In [2]:
```
import pandas as pd    # to load dataset
import numpy as np     # for mathematic equation
from nltk.corpus import stopwords   # to get collection of stopwords
from sklearn.model_selection import train_test_split       # for splitting dataset
from tensorflow.keras.preprocessing.text import Tokenizer  # to encode text to int
from tensorflow.keras.preprocessing.sequence import pad_sequences   # to do padding or truncating
from tensorflow.keras.models import Sequential     # the model
from tensorflow.keras.layers import Embedding, LSTM, Dense # layers of the architecture
from tensorflow.keras.callbacks import ModelCheckpoint   # save model
from tensorflow.keras.models import load_model   # load saved model
import re
```

In [3]:
```
data = pd.read_csv('IMDB Dataset.csv')

print(data)
```

```
                                                  review sentiment
0      One of the other reviewers has mentioned that ...  positive
1      A wonderful little production. <br /><br />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
...                                                  ...       ...
49995  I thought this movie did a down right good job...  positive
49996  Bad plot, bad dialogue, bad acting, idiotic di...  negative
49997  I am a Catholic taught in parochial elementary...  negative
49998  I'm going to have to disagree with the previou...  negative
49999  No one expects the Star Trek movies to be high...  negative

[50000 rows x 2 columns]
```

In [4]:
```
english_stops = set(stopwords.words('english'))
```

In [5]:
```
def load_dataset():
    df = pd.read_csv('IMDB Dataset.csv')
    x_data = df['review']       # Reviews/Input
    y_data = df['sentiment']    # Sentiment/Output

    # PRE-PROCESS REVIEW
    x_data = x_data.replace({'<.*?>': ''}, regex = True)          # remove html tag
    x_data = x_data.replace({'[^A-Za-z]': ' '}, regex = True)     # remove non alphabet
    x_data = x_data.apply(lambda review: [w for w in review.split() if w not in english_stops])  # remove stop wor
    x_data = x_data.apply(lambda review: [w.lower() for w in review])   # lower case

    # ENCODE SENTIMENT -> 0 & 1
    y_data = y_data.replace('positive', 1)
    y_data = y_data.replace('negative', 0)

    return x_data, y_data

x_data, y_data = load_dataset()

print('Reviews')
print(x_data, '\n')
print('Sentiment')
print(y_data)
```

```
Reviews
0        [one, reviewers, mentioned, watching, oz, epis...
1        [a, wonderful, little, production, the, filmin...
2        [i, thought, wonderful, way, spend, time, hot,...
3        [basically, family, little, boy, jake, thinks,...
4        [petter, mattei, love, time, money, visually, ...
                              ...
49995    [i, thought, movie, right, good, job, it, crea...
49996    [bad, plot, bad, dialogue, bad, acting, idioti...
49997    [i, catholic, taught, parochial, elementary, s...
49998    [i, going, disagree, previous, comment, side, ...
49999    [no, one, expects, star, trek, movies, high, a...
Name: review, Length: 50000, dtype: object

Sentiment
0        1
1        1
2        1
3        0
4        1
        ..
49995    1
49996    0
49997    0
49998    0
49999    0
Name: sentiment, Length: 50000, dtype: int64
```

In [6]:

```
x_train, x_test, y_train, y_test = train_test_split(x_data, y_data, test_size = 0.2)

print('Train Set')
print(x_train, '\n')
print(x_test, '\n')
print('Test Set')
print(y_train, '\n')
print(y_test)
```

```
Train Set
42357    [i, voted, mainly, hitchcock, agreed, direct, ...
2872     [any, film, school, student, could, made, film...
16071    [spider, man, opinion, best, superhero, ever, ...
19012    [after, spending, half, hour, examining, rumor...
24494    [the, thing, john, carpenter, best, movie, mer...
                               ...
2344     [i, heard, movie, bad, they, even, warned, ter...
32560    [this, tv, last, night, i, painfully, forced, ...
4291     [this, movie, one, many, kung, fu, action, fil...
20075    [this, movie, excellent, i, expecting, live, h...
21247    [hello, lovely, dirty, dancing, fans, i, came,...
Name: review, Length: 40000, dtype: object

36807    [this, movie, story, it, bunch, guys, tortures...
40294    [bacall, well, especially, considering, nd, fi...
17121    [a, young, woman, jodie, foster, witnessing, m...
49552    [when, film, gets, right, really, gets, right,...
11600    [along, cops, the, goat, one, keaton, two, fun...
                               ...
48632    [perfect, families, small, children, looking, ...
30387    [joe, first, released, us, summer, despite, re...
30179    [the, crimson, rivers, one, directed, top, eve...
5583     [we, know, movie, never, complete, justice, bo...
9715     [put, movie, misery, burn, negatives, what, i,...
Name: review, Length: 10000, dtype: object

Test Set
42357    0
2872     0
16071    1
19012    0
24494    1
        ..
2344     0
32560    0
4291     1
20075    1
21247    1
Name: sentiment, Length: 40000, dtype: int64

36807    1
40294    1
17121    0
49552    1
11600    1
        ..
48632    1
30387    0
30179    0
5583     0
9715     0
Name: sentiment, Length: 10000, dtype: int64
```

In [7]:

```
def get_max_length():
    review_length = []
    for review in x_train:
        review_length.append(len(review))

    return int(np.ceil(np.mean(review_length)))
```

In [8]:

```
# ENCODE REVIEW
token = Tokenizer(lower=False)    # no need lower, because already lowered the data in load_data()
token.fit_on_texts(x_train)
x_train = token.texts_to_sequences(x_train)
x_test = token.texts_to_sequences(x_test)

max_length = get_max_length()

x_train = pad_sequences(x_train, maxlen=max_length, padding='post', truncating='post')
x_test = pad_sequences(x_test, maxlen=max_length, padding='post', truncating='post')

total_words = len(token.word_index) + 1   # add 1 because of 0 padding

print('Encoded X Train\n', x_train, '\n')
print('Encoded X Test\n', x_test, '\n')
print('Maximum review length: ', max_length)
```

```
Encoded X Train
 [[    1  5471  1331 ...     0     0     0]
 [ 1602     4   272 ...     0     0     0]
 [ 4501    52   568 ...   674   674   583]
 ...
 [    8     3     5 ... 19318  3981    55]
 [    8     3   225 ...     0     0     0]
 [ 4790  1205  1544 ...     0     0     0]]

Encoded X Test
 [[    8     3    13 ...     0     0     0]
 [ 7044    16   166 ...   677   648    86]
 [   38    97   152 ...    88    33  2633]
 ...
 [    2 15639  7887 ...     4     7   768]
 [  230    47     3 ...     0     0     0]
 [  180     3  4484 ...     0     0     0]]

Maximum review length:  130
```

In [11]:
```
EMBED_DIM = 32
LSTM_OUT = 64

model = Sequential()
model.add(Embedding(total_words, EMBED_DIM, input_length = max_length))
model.add(LSTM(LSTM_OUT))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])

print(model.summary())
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 130, 32)           2956256

 lstm (LSTM)                 (None, 64)                24832

 dense (Dense)               (None, 1)                 65

=================================================================
Total params: 2,981,153
Trainable params: 2,981,153
Non-trainable params: 0
_________________________________________________________________
None
```

In [12]:
```
checkpoint = ModelCheckpoint(
    'models/LSTM.h5',
    monitor='accuracy',
    save_best_only=True,
    verbose=1
)
```

In [13]:
```
model.fit(x_train, y_train, batch_size = 128, epochs = 5, callbacks=[checkpoint])
```

```
Epoch 1/5
313/313 [==============================] - ETA: 0s - loss: 0.4698 - accuracy: 0.7623
Epoch 1: accuracy improved from -inf to 0.76230, saving model to models\LSTM.h5
313/313 [==============================] - 107s 323ms/step - loss: 0.4698 - accuracy: 0.7623
Epoch 2/5
313/313 [==============================] - ETA: 0s - loss: 0.2187 - accuracy: 0.9202
Epoch 2: accuracy improved from 0.76230 to 0.92020, saving model to models\LSTM.h5
313/313 [==============================] - 91s 290ms/step - loss: 0.2187 - accuracy: 0.9202
Epoch 3/5
313/313 [==============================] - ETA: 0s - loss: 0.1286 - accuracy: 0.9586
Epoch 3: accuracy improved from 0.92020 to 0.95860, saving model to models\LSTM.h5
313/313 [==============================] - 89s 285ms/step - loss: 0.1286 - accuracy: 0.9586
Epoch 4/5
313/313 [==============================] - ETA: 0s - loss: 0.0772 - accuracy: 0.9782
Epoch 4: accuracy improved from 0.95860 to 0.97817, saving model to models\LSTM.h5
313/313 [==============================] - 85s 271ms/step - loss: 0.0772 - accuracy: 0.9782
Epoch 5/5
313/313 [==============================] - ETA: 0s - loss: 0.0506 - accuracy: 0.9877
Epoch 5: accuracy improved from 0.97817 to 0.98773, saving model to models\LSTM.h5
313/313 [==============================] - 86s 275ms/step - loss: 0.0506 - accuracy: 0.9877
```

Out[13]:
```
<keras.callbacks.History at 0x1ff6f3d2ce0>
```

In [15]:
```
loaded_model = load_model('models/LSTM.h5')
```

In [16]:
```python
review = str(input('Movie Review: '))
```
```
Movie Review: 36807
```

In [17]:
```python
# Pre-process input
regex = re.compile(r'[^a-zA-Z\s]')
review = regex.sub('', review)
print('Cleaned: ', review)

words = review.split(' ')
filtered = [w for w in words if w not in english_stops]
filtered = ' '.join(filtered)
filtered = [filtered.lower()]

print('Filtered: ', filtered)
```
```
Cleaned:
Filtered:  ['']
```

In [18]:
```python
tokenize_words = token.texts_to_sequences(filtered)
tokenize_words = pad_sequences(tokenize_words, maxlen=max_length, padding='post', truncating='post')
print(tokenize_words)
```
```
[[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]]
```

In [19]:
```python
result = loaded_model.predict(tokenize_words)
print(result)
```
```
1/1 [==============================] - 3s 3s/step
[[0.9666963]]
```

In [20]:
```python
if result >= 0.7:
    print('positive')
else:
    print('negative')
```
```
positive
```

In [ ]:
```python
```