Task 1: Simple Bidirectional LSTM model

Q. What are the precision, recall and F1 score on the dev data?

Precision – 85.32%, Recall – 71.10%, F1 score – 77.57%

```
C:\Users\dipal\Documents\CSCI544_HW\HW4\hw4>perl conll03eval.txt < dev1.out
processed 51578 tokens with 5942 phrases; found: 4952 phrases; correct: 4225.
accuracy:  95.31%; precision:  85.32%; recall:  71.10%; FB1:  77.57
              LOC: precision:  93.36%; recall:  78.01%; FB1:  84.99  1535
             MISC: precision:  89.25%; recall:  76.57%; FB1:  82.43  791
              ORG: precision:  78.04%; recall:  65.18%; FB1:  71.03  1120
              PER: precision:  80.48%; recall:  65.80%; FB1:  72.40  1506
```

First converted input data into a list of sentences and their tags. Padded sentences with 0 up to max length of the largest sentence in dataset. Also Padded tags with -1 to match the length of sentence input. Mentioned in loss function to ignore index as -1 to not consider padding as output.

Model architecture used in code is as follows:

1. Embedding layer – Embedding layer has total embeddings equal to size of word vocab of train data.
2. BLSTM layer – Added LSTM layer with the parameters given in the question.
3. Dropout layer – Added separate dropout layer for BLSTM dropout with dropout rate given in question.
4. Linear layer – Added linear layer with the parameters given in the question.
5. ELU layer – Added ELU layer with the parameters given in the question.
6. Classifier layer – Added linear layer as classifier layer in model to have output classes equal to number of tags.

Later I flattened the output given from the model and used torch.max to predict the tag for every word in the sentence correctly.

Hyperparameters used –

1. Batch size - 8
   - Tried multiple combinations of batch sizes including 4, 8, 16, 32 and 64. Chose 8 as it gave the best output.
2. Number of epochs – 30
   - Tried multiple values of batch sizes and observed the losses in training, at 30 epochs my model gave the best result.
3. Class weights
   - Added class weights parameter in the optimizer and gave less weight to tag ‘O’ as the dataset was imbalanced and tag ‘O’ had a large number of instances than any other classes.
4. Learning rate - 1

- Tried multiple values of learning rate and observed the losses in training, at this learning rate my model gave the best result

5. Learning rate scheduler
   - Used ReduceLROnPlateau scheduler to decrease the learning rate after no improvement in validation loss during epochs.
   - Scheduler multiplied learning rate by factor of 0.5 when no improvement is seen after 3 epochs.

Task 2: Simple Bidirectional LSTM model

Q. What are the precision, recall and F1 score on the dev data?

Precision – 85.68%, Recall – 86.79%, F1 score – 86.23%

```
C:\Users\dipal\Documents\CSCI544_HW\HW4\hw4>perl conll03eval.txt < dev2.out
processed 51578 tokens with 5942 phrases; found: 6019 phrases; correct: 5157.
accuracy:  97.23%; precision:  85.68%; recall:  86.79%; FB1:  86.23
              LOC: precision:  93.97%; recall:  90.80%; FB1:  92.36  1775
             MISC: precision:  82.41%; recall:  79.28%; FB1:  80.82  887
              ORG: precision:  85.45%; recall:  79.27%; FB1:  82.24  1244
              PER: precision:  80.22%; recall:  92.02%; FB1:  85.71  2113
```

First converted input data into list of sentences and their tags. Padded sentences with 0 up to max length of largest sentence in dataset. Also Padded tags with -1 to match the length of sentence input. Mentioned in loss function to ignore index as -1 to not consider padding as output.

Model architecture used in code is as follows:

1. Embedding layer – Passed pre-trained embedding matrix from GloVe dataset to embedding layer. Set parameter Freeze = false to let the embedding learn more from input during training.
2. Dropout layer – Added dropout layer with p =0.2 before sending input to BLSTM layer to improve the model efficiency.
3. BLSTM layer – Added LSTM layer with the parameters given in the question.
4. Dropout layer – Added separate dropout layer for BLSTM dropout with dropout rate given in question.
5. Linear layer – Added linear layer with the parameters given in the question.
6. Dropout layer – Again added dropout layer with p =0.2 before sending input to ELU layer to improve the model efficiency.
7. ELU layer – Added ELU layer with the parameters given in the question.
8. Classifier layer – Added linear layer as classifier layer in model to have output classes equal to number of tags.

Later I flattened the output given from the model and used torch.max to predict the tag for every word in the sentence correctly.

Hyperparameters used –

1. Batch size - 8
   - Tried multiple combinations of batch sizes including 4, 8, 16, 32 and 64. Chose 8 as it gave the best output.
2. Number of epochs – 50
   - Tried multiple values of batch sizes and observed the losses in training, at 50 epochs my model gave the best result.
3. Class weights
   - Added class weights parameter in the optimizer and gave less weight to tag ‘O’ as the dataset was imbalanced and tag ‘O’ had a large number of instances than any other classes.
   - Also gave some other tags more weight as they had very less instances in dataset and their predictions were not up to the mark.
4. Learning rate - 1
   - Tried multiple values of learning rate and observed the losses in training, at this learning rate my model gave the best result
5. Learning rate scheduler
   - Used OneCycleLR scheduler to adjust the learning linearly as per the number of epochs with limit of maximum learning rate as 1.