```
import numpy as np
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
```

```
# Load the IMDb dataset
max_features = 5000  # Vocabulary size
max_len = 200  # Maximum length of a review
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=max_features)
```

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

```
# Pad sequences to ensure each review is of the same length (max_len)
x_train = pad_sequences(x_train, maxlen=max_len)
x_test = pad_sequences(x_test, maxlen=max_len)
```

```
# Build the LSTM model
model = Sequential()
model.add(Embedding(max_features, 128, input_length=max_len))  # Embedding layer
model.add(LSTM(128))  # LSTM layer
model.add(Dense(1, activation='sigmoid'))  # Output layer
```

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

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

```
# Train the model
model.fit(x_train, y_train, epochs=3, batch_size=64, validation_data=(x_test, y_test))
```

```
Epoch 1/3
391/391 ━━━━━━━━━━━━━━━━━━━━ 286s 726ms/step - accuracy: 0.6390 - loss: 0.6770 - val_accuracy: 0.8221 - val_loss: 0.4105
Epoch 2/3
391/391 ━━━━━━━━━━━━━━━━━━━━ 323s 729ms/step - accuracy: 0.8483 - loss: 0.3566 - val_accuracy: 0.8401 - val_loss: 0.3621
Epoch 3/3
391/391 ━━━━━━━━━━━━━━━━━━━━ 310s 699ms/step - accuracy: 0.8890 - loss: 0.2781 - val_accuracy: 0.8724 - val_loss: 0.3098
<keras.src.callbacks.history.History at 0x786c0a99a5f0>
```

```
# Function to preprocess and pad a new review
def preprocess_review(review, max_len=200):
    word_index = imdb.get_word_index()  # Get the word index from IMDb
    tokens = [word_index.get(word, 0) for word in review.lower().split()]  # Convert words to integers
    return pad_sequences([tokens], maxlen=max_len)  # Pad the sequence
```

```
# Function to predict sentiment of a new review
def predict_sentiment(review):
    processed_review = preprocess_review(review)
    prediction = model.predict(processed_review)[0][0]  # Get prediction
    sentiment = "Positive" if prediction > 0.5 else "Negative"
    print(f"Review: {review}")
    print(f"Predicted Sentiment: {sentiment} (Confidence: {prediction:.4f})")
```

```
# Example reviews
new_review_1 = "This movie was absolutely amazing! I loved it."
new_review_2 = "This movie is not good. Do not watch. "
```

```
# Test with new reviews
predict_sentiment(new_review_1)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step
Review: This movie was absolutely amazing! I loved it.
Predicted Sentiment: Positive (Confidence: 0.5620)
```

```
predict_sentiment(new_review_2)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
Review: This movie is not good. Do not watch.
Predicted Sentiment: Negative (Confidence: 0.4064)
```

```
# Save the model to a file
model.save('sentiment_analysis_model.h5')
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered
```

```
# Save the model to Google Drive
model.save('/content/drive/My Drive/sentiment_analysis_model.h5')
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered
```