Atharva Vaidya
23CS60R41

Task 1: word2vec nn, rnn and lstm report

Task 1 – A

Word2Vec with NN layers

Results:

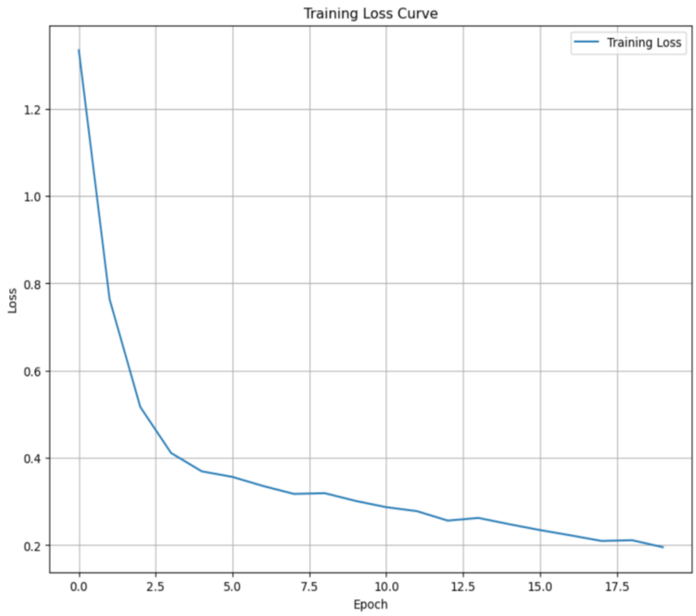
```
Test Accuracy: 0.88

Test F1 Score: 0.8808079062356072

Confusion matrix:

[[108   5   9   3]
 [  1 117   3   4]
 [  4   2 107  12]
 [  0   2  15 108]]
```

Classification Report:

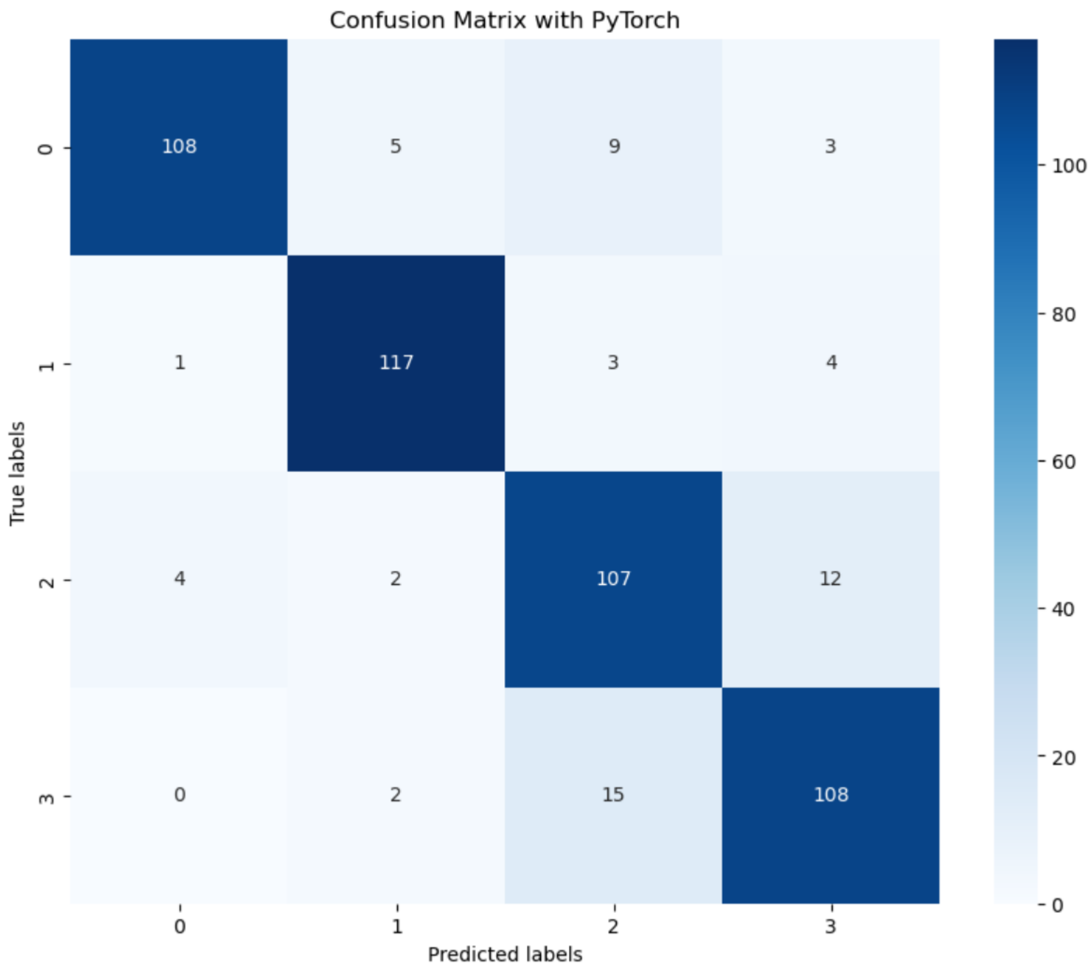
	precision	recall	f1-score	support
0	0.96	0.86	0.91	125
1	0.93	0.94	0.93	125
2	0.80	0.86	0.83	125
3	0.85	0.86	0.86	125
accuracy			0.88	500
macro avg	0.88	0.88	0.88	500
weighted avg	0.88	0.88	0.88	500

Loss per epoch :

```
Epoch 1/20 - Loss: 1.3342490419745445
Epoch 2/20 - Loss: 0.7636880492791533
Epoch 3/20 - Loss: 0.5165558280423284
Epoch 4/20 - Loss: 0.4112724685110152
Epoch 5/20 - Loss: 0.3691287524998188
Epoch 6/20 - Loss: 0.356366487685591
Epoch 7/20 - Loss: 0.335458823479712
Epoch 8/20 - Loss: 0.3174627139233053
Epoch 9/20 - Loss: 0.3191394293680787
Epoch 10/20 - Loss: 0.3014697222970426
Epoch 11/20 - Loss: 0.28707222640514374
Epoch 12/20 - Loss: 0.2778952154330909
Epoch 13/20 - Loss: 0.25629640673287213
Epoch 14/20 - Loss: 0.26239574677310884
Epoch 15/20 - Loss: 0.24810648243874311
Epoch 16/20 - Loss: 0.2346387798897922
Epoch 17/20 - Loss: 0.22247639182023704
Epoch 18/20 - Loss: 0.20974952867254615
Epoch 19/20 - Loss: 0.21135730133391917
Epoch 20/20 - Loss: 0.19533128989860415
```



Confusion Matrix:



Hyperparameter values:

1) Word2Vec Embedding Dimension: 300

2) Batch Size: 64

3) Neural Network Layers and Dimensions:

    Input Layer Dimension: 300

    First Hidden Layer Dimension: 256

    Second Hidden Layer Dimension: 128

    Third Hidden Layer Dimension: 64

    Output Layer Dimension: Number of unique labels

4) Dropout Rate: 0.5

5) Learning Rate for Adam Optimizer: 0.001

6) Number of Epochs: 20

7) Loss Function: Cross-Entropy Loss (nn.CrossEntropyLoss())

Challenges faced and resolution:

1) One challenge was selecting appropriate hyperparameters, such as the batch size and learning rate, to ensure the model trains effectively without overfitting or underfitting. I addressed this by starting with commonly used default values and then adjusting them based on the performance of the model on the training and validation datasets.

2) Handling the text preprocessing was another challenge, especially ensuring that the text data is cleaned and tokenized effectively to be suitable for the Word2Vec model and the neural network. This involved removing unnecessary characters, HTML tags, and ensuring that the text is tokenized consistently. To resolve this, I used regular expressions for cleaning and the NLTK library for tokenization, ensuring the text was appropriately prepared for feature extraction and modelling.

(Task 1- B is on the next page)

Task 1-B ()

