In [3]:
```
import pandas as pd
import csv
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Embedding, LSTM, Dense
from sklearn.model_selection import train_test_split
from keras.models import load_model
```

```
WARNING:tensorflow:From C:\Users\Ownerqp\AppData\Local\Programs\Python\Pyt
hon311\Lib\site-packages\keras\src\losses.py:2976: The name tf.losses.spar
se_softmax_cross_entropy is deprecated. Please use tf.compat.v1.losses.spa
rse_softmax_cross_entropy instead.
```

In [4]:
```
# Set the maximum number of features (words in the vocabulary) and maximum l
df=pd.read_csv('IMDB Dataset.csv')

df.head()
```

Out[4]:

	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 [5]:
```
max_features = 10000
maxlen = 200

# Initialize a tokenizer and fit it on the movie review text data
tokenizer = Tokenizer(num_words=max_features)
tokenizer.fit_on_texts(df['review'])

# Convert the text data into sequences of integers and pad them to ensure ur
X = tokenizer.texts_to_sequences(df['review'])
X = pad_sequences(X, maxlen=maxlen)
y = df['sentiment']
```

In [6]:
```
# Convert the categorical sentiment labels into one-hot encoded vectors
y = pd.get_dummies(y)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra
```

In [6]:
```
# # Initialize a Sequential model
# model = Sequential()

# # Add an Embedding layer to convert word indices to dense vectors
# model.add(Embedding(input_dim=max_features, output_dim=128, input_length=m

# # Add an LSTM layer with dropout for sequence processing
# model.add(LSTM(units=64, dropout=0.2, recurrent_dropout=0.2))

# # Add a Dense layer for classification with softmax activation
# model.add(Dense(units=2, activation='softmax'))
```

```
WARNING:tensorflow:From C:\Users\Ownerqp\AppData\Local\Programs\Python\Pyt
hon311\Lib\site-packages\keras\src\backend.py:873: The name tf.get_default
_graph is deprecated. Please use tf.compat.v1.get_default_graph instead.
```

In [9]:
```
# Initialize a Sequential model using the list format
model = Sequential([
    # Add an Embedding layer to convert word indices to dense vectors
    Embedding(input_dim=max_features, output_dim=128, input_length=maxlen),
    # Add an LSTM layer with dropout for sequence processing
    LSTM(units=64, dropout=0.2, recurrent_dropout=0.2),
    # Add a Dense layer for classification with softmax activation
    Dense(units=2, activation='softmax')
])
```

In [10]:
```
# Compile the model with appropriate optimizer, loss function, and metrics
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['

# Train the model
model.fit(X_train, y_train, batch_size=32, epochs=5, validation_data=(X_tes
```

```
WARNING:tensorflow:From C:\Users\Ownerqp\AppData\Local\Programs\Python\Pyt
hon311\Lib\site-packages\keras\src\optimizers\__init__.py:309: The name t
f.train.Optimizer is deprecated. Please use tf.compat.v1.train.Optimizer i
nstead.

Epoch 1/5
WARNING:tensorflow:From C:\Users\Ownerqp\AppData\Local\Programs\Python\Pyt
hon311\Lib\site-packages\keras\src\utils\tf_utils.py:492: The name tf.ragg
ed.RaggedTensorValue is deprecated. Please use tf.compat.v1.ragged.RaggedT
ensorValue instead.

WARNING:tensorflow:From C:\Users\Ownerqp\AppData\Local\Programs\Python\Pyt
hon311\Lib\site-packages\keras\src\engine\base_layer_utils.py:384: The nam
e tf.executing_eagerly_outside_functions is deprecated. Please use tf.comp
at.v1.executing_eagerly_outside_functions instead.

1250/1250 [==============================] - 376s 293ms/step - loss: 0.368
2 - accuracy: 0.8369 - val_loss: 0.2912 - val_accuracy: 0.8782
Epoch 2/5
1250/1250 [==============================] - 364s 291ms/step - loss: 0.235
5 - accuracy: 0.9080 - val_loss: 0.2682 - val_accuracy: 0.8854
Epoch 3/5
1250/1250 [==============================] - 371s 296ms/step - loss: 0.176
1 - accuracy: 0.9330 - val_loss: 0.3096 - val_accuracy: 0.8864
Epoch 4/5
1250/1250 [==============================] - 370s 296ms/step - loss: 0.142
4 - accuracy: 0.9474 - val_loss: 0.3119 - val_accuracy: 0.8788
Epoch 5/5
1250/1250 [==============================] - 373s 298ms/step - loss: 0.108
3 - accuracy: 0.9617 - val_loss: 0.3374 - val_accuracy: 0.8838
```

