```
# Import necessary libraries
import pandas as pd
import numpy as np
import re
import pickle
import matplotlib.pyplot as plt
from textblob import TextBlob
from nltk.corpus import stopwords
from nltk import download
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout

# Download NLTK stopwords
download('stopwords')

# -------------------------------------
# STEP 1: Load the Dataset
# -------------------------------------
df=pd.read_csv("/content/drive/MyDrive/fake_or_real_news_large.csv")


# View data structure
print("Dataset preview:")
print(df.head())

# -------------------------------------
# STEP 2: Preprocess the Text
# -------------------------------------
print("Preprocessing text...")

def clean_text(text):
    text = re.sub(r'[^\w\s]', '', text.lower())  # Remove punctuation, lowercase
    return " ".join([word for word in text.split() if word not in stopwords.words('englis

# Apply cleaning and sentiment
df['clean_text'] = df['text'].apply(lambda x: clean_text(str(x)))
df['sentiment'] = df['text'].apply(lambda x: TextBlob(str(x)).sentiment.polarity)

# Map 'FAKE' and 'REAL' to 0 and 1
df['label'] = df['label'].map({'FAKE': 0, 'REAL': 1})

# -------------------------------------
# STEP 3: Tokenization and Sequencing
# -------------------------------------
print("Tokenizing text...")

tokenizer = Tokenizer(num_words=5000)
tokenizer.fit_on_texts(df['clean_text'])

X = tokenizer.texts_to_sequences(df['clean_text'])
X = pad_sequences(X, maxlen=300)
y = df['label'].values

# -------------------------------------
# STEP 4: Split Data for Training and Testing
# -------------------------------------
```

```
print("Splitting train and test data...")
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# -------------------------------------
# STEP 5: Build the LSTM Model
# -------------------------------------
print("Building LSTM model...")

model = Sequential([
    Embedding(input_dim=5000, output_dim=64, input_length=300),
    LSTM(64, dropout=0.2, recurrent_dropout=0.2),
    Dense(1, activation='sigmoid')
])

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

# -------------------------------------
# STEP 6: Train the Model
# -------------------------------------
print("Training model...")
history = model.fit(X_train, y_train, epochs=5, batch_size=64, validation_data=(X_test, y

# -------------------------------------
# STEP 7: Evaluate the Model
# -------------------------------------
print("Evaluating model...")
y_pred = (model.predict(X_test) > 0.5).astype("int32")

# Print performance metrics
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))

# -------------------------------------
# STEP 8: Plot Accuracy and Loss
# -------------------------------------
plt.figure(figsize=(12, 5))

# Accuracy plot
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy', marker='o')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy', marker='x')
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

# Loss plot
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss', marker='o')
plt.plot(history.history['val_loss'], label='Validation Loss', marker='x')
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.tight_layout()
plt.savefig("training_metrics.png")
plt.show()

# -------------------------------------
# STEP 9: Save the Model and Tokenizer
# -------------------------------------
model.save("lstm_model.h5")
```

```
with open("tokenizer.pkl", "wb") as f:
    pickle.dump(tokenizer, f)

print("Model and tokenizer saved successfully")
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
Dataset preview:
             title                                               text label
0  News headline 0  This is the body of news article number 0. It ...  REAL
1  News headline 1  This is the body of news article number 1. It ...  FAKE
2  News headline 2  This is the body of news article number 2. It ...  FAKE
3  News headline 3  This is the body of news article number 3. It ...  FAKE
4  News headline 4  This is the body of news article number 4. It ...  REAL
Preprocessing text...
Tokenizing text...
Splitting train and test data...
Building LSTM model...
Training model...
Epoch 1/5
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/embedding.py:90:
  warnings.warn(
2/2 ━━━━━━━━━━━━━━━━━━━━ 15s 2s/step - accuracy: 0.4292 - loss: 0.6934 - v
Epoch 2/5
2/2 ━━━━━━━━━━━━━━━━━━━━ 3s 1s/step - accuracy: 0.6219 - loss: 0.6922 - va
Epoch 3/5
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 1s/step - accuracy: 0.4917 - loss: 0.6931 - va
Epoch 4/5
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 632ms/step - accuracy: 0.4948 - loss: 0.6924 -
Epoch 5/5
2/2 ━━━━━━━━━━━━━━━━━━━━ 1s 652ms/step - accuracy: 0.4948 - loss: 0.6919 -
Evaluating model...
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 724ms/step
Accuracy: 0.4

Classification Report:
              precision    recall  f1-score   support
```