iNLP Assignment-2

Report

Akshit Sharma
2021101029

Feed Forward Neural Network POS Tagger

Dev set metrics on 3 different configurations:

Columns for Confusion matrices: [1: 'ADJ', 2: 'ADP', 3: 'ADV', 4: 'AUX', 5: 'CCONJ', 6: 'DET', 7: 'INTJ', 8: 'NOUN', 9: 'NUM', 10: 'PART', 11: 'PRON', 12: 'PROPN', 13: 'VERB']

1. epochs=10, embedding_size=30, hidden_layer_size=50, learning_rate=0.01, batch_size=5, p=2, s=2, ReLU, 1 hidden layer

```
Accuracy Score on dev set: 0.9435897435897436
F1-Score(micro) on dev set: 0.9435897435897436
F1-Score(macro) on dev set: 0.8536349282747082
Recall(micro) Score on dev set: 0.9435897435897436
Recall(macro) Score on dev set: 0.8306504893793248
Precision(micro) Score on dev set: 0.9435897435897436
Precision(macro) Score on dev set: 0.8921144452986771
Confusion matrix for dev set:
 [[ 168    0    0    0    0    1    0    4    0    1    0    2   50]
 [   0 1382    0    1    0    0    0    0    0    0    0    0   31]
 [   0    0   44    0    0    0    0    1    0    0    0    0   14]
 [   0    0    0  252    0    0    0    0    0    0    0    0   12]
 [   0    0    0    0  105    0    0    0    0    0    0    0    2]
 [   0    0    0    0    0  500    0    0    0    0    7    0   60]
 [   0    0    0    0    0    0   26    0    0    0    0    0    9]
 [   0    0    0    0    0    0    0 1099    1    0    0    4   33]
 [   0    0    0    0    0    0    0    0  111    0    0    0   20]
 [   1    2    0    0    0    0    0    0    0    0    0    0   70]
 [   0    0    0    0    0    1    0    0    0    0  408    0    4]
 [   0    0    0    0    0    0    0    2    0    0    0 1511   38]
 [   0    0    0    1    0    0    0    2    0    0    0    0  650]]
```

2. epochs=10, embedding_size=30, hidden_size=50, learning_rate=0.01 , batch_size=5, p=2, s=2, Tanh, 2 hidden layers (same size)

```
Accuracy Score on dev set: 0.9467571644042232
F1-Score(micro) on dev set: 0.9467571644042232
F1-Score(macro) on dev set: 0.8964707463601359
Recall(micro) Score on dev set: 0.9467571644042232
Recall(macro) Score on dev set: 0.894897492634465
Precision(micro) Score on dev set: 0.9467571644042232
Precision(macro) Score on dev set: 0.8998116637685564
Confusion matrix for dev set:
 [[ 168    4   23    0    0    1    0   13    3    0    0   10    4]
 [   0 1378    0    3    0    8    0    4    0    9    0    2   10]
 [   4    0   33    2    1    3    0    3    3    0    2    0    8]
 [   0    0    1  255    1    0    0    0    0    0    0    0    7]
 [   0    2    0    1  104    0    0    0    0    0    0    0    0]
 [   1   17    2    0    0  541    0    0    1    0    2    0    3]
 [   0    0    0    1    0    0   34    0    0    0    0    0    0]
 [   6    1    1    0    0    3    1 1091    3    0    0   15   16]
 [   0    2    4    0    0    0    0    3  108    0    0    0   14]
 [   2    4    0    3    0    0    0    1    0   63    0    0    0]
 [   0    0    1    1    0    6    0    0    0    0  400    1    4]
 [   3    5    1    1    4    0    0    5   10    0    0 1513    9]
 [   0   41    1    2    0    0    0   14    3    0    1    2  589]]
```

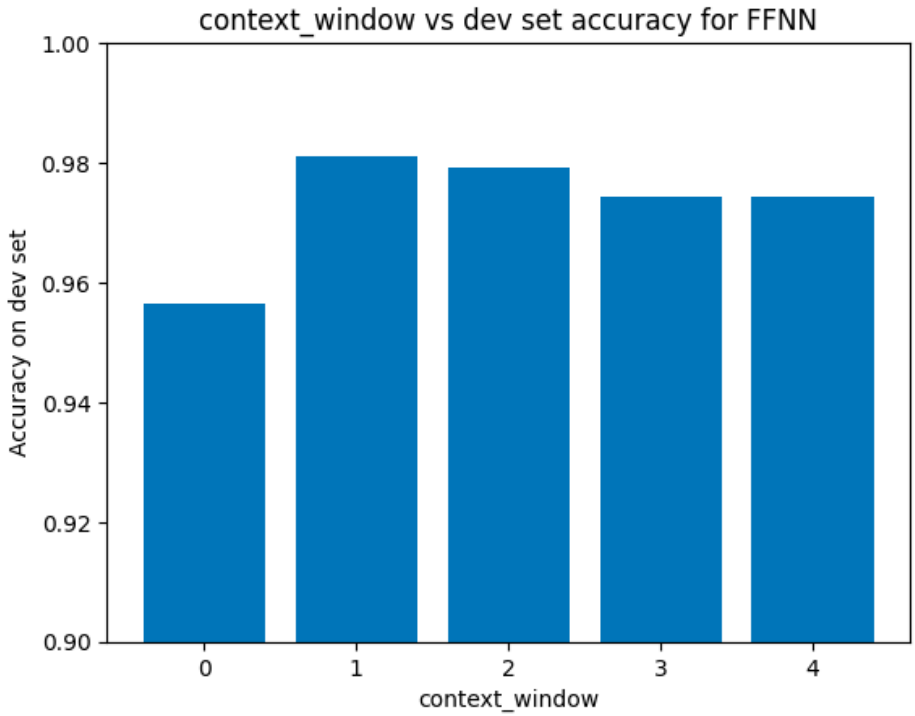
3. epochs=10, embedding_size=20, hidden_size=120, learning_rate=0.01 , batch_size=10, p=4, s=4, ReLu, 1 hidden layer

```
Accuracy Score on dev set: 0.9597285067873303
F1-Score(micro) on dev set: 0.9597285067873303
F1-Score(macro) on dev set: 0.9383967199957396
Recall(micro) Score on dev set: 0.9597285067873303
Recall(macro) Score on dev set: 0.9201097962278777
Precision(micro) Score on dev set: 0.9597285067873303
Precision(macro) Score on dev set: 0.965264128172647
Confusion matrix for dev set:
 [[ 188    0    0    0    0    0    0    8    0    0    0    2   28]
 [   0 1378    0    0    0    0    0    0    0    6   21    0    9]
 [   2    0   44    0    0    0    0    2    0    0    0    0   11]
 [   0    0    0  242    0    0    0    0    0    0    1    0   21]
 [   0    0    0    0  106    0    0    0    0    0    0    0    1]
 [   0    7    0    0    1  534    0    0    0    0   16    0    9]
 [   0    0    0    0    0    0   34    0    0    0    0    0    1]
 [   0    1    0    0    0    0    0 1093    0    0    0    8   35]
 [   0    0    0    0    0    0    0    1  103    0    0    0   27]
 [   0    3    0    0    0    0    0    0    0   64    0    0    6]
 [   0    0    0    0    0    1    0    0    0    0  409    0    3]
 [   0    0    0    0    1    0    0    2    0    0    0 1518   30]
 [   0    0    0    1    0    0    0    2    0    0    0    0  650]]
```

Test set metrics for best accuracy configuration (3rd one) (from above):

```
Accuracy Score on test set: 0.958966565349544
F1-Score(micro) on test set: 0.958966565349544
F1-Score(macro) on test set: 0.9294211099798706
Recall(micro) Score on test set: 0.958966565349544
Recall(macro) Score on test set: 0.9103940064232436
Precision(micro) Score on test set: 0.958966565349544
Precision(macro) Score on test set: 0.9617202131153981
Confusion matrix for test set:
 [[ 199    0    0    0    0    0    0    3    0    0    0    2   16]
 [   0 1389    0    0    0    0    0    0    0    1   30    0   14]
 [   7    0   45    0    0    0    0    0    0    1    0    1   22]
 [   0    0    0  230    0    0    0    1    0    0    2    0   23]
 [   0    0    1    0  106    0    0    0    0    0    0    0    2]
 [   1    1    0    0    0  494    0    0    0    0    3    2   11]
 [   0    0    0    0    0    0   34    0    0    0    0    0    2]
 [   1    1    0    0    0    0    0 1124    1    0    0    1   38]
 [   0    0    0    0    0    0    0    0   94    0    0    1   32]
 [   0    2    0    0    0    0    0    0    0   52    0    0    2]
 [   0    0    0    0    0    2    1    0    1    0  385    0    3]
 [   1    0    0    0    0    0    0    2    0    0    0 1530   34]
 [   0    0    0    0    0    0    0    1    0    0    0    0  628]]
```

