Part 3 – Report

Name

Simon Raviv

Note

I have implemented batching using Pytorch LSTMCell as restricted in the requirements.

The requirements limit us to use 5 epochs. It imposes restrictions on the maximum accuracy we can get. I believe that without this restrictions, higher accuracies are possible. In addition, using smaller batches and training for longer time, can get higher accuracy as well.

Due to the big number of models, it was too much time consuming to conduct proper optimization on the hyperparameters. My strategy in this part was to do optimization on representation a (not too much, due to the amount of time needed) and tune it for other parts with the time I had.

Models parameters

I will present the models and parameters with Pytorch model structs.

Common between all the models:

- Optimizer: ADAM
- Learning rate scheduling: Linear (When used)
- Dropout: Implemented for linear/MLP layers, but not used due to the need to use more epochs to get high accuracies.
- Weight decay: Implemented, but not used due to the need to use more epochs to get high accuracies.
- Epochs: 5
- Train/dev ratio: 0.9/0.1

POS

Representation a

- batch-size=32
- embedding-dim=100
- lstm-hidden-dim=200
- lr=0.003
- sched-step=1
- sched-gamma=1

```
BiLSTMTagger(
  (word_representation): Embedding(37864, 100, padding_idx=34055)
  (bi_lstm_layer_1): BiLSTM(
    (lstm_forward): LSTM(
      (lstm_cell): LSTMCell(100, 200)
    )
    (lstm_backward): LSTM(
      (lstm_cell): LSTMCell(100, 200)
    )
  )
  (bi_lstm_layer_2): BiLSTM(
    (lstm_forward): LSTM(
      (lstm_cell): LSTMCell(400, 200)
    )
    (lstm_backward): LSTM(
      (lstm_cell): LSTMCell(400, 200)
    )
  )
  (fc): Sequential(
    (0): Linear(in_features=400, out_features=46, bias=True)
    (1): Dropout(p=0.0, inplace=False)
  )
  (softmax): Softmax(dim=2)
)
```

Representation b

- batch-size=50
- embedding-dim=50
- lstm-hidden-dim=100
- lr=0.01
- sched-step=1
- sched-gamma=0.7

```
BiLSTMTagger(
  (word_representation): CharacterLSTM(
    (embedding): Embedding(87, 50, padding_idx=73)
    (lstm): LSTM(
      (lstm_cell): LSTMCell(50, 100)
    )
  )
  (bi_lstm_layer_1): BiLSTM(
    (lstm_forward): LSTM(
      (lstm_cell): LSTMCell(100, 100)
    )
    (lstm_backward): LSTM(
      (lstm_cell): LSTMCell(100, 100)
    )
  )
  (bi_lstm_layer_2): BiLSTM(
    (lstm_forward): LSTM(
      (lstm_cell): LSTMCell(200, 100)
    )
    (lstm_backward): LSTM(
      (lstm_cell): LSTMCell(200, 100)
    )
  )
  (fc): Sequential(
    (0): Linear(in_features=200, out_features=46, bias=True)
    (1): Dropout(p=0.0, inplace=False)
  )
  (softmax): Softmax(dim=2)
)
```

Representation c – Best Model

- batch-size=32
- embedding-dim=100
- lstm-hidden-dim=200
- lr=0.005
- sched-step=1
- sched-gamma=1

```
BiLSTMTagger(
  (word_representation): CBOWSubword(
    (embedding): Embedding(44139, 100, padding_idx=38780)
  )
  (bi_lstm_layer_1): BiLSTM(
    (lstm_forward): LSTM(
      (lstm_cell): LSTMCell(100, 200)
    )
    (lstm_backward): LSTM(
      (lstm_cell): LSTMCell(100, 200)
    )
  )
  (bi_lstm_layer_2): BiLSTM(
    (lstm_forward): LSTM(
      (lstm_cell): LSTMCell(400, 200)
    )
    (lstm_backward): LSTM(
      (lstm_cell): LSTMCell(400, 200)
    )
  )
  (fc): Sequential(
    (0): Linear(in_features=400, out_features=46, bias=True)
    (1): Dropout(p=0.0, inplace=False)
  )
  (softmax): Softmax(dim=2)
)
```

Representation d

- batch-size=50
- embedding-dim=50
- lstm-hidden-dim=100
- lr=0.01
- sched-step=1
- sched-gamma=0.7

```
BiLSTMTagger(
  (word_representation): WordEmbeddingCharacterLSTM(
    (word_embedding): Embedding(37864, 50, padding_idx=1005)
    (char_lstm): CharacterLSTM(
      (embedding): Embedding(87, 25, padding_idx=36)
      (lstm): LSTM(
        (lstm_cell): LSTMCell(25, 100)
      )
    )
    (fc): Sequential(
      (0): Linear(in_features=150, out_features=75, bias=True)
      (1): Dropout(p=0.0, inplace=False)
    )
  )
  (bi_lstm_layer_1): BiLSTM(
    (lstm_forward): LSTM(
      (lstm_cell): LSTMCell(75, 100)
    )
    (lstm_backward): LSTM(
      (lstm_cell): LSTMCell(75, 100)
    )
  )
  (bi_lstm_layer_2): BiLSTM(
    (lstm_forward): LSTM(
      (lstm_cell): LSTMCell(200, 100)
    )
    (lstm_backward): LSTM(
      (lstm_cell): LSTMCell(200, 100)
    )
  )
  (fc): Sequential(
    (0): Linear(in_features=200, out_features=46, bias=True)
    (1): Dropout(p=0.0, inplace=False)
  )
  (softmax): Softmax(dim=2)
)
```

NER

Representation a

- batch-size=32
- embedding-dim=50
- lstm-hidden-dim=50
- lr=0.008
- sched-step=1
- sched-gamma=1

```
BiLSTMTagger(
  (word_representation): Embedding(22167, 50, padding_idx=1410)
  (bi_lstm_layer_1): BiLSTM(
    (lstm_forward): LSTM(
      (lstm_cell): LSTMCell(50, 50)
    )
    (lstm_backward): LSTM(
      (lstm_cell): LSTMCell(50, 50)
    )
  )
  (bi_lstm_layer_2): BiLSTM(
    (lstm_forward): LSTM(
      (lstm_cell): LSTMCell(100, 50)
    )
    (lstm_backward): LSTM(
      (lstm_cell): LSTMCell(100, 50)
    )
  )
  (fc): Sequential(
    (0): Linear(in_features=100, out_features=6, bias=True)
    (1): Dropout(p=0.0, inplace=False)
  )
  (softmax): Softmax(dim=2)
)
```

Representation b

- batch-size=50
- embedding-dim=32
- lstm-hidden-dim=50
- lr=0.01
- sched-step=1
- sched-gamma=0.7

```
BiLSTMTagger(
  (word_representation): CharacterLSTM(
    (embedding): Embedding(88, 32, padding_idx=46)
    (lstm): LSTM(
      (lstm_cell): LSTMCell(32, 50)
    )
  )
  (bi_lstm_layer_1): BiLSTM(
    (lstm_forward): LSTM(
      (lstm_cell): LSTMCell(50, 50)
    )
    (lstm_backward): LSTM(
      (lstm_cell): LSTMCell(50, 50)
    )
  )
  (bi_lstm_layer_2): BiLSTM(
    (lstm_forward): LSTM(
      (lstm_cell): LSTMCell(100, 50)
    )
    (lstm_backward): LSTM(
      (lstm_cell): LSTMCell(100, 50)
    )
  )
  (fc): Sequential(
    (0): Linear(in_features=100, out_features=6, bias=True)
    (1): Dropout(p=0.0, inplace=False)
  )
  (softmax): Softmax(dim=2)
)
```

Representation c – Best Model

- batch-size=32
- embedding-dim=50
- lstm-hidden-dim=50
- lr=0.008
- sched-step=1
- sched-gamma=1

```
BiLSTMTagger(
  (word_representation): CBOWSubword(
    (embedding): Embedding(28750, 50, padding_idx=3433)
  )
  (bi_lstm_layer_1): BiLSTM(
    (lstm_forward): LSTM(
      (lstm_cell): LSTMCell(50, 50)
    )
    (lstm_backward): LSTM(
      (lstm_cell): LSTMCell(50, 50)
    )
  )
  (bi_lstm_layer_2): BiLSTM(
    (lstm_forward): LSTM(
      (lstm_cell): LSTMCell(100, 50)
    )
    (lstm_backward): LSTM(
      (lstm_cell): LSTMCell(100, 50)
    )
  )
  (fc): Sequential(
    (0): Linear(in_features=100, out_features=6, bias=True)
    (1): Dropout(p=0.0, inplace=False)
  )
  (softmax): Softmax(dim=2)
)
```

