Movie Review Sentiment Analysis Project

Vishwa Vijayasankar

October 15, 2024

1.1

The model contains an embedding layer, simpleRNN layer, and a feedforward layer. After 20 training epochs, with an Adam optimizer using a learning rate 10^{-3} as suggested, I obtained a training accuracy of 95.33%, and a test accuracy of 70.57%.

Metric	Accuracy	Loss
Train (20th Epoch)	0.953	0.1224
Test	0.706	1.0037

1.2.1

The model I used here was mostly identical to 1.1, save for swapping the simpleRNN layer with an LSTM layer.

After 20 training epochs using the same recommended hyperparameters, I obtained a training accuracy of 95.57%, and a test accuracy of 73.33%.

Metric	Accuracy	Loss
Train (20th Epoch)	0.946	1.430
Test	0.733	0.8557

1.2.2

With my trained models, I can evaluate them on a truncated version of the test set based on varying sequence lengths. I set a lower limit of sequence length (by character) to see what would happen as I restrict the sequence lengths to be longer and longer.

As you can see, initially, the LSTM does not perform as well as the RNN, but with longer sequences, the accuracy improves. However, as we go beyond 5000 or 6000 characters, the LSTM's accuracy collapses. This could be due to discrepancies between the training and test data.

Model	3000	4000	5000	6000	7000
RNN	0.660	0.650	0.627	0.580	0.571
LSTM	0.646	0.655	0.644	0.500	0.429

2.1

Let, for every possible value of y_j,

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

Show that

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

There are two parts to showing equivalence. First, the recursive call in the second equation can be expressed as a summation. Then, we modify the max function to work with the iterative version.

$$v_j(y_j) = s(x, j, y_{j-1}, y_j) + v_{j-1}(y_{j-1})$$
$$v_j(y_j) = s(x, j, y_{j-1}, y_j) + s(x, j-1, y_{j-2}, y_{j-1}) + v_{j-2}(y_{j-2})$$

Continue this until $j = 1$:

$$v_j(y_j) = s(x, j, y_{j-1}, y_j) + \ldots + s(x, 0, y_0, y_1).$$

2.2

The Viterbi algorithm computes the optimal sequence $\hat{y}$ by maximizing the transition probabilities over all possible tags. The algorithm has a time complexity of $O(n|K|^2)$.

3.1.1

Below are the results of running the model on the dev and test data.

Set	Precision	Recall	F_1
Test	53.2%	37.31%	43.96
Dev	59.80%	41.25%	48.82

3.1.2

Below is the detailed breakdown of results for both the dev and test sets.

Category	Precision (%)	Recall (%)	F_1
Overall	59.80	41.25	48.82
LOC	87.53	58.31	69.99
MISC	69.94	63.89	66.78
ORG	36.04	42.65	39.07
PER	49.43	11.90	19.18

Category	Precision (%)	Recall (%)	FB1 (%)
Overall	53.28	37.41	43.96
LOC	86.52	55.88	67.91
MISC	54.45	50.64	52.48
ORG	37.26	44.93	40.74
PER	32.75	4.68	8.19