INLP Assignment 2

Name	Rhythm
Roll No.	2021101081

Feed Forward Neural POS Tagger

Preprocessing

The vocabulary is build using the train sentences. Special tokens like the start, end, unknown tokens are also added to vocab. The class dictionary is also made. All the tokens and the class will be represented using these dictionaries

Dataset creation

A `class CustomDataset(Dataset)` was constructed. It would take the sentences from the conllu files , p, s as arguements and provided with an index, return the (p + s + 1) token indexes and target POS tag of the target token. Dataloader is build with the corresponding dataset for the train, dev and test sets.

Model

A `class FeedForwardNN(nn.Module)` was contructed. The `def __init__` takes in the hyperparameters for the model and initialise the layers accordingly. The `def forward` takes in the training set and Outputs the logits corresponding to each class. The `def predict` applies softmax on the forward function logits and returns the predicted POS tags.

Hyperparameters in control:

- embedding dim:- *kept fixed at 100*
- Hidden layers:- No of layers in the Neaural Net
- Hidden dimensions:- No of neurons in each hidden layer

- Activation function:- The activation function used after each layer (’relu’ or ‘tanh’)
- Context sizes:- The p = s = context size parameter (same used for the train dataset)

Results

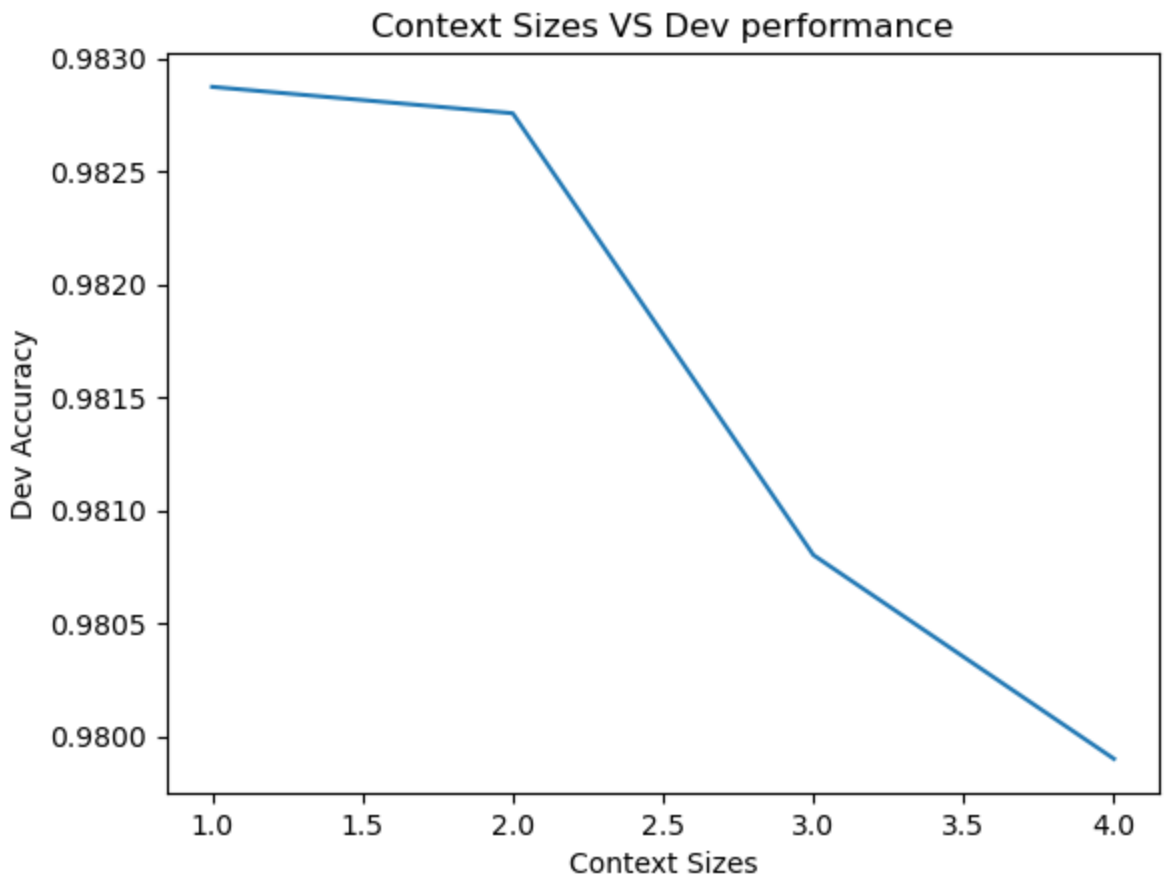
The following were the Dev set performances for Combinations of the Hyperparameters

Context Size	Hidden Layers	Activation	Accuracy
1	2	relu	0.982873
2	1	tanh	0.982756
1	1	tanh	0.982572
2	2	tanh	0.982305
1	1	relu	0.981888
2	1	relu	0.981587
1	2	tanh	0.980919
3	1	tanh	0.980803
3	2	tanh	0.980803
4	1	tanh	0.979901
3	2	relu	0.979183
2	2	relu	0.978733
4	2	tanh	0.976896
3	1	relu	0.975995
4	2	relu	0.974826
4	1	relu	0.974676

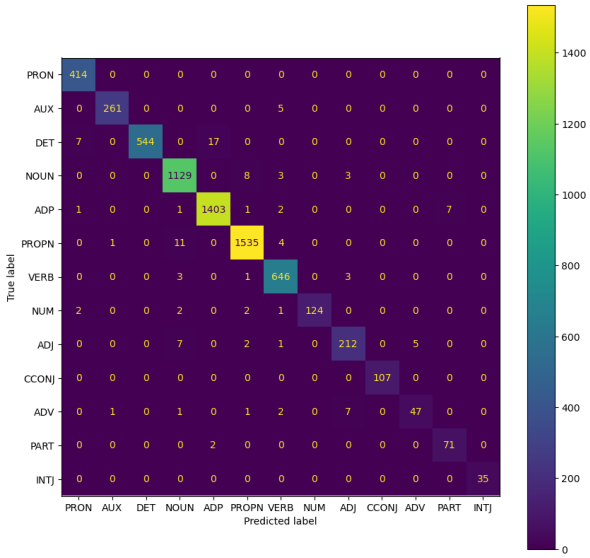
Note:

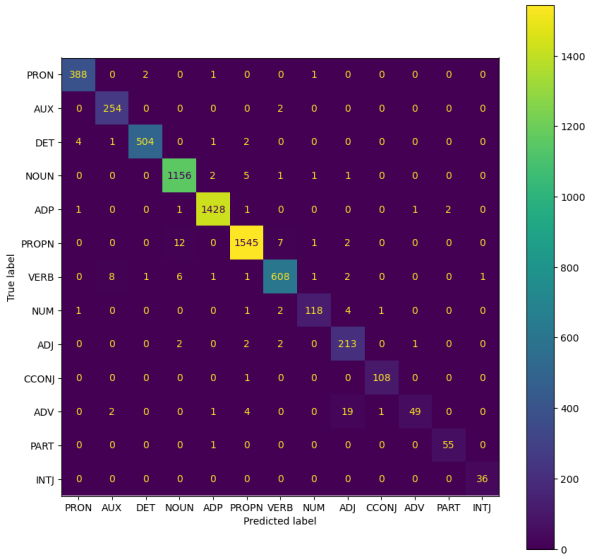
- **The best model performed with** `Accuracy: 0.9821, F1 Score: 0.9602, recall: 0.9821` **on the Test set**
- **The best model performed with** `Accuracy: 0.9828, F1 Score: 0.9688, recall: 0.9828` **on the Dev set**

The following is the Best Dev set accuracy VS the Context size



Confusion Matrices on the Dev and the Test set





Analysis

The model clearly performed better for context sizes 1,2 as compared to longer context sizes. The model was trained for 10 epochs and the version performing best on the dev set was choosen to prevent the case of overfitting on the train set.

The model architecture with different no. of hidden layers has a mixed performance impact and nearly all the architectures perform with over **97%** accuracy on the dev and test set.

On looking at the confusion matrices, it is visible that the about ~25% of the words with ADV class are misclassified as ADJ which is very high compared to any other misclassification of any other class. The reason could be that bot ADJ and ADV describe words and hence play similar roles. The data isnt diverse much to capture their different yet similar roles.

RNN POS Tagger

Preprocessing

The vocabulary is build using the train sentences. Special tokens like the start, end, unknown, pad are also added to vocab. The class dictionary is also made. A special class is added that will help to ignore the special tokens like start, end, padding while calculating losses. All the tokens and the class will be represented using these dictionaries.

Dataset creation

A `class CustomDataset(Dataset)` was constructed. It would take the sentences from the conllu files as arguements and provided with an index, return indexed sentence with token ids and having start and end tokens and padded to max length sentence in the train set.

Model

A `class RNNModel(nn.Module)` was contructed. The `def __init__` takes in the hyperparameters for the model and initialise the layers accordingly. The `def forward` takes in the training set and Outputs the logits corresponding to each class. The `def predict` applies softmax on the forward function logits and returns the predicted POS tags. Above the base seq model, a classifying MLP with 1 Hidden layer is also build.

Hyperparameters in control:

- embedding dim:- *kept fixed at 100*
- Num layers:- No of stacks in the RNN model
- Hidden dimensions:- No of neurons in the hidden layer of the RNN
- Activation function:- The activation function used in the classification layer above the rnn ('relu' or 'tanh')
- Classification Dimension:- The size of the classification layer
- model type:- rnn/gru/lstm
- bidirectionality:- True/ False

Results

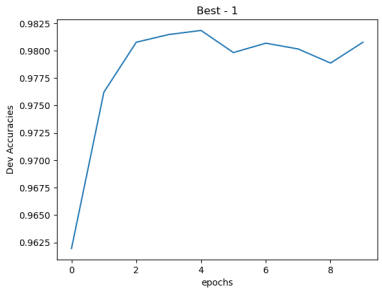
The following were the Dev set performances for Combinations of the Hyperparameters (showing 5 best models)

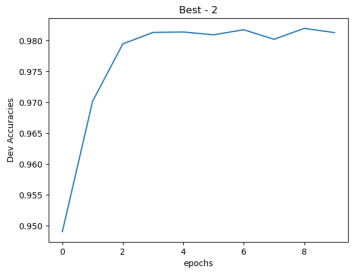
Model	Bidirectionality	Num Layers	Hidden dims	Activation	Classification dim	Accuracy
gru	True	1	128	tanh	128	0.984657
lstm	True	1	64	tanh	64	0.984607
rnn	True	1	64	relu	128	0.984299
lstm	True	1	128	tanh	64	0.983812
gru	True	1	128	tanh	64	0.983732

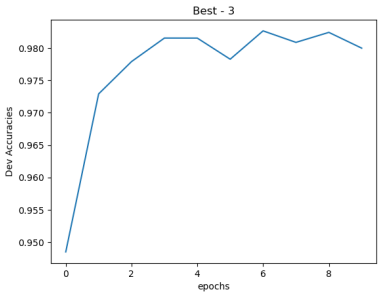
Note:

- **The best model performed with** `Accuracy: 0.9863, F1 Score: 0.9665, recall: 0.9863` **on the Test set**
- **The best model performed with** `Accuracy: 0.9837, F1 Score: 0.9714, recall: 0.9837` **on the Dev set**

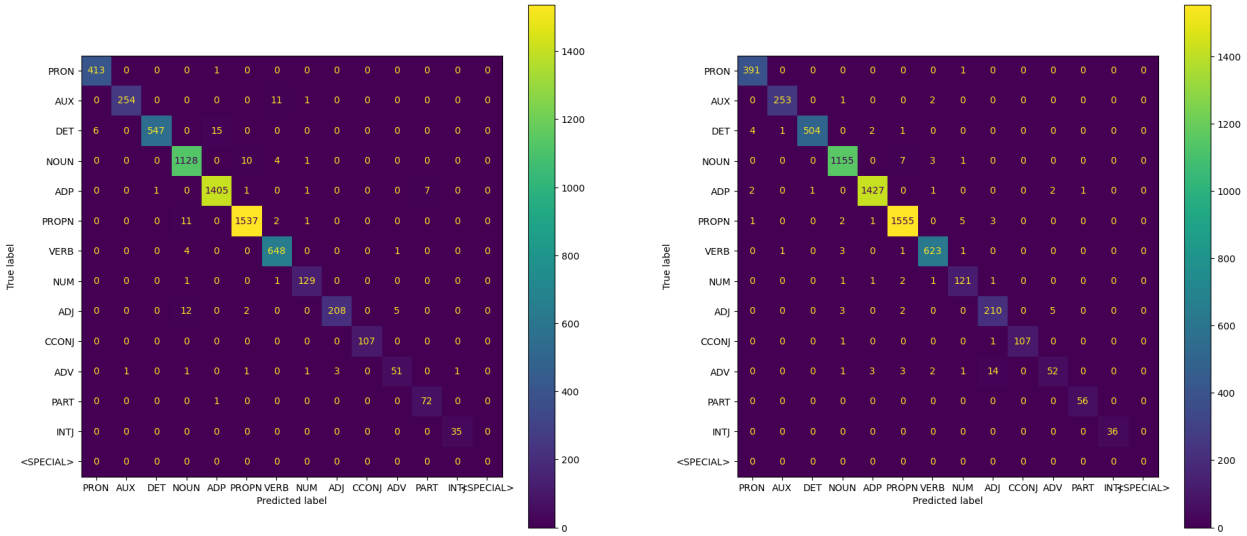
The following is the epochs VS dev set accuracies for the best 3 models







Confusion Matrices on the Dev and the Test set



Analysis

The Bidirectional Seq Models clearly outperforms the sequential models. This is since the POS tagging is also dependent on the sucessive tokens just like the previous tokens, with can only be captured using the bidirectional models.

The GRU and LSTM models have more Learning capacity, yet slower convergence than traditional RNN models. Also the models with 2 Num of layers were overfitting on the train set and thus performed poorly on the dev/test set.

On looking at the confusion matrices, it is visible that the about ~25% of the words with ADV class are misclassified as ADJ which is very high compared to any other misclassification of any other class. The reason could be that bot ADJ and ADV describe words and hence play similar roles. The data isnt diverse much to capture their different yet similar roles.