Relation Classification in Natural Language Processing

GRU with Attention

- **GRU Implementation:**
  - **Forward GRU Layer -**
    This layer is used to calculate the forward representation of the sentence. Every forward layer returns the sequences of all previous layer representations.
  - **Backward GRU Layer -**
    In this GRU layer, to get backward representation, go_backwards flag is set True which reverses whatever sequence that you give to the GRU.
  - **Bidirectional Layer -**
    - This layer takes both forward and backward layer representations to create a concatenated sequence which has both forward and backward sequence of the input given to this layer. The GRU is implemented with the help of this layer which takes a sequence as input.
    - Word and POS (Parts of Speech) embeddings which already have the dependency structure, are extracted using embedding_lookup from inputs and pos_inputs respectively. Before feeding input to this layer, masking is performed so that the padded tokens (to maintain a same sequence length) are avoided in calculation of sequence representation.
    - This model output is fed to the attention function (explained in detail below). Output of the attention function is fed to a classifier layer defined as decoder layer which basically classifies each sequence among the 19 relations. Relations are taken as defined in [1]
    - Attention and regularization loss are calculated as mentioned in [2]. Logits (output of the classifier layer) is used to calculate the regularization loss using the model trainable variables with lambda value 10^{-5}.

- **Attention Implementation:**
  - Output of the Bidirectional GRU is fed to the attention function.
  - Firstly, the rnn output is converted using a non-linear transform (i.e. tanh), then the result is multiplied using tensordot with a hyperparameter of the model i.e. omega.
  - Now, to calculate the attention of one specific word wrt to the entire sequence, I have calculated softmax probabilities. Secondly, I have multiplied the softmax probabilities with the input to attention function. Then I have calculated the weighted sum of the result obtained. To get final representation of the sequence, non-linear transformation of weighted sum is done.

- **F1 score:** This Bidirectional GRU yielded f1 score of 0.57 having word, parts-of-speech and dependency structure.

GRU Experiment 1

Observation:

In this experiment, I have taken **only word embeddings** in training our model. Therefore, the f1 score comes least compared to other experiments. This shows that omitting dependency info i.e shortest path and pos tags results in low f1 score since they provide external information regarding any sequence.

Epochs	Train Loss	Validation Loss	F1 Score
Epoch 1	2.1421	2.6192	0.3692
Epoch 5	0.849	1.9361	0.52

GRU Experiment 2

Observation:

In this experiment, I have taken **word embeddings and pos tags** in training our model. Therefore, the f1 score comes less compared to the f1 score when we trained our model using word, pos embeddings and dependency structure. This shows that omitting dependency info i.e shortest path results in low f1 score.

Epochs	Train Loss	Validation Loss	F1 Score
Epoch 1	4.5182	3.7413	0.3451
Epoch 5	0.8521	1.8027	0.5420

GRU Experiment-3

Observation:

In this experiment, I have taken **word embeddings and dependency info** in training our model. Contrastingly without pos tags, the f1 score comes out to **best** among all configurations. Since adding pos tags to the sequence led to adding up of irrelevant information and creating a complex sequence representation which the model eventually was not able to understand intuitively and therefore produced incorrect results. (i.e. when pos tags were included)

Epochs	Train Loss	Validation Loss	F1 Score
Epoch 1	2.5026	2.5314	0.3249
Epoch 5	0.6575	1.7612	0.587 (BEST)

Advanced Model

Justification / Modification: [4]

- The model that I have used is basically a connectionist Bi-directional RNNs which outperforms the basic Bi-directional GRU model.
- This connectionist Bi-directional model introduces a cross-entropy and regularization loss for its better results.
- This model is best for situations when we are faced with a sentence classification task since it combines the output of all hidden layer representations to obtain the final decision.
- In contrast to our basic Bi-GRU model, this model is specifically used for sentence modelling, i.e. it predicts an output vector i.e. the final sequence representation only after going through the entire sentence and not only after processing each word. This is why this model is considered to be the best for relation extraction.
- This model takes the embeddings of the succeeding and preceding words along with the centre word at any time step t since words other than centre word provide more contextual information. This is why I chose this model over a traditional Bi-directional GRU model since it introduces the concept of including contextually important information also.
- This RNN model incorporates both history vector as well as future vector, in other words, includes more information for the word at time step t for better results.

Implementation:

- To get a brief and high-level view of the model implemented, you can refer to the README file.
- Otherwise, you can refer to the connectionist RNN model in the paper [3]. This paper has defined two neural architectures but I have implemented RNN individually.

Experiments:

In this experiment I used **word embeddings and dependency info** to train the model for **20 epochs** which uses **100D pretrained** glove embeddings. (Table 1)

Epochs	Train Loss	Validation Loss	F1 Score
Epoch 1	5.87	2.6964	0.3617
Epoch 20	0.3886	1.23	0.590

Table 1

In this experiment I used **word & pos embeddings and dependency info** to train the model for **20 epochs** which uses **100D pretrained** glove embeddings. (Table 2)

Epochs	Train Loss	Validation Loss	F1 Score
Epoch 1	7.0268	2.837	0.2781
Epoch 20	0.2154	2.152	0.5888

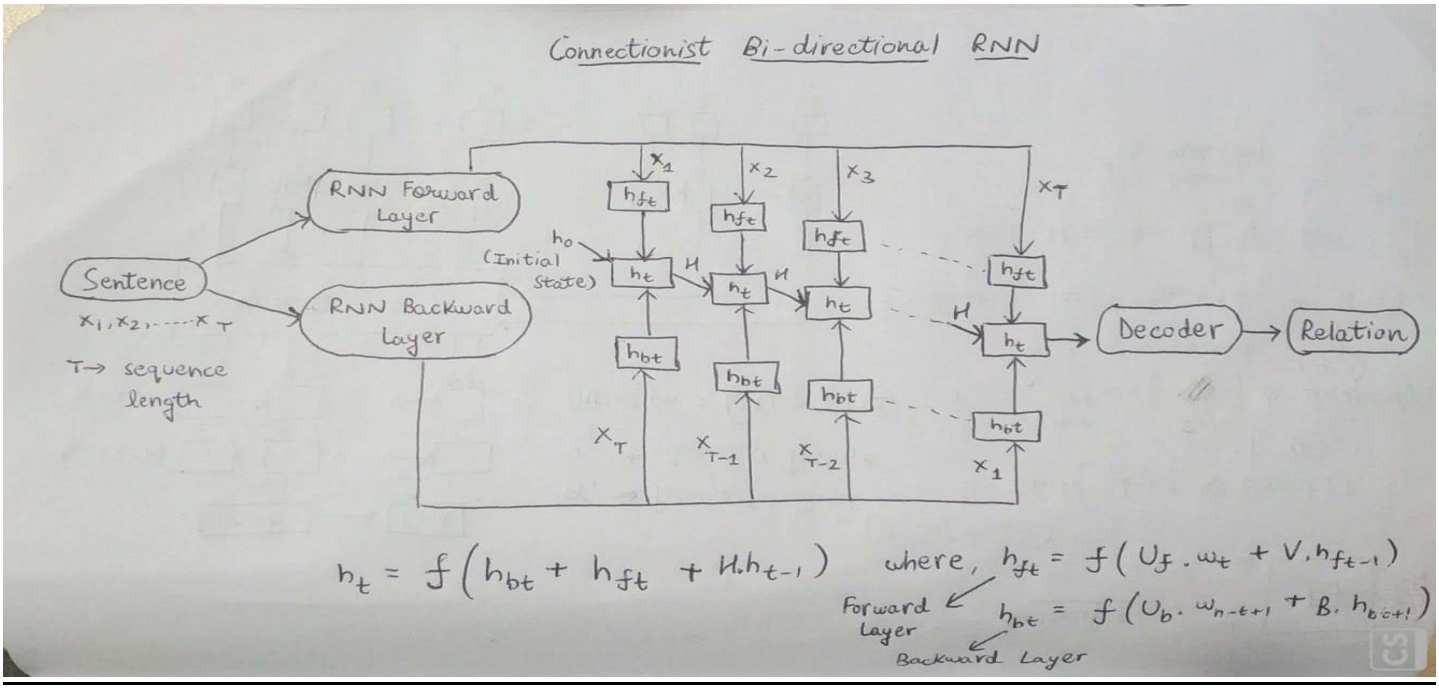
Table 2

In this experiment I used **word and dependency info** to train the model for **20 epochs** which uses **300D pretrained** glove embeddings. (Table 3) This experiment had by far the **best f1 score**.

Epochs	Train Loss	Validation Loss	F1 Score
Epoch 1	5.4319	2.6291	0.4726
Epoch 20	0.3256	2.0145	0.5928 (BEST)

Table 3

Model Architecture:



References:

[1] SemEval-2010 Task 8: Multi-Way Classification of Semantic Relations Between Pairs of Nominals

[2] Attention-Based Bidirectional Long Short-Term Memory Networks for Relation Classification

[3] Combining Recurrent and Convolutional Neural Networks for Relation Classification

[4] Discussed advanced model ideas with Abhijat Shrivatava (112584928)