CODE:

```
import tensorflow as tf
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
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.naive_bayes import MultinomialNB
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem import SnowballStemmer

# Load the dataset
df = pd.read_csv('train1.csv')
pd.set_option('display.max_colwidth',-1)
df.head()
df.tail()
df.shape

# Preprocess the text data
stop_words = set(stopwords.words('english'))
stemmer = SnowballStemmer('english')
df['Sentence1'] = df['Sentence1'].apply(lambda x: ' '.join([stemmer.stem(word.lower()) for word in x.split() if word.lower() not in stop_words]))
```

```
df['Sentence2'] = df['Sentence2'].apply(lambda x: ' '.join([stemmer.stem(word.lower()) for word in x.split() if word.lower() not in stop_words]))


#Long-Short Term Memory(LSTM) Model

# Split the dataset into training and testing sets

train_size = int(len(df) * 0.8)

train_data = df[:train_size]

test_data = df[train_size:]


# