In [1]:
```
import tensorflow as tf
from tensorflow.keras.datasets import imdb
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense
from tensorflow.keras.preprocessing.sequence import pad_sequences
```

In [2]:
```
# Set the parameters
max_features = 10000
maxlen = 100
batch_size = 32
# Number of words to consider as features
# Cut texts after thisnumber of words (among top max_features most commonwords
```

In [3]:
```
# Load the IMDB dataset
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=max_features)
```

In [4]:
```
# Pad sequences to have a consistent length for the input to the RNN
x_train = pad_sequences(x_train, maxlen=maxlen)
x_test = pad_sequences(x_test, maxlen=maxlen)
```

In [5]:
```
# Build the RNN model with LSTM
model = Sequential()
model.add(Embedding(max_features, 128))
model.add(LSTM(64, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(1, activation='sigmoid'))
```

In [6]:
```
# Compile the model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy
```

In [7]:
```
# Train the model
model.fit(x_train, y_train, batch_size=batch_size, epochs=5, validation_data=(
```

```
Epoch 1/5
782/782 ━━━━━━━━━━━━━━━━━━━━ 111s 134ms/step - accuracy: 0.7255 - loss: 0.53
75 - val_accuracy: 0.7850 - val_loss: 0.4572
Epoch 2/5
782/782 ━━━━━━━━━━━━━━━━━━━━ 521s 666ms/step - accuracy: 0.8344 - loss: 0.38
19 - val_accuracy: 0.8351 - val_loss: 0.3750
Epoch 3/5
782/782 ━━━━━━━━━━━━━━━━━━━━ 103s 131ms/step - accuracy: 0.8822 - loss: 0.28
86 - val_accuracy: 0.8398 - val_loss: 0.3712
Epoch 4/5
782/782 ━━━━━━━━━━━━━━━━━━━━ 104s 133ms/step - accuracy: 0.8996 - loss: 0.25
08 - val_accuracy: 0.8364 - val_loss: 0.3962
Epoch 5/5
782/782 ━━━━━━━━━━━━━━━━━━━━ 104s 133ms/step - accuracy: 0.9246 - loss: 0.19
63 - val_accuracy: 0.8396 - val_loss: 0.4253
```

Out[7]:
```
<keras.src.callbacks.history.History at 0x19f4ea79f30>
```

In [8]:
```
# Evaluate the model
score, acc = model.evaluate(x_test, y_test, batch_size=batch_size)
print(f'Test score: {score}')
print(f'Test accuracy: {acc}')
```

```
782/782 ━━━━━━━━━━━━━━━━━━━━ 22s 28ms/step - accuracy: 0.8368 - loss: 0.4366
Test score: 0.42526325583457947
Test accuracy: 0.8396000266075134
```

In [ ]: