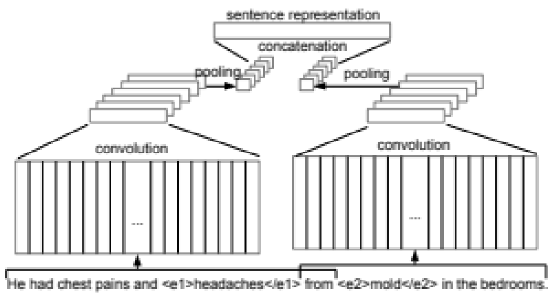
sentence representation
concatenation
pooling
pooling
convolution
convolution
He had chest pains and <e1>headaches</e1> from <e2>mold</e2> in the bedrooms.

Fig:1.1: CNN with Max Pooling Layers.

My work is inspired by [1], wherein the publisher has employed 2 custom CNNs on the base input sentences to extract features on them. The output of each of these CNN's is fed to Max-Pooling layers with pool_size= 2, stride =2 resulting in coarse representations of the learned features, when concatted together,to generate a sentence representation.

For the ensemble, I also utilised BiLSTM model on the original word_embeddings , followed by attention(same as one employed in Basic Model,defined earlier). The output from both the CNN as well as Att-BiLSTM is then merged together and flattened out so as to recover the 2-D output , which we finally pass through Dense feed forward layers to obtain logits for classification .

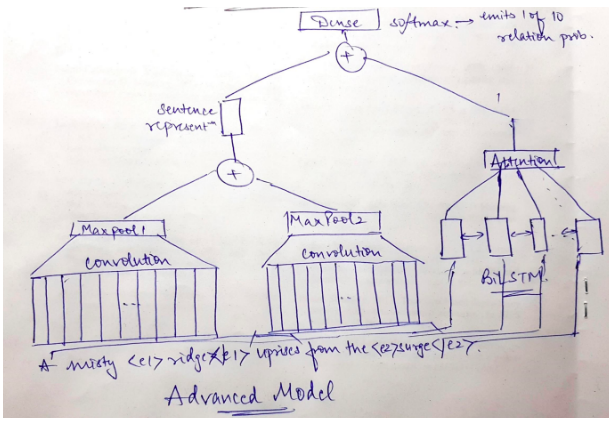
Dense
softmax → emits 1 of 10 relation prob.
Sentence represent^n
Attention
Maxpool1
MaxPool2
convolution
convolution
BiLSTM
A misty <e1>ridge</e1> uprises from the <e2>surge</e2>.
Advanced Model

Fig:1.1: BiDirectional GRU Model with Attention.

Full Architecture described in the README (for Advanced Model)

3 Conclusion:

We see that the advanced Model (ensemble of CNN plus BiLSTM) performs better than the Basic BiGRU Model. The performance of the BiGRU model varies on increasing epochs , however with Advanced Ensemble Model we see a linear increase upto 15 epochs, with F1 score ranging from 0.57-0.63.

4 References:

[1] Combining Recurrent and Convolutional Neural Networks for Relation Classification : Ngoc Thang Vu1, and Heike Adel and Pankaj Gupta and Hinrich Schutze