Sentiment Analysis

The entire report is divided into three sections: the preprocessing of the data, the development and training of the model, and its results and findings.

Preprocessing of data:

1. To load the data, we employ the OS library function. The first function, "load," returns reviews from each file, and the second, "load_1," provides a list of data for each positive and negative that has been joined together and will be used later to build the dataset.
2. Subsequently, we produced two datasets with the corresponding labels. The first is test data in test_NLP.py and training data in train_NLP.py.
3. Using the 'preprocess_review' function, we then eliminated all special characters and HTML tags from the dataset.
4. Next, we used the NLTK library's (English Stopwords Collection), 'remove_stop_words' function to remove every stopword from the dataset.
5. Then, using the keras tokenizer to tokenize every word in the document, we run the fit_transform operation on the training set. The tokenizer model is then saved as a .pkl file so that it may be used again with test data.
6. At the last step we create training data and validation data split in the ration of 80:20.

The training and development of the model:

1. The model was created using one embedding layer, one conv1D layer with parameters, filter_size=32, kernal_size=2 and activation function as leaky Relu.
2. Next layer is GlobalAverageMaxPooling1D layer.
3. Next three layers are Dense layer with 512, 256 and 64 neurons respectively with each layer having a dropout of 10% and activation function as leaky Relu.
4. At the last layer we have 1 neuron with sigmoid activation function which classifies the movie review as positive or negative.
5. For loss calculation we use ‘binary_crossentropy’ function along with ‘adam’ optimizer with learning rate of 0.001.
6. We train model with 10 epochs and 100 batch size and we accuracy as the metric for evaluation. Below fig. shows the performance of model at each epoch.
7. We store the model in .h5 format to use in test_NLP.py file for evaluation on test data.

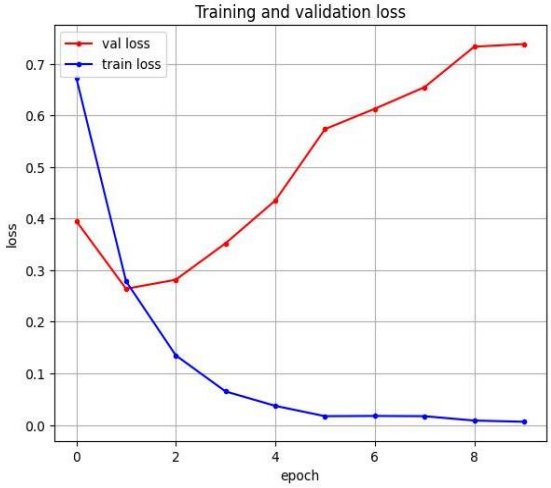
```
Vinamra@LAPTOP-I6POQ8U4 MINGW64 /e/Waterloo Courses/ECE 657/Assigments/Assignment 3/Answer 3
$ python train_NLP.py
Epoch 1/10
200/200 - 24s - loss: 0.6724 - accuracy: 0.5350 - val_loss: 0.3950 - val_accuracy: 0.8342 - 24s/epoch - 119ms/step
Epoch 2/10
200/200 - 23s - loss: 0.2782 - accuracy: 0.8856 - val_loss: 0.2637 - val_accuracy: 0.8938 - 23s/epoch - 116ms/step
Epoch 3/10
200/200 - 23s - loss: 0.1348 - accuracy: 0.9502 - val_loss: 0.2814 - val_accuracy: 0.8958 - 23s/epoch - 116ms/step
Epoch 4/10
200/200 - 24s - loss: 0.0649 - accuracy: 0.9791 - val_loss: 0.3520 - val_accuracy: 0.8912 - 24s/epoch - 118ms/step
Epoch 5/10
200/200 - 24s - loss: 0.0370 - accuracy: 0.9886 - val_loss: 0.4348 - val_accuracy: 0.8842 - 24s/epoch - 118ms/step
Epoch 6/10
200/200 - 23s - loss: 0.0169 - accuracy: 0.9946 - val_loss: 0.5732 - val_accuracy: 0.8764 - 23s/epoch - 117ms/step
Epoch 7/10
200/200 - 23s - loss: 0.0174 - accuracy: 0.9943 - val_loss: 0.6126 - val_accuracy: 0.8734 - 23s/epoch - 116ms/step
Epoch 8/10
200/200 - 23s - loss: 0.0169 - accuracy: 0.9945 - val_loss: 0.6544 - val_accuracy: 0.8510 - 23s/epoch - 116ms/step
Epoch 9/10
200/200 - 23s - loss: 0.0085 - accuracy: 0.9972 - val_loss: 0.7329 - val_accuracy: 0.8806 - 23s/epoch - 116ms/step
Epoch 10/10
200/200 - 23s - loss: 0.0061 - accuracy: 0.9982 - val_loss: 0.7380 - val_accuracy: 0.8794 - 23s/epoch - 116ms/step
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 1459, 100)         10955700

 conv1d (Conv1D)             (None, 1458, 32)          6432

 global_average_pooling1d (G  (None, 32)               0
 lobalAveragePooling1D)

 dropout (Dropout)           (None, 32)                0

 dense (Dense)               (None, 512)               16896

 dropout_1 (Dropout)         (None, 512)               0

 dense_1 (Dense)             (None, 256)               131328

 dropout_2 (Dropout)         (None, 256)               0

 dense_2 (Dense)             (None, 64)                16448

 dense_3 (Dense)             (None, 1)                 65

=================================================================
Total params: 11,126,869
Trainable params: 11,126,869
Non-trainable params: 0
_________________________________________________________________
```

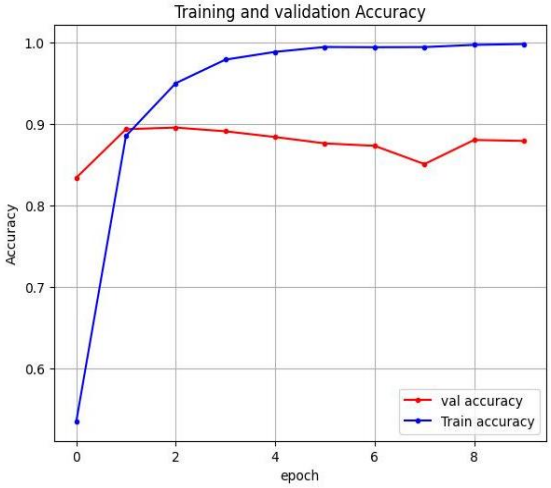
Results and findings:

1. The model was able to achieve 85.868% accuracy on test data with loss of 85.37% as shown in figure below.

```
Vinamra@LAPTOP-I6POQ8U4 MINGW64 /e/Waterloo Courses/ECE 657/Assigments/Assignment 3/Answer 3
$ python test_NLP.py
782/782 [==============================] - 5s 6ms/step - loss: 0.8537 - accuracy: 0.8587
Accuracy on test set: 85.8680009841919
```

2. The below graph shows the training summary of NLP model. It can be observed from the graph that after the 2nd epoch the validation loss is increasing which indicates that model is overfitting on training data and training has stopped after 2nd epoch. Moreover, after training for 10 epochs the accuracy on training set became 99.82% with loss of 0.61% and validation accuracy became 87.94% with loss of 73.8%. Thus, we can conclude the model is working fine with regards to the test and validation data accuracies as they are pretty close but definitely overfitting on training data based on difference between the losses of validation and training data.


Training and validation loss
val loss
train loss
loss
epoch
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
2
4
6
8


Training and validation Accuracy
val accuracy
Train accuracy
Accuracy
epoch
0.6
0.7
0.8
0.9
1.0
0
2
4
6
8