Assignment 3 Report

Daniel Hsieh, Lucas Schmitt, Devasha Trivedi

June 14, 2023

1. Programming: Text Classification with Neural Networks

Model	Training Set Accuracy	Test Set Accuracy	
RNN	0.986	0.661	
LSTM	0.942	0.722	

We decided to use 10 epochs instead of 20 when fitting the model because after the 10th epoch, there was not much improvement. Through these results, we can see that the RNN and LSTM models were both close in accuracy on the training set, and the LSTM model outpeforms the RNN model on the test set. Given what we expected in terms of performance, the LSTM model outperforming the RNN model is not a surprise and fits with our expectations.

2. Deriving the Viterbi Algorithm

2.1.

We are given that for every possible value of y_j ,

$$v_j(y_j) = \max_{y_1 \ , \quad \ldots \ , \ y_{j-1}} s(x,\ i,\ y_{j-1},\ \ y_i)$$

This is equivalent to

$$\max_{y_{j-1}}[\max_{y_1 \ , \quad \ldots \ , \ y_{j-2}} s(x,\ j,\ y_{j-1},\ \ y_j) + s(x,\ i,\ y_{i-1},\ \ y_i)]$$

which equals

$$\max_{y_{j-1}}[s(x,\ j,\ y_{j-1},\ \ y_j) + max_{y_1 \ , \quad \ldots \ , \ y_{j-2}} s(x,\ i,\ y_{j-1},\ \ y_i)]$$

From the initial equation we can get that

$$\max_{y_1 \ , \quad \ldots \ , \ y_{j-2}} s(x,\ i,\ y_{j-1},\ \ y_i)$$

is equivalent to

$$v_{j-1}(y_{j-1})$$

so we can substitute that back into the equation to get

$$v_j(y_j) = \max_{y_{j-1}}[s(x,\ j,\ y_{j-1},\ \ y_j) + v_{j-1}(y_{j-1})].$$

Thus we have shown that for every possible value of y_j, where

$$v_j(y_j) = \max_{y_1 \ , \quad \ldots \ , \ y_{j-1}} s(x,\ i,\ y_{j-1},\ \ y_i) \ , v_j(y_j) = \max_{y_{j-1}}[s(x,\ i,\ y_{j-1},\ \ y_j) + v_{j-1}(y_{j-1})]$$

is true.

2.2.

The Viterbi algorithm can be represented as a matrix of size T x n where T is the total number of possible tags or classes and n is the length of the sequence or sentence. For each square in the T x n matrix, we need to look through T different things/classes with argmax. Therefore, the time complexity of the Viterbi algorithm is O(n|T|^2).

3. The Viterbi Algorithm

In this section, we implement the Viterbi algorithm for named entity recognition. We do this by implementing a decode() fucntion. This function takes an input length, set of tags, and score function. For the viterbi algorithm, we create two matricies of size (len(tagset) x len(input)). These matricies keep track of the optimal score and node choice at any [tag][token] pair. From there. We populate these matrices using the provided score function, such that:

$$score[i][j] = max_k(scores[k, j-1] + score(tagset[i], tagset[k], j)$$
$$tags[i][j] = argmax_k(scores[k, j-1] + score(tagset[i], tagset[k], j) \text{ For all tags } i \text{ and input token } j$$

We can then backtrack these matrices, starting with the tag that gives us the maximum score for $score[i][len(input)]$, where the optimal tag for input token $j-1$ is:

$$optimal\_tag[j-1] = tags[optimal\_tag[j-1]][j]$$

Using this method, we backward propagate the optimal path, providing us with the optimal solution for this algorithm.
Using this method, we get the following accuracy, precision, recall, and F_1:

%	ner.dev	ner.test
Accuracy	89.61	88.02
Percision	59.80	53.28
Recall	41.25	37.41
F_1	48.82	43.96

We get the following full output for the dev file:

```
processed 51578 tokens with 5917 phrases; found: 4082 phrases; correct: 2441.
accuracy:  41.38%; (non-O)
accuracy:  89.61%; precision:  59.80%; recall:  41.25%; FB1:  48.82
              LOC: precision:  87.53%; recall:  58.31%; FB1:  69.99  1219
             MISC: precision:  69.94%; recall:  63.89%; FB1:  66.78  835
              ORG: precision:  36.04%; recall:  42.65%; FB1:  39.07  1587
              PER: precision:  49.43%; recall:  11.90%; FB1:  19.18  441
```

and the following full output for the test file:

```
processed 46666 tokens with 5616 phrases; found: 3943 phrases; correct: 2101.
accuracy:  37.73%; (non-O)
accuracy:  88.02%; precision:  53.28%; recall:  37.41%; FB1:  43.96
              LOC: precision:  86.52%; recall:  55.88%; FB1:  67.91  1076
             MISC: precision:  54.45%; recall:  50.64%; FB1:  52.48  652
              ORG: precision:  37.26%; recall:  44.93%; FB1:  40.74  1986
              PER: precision:  32.75%; recall:   4.68%; FB1:   8.19  229
```