```
import tensorflow as tf
from tensorflow.keras.datasets import imdb
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, Dense, LSTM, Dropout, Flatten
from tensorflow.keras.optimizers import Adam

# Load data
num_words = 10000
max_len = 200
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=num_words)

# Pad sequences to ensure uniform length
x_train = pad_sequences(x_train, maxlen=max_len)
x_test = pad_sequences(x_test, maxlen=max_len)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/imdb.npz
17464789/17464789 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

```
# Build the model
model = Sequential()
model.add(Embedding(input_dim=num_words, output_dim=128, input_length=max_len))
model.add(LSTM(128, return_sequences=False))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))

# Compile the model
model.compile(optimizer=Adam(learning_rate=0.001), loss='binary_crossentropy', metrics=['accuracy'])
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument `input_length` is deprecated. J
  warnings.warn(
```

```
# Train the model
history = model.fit(x_train, y_train, epochs=2, batch_size=64, validation_data=(x_test, y_test))
```

```
Epoch 1/2
391/391 ━━━━━━━━━━━━━━━━━━━━ 345s 883ms/step - accuracy: 0.8569 - loss: 0.3491 - val_accuracy: 0.8693 - val_loss: 0.3191
Epoch 2/2
391/391 ━━━━━━━━━━━━━━━━━━━━ 387s 896ms/step - accuracy: 0.9143 - loss: 0.2322 - val_accuracy: 0.8590 - val_loss: 0.3340
```

```
# Evaluate on test data
test_loss, test_acc = model.evaluate(x_test, y_test)

print(f'Test Loss: {test_loss}')
print(f'Test Accuracy: {test_acc}')
```

```
782/782 ━━━━━━━━━━━━━━━━━━━━ 125s 161ms/step - accuracy: 0.8583 - loss: 0.3381
Test Loss: 0.333985298871994
Test Accuracy: 0.8590400218963623
```

```
# Function to preprocess and predict
def predict_review(review):
    review_seq = imdb.get_word_index()
    review_words = [review_seq.get(word, 0) for word in review.split()]
    review_padded = pad_sequences([review_words], maxlen=max_len)
    prediction = model.predict(review_padded)
    return "Positive" if prediction > 0.5 else "Negative"

# Example prediction
review_text = "The movie was excellent and the performances were outstanding"
print(predict_review(review_text))
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/imdb_word_index.json
1641221/1641221 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 430ms/step
Positive
```