CSCI544: Homework Assignment No 4

Name: Abhishek Mishra
USC Id: 3432233339

Task 1: Simple Bidirectional LSTM model (40points)

First read all the data files (train, dev/ validation and test). Create a list of list where the list first dimension is number of sentences in the dataset and second dimension states the number of words present in each sentence.

BiLSTM network architecture:

Embedding Layer: vocab size * embedding_dim = 30292 * 100
LSTM Layer: embedding_dim = 100, hidden_dim = 256, num_lstm_layers = 1, bidirectional = True
Dropout: 0.33
Linear Layer: input_dim= 512 (hidden_dim * 2(bidirectional)), linear_out_dim = 128
Elu Layer: alpha = 0.01
Classifier Layer: input_dim = 128, out_dim = 9 (Number of classes/NER – tags)

Created a custom data loader to retrieve sentences for training the model. Also used a custom collator function to manage each batch. Here as we know the sentences are of variable size that's why we need to pad the sentence with the max length of sentence present in that batch. But we also need to tell the network that we don't want to calculate loss for padding. To handle that we used pad_packed_sequence which takes a list containing lengths of each sentence. So now network knows for which index value padding starts and not to calculate the gradients.

Created a dictionary which stores all the words present in all the datasets (train, dev, test) as the key and value as index in the dictionary. We don't need to insert <UNK> as the vocab is generated using all the dataset's words. Same goes for the label/NER tags dictionary.

We also used the class weights. As 'O' NER-tag was overpowering the model was predicting every word as 'O' initially. We call such situation imbalanced class. To mitigate such issues, we used class weights. To calculate the class weights, divide the total number of labels by their respective class frequency. Also, this score is multiplied by a number which is a hyperparameter. Now we take log of that score if the value if less than 1.0 we give those classes 1.0 and rest similar to score.

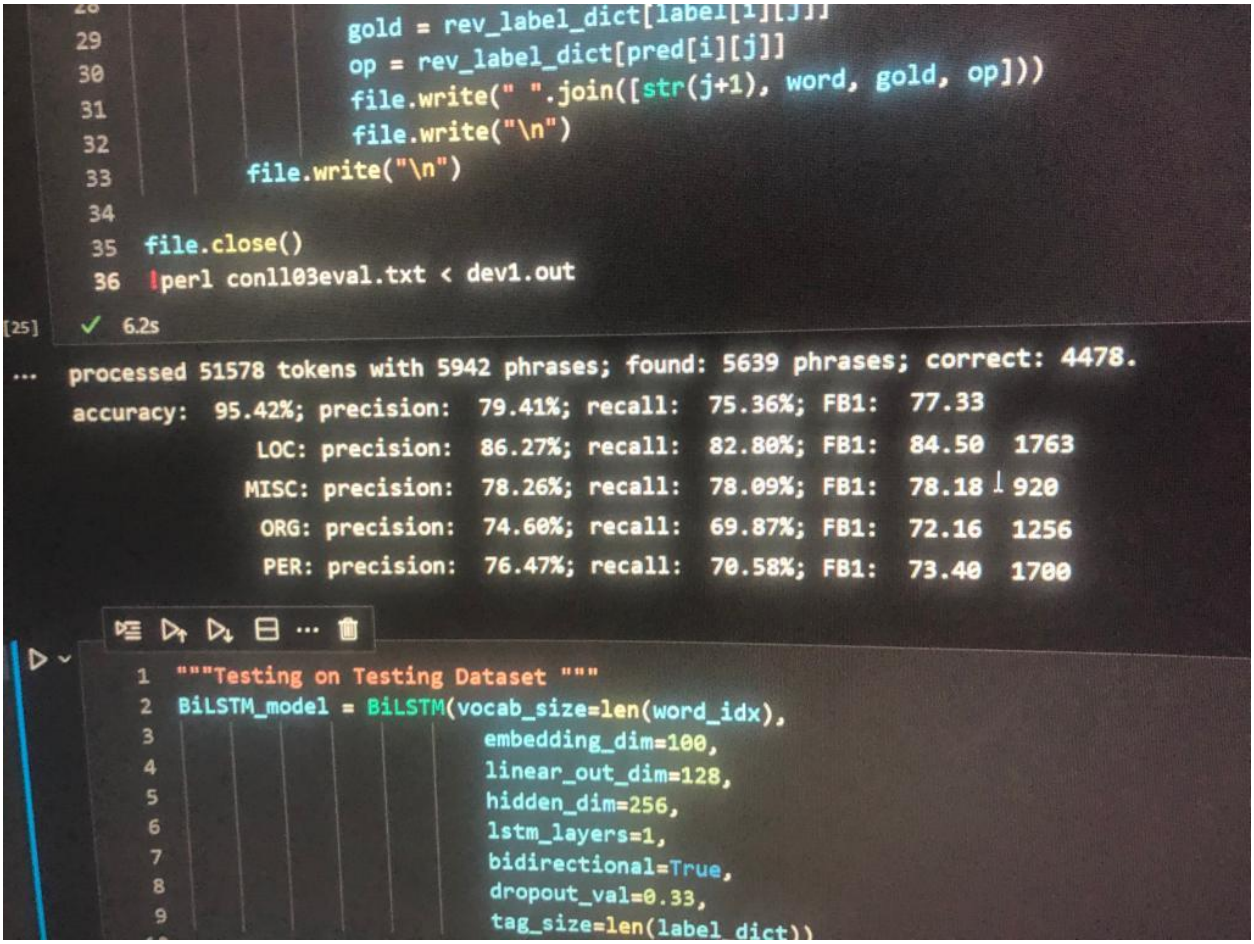
We train our BiLSTM model for 200 epochs with Cross Entropy loss and SGD optimizer. We pass our class weights to Loss Function. In SGD we use learning rate equal to 0.1 and momentum of 0.9.