Out[10]:
```
<keras.src.callbacks.History at 0x2441827e390>
```

In [11]:
```
#save the trained model
model.save("sentiment_model.h5")
```

```
C:\Users\Ownerqp\AppData\Local\Programs\Python\Python311\Lib\site-packages
\keras\src\engine\training.py:3103: UserWarning: You are saving your model
as an HDF5 file via `model.save()`. This file format is considered legacy.
We recommend using instead the native Keras format, e.g. `model.save('my_m
odel.keras')`.
  saving_api.save_model(
```

In [12]:
```
loss, accuracy = model.evaluate(X_test, y_test)
print("Test Loss:", loss)
print("Test Accuracy:", accuracy)
```

```
313/313 [==============================] - 16s 50ms/step - loss: 0.3374 -
accuracy: 0.8838
Test Loss: 0.3374056816101074
Test Accuracy: 0.8838000297546387
```

In [13]:
```
# Assuming `new_data` is a list containing new reviews
new_data = [
    "This movie was fantastic. I loved every moment of it!",
    "Terrible movie. I regret wasting my time watching it."
    "This movie was awesome. I loved each shoot of movie!",
    "Horrific movie. I regret wasting my money on it."
]

# Tokenize and pad sequences for new data
new_sequences = tokenizer.texts_to_sequences(new_data)
new_sequences = pad_sequences(new_sequences, maxlen=maxlen)

# Use the trained model to predict sentiment
predictions = model.predict(new_sequences)

# Decode predictions
labels = ['Negative', 'Positive']
for i, prediction in enumerate(predictions):
    label = labels[prediction.argmax()]
    print(f"Review: {new_data[i]}")
    print(f"Predicted sentiment: {label}")
    print()
```

```
1/1 [==============================] - 1s 1s/step
Review: This movie was fantastic. I loved every moment of it!
Predicted sentiment: Positive

Review: Terrible movie. I regret wasting my time watching it.This movie wa
s awesome. I loved each shoot of movie!
Predicted sentiment: Negative

Review: Horrific movie. I regret wasting my money on it.
Predicted sentiment: Negative
```

In [1]:
```
# Take input from the user
user_input = input("Enter a movie review: ")

# Tokenize and pad sequence for the user input
input_sequence = tokenizer.texts_to_sequences([user_input])
input_sequence = pad_sequences(input_sequence, maxlen=maxlen)

# Use the trained model to predict sentiment
prediction = model.predict(input_sequence)

# Decode the prediction
label = labels[prediction.argmax()]

# Print the predicted sentiment
print("Predicted sentiment:", label)
```

```
Enter a movie review: This is the kind of movie that is impossible to do j
ustice, just by talking about it! It is the kind of experience you had onc
e.. but you never thought you would get again.. until this movie proves yo
u WRONG!!  This movie takes the aspects of the first movie and improves up
on them in almost every way possible, already writing itself into the book
s of greatest sequels of ALL TIME!!

---------------------------------------------------------------------------
-
NameError                                 Traceback (most recent call las
t)
Cell In[1], line 5
      2 user_input = input("Enter a movie review: ")
      4 # Tokenize and pad sequence for the user input
----> 5 input_sequence = tokenizer.texts_to_sequences([user_input])
      6 input_sequence = pad_sequences(input_sequence, maxlen=maxlen)
      8 # Use the trained model to predict sentiment

NameError: name 'tokenizer' is not defined
```

In [ ]: