INLP Assignment 2: Neural POS Tagging

Introduction

The task for this assignment was split into two portions, first utilizing a simple Feed-Forward Network (FFN) to solve the task; and next using a RNN based model.

The dataset used for this task was `en_atis` from the larger Universal Dependencies (UD) dataset. This contained above 4000 training sentences, with the central theme of inquiring about airplane tickets. The models trained on this data were evaludated on a test set of around 600 sentences.

The code for this assignment can be found on GitHub.

Data Preparation

Feed-Forward Neural Network

The model(s) were trained to consider a window of the p previous tokens, the specific token t itself, and the s next tokens, to predict the POS tag of token t. Such windows had to be generated for all sentences in the training, test and validation sets. The windows were then padded to ensure the same uniform length $(p + s + 1)$.

To account for the possibility of unkown vocabulary items, words with a frequency lesser than 3 were excluded from the final training vocabulary.

Both POS tags and vocabulary items were separately encoded by Label Encoders.

Recurrent Neural Network

The model(s) were trained to take a sequence of tokens to predict (in sequence) each POS tag of the tokens. As this sequence of tokens has no fixed length, it was necessary to artificially set a maximum length, and pad the tokens of the sentences and target values as necessary.

We repeated the same steps of excluding words with a frequency lesser than 3, to account for unknown tokens; and also encoded POS tags and the tokens using Label Encoders.

Model Architecture

Feed-Forward Neural Networks

Configuration 1:

Layer	Input Dimension	Output Dimension
Embedding	*Size of Training Vocabulary*	256
Flatten	NA	NA
Linear	$256 * (p + s + 1)$	256
ReLU	NA	NA
Linear	256	128
ReLU	NA	NA
Linear	128	*Number of Distinct Target Labels*

Configuration 2:

Layer	Input Dimension	Output Dimension
Embedding	*Size of Training Vocabulary*	256
Flatten	NA	NA
Linear	$256 * (p + s + 1)$	128
ReLU	NA	NA
Linear	128	64
ReLU	NA	NA
Linear	64	*Number of Distinct Target Labels*

Configuration 3:

Layer	Input Dimension	Output Dimension
Embedding	*Size of Training Vocabulary*	256
Flatten	NA	NA
Linear	$256 * (p + s + 1)$	256
Leaky ReLU	NA	NA
Linear	256	128
Leaky ReLU	NA	NA
Linear	128	*Number of Distinct Target Labels*

Recurrent Neural Networks

Configuration 1:

Layer	Input Dimension	Output Dimension
Embedding	*Size of Training Vocabulary*	256
RNN	256	256
Linear	256	*Number of Distinct Target Labels*

Configuration 2:

Layer	Input Dimension	Output Dimension
Embedding	*Size of Training Vocabulary*	256
LSTM	256	256
Linear	256	*Number of Distinct Target Labels*

Configuration 3:

Layer	Input Dimension	Output Dimension
Embedding	*Size of Training Vocabulary*	256
GRU	256	256
Linear	256	*Number of Distinct Target Labels*

