In [1]:
```
# Credits: https://machinelearningmastery.com/sequence-classification-lstm-recurrent-neural-networks-python-keras/
# LSTM for sequence classification in the IMDB dataset
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.embeddings import Embedding
from keras.preprocessing import sequence
import warnings
warnings.filterwarnings("ignore")
# fix random seed for reproducibility
numpy.random.seed(7)
```

```
Using TensorFlow backend.
```

In [2]:
```
#Dataset taken from : https://keras.io/datasets/#imdb-movie-reviews-sentiment-classification

# load the dataset but only keep the top n words, zero the rest
top_words = 5000
(X_train, y_train), (X_test, y_test) = imdb.load_data(nb_words=top_words)
```

```
/usr/local/lib/python3.6/dist-packages/keras/datasets/imdb.py:49: UserWarning: The `nb_words` argument in `load_data` has been renamed `num_words`.
  warnings.warn('The `nb_words` argument in `load_data` '
```

In [3]:
```
print(X_train[1])
print(type(X_train[1]))
print(len(X_train[1]))
```

```
[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, 2
```

```
1, 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, 490
1, 19, 4, 1002, 5, 89, 29, 952, 46, 37, 4, 455, 9, 45, 43, 38, 1543, 19
05, 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, 22
0, 175, 136, 50, 9, 4373, 228, 2, 5, 2, 656, 245, 2350, 5, 4, 2, 131, 1
52, 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]
<class 'list'>
189
```

In [4]:

```
# truncate and/or pad input sequences
# maximum length review be 600
max_review_length = 600
#padding maximum length review to X_train and X_test
X_train = sequence.pad_sequences(X_train, maxlen=max_review_length)
X_test = sequence.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
   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    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]
```

In [5]:
```
# create the model
embedding_vecor_length = 32
model = Sequential()
model.add(Embedding(top_words, embedding_vecor_length, input_length=max
_review_length))
model.add(LSTM(100))
#Using Sigmoid Activation
model.add(Dense(1, activation='sigmoid'))
# Optimizer used : Adam
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['a
```

```
ccuracy'])
print(model.summary())
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 600, 32)           160000
_________________________________________________________________
lstm_1 (LSTM)                (None, 100)               53200
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 101
=================================================================
Total params: 213,301
Trainable params: 213,301
Non-trainable params: 0
_________________________________________________________________
None
```

In [6]:

```
model.fit(X_train, y_train, nb_epoch=10, batch_size=64)
# Final evaluation of the model
scores = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
```

```
/usr/local/lib/python3.6/dist-packages/keras/models.py:981: UserWarnin
g: The `nb_epoch` argument in `fit` has been renamed `epochs`.
  warnings.warn('The `nb_epoch` argument in `fit` '
```

```
Epoch 1/10
25000/25000 [==============================] - 295s 12ms/step - loss:
0.4567 - acc: 0.7825
Epoch 2/10
20160/25000 [=====================>......] - ETA: 57s - loss: 0.3694
- acc: 0.834325000/25000 [==============================] - 295s 12ms/s
tep - loss: 0.3647 - acc: 0.8385
Epoch 3/10
25000/25000 [==============================] - 297s 12ms/step - loss:
0.5037 - acc: 0.7639
Epoch 4/10
 4416/25000 [====>.........................] - ETA: 4:03 - loss: 0.3698
- acc: 0.838525000/25000 [==============================] - 297s 12ms/s
```

```
tep - loss: 0.3554 - acc: 0.8503
Epoch 5/10
24832/25000 [============================>.] - ETA: 1s - loss: 0.2673 -
acc: 0.896125000/25000 [==============================] - 296s 12ms/ste
p - loss: 0.2671 - acc: 0.8961
Epoch 6/10
25000/25000 [==============================] - 297s 12ms/step - loss:
0.2477 - acc: 0.9051
Epoch 7/10
 5824/25000 [=====>........................] - ETA: 3:47 - loss: 0.1967
- acc: 0.926525000/25000 [==============================] - 296s 12ms/s
tep - loss: 0.2131 - acc: 0.9180
Epoch 8/10
25000/25000 [==============================] - 296s 12ms/step - loss:
0.1861 - acc: 0.9304
Epoch 9/10
  256/25000 [..............................] - ETA: 4:51 - loss: 0.1830
- acc: 0.929725000/25000 [==============================] - 297s 12ms/s
tep - loss: 0.1659 - acc: 0.9392
Epoch 10/10
22784/25000 [==========================>...] - ETA: 26s - loss: 0.1636
- acc: 0.938125000/25000 [==============================] - 296s 12ms/s
tep - loss: 0.1657 - acc: 0.9374
Accuracy: 86.24%
```

Observation:

1. The training accuracy at the end of fist epoch is 78%.
2. At the end of second epcoch it is 83%.
3. At last it is 93%.