Kruthi Channamallikarjuna (channama@usc.edu)

Sequence To Sequence Bi-LSTM For POS Tagging

1. MODEL

To begin with, I implemented the basic RNN network model (with the hint given in the assignment document) which after fine tuning of parameters gave me an accuracy of 91.7% for Japanese and 93.2% for Italian. As a normal RNN network contains a single neural network layer, it makes it impossible to learn to connect long information known as Long-term dependencies (LTD). So, I chose to implement a Bi-LSTM model to solve the LTD issue by remembering information for a longer time than RNN.
Following is the flow of implementation:

- Data Preprocessing
    - Word Embeddings: Assigning each word and it's POS tag with term index and tag index.
    - The data should be transformed for each sequence to have same length by padding smaller sentences with 0, this is done in BuildMatrices function.
- Sequence Model
    - Embedding:
      An Embedding layer of 150 dimension vectors for each term is initialized. They are trained through back propagation.
    - Bi – LSTM:
      The Bi LSTM layer is inputted with the output of the embedding layer, in which each LSTM cell is initialized with 75 units .
      A Dropout wrapper with output keep probability of 0.9 is applied over these cells to fine tune the model.
      The forward and backward outputs of the LSTM are concatenated and the output is reshaped to get the logits of desired shape to output number of tags.
- Training
  The training involved a learning rate of 0.01 with Adam Optimizer and a sequence to sequence loss function.

2. FINE TUNING WITH HYPERPARAMETERS

I started training the models with a very low learning rate of 0.001, with low embedding sizes as much as 30 and cell units initialized to 50.

Even though the number of iterations was more with the given train time, the model did not show good accuracy. Adding more dimensions to the embedding vector and increasing the cell units helped get more accurate results for both the languages. An increased learning rate in steps helped me understand accuracy improved with it.

After tuning the model with different values, following are the hyper parameters used:

- Learning Rate: 0.01
- Optimizer: Adam Optimizer
- Loss Function: seq2seq.sequence_loss
- Batch Size: 40
- Maximum Train Time: 11 minutes

This helped me improve the accuracy of Japanese to 95% and Italian to 95.4% and improve the performance for secret language as well.