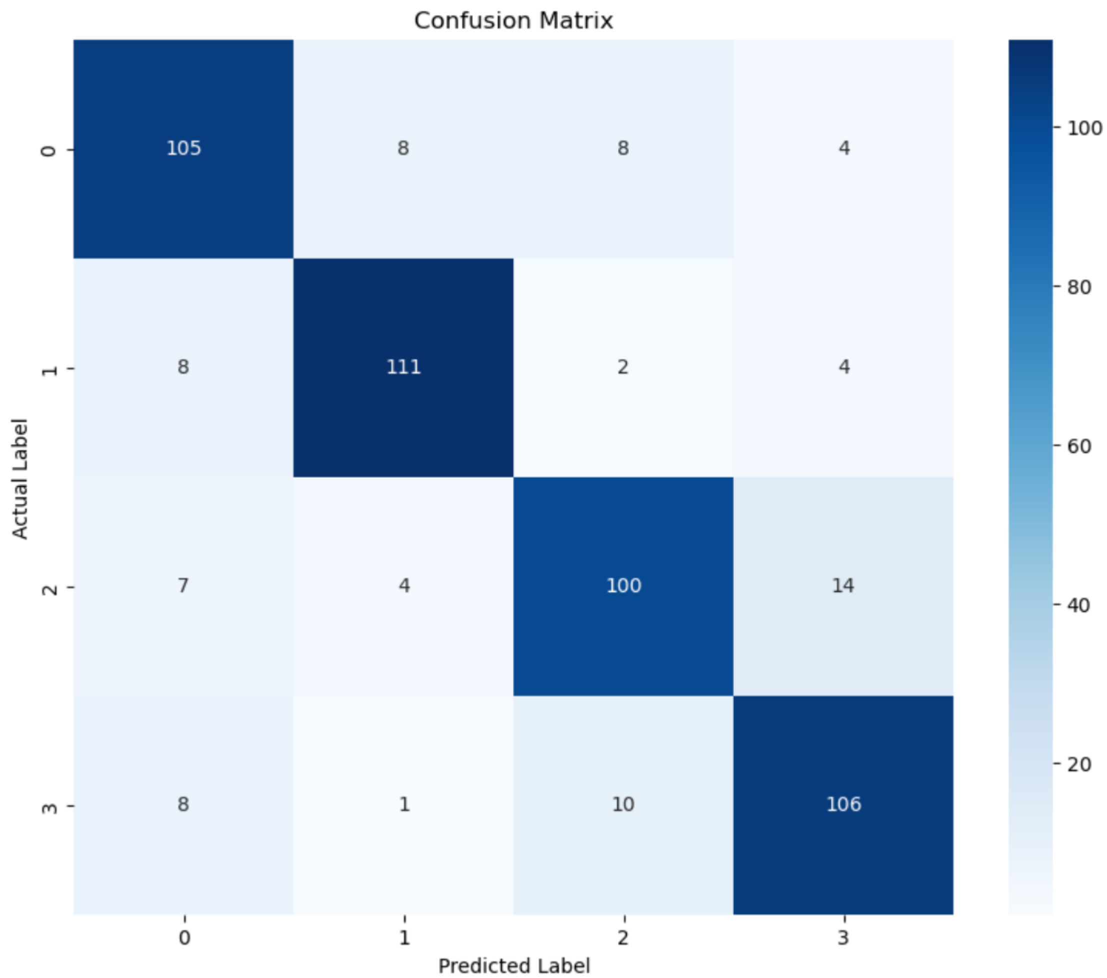
Vanilla RNN and LSTM

Vanilla RNN:

Classification Report :

```
Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.84      0.83       125
           1       0.90      0.89      0.89       125
           2       0.83      0.80      0.82       125
           3       0.83      0.85      0.84       125

    accuracy                           0.84       500
   macro avg       0.84      0.84      0.84       500
weighted avg       0.84      0.84      0.84       500
```

Confusion Matrix:



Hyperparameters used:

- **vocab_size**: Length of the vocabulary + 1 (for padding)
- **embedding_dim**: 300 (dimension of the word vectors)
- **hidden_size**: 128 (dimension of the RNN hidden state)
- **num_classes**: Number of unique classes in the dataset = 4
- **bidirectional**: **True** (using a bidirectional RNN)
- **dropout**: 0.5 (dropout rate for regularization)

loss per epoch :

```
Epoch 1/20, Loss: 0.1591
Epoch 2/20, Loss: 0.1620
Epoch 3/20, Loss: 0.1672
Epoch 4/20, Loss: 0.1635
Epoch 5/20, Loss: 0.1622
Epoch 6/20, Loss: 0.1605
Epoch 7/20, Loss: 0.1629
Epoch 8/20, Loss: 0.1615
Epoch 9/20, Loss: 0.1656
Epoch 10/20, Loss: 0.1671
Epoch 11/20, Loss: 0.1577
Epoch 12/20, Loss: 0.1620
Epoch 13/20, Loss: 0.1677
Epoch 14/20, Loss: 0.1660
Epoch 15/20, Loss: 0.1636
Epoch 16/20, Loss: 0.1614
Epoch 17/20, Loss: 0.1652
Epoch 18/20, Loss: 0.1635
Epoch 19/20, Loss: 0.1606
Epoch 20/20, Loss: 0.1636
```

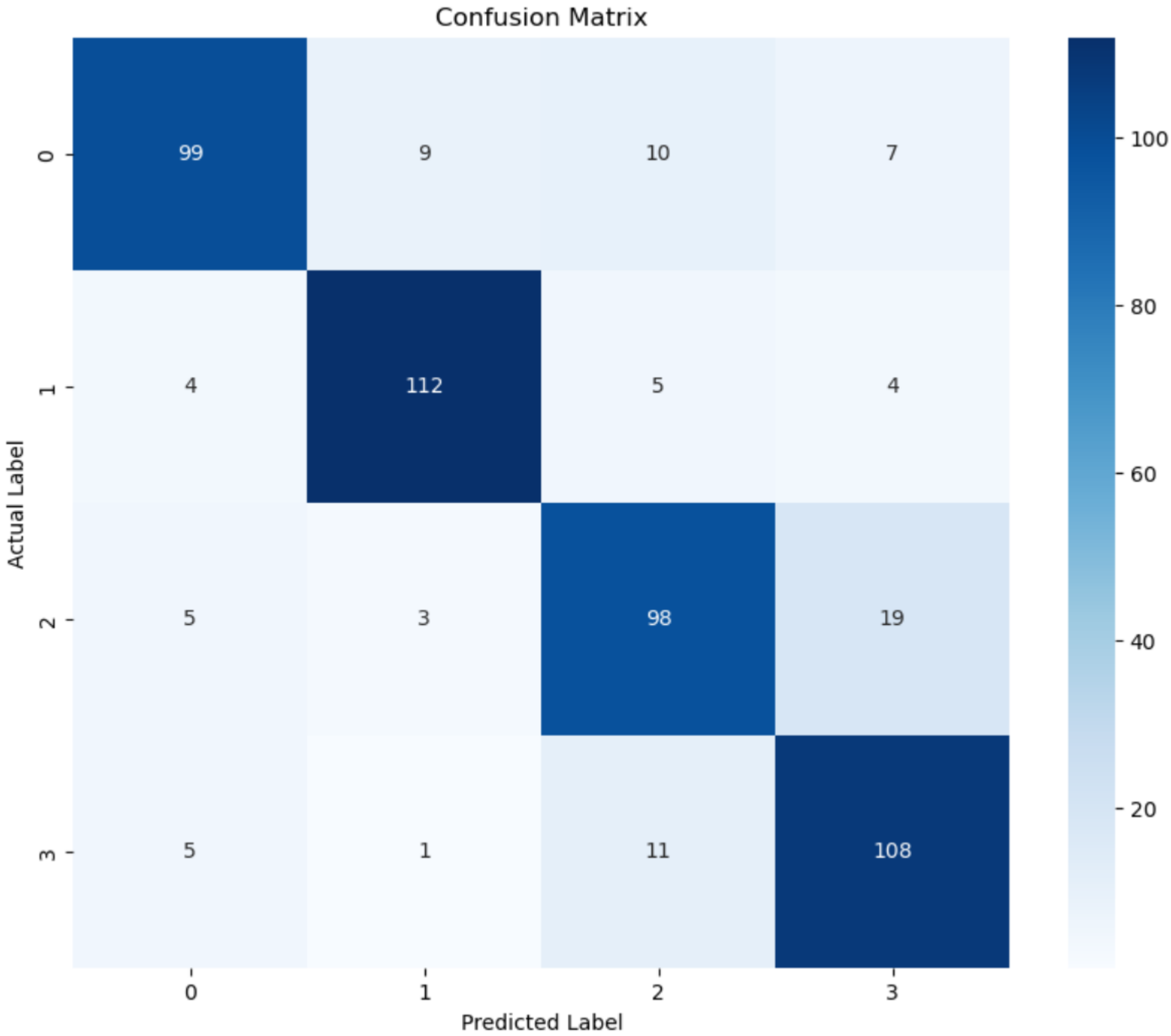
LSTM:

Classification Report:

Classification Report:

	precision	recall	f1-score	support
0	0.88	0.79	0.83	125
1	0.90	0.90	0.90	125
2	0.79	0.78	0.79	125
3	0.78	0.86	0.82	125
accuracy			0.83	500
macro avg	0.84	0.83	0.83	500
weighted avg	0.84	0.83	0.83	500

Confusion Matrix:

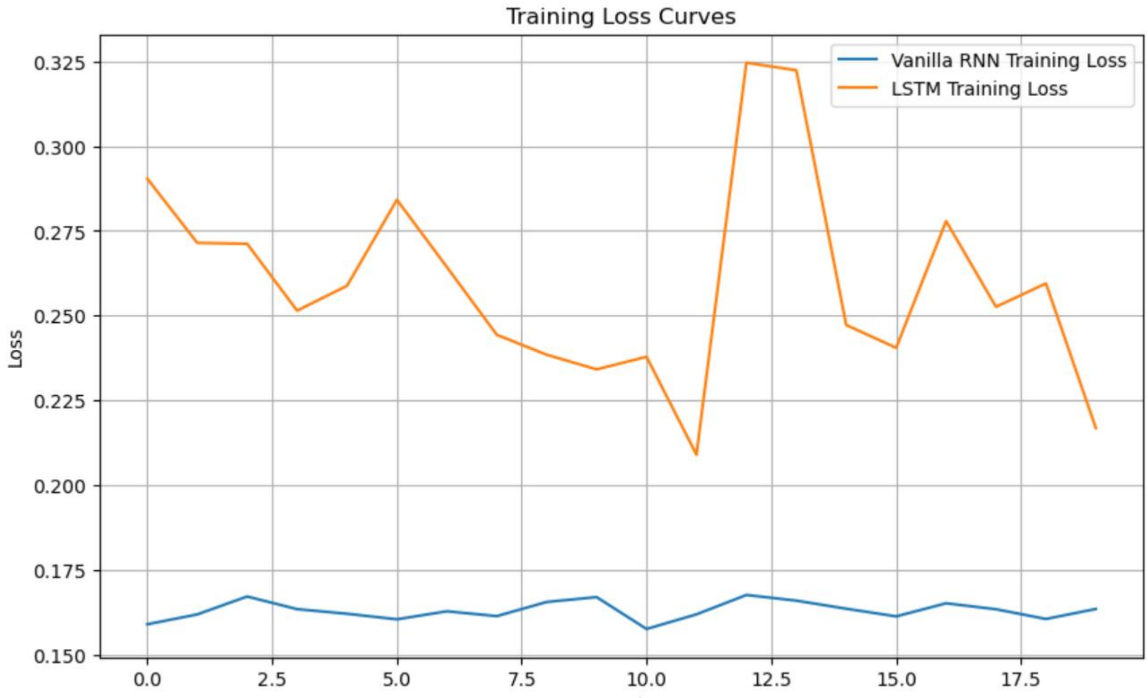


Loss per epoch :

```
Epoch 1, Loss: 0.29038958239834756
Epoch 2, Loss: 0.271456035785377
Epoch 3, Loss: 0.27120111091062427
Epoch 4, Loss: 0.2514864052645862
Epoch 5, Loss: 0.2588131108786911
Epoch 6, Loss: 0.28415487986057997
Epoch 7, Loss: 0.26433773105964065
Epoch 8, Loss: 0.24437629792373627
Epoch 9, Loss: 0.23849304905161262
Epoch 10, Loss: 0.23418563976883888
Epoch 11, Loss: 0.23788254405371845
Epoch 12, Loss: 0.20903933269437402
Epoch 13, Loss: 0.3245592007879168
Epoch 14, Loss: 0.32232873793691397
Epoch 15, Loss: 0.24725464056245983
Epoch 16, Loss: 0.24053579126484692
Epoch 17, Loss: 0.277883832808584
Epoch 18, Loss: 0.2526153514627367
Epoch 19, Loss: 0.25950646214187145
Epoch 20, Loss: 0.21687763812951744
```

Hyperparameters used:

- **vocab_size**: Length of the vocabulary + 1 (for padding)
- **embedding_dim**: 300 (dimension of the word vectors)
- **hidden_dim**: 128 (number of features in the hidden state of the LSTM)
- **num_classes**: Number of unique classes in the dataset (derived from **train_df['label'].nunique()**)
- **bidirectional**: **True** (using a bidirectional LSTM)
- **dropout**: 0.5 (dropout rate for regularization)
- **learning_rate**: 0.001 (learning rate for the optimizer)
- **epochs**: 20 (number of epochs for training)



Challenges faced and resolution:

1) One challenge faced while designing the Vanilla RNN and LSTM models was selecting the appropriate architecture and hyperparameters. To resolve this, I considered the size of the dataset, the complexity of the task, and the computational resources available. I opted for relatively modest network sizes (hidden dimensions) and experimented with different dropout rates to prevent overfitting while ensuring the models were complex enough to capture the patterns in the data.

2) Another challenge was ensuring that the models could handle variable-length text inputs efficiently. To address this, I used padding and truncation strategies to regularize the input size, ensuring that all text inputs were converted to a uniform length before being fed into the models.