HW 4 - Report

MRINAL KADAM
USC ID: 3135945534

Answers to the questions in the assignment:

Hyper-parameters for all tasks combined (Common):

The hyper-parameters of the network as mentioned in the HW PDF are listed below:

```
embedding_dimension = 100
character_embedding_dimension = 30
lstm_hidden_dimension = 256
dropout = 0.33
output_dimension = 128
```

Task 1: Simple Bidirectional LSTM model

Other parameters & hyper-parameters selected by me:

(Selected after also referencing the paper
https://arxiv.org/pdf/1603.01354.pdf
by Xuezhe Ma and Eduard Hovy)

```
num_epochs = 50
batch_size = 16
learning_rate = 0.015
momentum = 0.9
decay_rate = 0.05
gradient_clip = 5.0

seed = 100
```

batch size: 16
learning rate: 0.015
learning rate scheduling: learning_rate / (1 + decay_rate * epoch)

Optimizer & Loss Function:

Optimizer: SGD

```
loss_fn = torch.nn.CrossEntropyLoss
```

Dev:

After 50 epochs,

processed 51578 tokens with 5942 phrases; found: 5425 phrases; correct: 4607.
accuracy: 96.25%; **precision**: **84.92%**; **recall**: **77.53%**; **FB1**: **81.06**
LOC: precision: 88.36%; recall: 88.41%; FB1: 88.38 1838
MISC: precision: 80.23%; recall: 74.40%; FB1: 77.21 855
ORG: precision: 78.18%; recall: 67.34%; FB1: 72.36 1155
PER: precision: 88.40%; recall: 75.68%; FB1: 81.54 1577

Task 2: Using GloVe word embeddings

Other parameters & hyper-parameters selected by me:

(Selected after also referencing the paper
https://arxiv.org/pdf/1603.01354.pdf
by Xuezhe Ma and Eduard Hovy)

```
num_epochs = 100
batch_size = 16
learning_rate = 0.015
momentum = 0.9
decay_rate = 0.05
gradient_clip = 5.0

seed = 100
```

batch size: 16
learning rate: 0.015
learning rate scheduling: learning_rate / (1 + decay_rate * epoch)

Optimizer & Loss Function:

Optimizer: SGD

```
loss_fn = torch.nn.CrossEntropyLoss
```

Dev:

After 100 epochs,

processed 51578 tokens with 5942 phrases; found: 5440 phrases; correct: 4850.
accuracy: 96.81%; **precision**: **89.15%**; **recall**: **81.62%**; **FB1**: **85.22**
LOC: precision: 91.88%; recall: 89.33%; FB1: 90.59 1786
MISC: precision: 85.86%; recall: 73.75%; FB1: 79.35 792
ORG: precision: 83.73%; recall: 73.30%; FB1: 78.17 1174

PER: precision: 91.59%; recall: 83.93%; FB1: 87.59 1688

Task 3(Bonus): LSTM-CNN model

Other parameters & hyper-parameters selected by me:

(Selected after also referencing the paper
https://arxiv.org/pdf/1603.01354.pdf
by Xuezhe Ma and Eduard Hovy)

```
num_epochs = 100
batch_size = 16
learning_rate = 0.015
momentum = 0.9
decay_rate = 0.05
gradient_clip = 5.0

out_channels = 30
in_channels=1,
out_channels=out_channels,
kernel_size=(3, character_embedding_dimension),
padding=(2, 0)

seed = 100
```

batch size: 16
learning rate: 0.015
learning rate scheduling: learning_rate / (1 + decay_rate * epoch)

number of CNN layers: 1
kernel size: (3,30)
output dimension of each CNN layer: 30

Optimizer & Loss Function:

Optimizer: SGD

```
loss_fn = torch.nn.CrossEntropyLoss
```

Dev:

After 100 epochs,

processed 51578 tokens with 5942 phrases; found: 6033 phrases; correct: 5285.
accuracy: 98.08%; **precision: 87.60%**; **recall**: **88.94%**; **FB1**: **88.27**
LOC: precision: 90.86%; recall: 93.63%; FB1: 92.23 1893

MISC: precision: 84.38%; recall: 80.26%; FB1: 82.27 877
ORG: precision: 78.48%; recall: 86.20%; FB1: 82.16 1473
PER: precision: 93.24%; recall: 90.61%; FB1: 91.91 1790

Brief description of my solution:

Task 1:
Random word embeddings were used without character level encoding.
Task 2:
Glove word embeddings were used without character level encoding.
Task 3:
Glove word embeddings were used with character level encoding.

All hyper-parameters were set after careful consideration and after referencing the paper that produced state of the art results- **https://arxiv.org/pdf/1603.01354.pdf**
by Xuezhe Ma and Eduard Hovy.

- The input file (train/dev/test) and the embedding file (Glove) were read according to the task being performed and formatted accordingly.
- Some data pre-processing was done, like making all digits zero and introducing <UNKNOWN> (for unknown words) and <PADDING> (for unequal lengths) tags.
- Word to ID and Label to ID dictionaries were created to give a unique number to every word and label encountered so that it could be accessed using that unique number later.
- Based on whether random or Glove embeddings were to be used, the words were mapped to their respective embeddings.
- A simple BiLSTM model was used for Tasks 1 & 2 whereas a BiLSTM-CNN model was used for Task 3.
- Batching was performed to reduce overfitting and to increase the computation speed as well as the performance of the model.
- The model was trained for the set number of epochs, and the model files were stored in the current working directory. When the test_flag was set to True, while submitting the final code, the already trained model was directly loaded, and tag predictions were made for dev and test files respectively.
- The performance metrics to be considered for this homework were precision, recall and mainly F1-score since we were given an imbalanced data set with majority NER tags being ‘O’.