In [2]:
```
import pandas as pd
import numpy as np
import re
import nltk
from nltk.corpus import stopwords

from numpy import array
from keras.preprocessing.text import one_hot
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers.core import Activation, Dropout, Dense
from keras.layers import Flatten
from keras.layers import GlobalMaxPooling1D
from keras.layers.embeddings import Embedding
from sklearn.model_selection import train_test_split
from keras.preprocessing.text import Tokenizer
```

In [6]:
```
movie_reviews = pd.read_csv(r"C:\Users\admin\Desktop\nlp theory da\IMDB Dataset.c

movie_reviews.isnull().values.any()

movie_reviews.shape
```

Out[6]:
```
(50000, 2)
```

In [7]:
```
movie_reviews.head()
```

Out[7]:

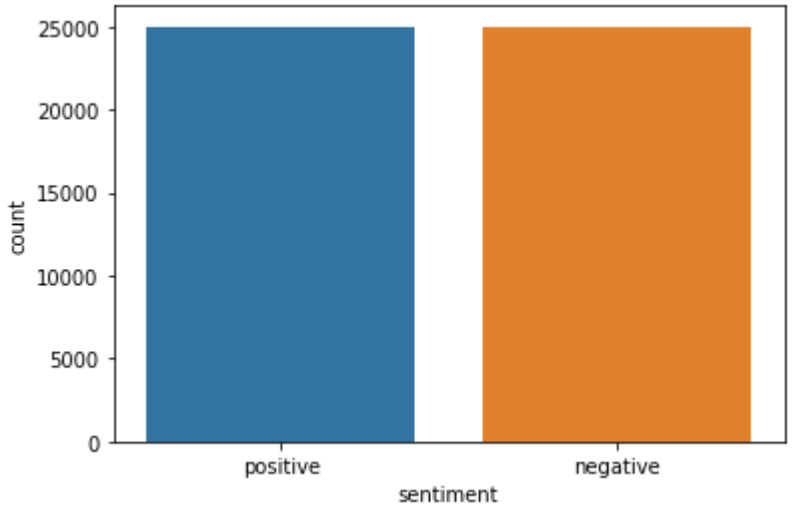
	review	sentiment
0	One of the other reviewers has mentioned that ...	positive
1	A wonderful little production. The...	positive
2	I thought this was a wonderful way to spend ti...	positive
3	Basically there's a family where a little boy ...	negative
4	Petter Mattei's "Love in the Time of Money" is...	positive

In [8]:
```
movie_reviews["review"][3]
```

Out[8]:
```
"Basically there's a family where a little boy (Jake) thinks there's a zombie i
n his closet & his parents are fighting all the time.<br /><br />This movie is
slower than a soap opera... and suddenly, Jake decides to become Rambo and kill
the zombie.<br /><br />OK, first of all when you're going to make a film you mu
st Decide if its a thriller or a drama! As a drama the movie is watchable. Pare
nts are divorcing & arguing like in real life. And then we have Jake with his c
loset which totally ruins all the film! I expected to see a BOOGEYMAN similar m
ovie, and instead i watched a drama with some meaningless thriller spots.<br />
<br />3 out of 10 just for the well playing parents & descent dialogs. As for t
he shots with Jake: just ignore them."
```

In [9]:
```
import seaborn as sns

sns.countplot(x='sentiment', data=movie_reviews)
```

Out[9]: <AxesSubplot:xlabel='sentiment', ylabel='count'>



In [10]:
```
def preprocess_text(sen):
    # Removing html tags
    sentence = remove_tags(sen)

    # Remove punctuations and numbers
    sentence = re.sub('[^a-zA-Z]', ' ', sentence)

    # Single character removal
    sentence = re.sub(r"\s+[a-zA-Z]\s+", ' ', sentence)

    # Removing multiple spaces
    sentence = re.sub(r'\s+', ' ', sentence)

    return sentence
```

In [11]:
```
TAG_RE = re.compile(r'<[^>]+>')

def remove_tags(text):
    return TAG_RE.sub('', text)
```

In [12]:
```
X = []
sentences = list(movie_reviews['review'])
for sen in sentences:
    X.append(preprocess_text(sen))
```

In [13]:
```
X[3]
```

Out[13]:
```
'Basically there a family where little boy Jake thinks there a zombie in his cl
oset his parents are fighting all the time This movie is slower than soap opera
and suddenly Jake decides to become Rambo and kill the zombie OK first of all w
hen you re going to make film you must Decide if its thriller or drama As drama
the movie is watchable Parents are divorcing arguing like in real life And then
we have Jake with his closet which totally ruins all the film expected to see B
OOGEYMAN similar movie and instead watched drama with some meaningless thriller
spots out of just for the well playing parents descent dialogs As for the shots
with Jake just ignore them '
```

In [14]:
```
y = movie_reviews['sentiment']

y = np.array(list(map(lambda x: 1 if x=="positive" else 0, y)))
```

In [15]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_
```

In [17]:
```
tokenizer = Tokenizer(num_words=5000)
tokenizer.fit_on_texts(X_train)

X_train = tokenizer.texts_to_sequences(X_train)
X_test = tokenizer.texts_to_sequences(X_test)
```

In [18]:
```
# Adding 1 because of reserved 0 index
vocab_size = len(tokenizer.word_index) + 1

maxlen = 100

X_train = pad_sequences(X_train, padding='post', maxlen=maxlen)
X_test = pad_sequences(X_test, padding='post', maxlen=maxlen)
```

In [22]:
```
from numpy import array
from numpy import asarray
from numpy import zeros

embeddings_dictionary = dict()
glove_file = open(r'C:\Users\admin\Desktop\nlp theory da\glove.6B.100d.txt', enc

for line in glove_file:
    records = line.split()
    word = records[0]
    vector_dimensions = asarray(records[1:], dtype='float32')
    embeddings_dictionary [word] = vector_dimensions
glove_file.close()
```

In [23]:
```
embedding_matrix = zeros((vocab_size, 100))
for word, index in tokenizer.word_index.items():
    embedding_vector = embeddings_dictionary.get(word)
    if embedding_vector is not None:
        embedding_matrix[index] = embedding_vector
```

In [26]:
```
from tensorflow.keras.layers import LSTM
model = Sequential()
embedding_layer = Embedding(vocab_size, 100, weights=[embedding_matrix], input_le
model.add(embedding_layer)
model.add(LSTM(128))

model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['acc'])
```

In [27]:
```
print(model.summary())
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 100, 100)          9254700
_________________________________________________________________
module_wrapper (ModuleWrappe (None, 128)               117248
_________________________________________________________________
dense (Dense)                (None, 1)                 129
=================================================================
Total params: 9,372,077
Trainable params: 117,377
Non-trainable params: 9,254,700
_________________________________________________________________
None
```

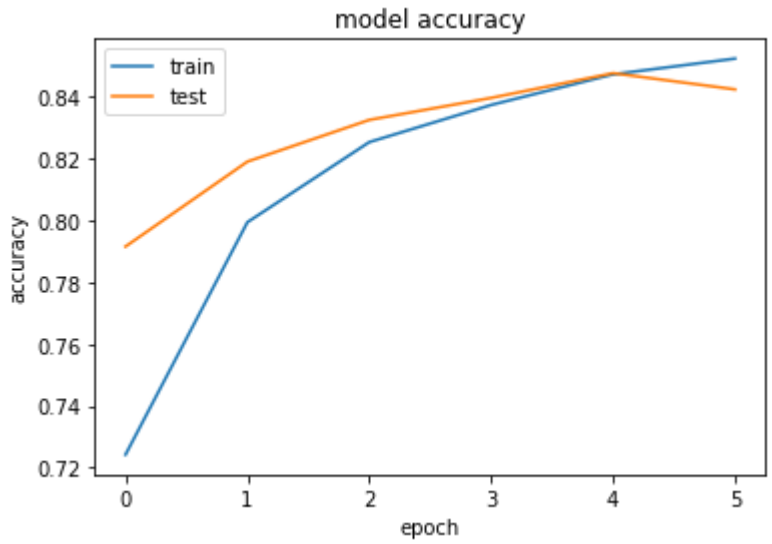
In [28]:
```
history = model.fit(X_train, y_train, batch_size=128, epochs=6, verbose=1, valida

score = model.evaluate(X_test, y_test, verbose=1)
```

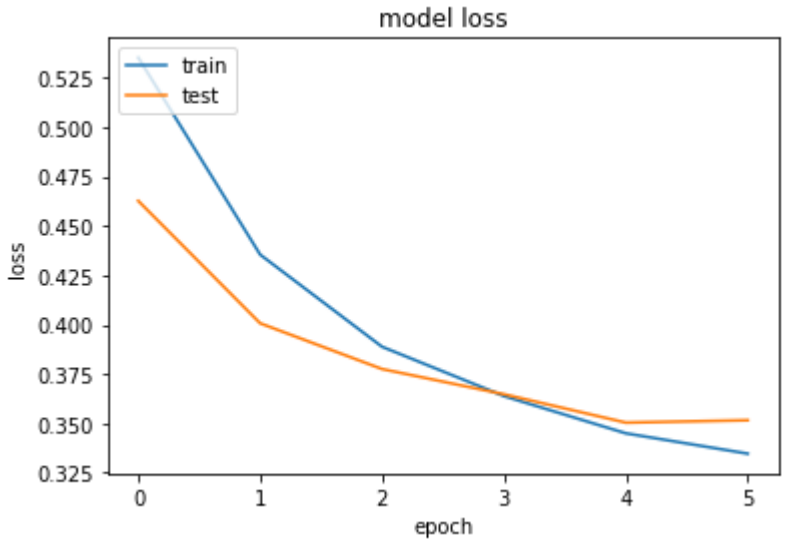
```
Epoch 1/6
WARNING:tensorflow:AutoGraph could not transform <bound method Dense.call of <k
eras.layers.core.Dense object at 0x0000025D32687E80>> and will run it as-is.
Please report this to the TensorFlow team. When filing the bug, set the verbosi
ty to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and attach the full outpu
t.
Cause: invalid syntax (tmpbqpbqjt9.py, line 48)
To silence this warning, decorate the function with @tf.autograph.experimental.
do_not_convert
WARNING: AutoGraph could not transform <bound method Dense.call of <keras.layer
s.core.Dense object at 0x0000025D32687E80>> and will run it as-is.
Please report this to the TensorFlow team. When filing the bug, set the verbosi
ty to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and attach the full outpu
t.
Cause: invalid syntax (tmpbqpbqjt9.py, line 48)
To silence this warning, decorate the function with @tf.autograph.experimental.
do_not_convert
250/250 [==============================] - 56s 157ms/step - loss: 0.5918 - acc:
0.6700 - val_loss: 0.4626 - val_acc: 0.7916
Epoch 2/6
250/250 [==============================] - 36s 143ms/step - loss: 0.4525 - acc:
0.7922 - val_loss: 0.4007 - val_acc: 0.8191
Epoch 3/6
250/250 [==============================] - 36s 143ms/step - loss: 0.3951 - acc:
0.8193 - val_loss: 0.3775 - val_acc: 0.8326
Epoch 4/6
250/250 [==============================] - 35s 141ms/step - loss: 0.3672 - acc:
0.8353 - val_loss: 0.3647 - val_acc: 0.8397
Epoch 5/6
250/250 [==============================] - 42s 167ms/step - loss: 0.3485 - acc:
0.8459 - val_loss: 0.3504 - val_acc: 0.8478
Epoch 6/6
250/250 [==============================] - 38s 153ms/step - loss: 0.3342 - acc:
0.8540 - val_loss: 0.3516 - val_acc: 0.8425
313/313 [==============================] - 8s 24ms/step - loss: 0.3477 - acc:
0.8436
```

In [29]:
```
print("Test Score:", score[0])
print("Test Accuracy:", score[1])
```

```
Test Score: 0.3477403223514557
Test Accuracy: 0.8435999751091003
```

In [30]:
```
import matplotlib.pyplot as plt

plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])

plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train','test'], loc='upper left')
plt.show()

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])

plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train','test'], loc='upper left')
plt.show()
```





In [31]:
```
instance = X[57]
print(instance)
```

```
I laughed all the way through this rotten movie It so unbelievable woman leaves
her husband after many years of marriage has breakdown in front of real estate
office What happens The office manager comes outside and offers her job Hilario
us Next thing you know the two women are going at it Yep they re lesbians Nothi
ng rings true in this Lifetime for Women with nothing better to do movie Clunky
dialogue like don want to spend the rest of my life feeling like had chance to
be happy and didn take it doesn help There a wealthy distant mother who disappr
oves of her daughter new relationship sassy black maid unbelievable that in the
year film gets made in which there a sassy black maid Hattie McDaniel must be t
urning in her grave The woman has husband who freaks out and wants custody of t
he snotty teenage kids Sheesh No cliche is left unturned
```

In [32]:
```
instance = tokenizer.texts_to_sequences(instance)

flat_list = []
for sublist in instance:
    for item in sublist:
        flat_list.append(item)

flat_list = [flat_list]

instance = pad_sequences(flat_list, padding='post', maxlen=maxlen)

model.predict(instance)
```

Out[32]:
```
array([[0.670702]], dtype=float32)
```

In [ ]:
```

```