INLP – Assignment 2

Course Coordinator: Manish Srivatsava

Name : Lakshmipathi Balaji

Roll No: 2021114007

Mail : lakshmipathi.balaji@gmail.com

Project details:

- POS tagging using LSTM/ GRU/ RNN.
- Hyperparameter tuning for best accuracy.

Procedure:

I have implemented a LSTM to do POS tagging using the given datasets – ud-treebanks-v2.11/UD_English-Atis/en_atis-ud-{train,dev,test}.conllu. The modularity in code is well maintained.

- **Things Learnt:** A clear understanding of the LSTMs and some background math of it.

Best Model Observed:

The hyperparameters used for the model -

```
# PARMAETERS
EMBEDDING_DIM = 300
BATCH_SIZE = 4
HIDDEN_DIM = 256
EPOCHS = 2
NUM_LAYERS = 3
```

Loss Function – NLLloss function (Negative log likelihood loss)

Optimiser – Adam (This optimiser auto adjusts learning rate so that parameter is not used in optimiser)

F1 scores, Accuracies, Loss per each epoch are given below -

```
Starting {0} Epoch 2023-03-09 14:59:27.550224
-1_Epoch Accuracy - 0
Accuracy of the network on the 6644 validation examples: 93 %
0_Epoch Accuracy - 93.10656231186033
Epoch 0 completed with loss 0.3708084225654602
Starting {1} Epoch 2023-03-09 14:59:43.303225
0_Epoch Accuracy - 93.10656231186033
Accuracy of the network on the 6644 validation examples: 94 %
1_Epoch Accuracy - 94.37086092715232
Epoch 1 completed with loss 0.09596716612577438
Accuracy of the network on the 6644 validation examples: 94 %
Accuracy of the network on the 6580 test examples: 94 %
              precision    recall  f1-score   support

           0       1.00      0.73      0.84       392
           1       0.97      0.91      0.94       256
           2       0.82      0.99      0.90       512
           3       0.98      0.97      0.97      1166
           4       0.95      1.00      0.98      1434
           5       0.95      0.99      0.97      1567
           6       0.94      0.87      0.91       629
           7       0.96      0.64      0.77       127
           8       0.94      0.95      0.95       220
           9       1.00      0.97      0.99       109
          10       0.91      0.79      0.85        76
          11       0.98      0.91      0.94        56
          12       0.97      1.00      0.99        36

    accuracy                           0.95      6580
   macro avg       0.95      0.90      0.92      6580
weighted avg       0.95      0.95      0.95      6580

[[ 286    0  105    0    0    0    0    1    0    0    0    0    0]
 [   0  233    0    0    0    0   23    0    0    0    0    0    0]
 [   0    0  508    1    0    2    1    0    0    0    0    0    0]
 [   0    0    1 1131    2   28    4    0    0    0    0    0    0]
 [   0    0    3    0 1429    0    0    0    0    0    1    1    0]
 [   0    0    1    6    0 1557    0    1    1    0    1    0    0]
 [   0    8    0    9   59    1  550    0    2    0    0    0    0]
 [   0    0    0    5    0   37    1   81    3    0    0    0    0]
 [   0    0    0    3    1    3    1    0  209    0    3    0    0]
 [   0    0    0    0    0    2    0    0    0  106    1    0    0]
 [   0    0    0    0    1    3    3    1    7    0   60    0    1]
 [   0    0    0    0    5    0    0    0    0    0    0   51    0]
 [   0    0    0    0    0    0    0    0    0    0    0    0   36]]
```

It can be observed that f1 score is high with good support and the confusion_matrix also shows that.

INLP – Assignment 2

Challenges Faced:

- I was not able to use the complete power of my inbuilt external gpu RTX-3060-4gb. After going through many I got to know about how I can use my gpu for this purpose.
- Though there was a little knowledge about LSTMs before I had to go through 3B1B videos and pytorch to get to know actual functionality and code of LSTMs.

Realisations:

- Intially I implemented a LSTM with pretrained embeddings with a embedding layer seperately for train,valid,test datasets but after some time I realised this is not a good way as the meaning the vectors for the same word may change as the dataset changed so I had to shift back to use embedding layer for each input.
- Then I passed sentence to embedding layer where each word is indexed by a dictionary storing all words and used individual dictionaries for 3 datasets and realised this will give vectors depending on as early as the word occurred in their respective datasets rather than the actual meaning.
- Then I finally got a conclusion to do the embeddings for each sentence where each word is indexed by training dataset dictionary for all of the datasets and treating new words as UNK.

Helpful links:

- 3b1b playlist on neural networks
- Pytorch tutorials.
- Intermediate models that are produced during hyperparameter tuning are here.models