Deep Learning

Assignment 5

Word Embedding

Aim

To develop a neural network model to perform classification on the reuters dataset with both trainable and loaded embedding layer and to test the accuracy of each model. Finally LSTM layers is added to the model with good accuracy to make it a further better model.

Problem Statement

The IMDB dataset consists of the reviews of movies. These reviews are to be classified based on the terms used in the reviews by a neural network model.

Proposed Solution

Dataset

The reuters dataset is used the most in the past decades. It consists of 21578 topics of texts without its topics and typographical errors. The purpose of this model is to classify the data according to its topics. These texts fall under 46 such categories that are to be classified.

Embedding layer

An embedding layer is created to which the dataset is passed. The dimensions of the embedding layer matrices are specified. This layer creates a unique 2D vector for every word in the given input. By this way it makes the process of language processing easier.

Dense Layer

Finally dense layers are attached to the model with relu activation. Since the output of the embedding layer has to be passed to the dense layer, the outputs of the embedding layer is flattened. Finally the model is built with the optimizer RMSProp and binary cross entropy loss and compiled.

Data preprocessing

The data was preprocessed by splitting it into test, train and validation datasets. Then the labels were converted to one hot encodings. Since the input

datasets could be of various string lengths they were padded with zeroes for a definite length.

Glove embedding

The Glove embedding layer was used for the loaded embedding. GloVe stands for global vectors for word representation. It is an unsupervised learning algorithm for generating word embedding by aggregating global word-word co-occurrence matrix from a corpus. The resulting embedding show interesting linear substructures of the word in vector space.

LSTM layer

It stands for Long Short Term Memory. It consists of feedback connections that help it to remember the sequences of input. Thus it can give a better accuracy than other deep learning models.

