Michael Royster

CSCI 544 – Homework #4

Task 1: Simple Bidirectional LSTM model

```
accuracy:  95.38%; precision:  77.46%; recall:  77.55%; FB1:  77.50
             LOC: precision:  85.27%; recall:  83.51%; FB1:  84.38  1799
            MISC: precision:  72.88%; recall:  73.75%; FB1:  73.32  933
             ORG: precision:  69.15%; recall:  69.87%; FB1:  69.51  1355
             PER: precision:  78.25%; recall:  79.10%; FB1:  78.67  1862
```

I used a batch_size of 4, learning_rate of 0.1 and a learning rate scheduling scheme to reduce the learning rate multiplicatively by 0.9 after each epoch. After many trial runs, it became clear that the large learning rate was necessary and that smaller batches also produced better results. The results from the example grading script are above.

Precision: 77.46

Recall: 77.55

F1: 77.50

Task2: Using GloVe word embeddings

```
accuracy:  98.54%; precision:  90.14%; recall:  92.58%; FB1:  91.34
             LOC: precision:  94.72%; recall:  95.75%; FB1:  95.24  1857
            MISC: precision:  82.43%; recall:  86.01%; FB1:  84.18  962
             ORG: precision:  83.58%; recall:  87.32%; FB1:  85.41  1401
             PER: precision:  94.42%; recall:  96.53%; FB1:  95.46  1883
```

I used the exact same hyperparameters and learning rate scheduling scheme for the LSTM utilizing the GloVe vectors. Adding the additional feature to identify whether the first letter was capitalized made a large difference in training. After beginning with the GloVe vectors and utilizing the capitalization feature my scores strongly outperformed the simple model.

Precision: 90.14

Recall: 92.58

F1: 91.34

Bonus Task: LSTM-CNN model

```
accuracy:  97.48%; precision:  84.02%; recall:  87.78%; FB1:  85.86
              LOC: precision:  91.41%; recall:  88.02%; FB1:  89.68  1769
             MISC: precision:  69.47%; recall:  79.72%; FB1:  74.24  1058
              ORG: precision:  75.10%; recall:  82.55%; FB1:  78.65  1474
              PER: precision:  92.13%; recall:  95.39%; FB1:  93.73  1907
```

I chose to implement the char-level CNN with two convolutional layers. Any more than this and I was getting greatly diminished returns in performance per training time. I began with an output dimension of 60 and kernel size of 5 and tapered to an output size of 30 and kernel size of 3 in the second layer. I attempted larger channels with larger kernel sizes, but they did not perform well. Interestingly, a smaller learning rate than was used in the standard BiLSTM performed better in the LSTM-CNN model. However, I was disappointed to be unable to tune the LSTM-CNN to outperform the BiLSTM with GloVe embeddings.

Precision: 84.02

Recall: 87.78

F1: 85.86