COMP6714 - Project

Identifying Hypernyms and Hyponyms

Task 1

# IMPLEMENTATION

Inputs: Golden list

Predict list

Output: F1 score

Method:

Given, golden and predict lists are of same dimensions (i.e. A x B).

Firstly, look at the tokens of first sentence for both lists.

The function extract\_tokens (List) reads the token list finds the starting token i.e. “B-TAR” or “B-HYP” (“B-\*” tokens). Then it checks if there are any “I-\*” tokens following to get the position of the next “O” token. The positions of the “B-\*” token and the last “I-\*” token is recorded as a tuple (token, B-\* token position, Next “O” token position) in a set. We use set to facilitate easy computation of the intersection.

The intersection of the two positional lists is then computed. The intersection value gives the True positives. The length of the positional list of the predicted tokens minus the intersection gives the false positives (the excess tokens identified by predictor). The length of the positional list of the golden tokens minus the intersection gives the false negatives (the tokens not found by predictor).

The true positives, false positives and false negatives for all sentences are aggregated for all comparison (as above) of the sentences in the golden and predicted list. Once the lists are exhausted the recall and precision values can be computed to finally determine the required F1 score (using formulae listed below).

Where,

And,

Task 2:

# IMPLEMENTATION

The original form of the Long Short Term Memory (LSTM) Cell updates the cell state by the following equation.

The variation to use in task 2 involves changing the input gate activation function to , so it is dependent on the forget gate activation function.

We have implemented this by modifying the LSTMCell function in torch (torch.nn.\_functions.rnn.LSTMCell) which can be found in the torch GitHub at the following address: <https://github.com/pytorch/pytorch/blob/c62490bf597ec93f308a8b0108522aa9b40701d9/torch/nn/_functions/rnn.py#L23>

The modification was to alter the input gate function to 1 – forgetgate.

Task 3

# IMPLEMENTATION

PERFORMANCE

Performance is compared against the base training case which uses the most recent model generated at the end of each epoch during training on the development set. The results from this method shows a gradual decrease in the loss although it does tend to fluctuate up and down every now and then.

# USING BEST F1 SCORE

When using the best F1 score we got the same result as the base case performance. This is most like because the F1 score is also improving with each epoch so the most recent model tends to be the model with the best F1 score.

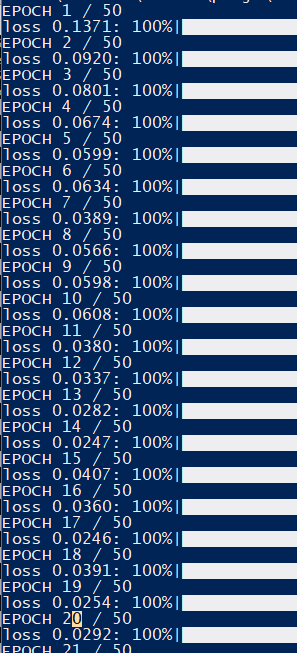
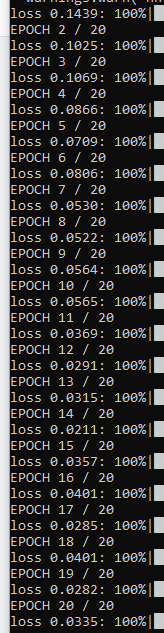
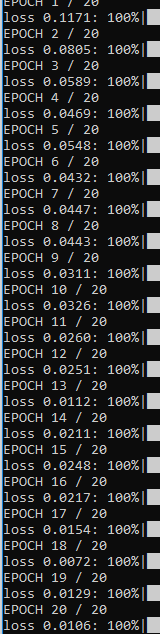
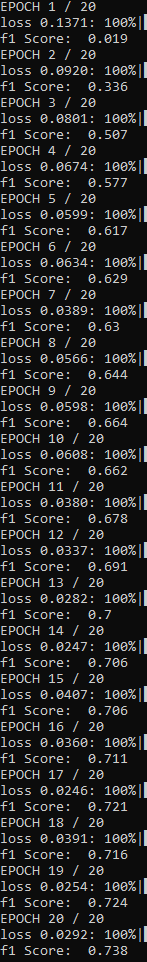
# USING RE-IMPLEMENTED LSTM CELL

The re-implemented LSTM Cell method tended to generate a worse performing model with higher losses than for the base case. The fluctuation going up and down at each epoch were more prominent too.

# ADDING CHAR BI-LSTM LAYER

This method was the best in terms of performance although the model took a significantly longer time to learn. The model generated through this method converged quicker with larger drops in the loss.

The results of the training performance for each case are shown below.

Base Case New LSTM Cell Char Bi-LSTM Best F1 Score