Now load the validation and testing dataset and test on the trained models. For convenience we have provided the models which gave us the best results.

For this Task
Accuracy: 95.42%
Precision: 79.41%
Recall: 75.36%
F1: 77.33

HyperParameters:

- Embedding Dimension = 100
- Hidden Dimension = 256
- Linear Output Dimension = 128
- Bidirectional = True
- Dropout = 0.33
- Number of LSTM layers = 1
- Batch Size = 4
- Learning Rate = 0.1
- Momentum = 0.9
- Epochs = 200



Task 2: Using GloVe word embeddings (60points)

For this task we need to download Glove word embeddings with 100 dimensions. Load that file and create an embedding matrix using the Vocab Dictionary we created in the Task -1. As we know glove word embedding doesn't have upper case words therefore to handle upper case words, we added the 5e-3 to the values for lower case words. The reason why because even they are upper case words, they are similar to the lower-case words.

After creating the embedding matrix, we need to assign this matrix to the embedding layer that's why we created a different class which take embedding matrix as parameter and load the embedding matrix in the embedding layer, rest the architecture of the network is same as in the task 1.

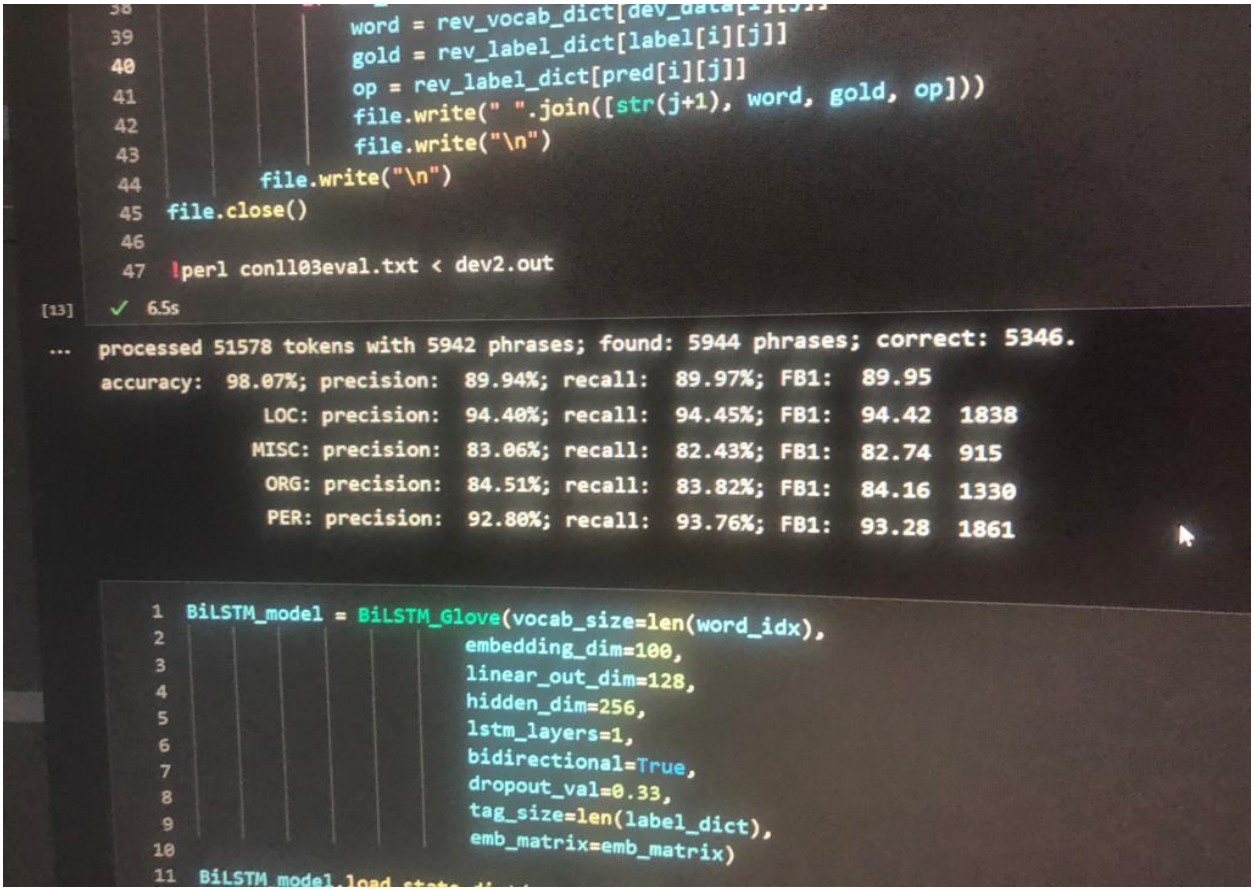
For training this network we use same hyper parameters except the batch size which is 8. We train our BiLSTM_glove model for 50 epochs.

Now test the trained model on validation dataset and testing dataset.

For Task-2:
Accuracy: 98.07%
Precision: 89.94%