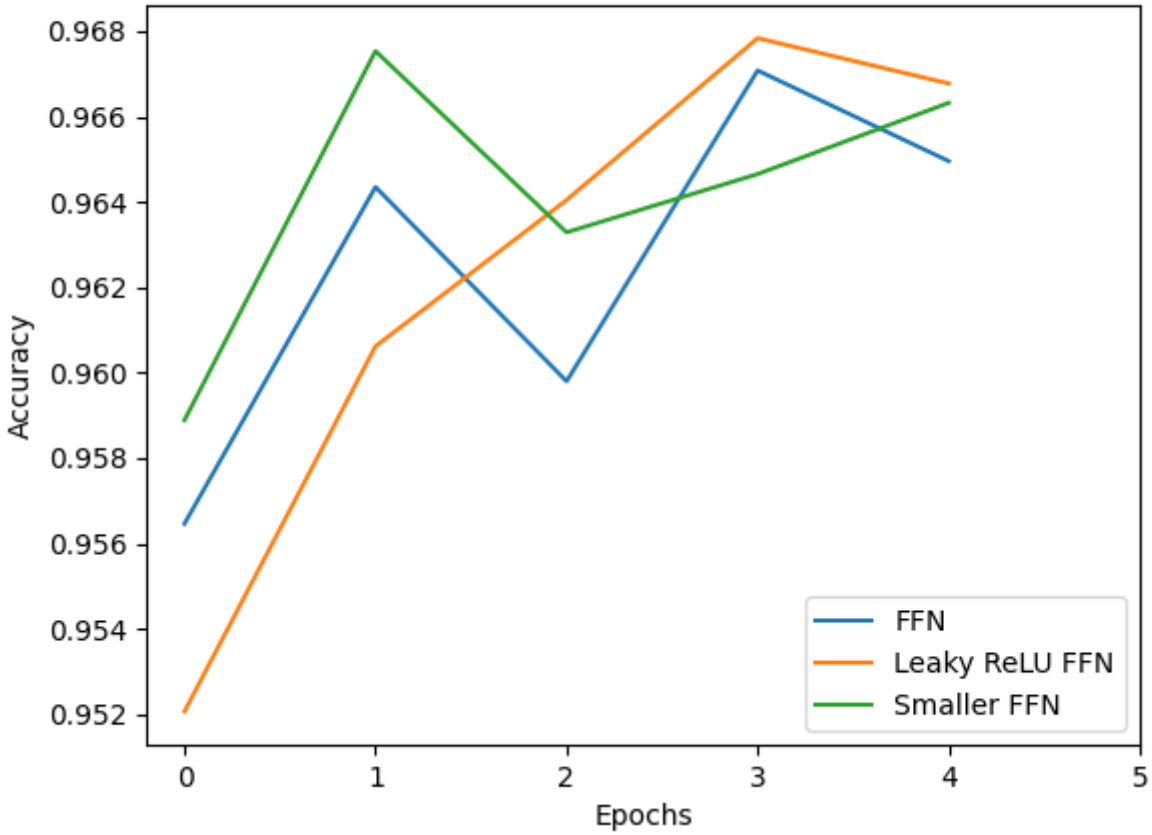
All models (FFNN / RNN) were trained with the following specifications:

- Epochs: 5
- Loss: Cross Entropy
- Optimizer: Adam

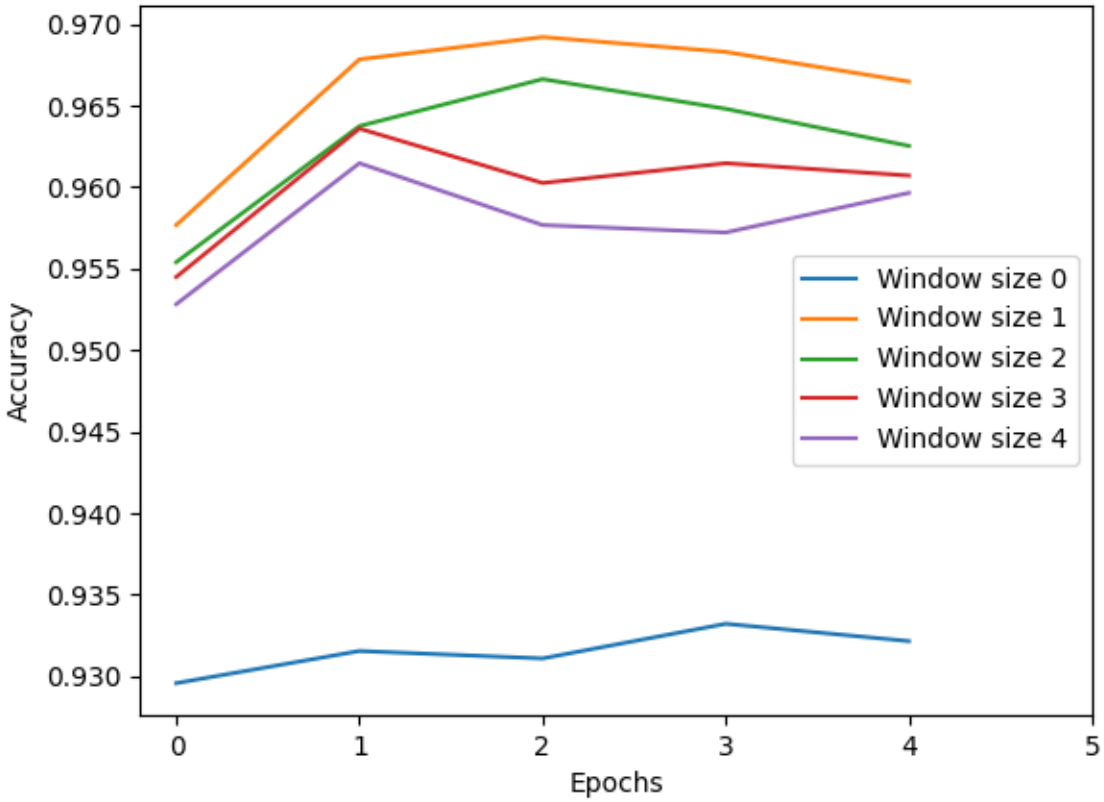
Observations

Feed-Forward Neural Networks

When evaluating the various configurations with $p = 2$, $s = 3$; we determined that Configuration 3, which utilizes the Leaky ReLU activation function, performed best on the validation set.



Subsequently, we used this model over various windows of size $w \in \{0...4\}$ and compared the accuracy on the validation set: ($p = s = w$)



From the above graph we see that the window size of $(p = s = 1)$ provides the best accuracy on the validation set.

Thus, we concluded that a model of Configuration 3, having window size of 1, was the best option for this task. We report some metrics of this model on the validation set and test set below:

Dev Set:

	precision	recall	f1-score	support
0	0.97	0.85	0.91	131
1	0.96	0.88	0.92	653
2	1.00	1.00	1.00	107
3	0.99	0.99	0.99	567
4	0.98	0.99	0.99	1551
5	0.96	0.97	0.97	1137
6	0.95	0.98	0.96	1414
7	0.85	1.00	0.92	73
8	0.99	0.96	0.97	264
9	1.00	0.91	0.96	35
10	0.70	0.81	0.75	59
11	0.99	0.99	0.99	413
12	0.94	0.88	0.91	226
accuracy			0.97	6630
macro avg	0.94	0.94	0.94	6630
weighted avg	0.97	0.97	0.96	6630

Confusion Matrix:

```
[[ 112    0    0    0    1   18    0    0    0    0    0    0    0]
 [   2  576    0    0    2    3   67    1    1    0    0    1    0]
 [   0    0  107    0    0    0    0    0    0    0    0    0    0]
 [   0    0    0  560    0    0    5    0    0    0    0    2    0]
 [   0    3    0    0 1537    8    0    0    0    0    0    0    3]
 [   0    6    0    0   16 1106    3    1    1    0    1    1    2]
 [   0    0    0    5    1    1 1386   11    0    0    9    0    1]
 [   0    0    0    0    0    0    0   73    0    0    0    0    0]
 [   0   11    0    0    0    0    0    0  253    0    0    0    0]
 [   0    3    0    0    0    0    0    0    0   32    0    0    0]
 [   0    0    0    0    2    2    0    0    1    0   48    0    6]
 [   0    1    0    3    0    0    0    0    0    0    0  409    0]
 [   1    1    0    0    2   11    1    0    0    0   11    0  199]]
```

Test Set:

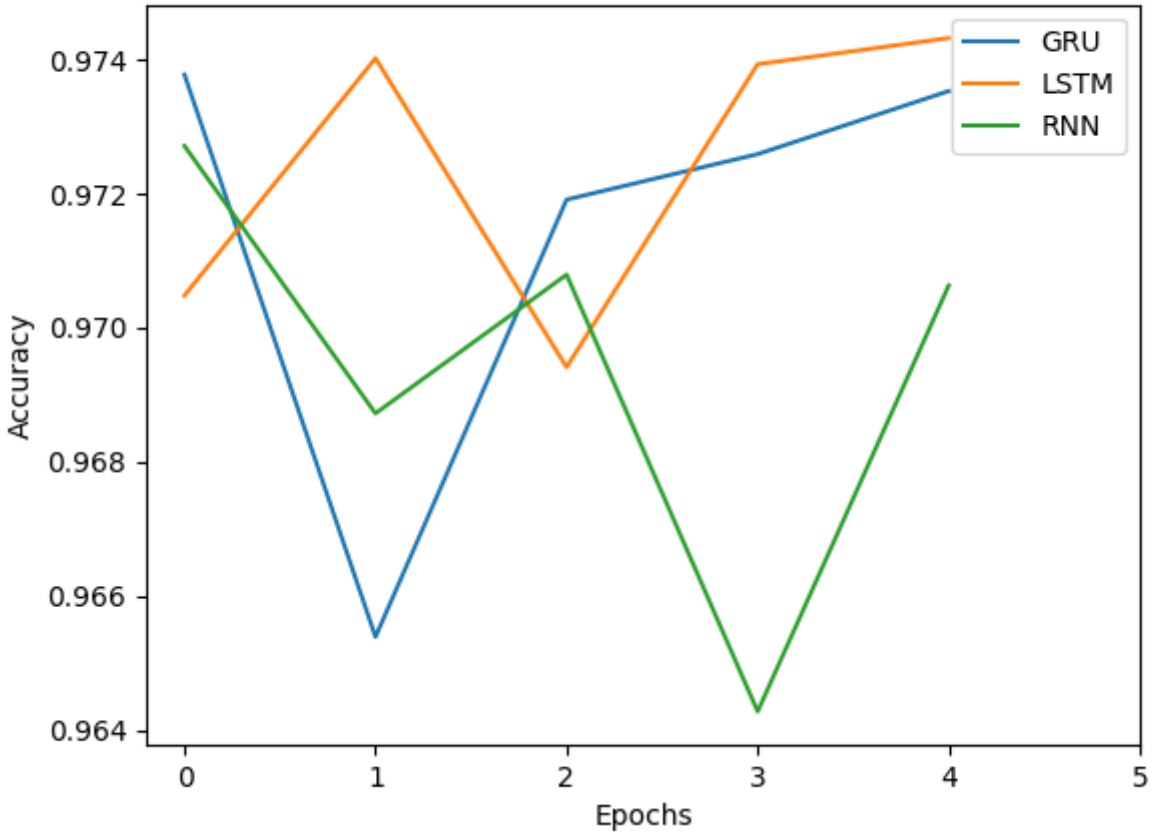
	precision	recall	f1-score	support
0	0.96	0.84	0.90	127
1	0.96	0.87	0.91	629
2	0.99	0.99	0.99	109
3	0.95	0.99	0.97	512
4	0.99	0.99	0.99	1567
5	0.97	0.97	0.97	1166
6	0.95	0.98	0.97	1434
7	0.92	1.00	0.96	56
8	0.98	0.99	0.98	256
9	0.92	0.94	0.93	36
10	0.82	0.80	0.81	76
11	0.99	0.98	0.99	392
12	0.93	0.95	0.94	220
accuracy			0.97	6580
macro avg	0.95	0.95	0.95	6580
weighted avg	0.97	0.97	0.97	6580

Confusion Matrix:

```
[[ 107    0    0    0    1   12    1    0    2    0    2    0    2]
 [   1  548    0    3    1    7   66    0    2    1    0    0    0]
 [   0    0  108    0    0    0    0    0    0    0    1    0    0]
 [   0    0    1  507    1    0    0    0    0    1    0    1    1]
 [   2    3    0    1 1545   15    0    0    0    0    0    0    1]
 [   0   11    0    0   14 1136    2    0    1    0    1    0    1]
 [   0    1    0   16    0    0 1405    5    0    0    6    1    0]
 [   0    0    0    0    0    0    0   56    0    0    0    0    0]
 [   0    2    0    0    0    0    0    0  254    0    0    0    0]
 [   0    2    0    0    0    0    0    0    0   34    0    0    0]
 [   0    1    0    0    2    1    0    0    1    0   61    0   10]
 [   1    1    0    7    0    0    0    0    0    0    0  383    0]
 [   0    2    0    0    2    3    1    0    0    1    3    0  208]]
```

Recurrent Neural Networks

When evaluating the various configurations, we determined that Configuration 2, which utilizes a LSTM layer, provides the best accuracy on the validation set; and is the best suited for this task.



We additionally note that the RNNs perform better than the FFNNs on the validation set. We report some metrics of the LSTM model on the validation set and the test set:

Dev Set:

	precision	recall	f1-score	support
1	0.89	0.98	0.93	91
2	0.98	0.95	0.96	55
3	0.99	0.89	0.94	184
4	0.98	0.99	0.98	423
5	0.97	0.99	0.98	567
6	0.99	1.00	1.00	601
7	0.96	0.98	0.97	210
8	0.98	0.95	0.97	63
9	0.94	0.90	0.92	91
10	1.00	1.00	1.00	62
11	0.81	0.81	0.81	16
12	0.91	0.74	0.82	27
13	1.00	1.00	1.00	9
accuracy			0.97	2399
macro avg	0.95	0.94	0.94	2399
weighted avg	0.97	0.97	0.97	2399

Confusion Matrix:

```
[[ 89   0   0   0   1   0   1   0   0   0   0   0   0]
 [  0  52   0   0   0   0   3   0   0   0   0   0   0]
 [ 11   0 164   0   8   0   0   1   0   0   0   0   0]
 [  0   0   0 417   0   1   2   0   3   0   0   0   0]
 [  0   0   2   0 563   0   0   0   0   0   0   2   0]
 [  0   0   0   1   0 600   0   0   0   0   0   0   0]
 [  0   1   0   3   1   0 205   0   0   0   0   0   0]
 [  0   0   0   0   1   2   0  60   0   0   0   0   0]
 [  0   0   0   3   1   1   1   0  82   0   3   0   0]
 [  0   0   0   0   0   0   0   0   0  62   0   0   0]
 [  0   0   0   0   0   0   1   0   2   0  13   0   0]
 [  0   0   0   0   7   0   0   0   0   0   0  20   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   9]]
```

