IMDB_review_EX1

```
#import
from keras.datasets import imdb
from keras.preprocessing import sequence
from keras import Sequential
from keras.layers import Embedding,LSTM,Dense,Dropout
import tensorflow as tf
from keras.preprocessing.sequence import pad_sequences
```

```
#load imdb data and split it intp training & testing datasets
vocabulary_size = 5000
(X_train, y_train), (X_test, y_test) = imdb.load_data(num_words = vocabulary_size)
# print('loaded dataset with {}training samples, {} test sample'.format((len(X_train),(len(X_test)))
print('loaded dataset with {}training samples, {} test sample'.format(str(len(X_train)),str(len(X_test))))
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/imdb.npz
17464789/17464789 [==============================] - 2s 0us/step
loaded dataset with 25000training samples, 25000 test sample
```

```
print("review:", X_train[0])
print("Label:", y_train[0])
```

```
review: [1, 14, 22, 16, 43, 530, 973, 1622, 1385, 65, 458, 4468, 66, 3941, 4, 173, 36, 256, 5, 25, 100, 43, 838, 112, 50, 670, 2, 9,
Label: 1
```

```
#decode the sentences to see the reviews as test
word_index = imdb.get_word_index()
#print(word_index)
reverse_word_index = dict( [(value,key) for (key,value) in word_index.items()])
decoded_review = ' '.join([reverse_word_index.get(i-3,'?')for i in X_train[0]])
print(X_train[0])
print(decoded_review)
decoded_review = ' '.join([reverse_word_index.get(i-3,'?')for i in X_train[2]])
print(X_train[2])
print(decoded_review)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/imdb_word_index.json
1641221/1641221 [==============================] - 1s 1us/step
[1, 14, 22, 16, 43, 530, 973, 1622, 1385, 65, 458, 4468, 66, 3941, 4, 173, 36, 256, 5, 25, 100, 43, 838, 112, 50, 670, 2, 9, 35, 48
? this film was just brilliant casting location scenery story direction everyone's really suited the part they played and you could
[1, 14, 47, 8, 30, 31, 7, 4, 249, 108, 7, 4, 2, 54, 61, 369, 13, 71, 149, 14, 22, 112, 4, 2401, 311, 12, 16, 3711, 33, 75, 43, 1829,
? this has to be one of the worst films of the ? when my friends i were watching this film being the target audience it was aimed at
```

```
#pad the sequences
max_words =500
X_train = sequence.pad_sequences(X_train, maxlen = max_words)
X_test = sequence.pad_sequences(X_test, maxlen = max_words)
print(X_train[0])
```

```
[   0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    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   16   43  530  973 1622 1385   65  458 4468
   66 3941    4  173   36  256    5   25  100   43  838  112   50  670
    2    9   35  480  284    5  150    4  172  112  167    2  336  385
   39    4  172 4536 1111   17  546   38   13  447    4  192   50   16
    6  147 2025   19   14   22    4 1920 4613  469    4   22   71   87
   12   16   43  530   38   76   15   13 1247    4   22   17  515   17
   12   16  626   18    2    5   62  386   12    8  316    8  106    5
```

```
    4 2223    2   16  480   66 3785   33    4  130   12   16   38  619
    5   25  124   51   36  135   48   25 1415   33    6   22   12  215
   28   77   52    5   14  407   16   82    2    8    4  107  117    2
   15  256    4    2    7 3766    5  723   36   71   43  530  476   26
  400  317   46    7    4    2 1029   13  104   88    4  381   15  297
   98   32 2071   56   26  141    6  194    2   18    4  226   22   21
  134  476   26  480    5  144   30    2   18   51   36   28  224   92
   25  104    4  226   65   16   38 1334   88   12   16  283    5   16
 4472  113  103   32   15   16    2   19  178   32]
```

```
_#inrialize the model
embedding_size = 32
model = tf.keras.Sequential()
model.add(tf.keras.layers.Embedding(vocabulary_size,embedding_size, input_length = max_words))
model.add(tf.keras.layers.LSTM(100))
model.add(tf.keras.layers.Dense(1,activation = 'sigmoid'))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 500, 32)           160000

 lstm (LSTM)                 (None, 100)               53200

 dense (Dense)               (None, 1)                 101

=================================================================
Total params: 213301 (833.21 KB)
Trainable params: 213301 (833.21 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
#compile model
model.compile( loss = 'binary_crossentropy',
              optimizer = 'adam',
               metrics = ['accuracy'])
```

```
#divide X_train in train and validation datasets
batch_size= 64
X_vaild , y_valid = X_train[:batch_size], y_train[:batch_size]
X_train_partial, y_train_partial = X_train[:batch_size:], y_train[:batch_size:]
```

```
class myCallback(tf.keras.callbacks.Callback):
  def on_epoch_end(self , epoch , logs ={}):
    DESIRED_ACC = 0.9
    if(logs.get('val_accuracy')>= DESIRED_ACC):
      print("\n stopping training as validation accuracy is reached to %.2f!" % DESIRED_ACC)
    self.model.stop_training= True
callbacks = myCallback()
```

```
#fit the model
num_epochs = 5
history = model.fit(X_train_partial, y_train_partial, validation_data=(X_vaild, y_valid), batch_size= batch_size, epochs = num_epochs,
```

```
Epoch 1/5
1/1 [==============================] - 4s 4s/step - loss: 0.6935 - accuracy: 0.4531 - val_loss: 0.6909 - val_accuracy: 0.6250
```

```
#test the model and print test accuracy score
scores =  model.evaluate (X_test, y_test)
print = ('Test accuracy:', scores[1])
```

```
782/782 [==============================] - 8s 10ms/step - loss: 0.6932 - accuracy: 0.4993
```