Natural Language Understanding Assignment 3

Udit Gupta
M.tech CDS
14454
udit2008.gupta@gmail.com

Dataset: Named entity recogniyion data; ner.txt
Libraries used: Keras(Tensorflow backend), Nltk, Numpy, sklearn-crfsuite
Preprocessing:The data is split into 70-10-20 train, validation and test. Converted the data in standard format each sentence is a list and each element of a list is a tuple of the word and its label.

Method 1: CRF

Metrics used for evaluation:Precision, Recall and F1-score.
Additional Features:

1. The POS tag of current word
2. Check whether previous word and next word are uppercase.
3. Check whether current word is in title case
4. The previous word and next word in lowercase.
5. The case(upper or lower) of current word.
6. Whether the current word is a digit.
7. The current word in lowercase.
8. The suffix of the current word.
9. Number of characters in the word

Training:L-BFGS algorithm with Elastic Net (L1 + L2) regularization for 100 iterations.**Training time:** 3.85 seconds.
Evaluation: This dataset has class imbalance. Instances corresponding to 'O' are much more than 'D' and 'T'. Naively predicting everything as 'O' gives 81% accuracy. So, Precision and recall have been used as metrics.

Evaluation

Label	Precision	Recall	F1-score	Support
'D'	0.779	0.710	0.743	990
'O'	0.951	0.974	0.962	11306
'T'	0.738	0.520	0.610	611
Average	0.928	0.932	**0.929**	12907
Average w/o 'O'	0.764	0.638	**0.692**	1601

Observations: The state transitions are shown below. It shows that it is highly likely to transition from Disease to Disease and Treatment to Treatment, but unlikely to transition from Treatment to Disease or Disease to Treatment.

```
Top likely transitions:
D      -> D       3.018441
O      -> O       2.253329
T      -> T       1.851464

Top unlikely transitions:
O      -> T       -1.849842
D      -> T       -3.022054
T      -> D       -5.158300
```

Important features:

```
Top positive:
5.327239 D         word.lower():depression
5.054657 D         word.lower():cataract
4.844047 T         word.lower():vaccination
4.394335 T         word.lower():slings
4.246045 D         word.lower():tumors
```

The model learns that some words such as depression are very likely to be a Disease. But it may also represent over-fitting. The important features **after turning off the word.tolower() feature** are shown below:

```
Top positive:
4.506283 O         word[-3:]:for
4.281012 O         word[-3:]:is
4.246555 T         word[-3:]:xel
3.989109 O         word[-3:]:on
```

This shows that the **suffix of the current word** is the second most important feature.
Conclusion: Though the overall F1-score is 93% it doesn't imply that the model is doing very well. This is because of class imbalance in the dataset. The final results are to be evaluated after removing label 'O'. The F1-score after removing label 'O' is 70%. This can be improved by using more sophisticated deep models using LSTMs.

Method 2: LSTM

Metric used for comparison: Accuracy on Training set and Test set.
Methodology:

1. Create a list of unique words in the dataset and map to an integer. Mapping called: word2index.

2. Create a list of unique tags in the dataset and map to an integer. Mapping called: tag2index.

3. Using the mappings word2index and tag2index, convert each sentence to sequence of integers. Pad each sequence to max size of sentence in dataset.

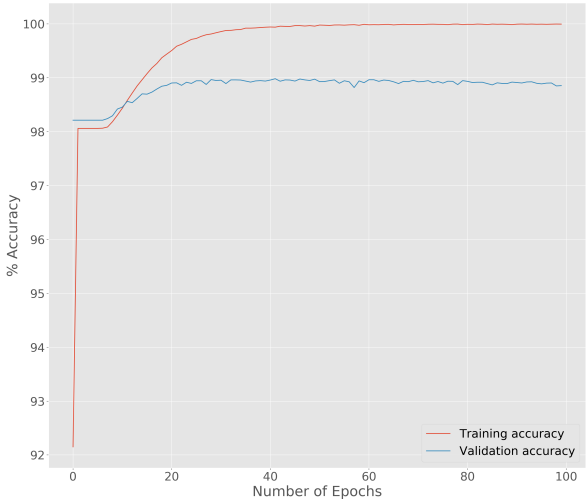
4. The tag sequence is converted to categorical form.

5. Generate 50 dimensional embeddings of the sequences.

6. Feed the sequences to Bidirectional LSTM, and finally use a dense layer with #tags output nodes and softmax activation.

Training: The model was compiled using categorical cross-entropy loss and rmsprop optimizer.

The model was trained for 100 epochs (batch size: 512).
Observations:The model gave accuracy of 98.7% on the test set.



Conclusion: LSTM gives a test set accuracy of 98.7% compared to 93% in CRF. So, sequence tagging using LSTMs performs better than the CRF technique used in 'Method 1'.

Method 3: LSTM-CRF

Metric used for comparison: Accuracy on Training set and Test set.
Methodology:

1. The input sequence is x = (x1,..., xm), i.e. the words of a sentence and the sequence of output states is s = (s1,..., sm), i.e. the named entity tags. In conditional random fields we model the conditional probability of the output state sequence give an input sequence.

2. This is done by defining a feature map $\Phi(x1, \ldots, xm, s1, \ldots, sm) \in \Re^d$ that maps an entire input sequence x paired with an entire state sequence s to some d-dimensional feature vector.

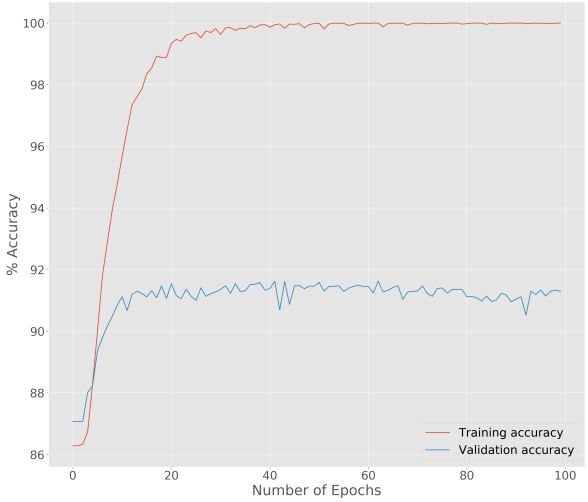
3. We can view the expression $w \cdot \Phi(x, s) = score_{crf(x,s)}$ as a scoring of how well the state sequence fits the given input sequence. (w is the parameter vector $w \in \Re^d$

4. Now, replace the linear scoring function by a non-linear neural network. So we define the score $score_{lstm-crf}(x, s) = \sum_{i=0}^{n} W_{s_{i-1},s_i} \cdot LSTM(x)_i + b_{s_{i-1},s_i}$

5. After constructing this score function, we optimize the conditional probability p(s—x; W,

b) like in the usual CRF and propagating back trough the network.

Training: Implemented using Keras contrib layer. # Epochs=100
Observations: The model gave an accuracy of 91% on test set.



Conclusions: The accuracy of LSTM-CRF model is not as good as that of plain LSTM model. However, the LSTM-CRF model may work well for a more sophisticated dataset in which number of tags are more and applying only a CRF model or LSTM model won't give good results.