Aneek Das

Fake Review Detector Task

March 26, 2017

Experiment Description

A dataset of positive and negative sentiments were provided, each of which was categorized into True or Fake reviews. A classifier was modeled that learned to classify the Fake and True reviews with maximum accuracy possible and good Precision, Recall and F measures.

Algorithm

Preparing the dataset (prep_data.py)

1. First of all the dataset had to be segregated in terms of Fake and True reviews and not sentiment wise. So, the True reviews from both positive and negative sentiments were merged and the same was done for Fake reviews.
2. Then the text converted to lowercase to avoid discrimination between same words in different cases. Then the text was parsed where it was tokenized into lexemes .
3. Further, the stop words (of, a, the ....) were removed to increase accuracy as these words are just used as connectors rather than providing meaning.
4. Then the words were Lemmatized to their base words. This helps discrimination between the same word having different affixes.
5. The words were converted to Number Encodings based on their order of presence in the lexicon.
6. The words were grouped into groups of 100 and the respective classification of [1, 0] for True and [0, 1] for Fake was appended for making the dataset.

7. Then the dataset was shuffled randomly .
8. The dataset was stored into a Pickle file for easy retrieval.

The LSTM Classifier (review_lstm.py)

1. First the dataset was loaded from the Pickle file.
2. Then the data and label part were segregated. Then the data part was Padded so that each data had a length of 100.
3. The data was then Reshaped into [-1, 100] and labels into [-1, 2] so as to have appropriate shape for the LSTM network.
4. The training data was 70% of the dataset and the testing data was the remaining 30% as provided in the problem transcript.
5. The LSTM model was trained and the metrics evaluated.(LSTM model discussed in detail below).

The LSTM Network

Why LSTM ?

A LSTM (Long Short Term Memory Network) is a variant of the vanilla RNN that has memory. It has gates that help it to decide which stuff to remember and what to forget. It also has an intermediate cell state that helps it make abstract representations of the data. LSTM’s have been known to be state of the art models for sequential data (especially text).

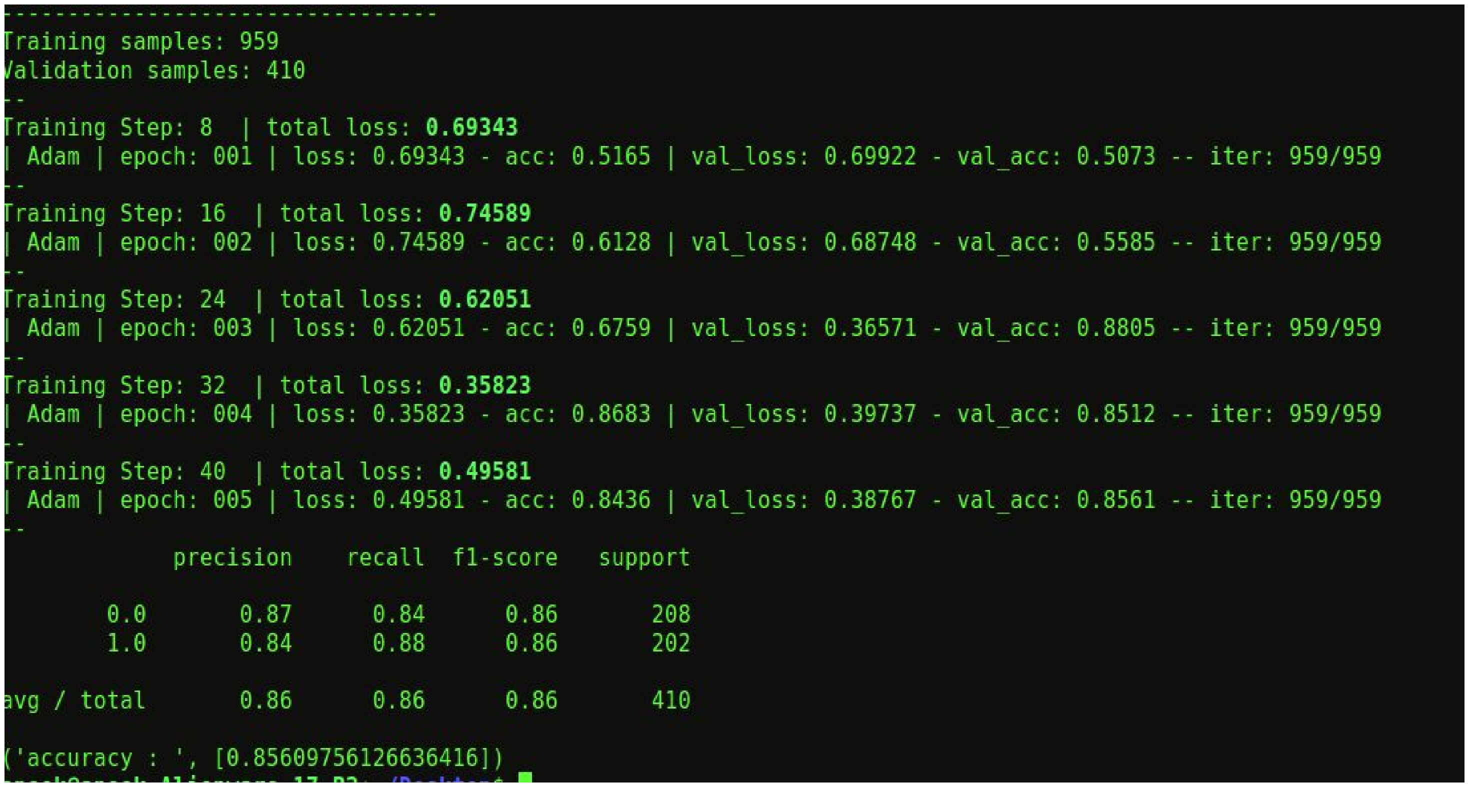
The Architecture

1. The network is a 9-Layer Network. It has 1 Input Layer, 1 Embedding Layer, 3 LSTM layers each having 1 Dropout Layer after it. The Output Layer is a Fully Connected Layer with 2 nodes (1 for each class).
2. The Input Layer has 100 nodes. The embedding layer takes each words and embeds them in space with similar words close to each other.
3. Each LSTM network has 512 nodes. A dropout of 0.5 is given so that the layer randomly drops half of it's nodes so that they don't become overly dependent on each other.
4. Softmax is used as activation to get binary classification output. While Adam is used as our optimizer as it includes Momentum and a Dynamic Learning Rate.
5. Loss is Categorical Cross-entropy.
6. The model was trained on a batch size of 128. It was trained of 5 Epochs.

Other models and the parameters

Several other models were tested. A Bi-Gram (Maximum E ntropy Model) model yielded 62% accuracy. A Bidirectional LSTM Network had an accuracy of 67% while a similar LSTM Network with 64 sequence length had an accuracy of 81%. So, the sequence length was increased to 100. The network was tested with LSTM Networks of 256 to 1024 LSTM nodes in each layer. While the 1024 Layer got a higher accuracy, it was not feasible as the training time was a lot. Each model was trained on several Epoch values of 1, 5, 7, 10 . It was observed for our current model if the Epochs was more than 5 , the model would overfit the training data and reduce accuracy on the test data. It was seen that the model over 10 iterations had 99.5% accuracy on the training data.

RESULTS



```
---------------------------------
Training samples: 959
Validation samples: 410
--
Training Step: 8  | total loss: 0.69343
| Adam | epoch: 001 | loss: 0.69343 - acc: 0.5165 | val_loss: 0.69922 - val_acc: 0.5073 -- iter: 959/959
--
Training Step: 16  | total loss: 0.74589
| Adam | epoch: 002 | loss: 0.74589 - acc: 0.6128 | val_loss: 0.68748 - val_acc: 0.5585 -- iter: 959/959
--
Training Step: 24  | total loss: 0.62051
| Adam | epoch: 003 | loss: 0.62051 - acc: 0.6759 | val_loss: 0.36571 - val_acc: 0.8805 -- iter: 959/959
--
Training Step: 32  | total loss: 0.35823
| Adam | epoch: 004 | loss: 0.35823 - acc: 0.8683 | val_loss: 0.39737 - val_acc: 0.8512 -- iter: 959/959
--
Training Step: 40  | total loss: 0.49581
| Adam | epoch: 005 | loss: 0.49581 - acc: 0.8436 | val_loss: 0.38767 - val_acc: 0.8561 -- iter: 959/959
--
             precision    recall  f1-score   support

        0.0       0.87      0.84      0.86       208
        1.0       0.84      0.88      0.86       202

avg / total       0.86      0.86      0.86       410

('accuracy : ', [0.85609756126636416])
```

So, our model has an accuracy of 85.6%. It may not be great but the best feature of the model is that it has almost 85% on both precision and recall for both the classes equally. So, it is a very well balanced model. The other metrics are given in the screenshot.