Sequence Models/POS Tagging Analysis

Amol Sharma

26th February 2019

This project explores sequence models by analyzing the performance of two models. The first is a simple tagger which (although it uses a neural network) labels words with their parts of speech independently of the context of the word. The second uses a Conditional Random Field method which does consider the context of the word. As part of the assignment, the properties of a Viterbi algorithm for second-order sequence models is also discussed.

1 Simple Twitter POS Tagger

A DatasetReader and Predictor were created to help implement facilitate the model. The reader iterates over the data, creating instances with the token and tag for each word. The custom predictor outputs prediction in tsv format.

Experiments

A grid-search was used to find a relatively good configuration. The results are given below.

Table 1: Grid Search for simple-twitter-tagger

Encoder Type	Batch Size	Optimizer	LR	Encoder Size	Train. Acc.	Val. Acc.
stacked_bidirectional_lstm	64	adagrad	0.05	512	1	0.8367
stacked_bidirectional_lstm	128	adagrad	0.01	512	0.9980	0.8614
rnn	128	adam	0.01	512	0.9611	0.7461
gru	128	sgd	0.05	512	0.4333	0.4329
stacked_bidirectional_lstm	128	adagrad	0.01	256	0.99802	0.8667

The model that performed best on the validation data was the last one. This model had a test-data accuracy of 0.8153.

2 CRF POS Tagging

The same DatasetReader and Predictor from part 1 were used in this part. Once again, a grid-search was used to find a good configuration. The same values were tested as they covered an acceptable range of variables. The results are shown on the next page.

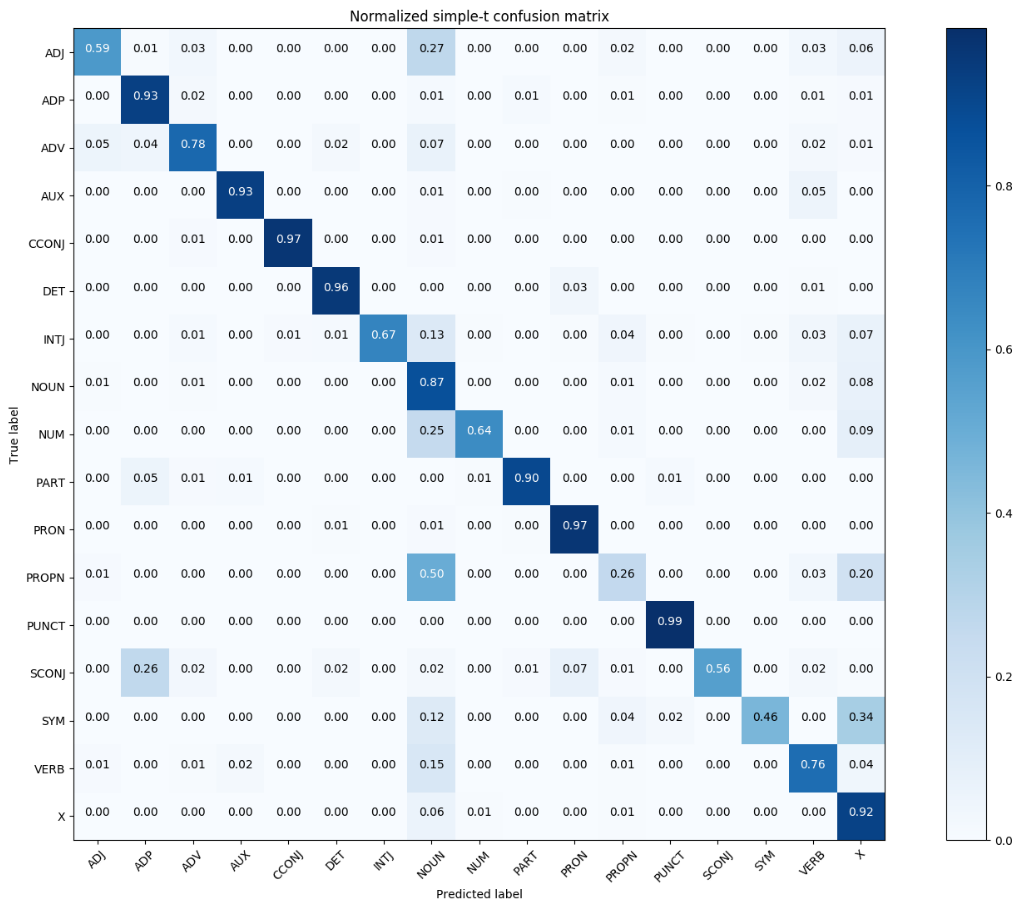
Table 2: Grid Search for crf-twitter-tagger

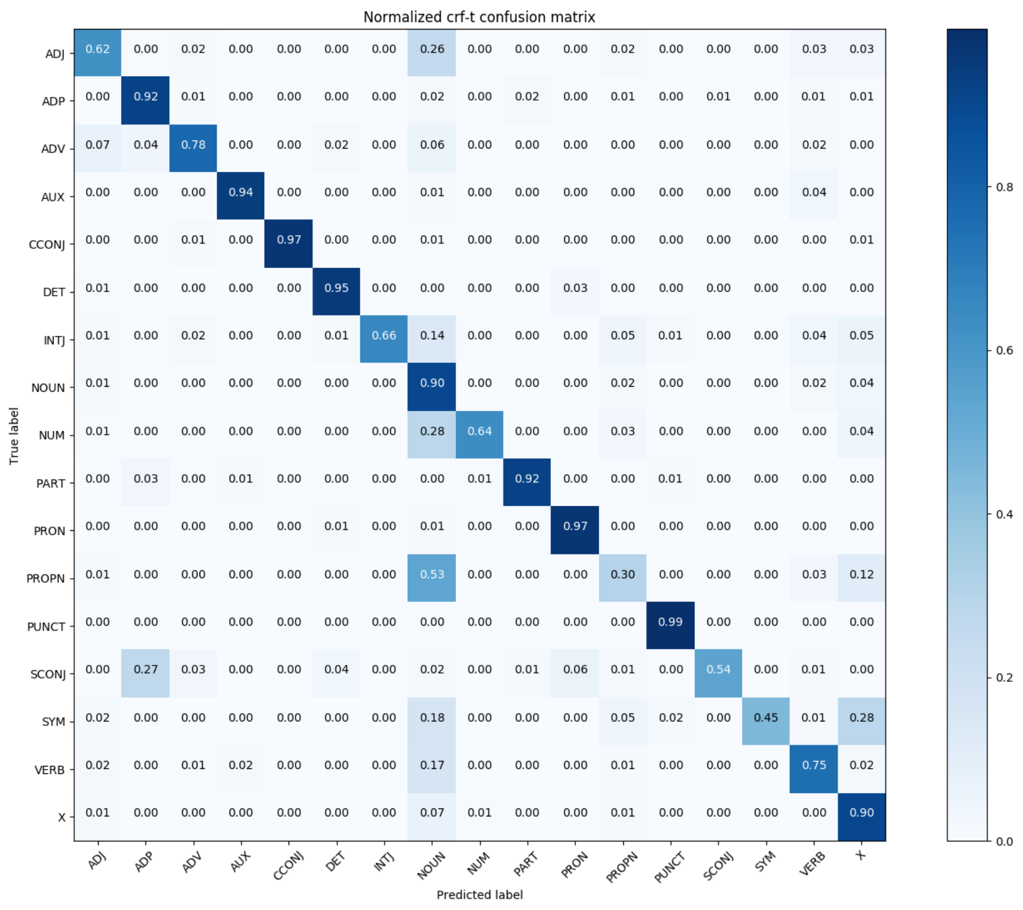
Encoder Type	Batch Size	Optimizer	LR	Encoder size	Train. Acc.	Val. Acc.
stacked_bidirectional_lstm	64	adagrad	0.05	512	1.0000	0.8339
stacked_bidirectional_lstm	128	adagrad	0.01	512	0.9989	0.8638
rnn	128	adam	0.01	512	0.9827	0.7737
gru	128	sgd	0.05	512	0.2009	0.2536
stacked_bidirectional_lstm	128	adagrad	0.01	256	0.9979	0.8666

This time as well, the last configuration was the most accurate on the validation data. This specific configuration's accuracy was almost identical to the simple tagger for validation data. However, the accuracy on test data was somewhat greater at 0.8226 (compared to 0.8153).

3 Automatic POS Tagging Analysis

The confusion matrices are given below.





Both matrices above are very similar. Both models show darker squares down the diagonal, indicating that both were pretty accurate. The CRF tagger seemed to perform better for identifying ADJ, NOUN, PART, and PROPN in terms of recall. The simple tagger was better at X while both very similar (difference of ≤ 0.01) for the rest.

Both models also had very similar mistakes shown by the squares shaded in the same places outside the diagonal. In both it seems that many parts of speech were incorrectly predicted to be nouns since there are 8 squares for the predicted NOUN label that are visibly shaded. This indicates the models lean towards labelling things as nouns. PROPN, SYM, and SCONJ had a spread in what they were identified as, indicating the models are not good at recognizing them.

PROPN stands out in the matrices as it is incorrectly identified more than it is correctly identified (30% of the time). In particular, it is labelled as a NOUN 50-53% of the time.

4 Viterbi

Recurrence

$$\begin{aligned}\heartsuit_i(y', y) &= max_{y'' \in \mathcal{L}} \left(s(\mathbf{x}, i-2, y'', y', y) + \heartsuit_{i-1}(y'', y') \right) \\ b_i(y', y) &= arg\ max_{y'' \in \mathcal{L}} \left(s(\mathbf{x}, i-2, y'', y', y) + \heartsuit_{i-1}(y'', y') \right)\end{aligned}$$

Space Complexity

The space complexity will be the number of boxes in the associated table. We will have l columns, and a row for every combination of two labels that could come before the current position. Therefore, the space complexity will be $|\mathcal{L}|^2 l$.

Time Complexity

Since we know the space complexity, the time complexity for this algorithm can be broken into (time taken per box × number of boxes). Therefore, we have $O\left(time\ per\ box \cdot |\mathcal{L}|^2 l\right)$. For each box, we know y' and y and are maximizing over y'', therefore we only have to look at the $|\mathcal{L}|$ possible labels for y''. Therefore, we get $O\left(|\mathcal{L}|^2 l \cdot |\mathcal{L}|\right) = O\left(|\mathcal{L}|^3 l\right)$