Submitted by: Himanshu Gupta

Detailed Description on the Assignment

Data Description

Training on Data used: trainSet

Testing on Data: candidateTestSet

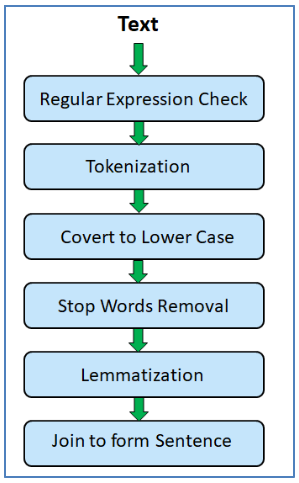
Result submitted in CSV file: candidateTestSetResult (search terms and their respective categories)

Model Selected: *Deep Learning Model with LSTM as hidden layer*

I have used the deep learning model with LSTM (RNN) as a hidden layer. The model is created using the sequential API with embedded layer dimension as 100. In order to reduce the over fitting regularization technique ‘SpatialDropout1D’ is used with ‘softmax’ as an activation function in output layer.

The early stopping of training model is supported using the callback, which will monitor the performance measure. I have used the epoch = 20 with bach_size = 64, the early stopping function will stop the training when there is no drop is loss and increase in accuracy independent of epoch value. So here the training got stopped at epoch = 14. The model is complied with ‘adam’ as an optimizer and loss = ‘categorical_crossentropy’ (as multiple classes (‘1419’) were present in the training dataset).

Pre-processing Steps



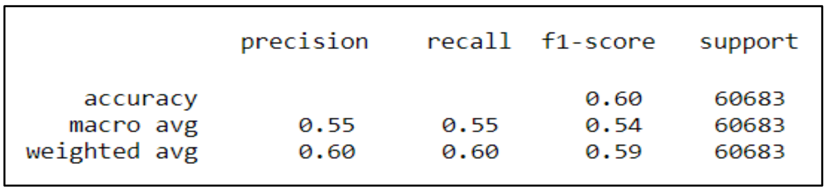
- The unwanted expressions are removed from the text.
- The texts are tokenised into words, so that further pre-processing on each word can be done easily.
- The tokens are converted from upper case to lower to make it uniform.
- The redundant words pertaining to articles, prepositions are removed from the tokenised words.
- The words are changed into base form using the technique ‘Lemmatization’.
- Eventually, the tokens are joined to form sentences (cleaned text).

Submitted by: Himanshu Gupta

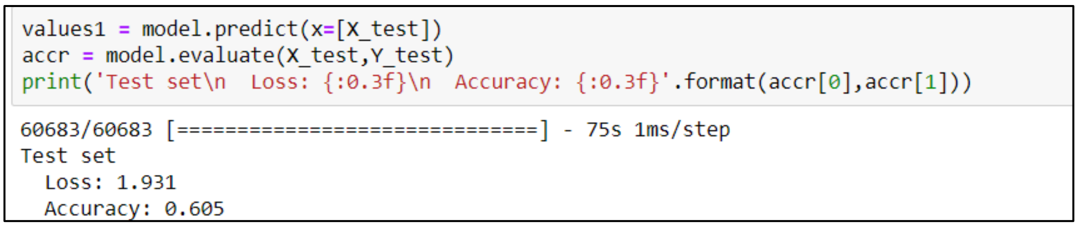
Method of Evaluation and Results

Evaluation Metrics: *Model accuracy, Model loss, Confusion Matrix, Precision, Recall and AUC.*

These metrics were used to evaluate the validation samples. Below are the results from these metrics:

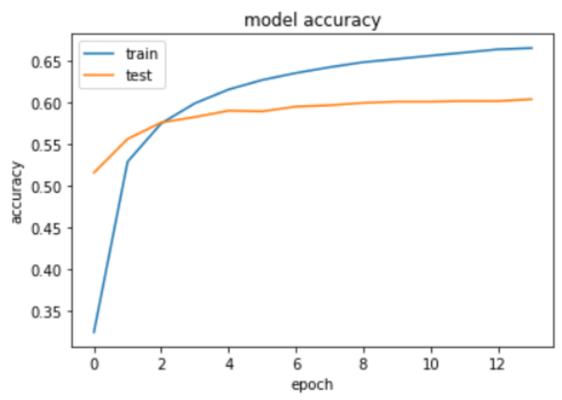


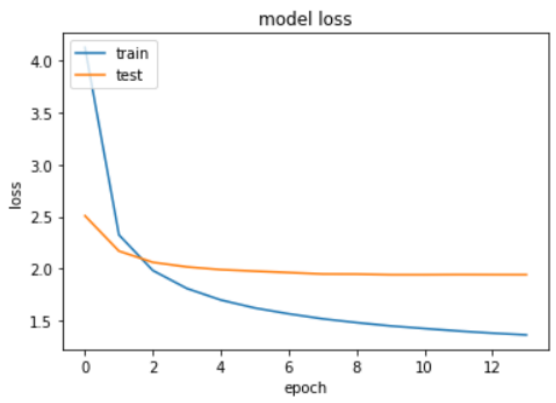
```
              precision    recall  f1-score   support

    accuracy                           0.60     60683
   macro avg       0.55      0.55      0.54     60683
weighted avg       0.60      0.60      0.59     60683
```

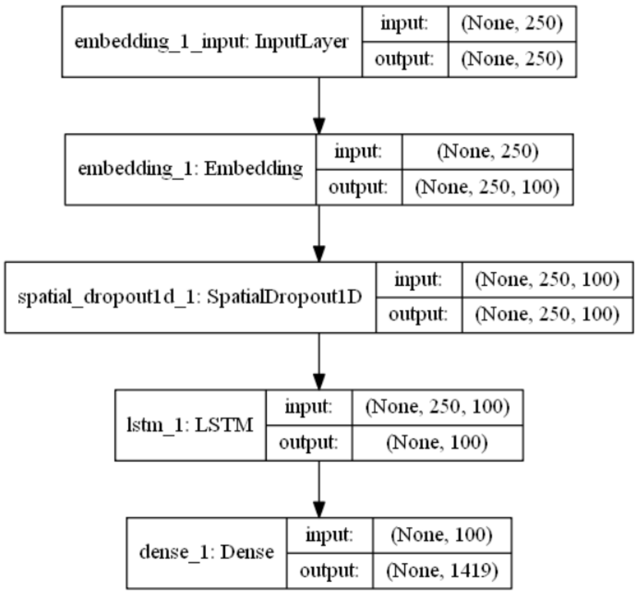


```
values1 = model.predict(x=[X_test])
accr = model.evaluate(X_test,Y_test)
print('Test set\n  Loss: {:0.3f}\n  Accuracy: {:0.3f}'.format(accr[0],accr[1]))
```

```
60683/60683 [==============================] - 75s 1ms/step
Test set
  Loss: 1.931
  Accuracy: 0.605
```







Model Plot figure

Challenge and Complexity

- The training was performed at epoch = 20, so it took significant amount of time for training.
- There were multiple classes (1419) present which made the classification a little intricate.

Future Work

- Various pre-trained word embedding models can be used during the model training using deep learning techniques (like Global vector trained on 6B words with dimension 200 => glove.6B.200d) can be employed as a future work.
- New hidden layers can also be added to this deep neural network which may increase the accuracy and decrease the loss.
- The number of features and word vector dimensions could be increased which will definitely increase the performance in prediction.

Limitation

- The number of features and dimension selected are less due to less computational power of my machine.
- A deep learning algorithm takes significant amount of time for training.