```
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Embedding, SpatialDropout1D
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.datasets import imdb
from sklearn.metrics import accuracy_score
```

```
# Load the IMDb dataset (keep only the top 10,000 most frequent words)
max_words = 10000
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=max_words)

# Check sample data
print(f"Sample Review (Token IDs): {x_train[0]}")
print(f"Sentiment: {y_train[0]}")  # 1 = positive, 0 = negative
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/imdb.npz
17464789/17464789 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Sample Review (Token IDs): [1, 14, 22, 16, 43, 530, 973, 1622, 1385, 65, 458, 4468, 66, 3941, 4, 173, 36, 256, 5, 25, 100, 43, 838, 112,
Sentiment: 1
```

```
max_sequence_length = 100  # Maximum review length

# Pad the sequences to ensure uniform input size
x_train = pad_sequences(x_train, maxlen=max_sequence_length)
x_test = pad_sequences(x_test, maxlen=max_sequence_length)

print(f"Padded Review: {x_train[0]}")
print(f"Shape of Training Data: {x_train.shape}")
```

```
Padded Review: [1415   33    6   22   12  215   28   77   52    5   14  407   16   82
    2    8    4  107  117 5952   15  256    4    2    7 3766    5  723
   36   71   43  530  476   26  400  317   46    7    4    2 1029   13
  104   88    4  381   15  297   98   32 2071   56   26  141    6  194
 7486   18    4  226   22   21  134  476   26  480    5  144   30 5535
   18   51   36   28  224   92   25  104    4  226   65   16   38 1334
   88   12   16  283    5   16 4472  113  103   32   15   16 5345   19
  178   32]
Shape of Training Data: (25000, 100)
```

```
embedding_dim = 100  # Dimension of the word embeddings

model = Sequential()
model.add(Embedding(input_dim=max_words, output_dim=embedding_dim, input_length=max_sequence_length))
model.add(SpatialDropout1D(0.2))  # Dropout to prevent overfitting
model.add(LSTM(100, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(1, activation='sigmoid'))  # Binary output (positive/negative)

# Compile the model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# Model summary
model.summary()
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument `input_length` is deprecated. Just
  warnings.warn(
Model: "sequential"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃         Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━┩
│ embedding (Embedding)                │ ?                           │     0 (unbuilt) │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ spatial_dropout1d (SpatialDropout1D) │ ?                           │     0 (unbuilt) │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ lstm (LSTM)                          │ ?                           │     0 (unbuilt) │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense (Dense)                        │ ?                           │     0 (unbuilt) │
└──────────────────────────────────────┴─────────────────────────────┴─────────────────┘
 Total params: 0 (0.00 B)
 Trainable params: 0 (0.00 B)
```

```
batch_size = 64
epochs = 3

history = model.fit(x_train, y_train, validation_data=(x_test, y_test),
                    batch_size=batch_size, epochs=epochs, verbose=1)
```

```
Epoch 1/3
391/391 ━━━━━━━━━━━━━━━━━━━━ 105s 262ms/step - accuracy: 0.7009 - loss: 0.5526 - val_accuracy: 0.8310 - val_loss: 0.3862
Epoch 2/3
391/391 ━━━━━━━━━━━━━━━━━━━━ 106s 270ms/step - accuracy: 0.8657 - loss: 0.3266 - val_accuracy: 0.8466 - val_loss: 0.3583
Epoch 3/3
391/391 ━━━━━━━━━━━━━━━━━━━━ 104s 265ms/step - accuracy: 0.8909 - loss: 0.2777 - val_accuracy: 0.8459 - val_loss: 0.3695
```

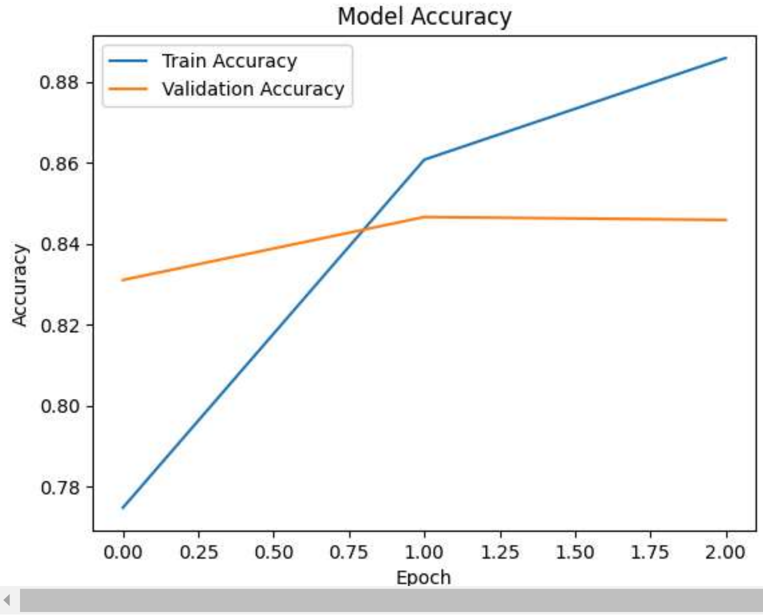
```
# Evaluate the model on test data
test_loss, test_accuracy = model.evaluate(x_test, y_test, verbose=0)
print(f"Test Accuracy: {test_accuracy:.2f}")

# Make predictions
y_pred = (model.predict(x_test) > 0.5).astype("int32")
print(f"Sample Prediction: {y_pred[0]}")
```

```
Test Accuracy: 0.85
782/782 ━━━━━━━━━━━━━━━━━━━━ 22s 28ms/step
Sample Prediction: [0]
```

```
import matplotlib.pyplot as plt

# Plot training & validation accuracy
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend()
plt.show()
```


Model Accuracy
Train Accuracy
Validation Accuracy
Accuracy
Epoch