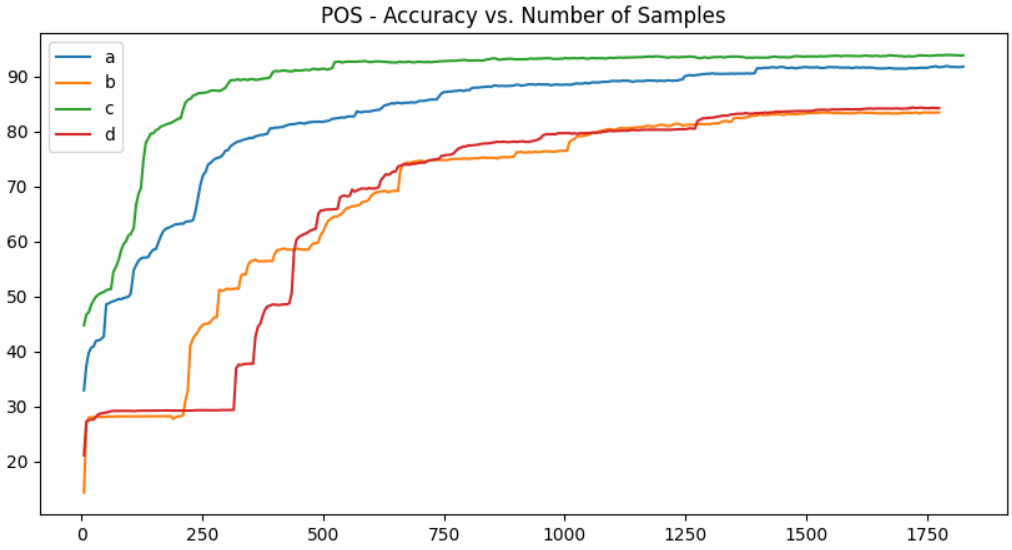
Representation d

- batch-size=50
- embedding-dim=32
- lstm-hidden-dim=50
- lr=0.01
- sched-step=1

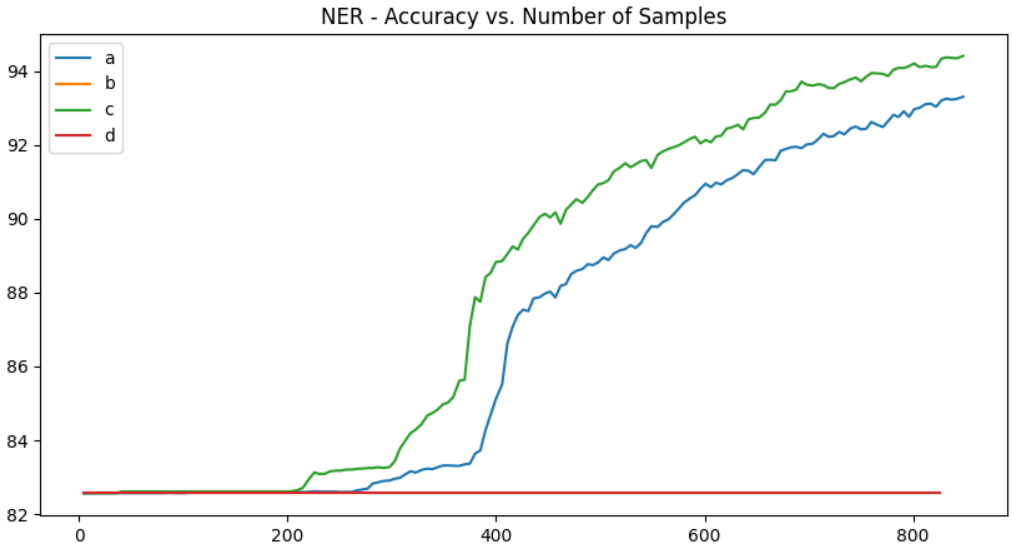
```
BiLSTMTagger(
  (word_representation): WordEmbeddingCharacterLSTM(
    (word_embedding): Embedding(22167, 32, padding_idx=9492)
    (char_lstm): CharacterLSTM(
      (embedding): Embedding(88, 16, padding_idx=36)
      (lstm): LSTM(
        (lstm_cell): LSTMCell(16, 50)
      )
    )
    (fc): Sequential(
      (0): Linear(in_features=82, out_features=41, bias=True)
      (1): Dropout(p=0.0, inplace=False)
    )
  )
  (bi_lstm_layer_1): BiLSTM(
    (lstm_forward): LSTM(
      (lstm_cell): LSTMCell(41, 50)
    )
    (lstm_backward): LSTM(
      (lstm_cell): LSTMCell(41, 50)
    )
  )
  (bi_lstm_layer_2): BiLSTM(
    (lstm_forward): LSTM(
      (lstm_cell): LSTMCell(100, 50)
    )
    (lstm_backward): LSTM(
      (lstm_cell): LSTMCell(100, 50)
    )
  )
  (fc): Sequential(
    (0): Linear(in_features=100, out_features=6, bias=True)
    (1): Dropout(p=0.0, inplace=False)
  )
  (softmax): Softmax(dim=2)
)
```

Graphs

POS



NER



Note: NER accuracy calculated regularly, with the common tag included in calculations.

Conclusion

- The best representation on both tasks was representation c – CBOW of word and sub word embeddings.
- Representation using Character LSTM showed very close results.

- Character LSTM and Word embedding with character LSTM didn’t train well on NER task using the parameters I was able to find.