Evaluation

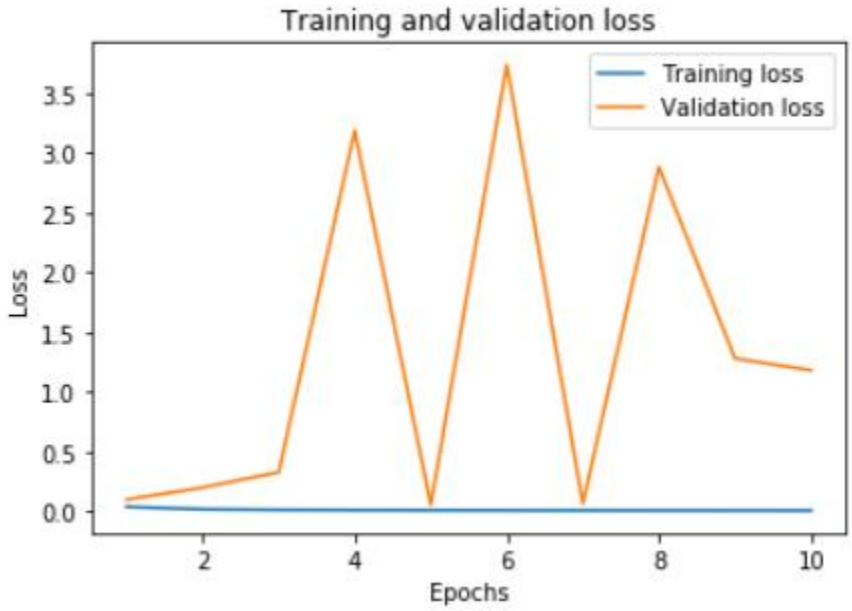
Trainable model

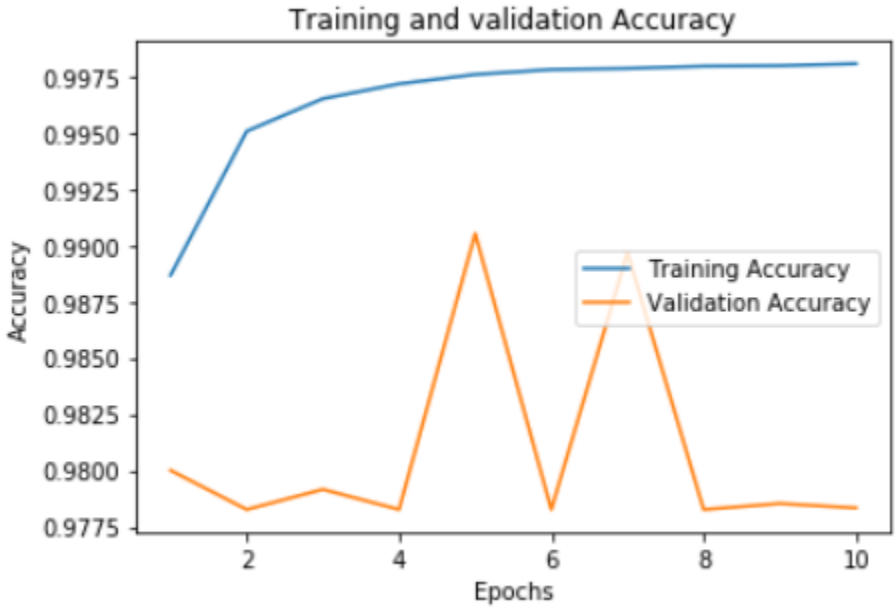
```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 2000, 8)           80000
_________________________________________________________________
batch_normalization_1 (Batch (None, 2000, 8)           32
_________________________________________________________________
flatten_1 (Flatten)          (None, 16000)             0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               8192512
_________________________________________________________________
dense_2 (Dense)              (None, 46)                23598
=================================================================
Total params: 8,296,142
Trainable params: 8,296,126
Non-trainable params: 16
_________________________________________________________________
```

Fitting the model

```
Epoch 6/10
8083/8083 [==============================] - 23s 3ms/step - loss: 0.0088 -
acc: 0.9978 - val_loss: 3.7361 - val_acc: 0.9783
Epoch 7/10
8083/8083 [==============================] - 22s 3ms/step - loss: 0.0083 -
acc: 0.9979 - val_loss: 0.0638 - val_acc: 0.9897
Epoch 8/10
```

```
8083/8083 [==============================] - 24s 3ms/step - loss: 0.0080 -
acc: 0.9980 - val_loss: 2.8806 - val_acc: 0.9783
Epoch 9/10
8083/8083 [==============================] - 23s 3ms/step - loss: 0.0076 -
acc: 0.9980 - val_loss: 1.2796 - val_acc: 0.9785
Epoch 10/10
8083/8083 [==============================] - 20s 3ms/step - loss: 0.0071 -
acc: 0.9981 - val_loss: 1.1797 - val_acc: 0.9783
```





Test accuracy – 97%

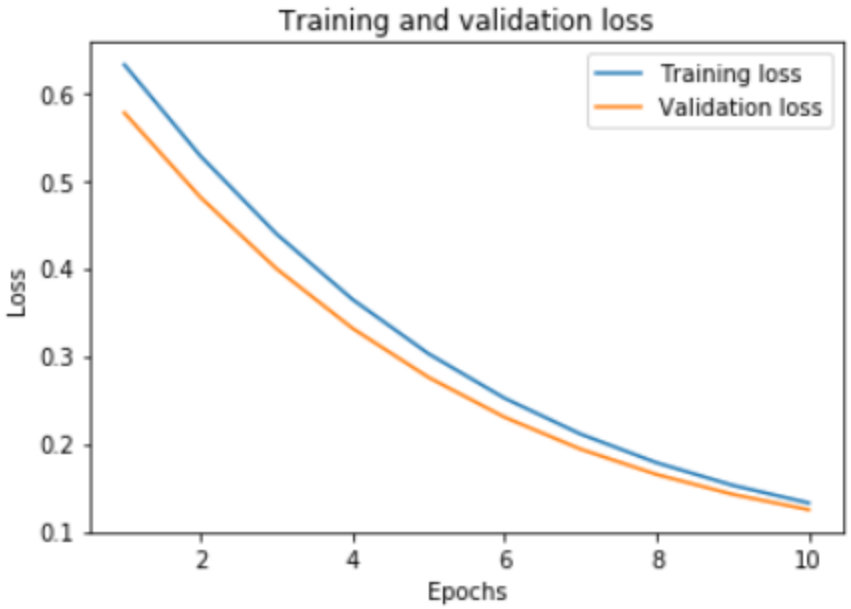
```
2246/2246 [==============================] - 1s 284us/step
[1.155681626985674, 0.9783766269683838]
```

Glove model

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_3 (Embedding)      (None, 2000, 100)         1000000
_________________________________________________________________
flatten_3 (Flatten)          (None, 200000)            0
_________________________________________________________________
dense_5 (Dense)              (None, 512)               102400512
_________________________________________________________________
dropout_3 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_6 (Dense)              (None, 46)                23598
=================================================================
Total params: 103,424,110
Trainable params: 103,424,110
Non-trainable params: 0
_________________________________________________________________
```

Fitting the model

```
Epoch 6/10
8083/8083 [==============================] - 178s 22ms/step - loss: 0.2519
- acc: 0.9783 - val_loss: 0.2302 - val_acc: 0.9783
Epoch 7/10
8083/8083 [==============================] - 179s 22ms/step - loss: 0.2109
- acc: 0.9783 - val_loss: 0.1938 - val_acc: 0.9783
Epoch 8/10
8083/8083 [==============================] - 170s 21ms/step - loss: 0.1783
- acc: 0.9783 - val_loss: 0.1649 - val_acc: 0.9783
Epoch 9/10
8083/8083 [==============================] - 172s 21ms/step - loss: 0.1525
- acc: 0.9783 - val_loss: 0.1423 - val_acc: 0.9783
Epoch 10/10
8083/8083 [==============================] - 168s 21ms/step - loss: 0.1325
- acc: 0.9783 - val_loss: 0.1247 - val_acc: 0.9783
```



Test accuracy – 97%

```
2246/2246 [==============================] - 3s 1ms/step
[0.12384231522586446, 0.9782605171203613]
```

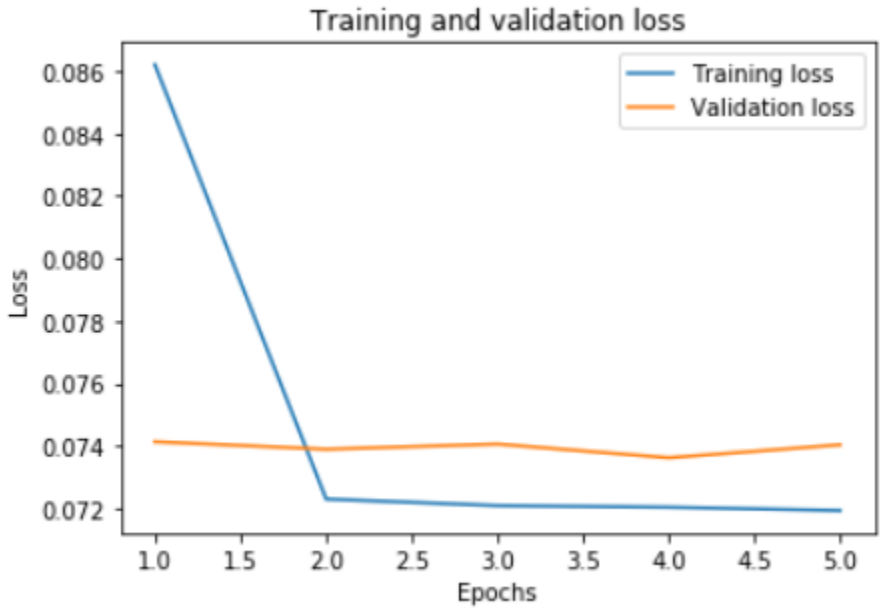
LSTM

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, None, 8)           80000
_________________________________________________________________
lstm_1 (LSTM)                (None, 128)               70144
_________________________________________________________________
dense_1 (Dense)              (None, 46)                5934
=================================================================
Total params: 156,078
Trainable params: 156,078
Non-trainable params: 0
_________________________________________________________________
```