Context Window vs Dev set accuracy graph



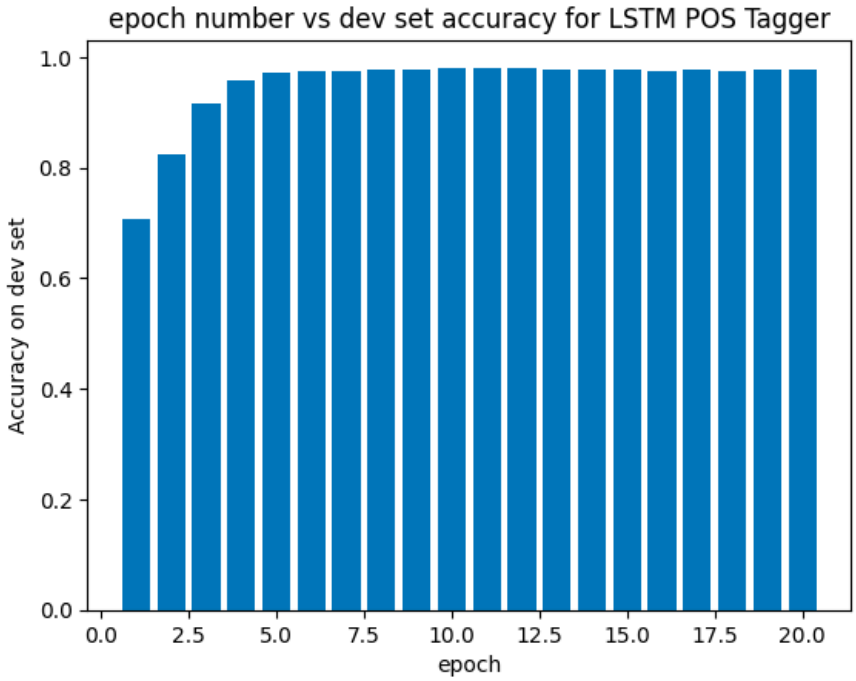
Recurrent Neural Network POS Tagger

Dev set metrics on 3 different configurations:

Columns for Confusion matrices: [0: 'NULL', 1: 'ADJ', 2: 'ADP', 3: 'ADV', 4: 'AUX', 5: 'CCONJ', 6: 'DET', 7: 'INTJ', 8: 'NOUN', 9: 'NUM', 10: 'PART', 11: 'PRON', 12: 'PROPN', 13: 'VERB']

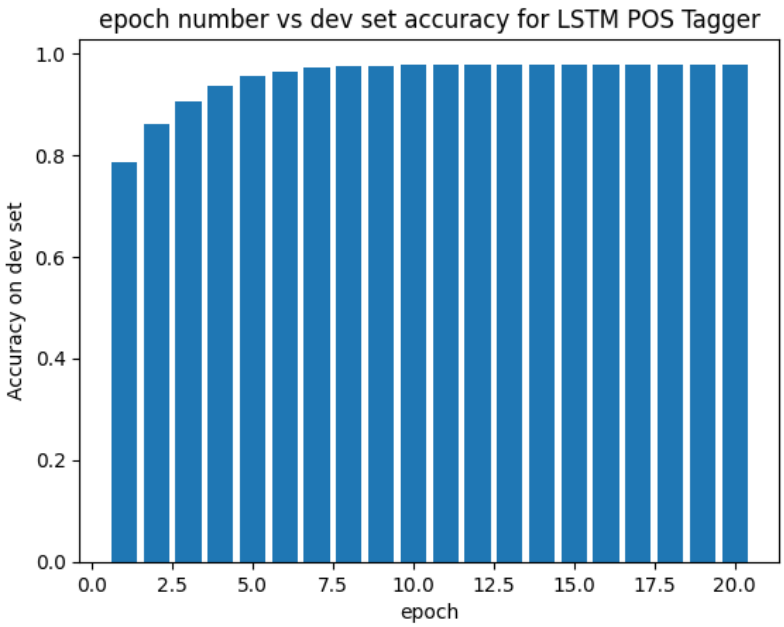
1. EMBEDDING_SIZE=128, HIDDEN_DIM=128, NUM_STACKS=2 ,BATCH_SIZE=128, lrate=0.001, EPOCHS=20

```
Accuracy Score on dev set: 0.977036344513264
F1-Score(micro) on dev set: 0.977036344513264
F1-Score(macro) on dev set: 0.903670587579702
Recall(micro) Score on dev set: 0.977036344513264
Recall(macro) Score on dev set: 0.9350139808054323
Precision(micro) Score on dev set: 0.977036344513264
Precision(macro) Score on dev set: 0.897072294028515 8
Confusion matrix for dev set:
 [[10484     0     0     0     0     0     0     0     0     0     0     0
      0     0]
 [    0   201     2     4     0     0     0     1    12     5     0     0
      1     0]
 [    0     0  1254     1     0     0     0     0     0     1   158     0
      0     0]
 [    0     6     0    46     0     0     0     0     1     6     0     0
      0     0]
 [    0     0     0     0   262     0     0     0     1     0     1     0
      0     0]
 [    0     0     0     0     0   107     0     0     0     0     0     0
      0     0]
 [    0     0    12     0     0     0   486     0     0     1     0    68
      0     0]
 [    0     0     0     0     0     0     0    35     0     0     0     0
      0     0]
 [    0     0     0     1     0     0     0     0  1102    10     0     0
     23     1]
 [    0     0     0     0     0     0     0     0     0   131     0     0
      0     0]
 [    0     0    15     0     0     0     0     0     0     0    58     0
      0     0]
 [    0     0     0     0     0     0     1     0     0     1     1   410
      0     0]
 [    0     0     0     0     0     0     0     0     3    18     0     0
   1530     0]
 [    0     0     4     0    11     0     0     0    14     9     0     0
      0   615]]
```