Test Set:

	precision	recall	f1-score	support
1	0.89	0.95	0.92	80
2	0.94	0.97	0.95	60
3	0.98	0.92	0.95	141
4	0.98	0.98	0.98	397
5	0.99	1.00	1.00	530
6	0.99	1.00	0.99	589
7	0.94	0.98	0.96	174
8	0.95	0.89	0.92	62
9	0.94	0.91	0.92	79
10	1.00	1.00	1.00	56
11	0.94	0.71	0.81	24
12	1.00	0.88	0.93	16
13	1.00	1.00	1.00	10
accuracy			0.98	2218
macro avg	0.97	0.94	0.95	2218
weighted avg	0.98	0.98	0.98	2218

Confusion Matrix:

```
[[ 76   0   2   0   1   0   0   1   0   0   0   0   0]
 [  0  58   0   0   0   0   2   0   0   0   0   0   0]
 [  9   0 130   0   1   0   1   0   0   0   0   0   0]
 [  0   1   0 388   0   4   2   0   2   0   0   0   0]
 [  0   0   0   0 530   0   0   0   0   0   0   0   0]
 [  0   0   0   1   0 588   0   0   0   0   0   0   0]
 [  0   1   0   2   0   0 170   1   0   0   0   0   0]
 [  0   2   0   1   0   1   3  55   0   0   0   0   0]
 [  0   0   0   3   0   1   1   1  72   0   1   0   0]
 [  0   0   0   0   0   0   0   0   0  56   0   0   0]
 [  0   0   0   1   0   2   1   0   3   0  17   0   0]
 [  0   0   0   0   2   0   0   0   0   0   0  14   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0  10]]
```

Analysis

The systems achieve high accuracy, precision, etc. on data that is specifically in the domain of airplane tickets. Its performance while generalizing for text outside this domain is varied;

We consider the sentence `i want a flight from memphis to seattle that arrives no later than 3 pm`, which brings to light some of the issues and erroneous nature of some outputs.

```
>i want a flight from memphis to seattle that arrives no later than 3 pm
i PRON
want VERB
a DET
flight NOUN
from ADP
memphis PROPN
to ADP
seattle PROPN
that ADP
arrives VERB
no DET
later ADJ
than ADP
3 NUM
pm NOUN
```

The initial portion of this sentence, as seen below, provides correct predictions for each of the tags.

```
>i want a flight from memphis
i PRON
want VERB
a DET
flight NOUN
from ADP
memphis PROPN
```

We notice that the word `no` has been tagged as a `DET`, which may appear as an incorrect tagging; however, we are required to note that the word has been tagged such in 5 separate training examples; and that the word does not appear in any other training sentences.

We modify the same sentence to `i want a flight from memphis to seattle that arrives no later than 11 pm` and note the erroneous output of `11`; which is misclassified as a `NOUN`. Here, `11` is being treated as an unknown token, as it is not observed any where in the training examples. We propose that this would be fixed by some special tokenization, etc., prior to the training of the model, which could not be implemented in time for the task.

```
>i want a flight from memphis to seattle that arrives no later than 11 pm
i PRON
want VERB
a DET
flight NOUN
from ADP
memphis PROPN
to ADP
seattle PROPN
that ADP
arrives VERB
no DET
later ADJ
than PART
11 NOUN
pm NOUN
```

If we considered other unknown tokens, as is the case with the sentence `i eat an apple`, we see that the word `apple` has been incorrectly tagged as `PRON`, and `eat` as an auxiliary verb `AUX`. The words `apple` and `eat` are not observed in the training corpus, which could serve as an explanation as to why the words are incorrectly tagged.

```
>i eat an apple
i PRON
eat AUX
an DET
apple PRON
```

Conclusion

We have seen various metrics for the Feed-Forward as well as Recurrent Neural Networks used for the task of POS tagging. We note that the models are able to predict the tags in a sentence with high accuracy, when the sentence was specifically in the domain of airplane tickets; the sentences provided in the training, test and validation sets all fell under this common theme.

As such, we may note improvements by increasing the number of samples, across various domains. Additionally, we may also be able to generalize to other domains by using different

word embeddings, such as GLoVE or BERT embeddings, etc., and by having a more rigorous preprocessing pipeline.