In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
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, LSTM, Dense, Bidirectional
from tensorflow.keras.utils import to_categorical
```

In [2]:
```
# Load the IMDB dataset
(train_data, train_labels), (test_data, test_labels) = imdb.load_data(num_words=10000)

# Display dataset shape
print(f'Training data shape: {train_data.shape}')
print(f'Training labels shape: {train_labels.shape}')
print(f'Testing data shape: {test_data.shape}')
print(f'Testing labels shape: {test_labels.shape}')
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/imdb.npz (https://storage.googleapis.com/tensorflow/tf-keras-datasets/imdb.npz)
17464789/17464789 [==============================] - 9s 1us/step
Training data shape: (25000,)
Training labels shape: (25000,)
Testing data shape: (25000,)
Testing labels shape: (25000,)
```

In [3]:
```
# Pad sequences to a maximum length of 500
max_length = 500
train_data = pad_sequences(train_data, maxlen=max_length)
test_data = pad_sequences(test_data, maxlen=max_length)

# Convert labels to categorical (if necessary)
train_labels = np.array(train_labels)
test_labels = np.array(test_labels)

# Check padded data shape
print(f'Padded training data shape: {train_data.shape}')
print(f'Padded testing data shape: {test_data.shape}')
```

```
Padded training data shape: (25000, 500)
Padded testing data shape: (25000, 500)
```

In [4]:
```
def create_model(is_bidirectional=False):
    model = Sequential()
    model.add(Embedding(input_dim=10000, output_dim=128, input_length=max_length))
    if is_bidirectional:
        model.add(Bidirectional(LSTM(64)))
    else:
        model.add(LSTM(64))
    model.add(Dense(1, activation='sigmoid'))  # Binary classification
    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model

# Create LSTM model
lstm_model = create_model()
lstm_model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 500, 128)          1280000

 lstm (LSTM)                 (None, 64)                49408

 dense (Dense)               (None, 1)                 65

=================================================================
Total params: 1,329,473
Trainable params: 1,329,473
Non-trainable params: 0
_________________________________________________________________
```

In [5]:
```
# Train LSTM model
lstm_history = lstm_model.fit(train_data, train_labels, epochs=5, batch_size=64, validation_split=0.2
```

```
Epoch 1/5
313/313 [==============================] - 227s 710ms/step - loss: 0.4482 - accuracy: 0.7889 - val_l
oss: 0.3199 - val_accuracy: 0.8662
Epoch 2/5
313/313 [==============================] - 370s 1s/step - loss: 0.2427 - accuracy: 0.9053 - val_los
s: 0.3345 - val_accuracy: 0.8526
Epoch 3/5
313/313 [==============================] - 497s 2s/step - loss: 0.1830 - accuracy: 0.9317 - val_los
s: 0.4105 - val_accuracy: 0.8656
Epoch 4/5
313/313 [==============================] - 231s 735ms/step - loss: 0.1300 - accuracy: 0.9534 - val_l
oss: 0.3595 - val_accuracy: 0.8628
Epoch 5/5
313/313 [==============================] - 247s 788ms/step - loss: 0.0962 - accuracy: 0.9674 - val_l
oss: 0.4457 - val_accuracy: 0.8542
```

In [6]:
```
# Evaluate LSTM model
lstm_loss, lstm_accuracy = lstm_model.evaluate(test_data, test_labels)
print(f'LSTM Test Accuracy: {lstm_accuracy:.2f}')
```

```
782/782 [==============================] - 91s 116ms/step - loss: 0.4756 - accuracy: 0.8485
LSTM Test Accuracy: 0.85
```

In [7]:
```
# Create BiLSTM model
bilstm_model = create_model(is_bidirectional=True)
bilstm_model.summary()

# Train BiLSTM model
bilstm_history = bilstm_model.fit(train_data, train_labels, epochs=5, batch_size=64, validation_split

# Evaluate BiLSTM model
bilstm_loss, bilstm_accuracy = bilstm_model.evaluate(test_data, test_labels)
print(f'BiLSTM Test Accuracy: {bilstm_accuracy:.2f}')
```

```
The history saving thread hit an unexpected error (OperationalError('database or disk is full')).His
tory will not be written to the database.
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_1 (Embedding)     (None, 500, 128)          1280000

 bidirectional (Bidirectiona  (None, 128)              98816
 l)

 dense_1 (Dense)             (None, 1)                 129

=================================================================
Total params: 1,378,945
Trainable params: 1,378,945
Non-trainable params: 0
_________________________________________________________________
Epoch 1/5
313/313 [==============================] - 481s 2s/step - loss: 0.4615 - accuracy: 0.7778 - val_los
s: 0.3427 - val_accuracy: 0.8578
Epoch 2/5
313/313 [==============================] - 1672s 5s/step - loss: 0.2683 - accuracy: 0.8947 - val_los
s: 0.3144 - val_accuracy: 0.8748
Epoch 3/5
313/313 [==============================] - 1849s 6s/step - loss: 0.1918 - accuracy: 0.9293 - val_los
s: 0.3092 - val_accuracy: 0.8686
Epoch 4/5
313/313 [==============================] - 446s 1s/step - loss: 0.1328 - accuracy: 0.9531 - val_los
s: 0.4077 - val_accuracy: 0.8698
Epoch 5/5
313/313 [==============================] - 458s 1s/step - loss: 0.1132 - accuracy: 0.9607 - val_los
s: 0.4178 - val_accuracy: 0.8438
782/782 [==============================] - 143s 182ms/step - loss: 0.4324 - accuracy: 0.8390
BiLSTM Test Accuracy: 0.84
```

In [8]:
```
# Plot training & validation accuracy
plt.plot(lstm_history.history['accuracy'], label='LSTM Training Accuracy')
plt.plot(lstm_history.history['val_accuracy'], label='LSTM Validation Accuracy')
plt.plot(bilstm_history.history['accuracy'], label='BiLSTM Training Accuracy')
plt.plot(bilstm_history.history['val_accuracy'], label='BiLSTM Validation Accuracy')
plt.title('Model Accuracy Comparison')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```