Fitting the model

```
Train on 8083 samples, validate on 899 samples
Epoch 1/5
8083/8083 [==============================] - 475s 59ms/step - loss: 0.0862
- acc: 0.9755 - val_loss: 0.0741 - val_acc: 0.9783
Epoch 2/5
```

```
8083/8083 [==============================] - 302s 37ms/step - loss: 0.0723
- acc: 0.9783 - val_loss: 0.0739 - val_acc: 0.9783
Epoch 3/5
8083/8083 [==============================] - 315s 39ms/step - loss: 0.0721
- acc: 0.9783 - val_loss: 0.0741 - val_acc: 0.9783
Epoch 4/5
8083/8083 [==============================] - 311s 38ms/step - loss: 0.0720
- acc: 0.9783 - val_loss: 0.0736 - val_acc: 0.9783
Epoch 5/5
8083/8083 [==============================] - 313s 39ms/step - loss: 0.0719
- acc: 0.9783 - val_loss: 0.0740 - val_acc: 0.9783
```



Testing accuracy – 97%

```
2246/2246 [==============================] - 17s 8ms/step
[0.07250244192653008, 0.9782605171203613]
```

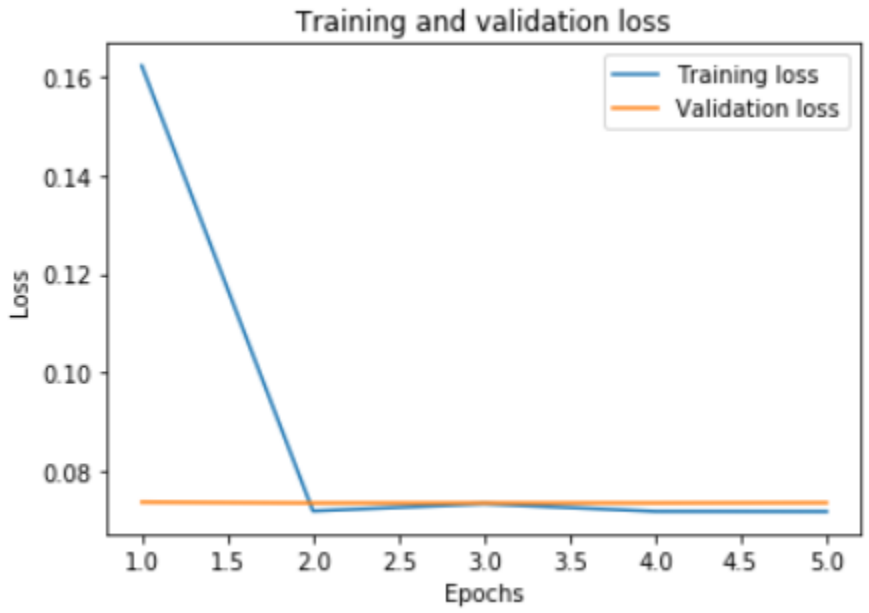
2 LSTM

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_2 (Embedding)      (None, None, 8)           80000
_________________________________________________________________
lstm_2 (LSTM)                (None, None, 25)          3400
_________________________________________________________________
lstm_3 (LSTM)                (None, 25)                5100
_________________________________________________________________
dense_2 (Dense)              (None, 46)                1196
=================================================================
```

```
Total params: 89,696
Trainable params: 89,696
Non-trainable params: 0
_________________________________________________________________
```

Fitting the model

```
Train on 8083 samples, validate on 899 samples
Epoch 1/5
8083/8083 [==============================] - 285s 35ms/step - loss: 0.1623
- acc: 0.9573 - val_loss: 0.0737 - val_acc: 0.9783
Epoch 2/5
8083/8083 [==============================] - 281s 35ms/step - loss: 0.0719
- acc: 0.9783 - val_loss: 0.0735 - val_acc: 0.9783
Epoch 3/5
8083/8083 [==============================] - 282s 35ms/step - loss: 0.0733
- acc: 0.9783 - val_loss: 0.0735 - val_acc: 0.9783
Epoch 4/5
8083/8083 [==============================] - 285s 35ms/step - loss: 0.0718
- acc: 0.9783 - val_loss: 0.0735 - val_acc: 0.9783
Epoch 5/5
8083/8083 [==============================] - 284s 35ms/step - loss: 0.0718
- acc: 0.9783 - val_loss: 0.0736 - val_acc: 0.9783
```



Testing accuracy – 97%

```
2246/2246 [==============================] - 10s 4ms/step
[0.07206905480616876, 0.9782605171203613]
```

Conclusion

All the models seemed to give the same high accuracy. This could be because the datasets were large and the models got trained well.

References

Code given by the Professor.

Keras code for reuters classification