HW4 Report

Answers and Scores

Task 1 (Simple BLSTM)
The output from the conll03eval.txt script for dev data:

```
processed 51578 tokens with 5942 phrases; found: 5524 phrases; correct: 4506.
accuracy:  95.65%; precision:  81.57%; recall:  75.83%; FB1:  78.60
          LOC: precision:  89.12%; recall:  80.68%; FB1:  84.69  1663
         MISC: precision:  82.93%; recall:  73.75%; FB1:  78.07  820
          ORG: precision:  70.77%; recall:  68.61%; FB1:  69.67  1300
          PER: precision:  81.79%; recall:  77.31%; FB1:  79.49  1741
```

Task 2 (GlOve BLSTM)
The output from the conll03eval.txt script for dev data:

```
processed 51578 tokens with 5942 phrases; found: 6059 phrases; correct: 5476.
accuracy:  98.56%; precision:  90.38%; recall:  92.16%; FB1:  91.26
          LOC: precision:  94.55%; recall:  95.37%; FB1:  94.96  1853
         MISC: precision:  83.72%; recall:  86.44%; FB1:  85.06  952
          ORG: precision:  83.61%; recall:  86.73%; FB1:  85.14  1391
          PER: precision:  94.69%; recall:  95.77%; FB1:  95.22  1863
```

Bonus Task
The output from the conll03eval.txt script for dev data:

```
processed 51578 tokens with 5942 phrases; found: 6037 phrases; correct: 5471.
accuracy:  98.54%; precision:  90.62%; recall:  92.07%; FB1:  91.34
          LOC: precision:  93.92%; recall:  95.86%; FB1:  94.88  1875
         MISC: precision:  83.14%; recall:  87.20%; FB1:  85.12  967
          ORG: precision:  86.75%; recall:  86.43%; FB1:  86.59  1336
          PER: precision:  93.98%; recall:  94.84%; FB1:  94.41  1859
```

HyperParameters

Task 1 (Simple BLSTM)

The parameters defined in the question are not mentioned

```
hidden_dim = 256
Learning rate = 1 with a multiplier of 0.1 after every 20 epochs
batch_size = 64
num_epochs = 40
```

Task 2 (GlOve BLSTM)

```
The parameters defined in the question are not mentioned
embedding_dim = 101
hidden_dim = 256
Learning rate = 1 with a multiplier of 0.1 after every 20 epochs
batch_size = 64
num_epochs = 40
```

Bonus Task

```
The parameters defined in the question are not mentioned
char_embedding_dim = 30
word_embedding_dim = 101
hidden_dim = 256
Learning rate = 1 for epochs 1-5, 0.1 for 6-30 and 0.01 for 31-50
batch_size = 64
num_epochs = 50
Kernel size = 3
Stride = 1
```

Explanations

Data Reading

Read the train, dev, and test data from the given dataset files and created tokenized sentences.

Task1

The following steps are followed for Task 1:

- A vocab is created from the training data sentences. 2 extra special tokens, "<pad>" (index=0) and "<unk>" (index=1), are added for handling the padding in the sentences and handling unknown words and words with frequency less than and equal to 2 respectively.
- We create numerical vectors for each sentence in train, dev and test data using the created vocab, where each word is represented by its index in the vocab. All the unknown words are given this index of 1. (corresponding to <unk>)
- We do similar numerical conversions for the labels list as well. (without the unknown case). For padding, we use -1 in the labels.

- A NERDataset class is created to pass to DataLoader. It also returns the lengths of each sentence other than the sentences and its labels. This helps in pack_padded_sequence and pad_packed_sequence in the model forward method.
- Pad_collate helps to pad each sentence and their labels to max length in that batch. Labels are padded with -1 and Sentences with 0 (corresponding to <pad> token).
- The model architecture is created as suggested. The initial embedding layer takes in the input and then the output is first pack_padded and then passed through LSTM layers and then pad_packed back. We then pass it though a dropout, linear, elu and final linear output layer. The cross entropy loss is used on the final output.
- The model trained is used to predict the dev and test files and write out in respective files

Task 2

The following steps are followed for Task 2:

- The provided glove file is used to create vocab and their vectors. The words not present in the vocab are mapped to <unk> and another <pad> token (index=0) is added to the vocab for padding index. The <unk> and <pad> are randomly initialized. The 100 size embedding is changed to 101 size embedding by adding 1 or 0 to the end to signify if the word is capitalized. There for each word in glove.txt we get 2 vector embeddings of size 101 in which the first 100 values are same and the last index value is 0 or 1.
- We create numerical vectors for each sentence in train, dev and test data using the created vocab, where each word is represented by its index in the vocab.
- We do similar numerical conversions for the labels list as well similar to Task 1.
- Rest of the steps are similar to that in the Task 1. Just the dimension of the embedding layer is different due to vocab size and embedding dim difference. The weights of the embedding layer are initialized using the modified vectors retrieved from the glove.txt as explained in step(1).
- We achieve much better results due to using the initial pretrained weights.

Bonus Task

The following steps are followed for the bonus task:

- We create a vocab of characters in a similar way we created for words in Task 1.
- We create numerical character vectors using this vocabulary.
- These vectors are then passed through a embedding layer and CNN layer after padding the vectors with 0 to match the length of largest word and sentence.
- The output if then concatenated with the embedding output of Task2.
- The concatented embedded vector is passed though the LSTM and other layers in same way as Task 2 to get the final output.