Recall: 89.97%
F1: 89.95

Hyper Parameters:

- Embedding Dimension = 100
- Hidden Dimension = 256
- Linear Output Dimension = 128
- Bidirectional = True
- Dropout = 0.33
- Number of LSTM layers = 1
- Batch Size = 8
- Learning Rate = 0.1
- Momentum = 0.9
- Epochs = 50



```
                word = rev_vocab_dict[dev_data[i][j]]
                gold = rev_label_dict[label[i][j]]
                op = rev_label_dict[pred[i][j]]
                file.write(" ".join([str(j+1), word, gold, op]))
                file.write("\n")
            file.write("\n")
file.close()

!perl conll03eval.txt < dev2.out
```

[13] ✓ 6.5s

```
processed 51578 tokens with 5942 phrases; found: 5944 phrases; correct: 5346.
accuracy:  98.07%; precision:  89.94%; recall:  89.97%; FB1:  89.95
              LOC: precision:  94.40%; recall:  94.45%; FB1:  94.42  1838
             MISC: precision:  83.06%; recall:  82.43%; FB1:  82.74  915
              ORG: precision:  84.51%; recall:  83.82%; FB1:  84.16  1330
              PER: precision:  92.80%; recall:  93.76%; FB1:  93.28  1861
```

```
BiLSTM_model = BiLSTM_Glove(vocab_size=len(word_idx),
                            embedding_dim=100,
                            linear_out_dim=128,
                            hidden_dim=256,
                            lstm_layers=1,
                            bidirectional=True,
                            dropout_val=0.33,
                            tag_size=len(label_dict),
                            emb_matrix=emb_matrix)
BiLSTM_model.load_state_dict(
```