Notebook Resources

- Name : Rishabh Patil
- SAP : 60009200056 (K/K2)
- Branch: K/K2

```
import numpy
from keras.datasets import imdb
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Embedding
from keras.preprocessing import sequence
from keras_preprocessing.sequence import pad_sequences
```

```
top_words = 5000
(X_train, y_train), (X_test, y_test) = imdb.load_data(num_words=top_words)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/imdb.npz
17464789/17464789 [==============================] - 3s 0us/step
```

```
print(X_train[1])
print(type(X_train[1]))
print(len(X_train[1]))
print(y_train[1])
print(type(y_train[1]))
```

```
[1, 194, 1153, 194, 2, 78, 228, 5, 6, 1463, 4369, 2, 134, 26, 4, 715, 8, 118, 1634, 14, 394,
<class 'list'>
189
0
<class 'numpy.int64'>
```

```
# truncate and/or pad input sequences
max_review_length = 600
X_train = pad_sequences(X_train, maxlen=max_review_length)
X_test = pad_sequences(X_test, maxlen=max_review_length)

print(X_train.shape)
print(X_train[1])
```

```
(25000, 600)
[   0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
```

✓ 7s completed at 11:03 AM

```
   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    1  194 1153  194    2   78  228    5    6
1463 4369    2  134   26    4  715    8  118 1634   14  394   20   13
 119  954  189  102    5  207  110 3103   21   14   69  188    8   30
  23    7    4  249  126   93    4  114    9 2300 1523    5  647    4
 116    9   35    2    4  229    9  340 1322    4  118    9    4  130
4901   19    4 1002    5   89   29  952   46   37    4  455    9   45
  43   38 1543 1905  398    4 1649   26    2    5  163   11 3215    2
   4 1153    9  194  775    7    2    2  349 2637  148  605    2    2
  15  123  125   68    2    2   15  349  165 4362   98    5    4  228
   9   43    2 1157   15  299  120    5  120  174   11  220  175  136
  50    9 4373  228    2    5    2  656  245 2350    5    4    2  131
 152  491   18    2   32    2 1212   14    9    6  371   78   22  625
  64 1382    9    8  168  145   23    4 1690   15   16    4 1355    5
  28    6   52  154  462   33   89   78  285   16  145   95]
```

```
import tensorflow as tf
# Create the model

embedding_vector_length = 32

model = Sequential()

model.add(Embedding(top_words + 1, embedding_vector_length, input_length=max_review_length))
model.add(LSTM(100))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

print(model.summary())
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 600, 32)           160032

 lstm (LSTM)                 (None, 100)               53200

 dense (Dense)               (None, 1)                 101

=================================================================
Total params: 213,333
Trainable params: 213,333
Non-trainable params: 0
_________________________________________________________________
None
```

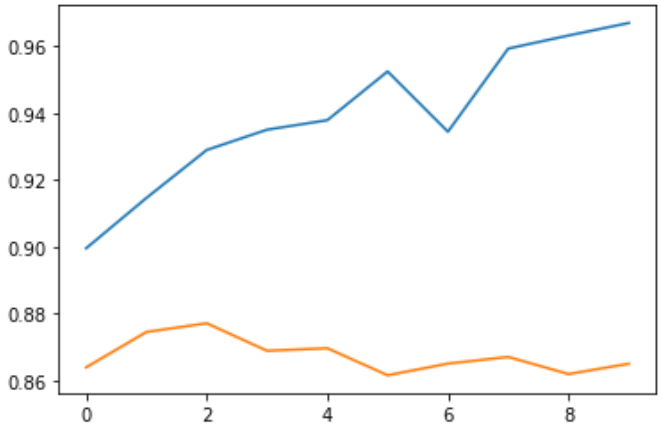
```
hist=model.fit(X_train, y_train, epochs=10, batch_size=64,verbose=1, validation_data=(X_test,y_test

# Final evaluation of the model
scores = model.evaluate(X_test, y_test, verbose=0)
```

```
print("Accuracy: %.2f%%" % (scores[1] 100))
```

```
Epoch 1/10
391/391  [==============================]  - 17s 43ms/step - loss: 0.2609 - accuracy: 0.8995 -
Epoch 2/10
391/391  [==============================]  - 18s 46ms/step - loss: 0.2212 - accuracy: 0.9146 -
Epoch 3/10
391/391  [==============================]  - 17s 44ms/step - loss: 0.1917 - accuracy: 0.9289 -
Epoch 4/10
391/391  [==============================]  - 17s 43ms/step - loss: 0.1727 - accuracy: 0.9350 -
Epoch 5/10
391/391  [==============================]  - 18s 46ms/step - loss: 0.1683 - accuracy: 0.9378 -
Epoch 6/10
391/391  [==============================]  - 17s 43ms/step - loss: 0.1316 - accuracy: 0.9524 -
Epoch 7/10
391/391  [==============================]  - 18s 47ms/step - loss: 0.1723 - accuracy: 0.9344 -
Epoch 8/10
391/391  [==============================]  - 18s 46ms/step - loss: 0.1164 - accuracy: 0.9593 -
Epoch 9/10
391/391  [==============================]  - 28s 71ms/step - loss: 0.1074 - accuracy: 0.9632 -
Epoch 10/10
391/391  [==============================]  - 18s 47ms/step - loss: 0.0950 - accuracy: 0.9670 -
Accuracy: 86.49%
```

```
import matplotlib.pyplot as plt
y1=hist.history['accuracy']
y2=hist.history['val_accuracy']
plt.plot(y1)
plt.plot(y2)
```

```
[<matplotlib.lines.Line2D at 0x7fd58dbcf9d0>]
```




```
print(X_test[1],y_test[1])
pred=model.predict(X_test)
print(pred)
```

```
[   0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
```

```
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    1   14   22 3443    6  176    7    2   88   12
 2679   23 1310    5  109  943    4  114    9   55  606    5  111    7
    4  139  193  273   23    4  172  270   11    2    2    4    2 2801
  109 1603   21    4   22 3861    8    6 1193 1330   10   10    4  105
  987   35  841    2   19  861 1074    5 1987    2   45   55  221   15
  670    2  526   14 1069    4  405    5 2438    7   27   85  108  131
    4    2    2 3884  405    9 3523  133    5   50   13  104   51   66
  166   14   22  157    9    4  530  239   34    2 2801   45  407   31
    7   41 3778  105   21   59  299   12   38  950    5 4521   15   45
  629  488 2733  127    6   52  292   17    4    2  185  132 1988    2
 1799  488 2693   47    6  392  173    4    2 4378  270 2352    4 1500
    7    4   65   55   73   11  346   14   20    9    6  976 2078    7
    2  861    2    5 4182   30 3127    2   56    4  841    5  990  692
    8    4 1669  398  229   10   10   13 2822  670    2   14    9   31
    7   27  111  108   15 2033   19    2 1429  875  551   14   22    9
 1193   21   45 4829    5   45  252    8    2    6  565  921 3639   39
    4  529   48   25  181    8   67   35 1732   22   49  238   60  135
 1162   14    9  290    4   58   10   10  472   45   55  878    8  169
   11  374    2   25  203   28    8  818   12  125    4 3077] 1
782/782  [==============================] - 7s 9ms/step
[[0.00619938]
 [0.999757  ]
 [0.13889685]
 ...
 [0.24086878]
 [0.07379474]
 [0.9452105 ]]
```