```
import pandas as pd
import numpy as np
import re
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
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
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt


def generate_fake_news_dataset(n_samples=1000):
    fake_patterns = [
        "A shocking revelation that will change everything",
        "Unbelievable truth hidden from the public",
        "Secret government conspiracy exposed",
        "Celebrities revealing underground networks",
        "Breaking news that mainstream media won't tell you"
    ]

    real_patterns = [
        "Scientific research confirms new findings",
        "Expert analysis provides insights",
        "Recent study highlights important trends",
        "Comprehensive report examines critical issues",
        "Researchers uncover significant data"
    ]

    data = {
        'text': [],
        'label': []
    }

    for _ in range(n_samples//2):

        data['text'].append(np.random.choice(fake_patterns) + " " +
                            ' '.join(np.random.choice(['dramatic', 'incredible', 'unbelievable']) for _ in range(5)))
        data['label'].append(1)


        data['text'].append(np.random.choice(real_patterns) + " " +
                            ' '.join(np.random.choice(['research', 'study', 'analysis']) for _ in range(5)))
        data['label'].append(0)

    df = pd.DataFrame(data)
    df.to_csv('fake_news_dataset.csv', index=False)
    return df


def preprocess_text(text):

    text = text.lower()


    text = re.sub(r'[^a-zA-Z\s]', '', text)


    text = re.sub(r'\s+', ' ', text).strip()

    return text


def main():

    df = generate_fake_news_dataset(n_samples=1000)
    print("Dataset Generated:")
    print(df['label'].value_counts())


    df['processed_text'] = df['text'].apply(preprocess_text)


    X_train, X_test, y_train, y_test = train_test_split(
        df['processed_text'],
        df['label'],
        test_size=0.2,
        random_state=42
    )


    tokenizer = Tokenizer(num_words=5000)
    tokenizer.fit_on_texts(X_train)

    X_train_seq = tokenizer.texts_to_sequences(X_train)
    X_test_seq = tokenizer.texts_to_sequences(X_test)


    max_length = 100
    X_train_pad = pad_sequences(X_train_seq, maxlen=max_length, padding='post')
    X_test_pad = pad_sequences(X_test_seq, maxlen=max_length, padding='post')


    model = Sequential([
        Embedding(5000, 128, input_length=max_length),
        LSTM(128, dropout=0.2),
        Dense(64, activation='relu'),
        Dropout(0.5),
        Dense(1, activation='sigmoid')
    ])

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


    early_stopping = EarlyStopping(
        monitor='val_loss',
        patience=5,
        restore_best_weights=True
    )

    history = model.fit(
        X_train_pad, y_train,
        epochs=20,
        batch_size=32,
        validation_split=0.2,
        callbacks=[early_stopping]
    )


    y_pred = model.predict(X_test_pad)
    y_pred_binary = (y_pred > 0.5).astype(int)

    print("\nClassification Report:")
    print(classification_report(y_test, y_pred_binary))

    cm = confusion_matrix(y_test, y_pred_binary)
    plt.figure(figsize=(8, 6))
    plt.imshow(cm, interpolation='nearest', cmap='Blues')
    plt.title('Confusion Matrix')
    plt.colorbar()
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.show()

if __name__ == "__main__":
    main()
```

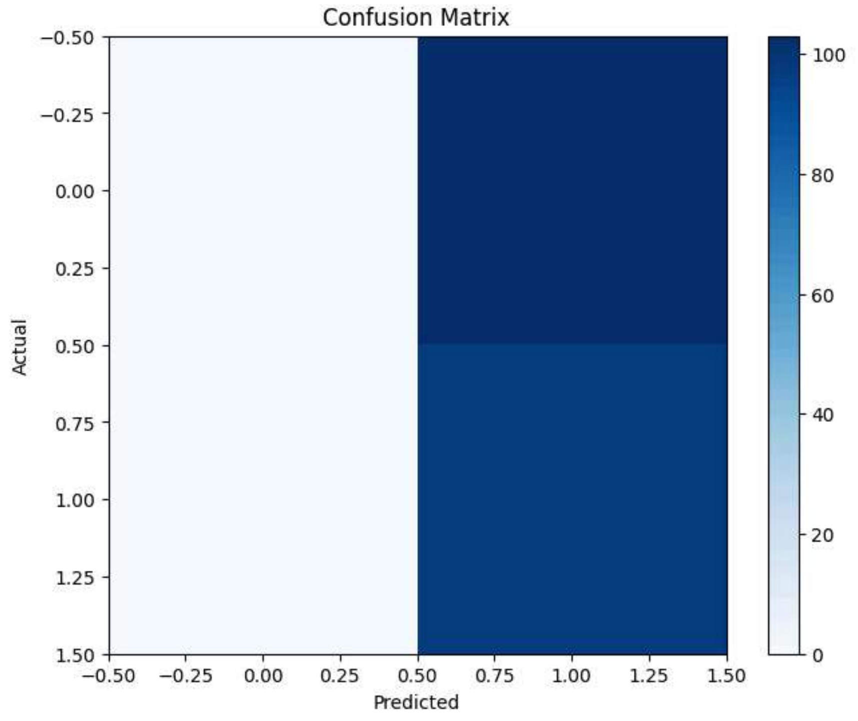
```
Dataset Generated:
label
1    500
0    500
Name: count, dtype: int64
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument `input_length` is deprecated. Just
  warnings.warn(
Epoch 1/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 7s 182ms/step - accuracy: 0.5008 - loss: 0.6933 - val_accuracy: 0.5188 - val_loss: 0.6924
Epoch 2/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 5s 276ms/step - accuracy: 0.4822 - loss: 0.6961 - val_accuracy: 0.5188 - val_loss: 0.6929
Epoch 3/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 3s 169ms/step - accuracy: 0.4885 - loss: 0.6937 - val_accuracy: 0.4812 - val_loss: 0.6934
Epoch 4/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 5s 171ms/step - accuracy: 0.5104 - loss: 0.6939 - val_accuracy: 0.5188 - val_loss: 0.6930
Epoch 5/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 7s 249ms/step - accuracy: 0.4997 - loss: 0.6938 - val_accuracy: 0.4812 - val_loss: 0.6935
Epoch 6/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 3s 172ms/step - accuracy: 0.5062 - loss: 0.6940 - val_accuracy: 0.4812 - val_loss: 0.6932
7/7 ━━━━━━━━━━━━━━━━━━━━ 1s 77ms/step

Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       103
           1       0.48      1.00      0.65        97

    accuracy                           0.48       200
   macro avg       0.24      0.50      0.33       200
weighted avg       0.24      0.48      0.32       200

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and be
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and be
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and be
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```


Confusion Matrix
Actual
Predicted