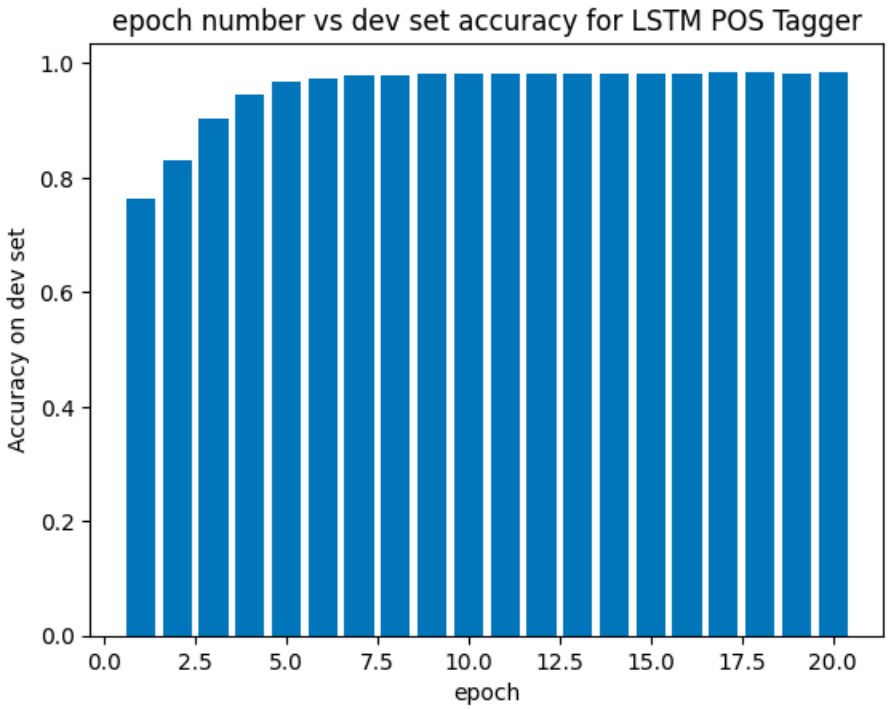
2. EMBEDDING_SIZE=128, HIDDEN_DIM=64, NUM_STACKS=1 ,BATCH_SIZE=128, lrate=0.001, EPOCHS=20

```
Accuracy Score on dev set: 0.9805422461142924
F1-Score(micro) on dev set: 0.9805422461142924
F1-Score(macro) on dev set: 0.9082468686054195
Recall(micro) Score on dev set: 0.9805422461142924
Recall(macro) Score on dev set: 0.9263664404695078
Precision(micro) Score on dev set: 0.9805422461142924
Precision(macro) Score on dev set: 0.9043947199310384
Confusion matrix for dev set:
 [[10484     0     0     0     0     0     0     0     0     0     0
      0     0]
 [    0   204     0     2     0     0     0     0    12     5     0     0
      3     0]
 [    0     0  1314     0     0     0     0     0     0     1    99     0
      0     0]
 [    0     5     0    43     0     0     0     0     2     9     0     0
      0     0]
 [    0     0     0     0   261     0     0     0     1     0     2     0
      0     0]
 [    0     0     0     0     0   107     0     0     0     0     0     0
      0     0]
 [    0     0    12     1     0     0   508     0     0     1     0    45
      0     0]
 [    0     0     0     1     0     0     0    34     0     0     0     0
      0     0]
 [    0     0     0     1     0     0     0     0  1098     9     0     0
     22     7]
 [    0     0     0     0     0     0     0     0     1   130     0     0
      0     0]
 [    0     0    23     0     0     0     0     0     0     0    50     0
      0     0]
 [    0     0     0     0     1     0     7     0     0     1     0   404
      0     0]
 [    0     2     0     0     0     0     0     0     6    18     0     0
   1525     0]
 [    0     0     1     0    11     0     0     0    13     9     0     0
      0   619]]
```



3.EMBEDDING_SIZE=256, HIDDEN_DIM=256,
NUM_STACKS=2 ,BATCH_SIZE=256, lrate=0.001, EPOCHS=20

```
Accuracy Score on dev set: 0.9829871620916397
F1-Score(micro) on dev set: 0.9829871620916397
F1-Score(macro) on dev set: 0.9123307343828196
Recall(micro) Score on dev set: 0.9829871620916397
Recall(macro) Score on dev set: 0.9213766615246425
Precision(micro) Score on dev set: 0.9829871620916397
Precision(macro) Score on dev set: 0.9128519863141598
Confusion matrix for dev set:
 [[12532     0     0     0     0     0     0     0     0     0     0     0
      0     0]
 [    0   210     0     0     0     0     0     0     9     5     0     0
      2     0]
 [    0     0  1359     0     0     0     0     0     0     1    54     0
      0     0]
 [    0     8     0    43     0     0     1     0     0     7     0     0
      0     0]
 [    0     0     0     0   255     0     0     0     1     0     2     0
      0     6]
 [    0     0     0     0     0   107     0     0     0     0     0     0
      0     0]
 [    0     0    16     0     0     0   496     0     0     1     0    54
      0     0]
 [    0     0     0     0     0     0     0    35     0     0     0     0
      0     0]
 [    0     1     0     0     1     0     0     0  1097    11     0     0
     24     3]
 [    0     0     0     0     0     0     0     0     0   131     0     0
      0     0]
 [    0     0    29     0     0     0     0     0     0     0    44     0
      0     0]
 [    0     0     1     0     1     0     4     0     0     1     0   406
      0     0]
 [    0     0     0     0     0     0     0     0     0    18     0     0
   1533     0]
 [    0    18    21     0     7     0     0     0    10     9     0     0
      0   588]]
```



Test set metrics for configuration having best accuracy score on dev set (from above):

```
Accuracy Score on test set: 0.9807336264753529
F1-Score(micro) on test set: 0.9807336264753529
F1-Score(macro) on test set: 0.9044259820360672
Recall(micro) Score on test set: 0.9807336264753529
Recall(macro) Score on test set: 0.9240416319799989
Precision(micro) Score on test set: 0.9807336264753529
Precision(macro) Score on test set: 0.903617847828231
Confusion matrix for test set:
 [[10704     0     0     0     0     0     0     0     0     0     0     0
      0     0]
 [    0   211     0     0     0     0     0     0     0     7     0     0
      2     0]
 [    0     1  1387     0     0     0     0     0     0     1    45     0
      0     0]
 [    0    19     2    47     0     0     0     0     0     5     0     0
      3     0]
 [    0     0     0     0   253     0     0     0     2     0     1     0
      0     0]
 [    0     0     0     0     0   109     0     0     0     0     0     0
      0     0]
 [    0     0     2     0     0     0   444     0     0     2     0    64
      0     0]
 [    0     0     0     0     0     0     0    36     0     0     0     0
      0     0]
 [    0     4     0     0     0     0     0     0  1125    10     0     0
     20     7]
 [    0     0     0     0     0     0     0     0     0   126     0     0
      0     1]
 [    0     0    14     0     0     0     0     0     0     0    42     0
      0     0]
 [    0     0     1     0     0     0     9     0     0     1     0   381
      0     0]
 [    0     1     0     0     0     0     0     0     3    30     0     0
   1533     0]
 [    0    24    18     0    11     0     0     0     6    17     0     0
      0   553]]
```

Analysis

- From the evaluation metrics obtained from multiple configurations shown above for the Feed Forward Neural Network POS tagger, we can say that the accuracy score is highest for p=s=1 and reduces as p and s increase. Also, having multiple hidden layers in the network and changing the activation function (ex-Tanh) does not make much change to the accuracy score. It shows that the complexity of predicting POS tags in not that high and even a simple network is good enough for the task. Another observation is the too large of embedding size may be irrelevant and lead to overfitting and thus a lower accuracy score. On the other hand, a somewhat large hidden layer size is needed for better performance. A small hidden layer is not able to capture sufficient information in the

model to predict tags. An observation for both models is that a large batch size performs better than a small one. So, we can say that larger batch sizes can lead to more stable updates to the model's parameters and faster convergence during training. Smaller batch sizes may introduce more noise into the parameters. In the LSTM POS tagger, we saw that more stacks (2) can improve the performance by capturing hierarchical representations of input sequences. But, adding more layers also increases the model's complexity and the risk of overfitting, seen for stacks=3 or 4. Too small of an embedding size may lead to loss in information and too large size can lead to high complexity of model and potential overfitting. So, an value of about 100-120 was found suitable for our case.

- From context window vs dev set accuracy graph, we notice that the accuracy is minimum when context window size is 0 (p=s=0). This is because we are treating words independently of their neighbourhood. In English, pos tags depend on context and with no context, our accuracy of predicting tags is low. Now, as context window size increases, the accuracy reduces (highest when p=s=1 and reduces for higher p and s). Context helps capture relevant syntactic and semantic information. But, large context may result in irrelevant and noisy information, which degrades performance. Too much context can introduce more ambiguity or make it harder for the model to focus on the most relevant information for POS tagging.

- It can be seen from the results of FFNN model and RNN model that the performance with LSTMs is better. LSTMs are specifically designed to capture long-range dependencies and sequential patterns in data. They have memory cells that can retain information over multiple time steps, making them more suitable for tasks involving sequential data like POS tagging. Sentences can vary in length, and the ability to handle variable-length inputs is crucial for tasks like POS tagging, which gives LSTMs their edge. Also, LSTMs can learn hierarchical representations of input sequences through their recurrent connections. They can automatically extract relevant features and capture intricate dependencies between words, leading to more effective representation learning for POS tagging.

- From the epoch vs dev set accuracy graphs obtained from multiple configurations, we can say that as epoch number increases i.e. model trains, the accuracy score on the dev set increases. After a point, the

accuracy stops changing much further and remains almost the same for the following epochs, which shows us that the model might be done training/ learning and we can stop.