Criteria

* How do you implement evaluate()?
* How does Modification 1 (i.e., storing model with best performance on the development set) affect the performance?
* How do you implement new\_LSTMCell()?
* How does Modification 2 (i.e., re-implemented LSTM cell) affect the performance?
* How do you implement get\_char\_sequence()?
* How does Modification 3 (i.e., adding Char BiLSTM layer) affect the performance?

Project - Identifying Hypernyms and Hyponyms

Tags: TAR corresponds to hyponyms and HYP corresponds to hypernyms.

Task 1:

IMPLEMENTATION

Implement :

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,

PERFORMANCE

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.

PERFORMANCE