NLP Assignment 2 Report

Mulukutla Krishna Praneet
2020113010

I have used a bi-directional LSTM for this task. Since the POS tag of a word can depend on context on both sides of the word, I determined that using a bi-directional LSTM, which can have a long memory if required, and also context from both sides of the word is appropriate.

For a typical model trained for around 50 epochs, the **Accuracy**, **Precision**, **Recall** and **F1 score** for the different classes of tags for training / validation / test sets are given here :

```
Evaluating on the training set:
Loss: 0.0059
Accuracy: 0.9983
Precision, Recall and F1 Score:
              precision    recall  f1-score   support

       <unk>       1.00      1.00      1.00     34777
        NOUN       1.00      1.00      1.00     23676
       PUNCT       1.00      1.00      1.00     23078
        VERB       1.00      1.00      1.00     18574
        PRON       1.00      1.00      1.00     17635
         ADP       1.00      1.00      1.00     16283
         DET       0.99      1.00      1.00     12946
       PROPN       1.00      1.00      1.00     12541
       <eos>       1.00      1.00      1.00     12476
         ADJ       1.00      1.00      1.00     12342
         AUX       1.00      1.00      1.00     10543
         ADV       1.00      1.00      1.00      6706
       CCONJ       1.00      1.00      1.00      5567
        PART       1.00      1.00      1.00      3999
         NUM       1.00      1.00      1.00      3843
       SCONJ       1.00      0.99      0.99       847
           X       0.99      1.00      0.99       688
        INTJ       1.00      0.99      1.00       599

    accuracy                           1.00    217120
```

```
   macro avg       1.00      1.00      1.00    217120
weighted avg       1.00      1.00      1.00    217120
```

```
Evaluating on the validation set:
Loss: 0.4186
Accuracy: 0.9256
Precision, Recall and F1 Score:
              precision    recall  f1-score   support

       <unk>       0.86      0.90      0.88      4194
        NOUN       0.99      0.99      0.99      3082
       PUNCT       0.94      0.91      0.92      2763
        VERB       0.99      0.99      0.99      2217
        PRON       0.94      0.98      0.96      2023
         ADP       0.98      0.99      0.99      1894
         DET       0.78      0.70      0.74      1878
       PROPN       1.00      1.00      1.00      2000
       <eos>       0.85      0.88      0.87      1787
         ADJ       0.96      0.99      0.97      1509
         AUX       0.92      0.88      0.90      1264
         ADV       1.00      0.99      0.99       779
       CCONJ       0.98      0.97      0.97       630
        PART       0.80      0.85      0.82       378
         NUM       0.91      0.87      0.89       402
       SCONJ       0.48      0.38      0.43       154
           X       0.75      0.72      0.73       115
        INTJ       0.80      0.60      0.68        67

    accuracy                           0.93     27136
   macro avg       0.88      0.87      0.87     27136
weighted avg       0.92      0.93      0.92     27136
```

```
Evaluating on the test set:
Loss: 0.3847
Accuracy: 0.9286
Precision, Recall and F1 Score:
              precision    recall  f1-score   support

       <unk>       0.87      0.89      0.88      4131
        NOUN       0.99      0.99      0.99      3105
       PUNCT       0.94      0.92      0.93      2653
        VERB       0.99      0.99      0.99      2158
        PRON       0.95      0.97      0.96      2018
         ADP       0.99      0.99      0.99      1896
```

```
          DET       0.78      0.74      0.76      2076
        PROPN       1.00      1.00      1.00      2076
        <eos>       0.85      0.89      0.87      1693
          ADJ       0.96      0.99      0.97      1495
          AUX       0.92      0.90      0.91      1225
          ADV       1.00      0.99      0.99       739
        CCONJ       0.96      0.97      0.97       629
         PART       0.83      0.80      0.81       536
          NUM       0.90      0.87      0.88       387
        SCONJ       0.44      0.38      0.41       139
            X       0.84      0.86      0.85       120
         INTJ       0.83      0.79      0.81        92

     accuracy                           0.93     27168
    macro avg       0.89      0.89      0.89     27168
 weighted avg       0.93      0.93      0.93     27168
```

(Note that the accuracy given at the top is the entire model's accuracy in general. The same quantity is also reported at the end, labelled 'accuracy')

Analysis

Scores

As mentioned earlier, I used bi-directional LSTM for long-term memory as well as context from both sides of the word.

From the scores of the training data, we see that, as expected, the model performs extremely well on the training data (almost 100% accuracy, precision, recall and F1 score). Actually, this may be a small sign of over-fitting.

Note that accuracy, gives a measure of the percentage of correctly classified data-points. Precision is high when the number of false positives is low. Recall is high when the number of false negatives is low. F1 score is the harmonic mean of precision and recall, and is high only when both precision and recall are high.

In the test and validation datasets, we see that model accuracy drops to around 93%, which is still a good score.

Now, we take a look at the weighted average of the precision scores and the recall scores in the test/validation sets. We see that these values are around 93% for both. This means that false positives and false negatives occur in fairly equal amounts, but not often. This is a healthy sign, and subsequently, the F1 score is also agreeably high (around 92%).

Now, if we take a closer look at these scores for individual classes, we observe that the SCONJ (subordinating conjunction) has extremely low scores (around 45%) for all these metrics. Examples of SCONJ are ‘if’, ‘while’, etc. This means that the model is unable to properly classify these words into SCONJ. It might be due to the fact that these words also appear in other classes, and generally in larger numbers. This can cause the model to classify these words incorrectly.

We also see that DET has somewhat low scores (around 75%) for all three metrics. I assume this is due to a similar reason.

We also observe that **open** classes like NOUN, PROPN, and VERB have high scores in all metrics. So we can observe a pattern here. Closed word classes like SCONJ and DET have lower scores, while open word classes like NOUNS have higher scores. As discussed before, the probable reason is the presence of the same words (from the closed word classes) in multiple classes in different contexts, making it harder for the model to learn their behavior.

Hyper-parameters

The **Hyper-parameters** used in the model are:

```
embedding_dim = 100
hidden_dim = 128
num_layers = 2
dropout = 0.2
batch_size = 32
num_epochs = 50
learning_rate = 0.001
```

Also note that the size of the vocabulary is 8866. I created 1-hot vectors using this vocabulary, and passed them to the embedding layer, which condensed them into a 100 sized vector representation.

There exist 2 bi-directional LSTM layers in this model. It is to be noted that increasing the number of layers increased training time, without significant enhancement in the scores obtained.

I also added dropout layers after the embedding layer and between the LSTM layer and the final fully connected layer. This ensures that the model does not over-fit, by dropping some layers.

The dimensions of the hidden layer are set at 128, once again, increasing this size did not fetch significant score improvements for the consequent training-time increase.

The batch size (for parallel computing on the GPU) is set to 32. Increasing batch size did not yield any improvements (sometimes even giving worse scores), while it was definitely faster.

The learning rate is set at 0.001, which is somewhat a standard value. Having a higher learning rate might lead to oscillations around the minimum without convergence.

With 50 epochs, the training data almost reached 100% accuracy, which is why I felt that increasing the number of epochs further might increase over-fitting.

As can be seen, I’ve experimented with the tuning of the hyper-parameters, and have found that this set of parameter values gives a good trade-off between training time and the accuracy scores.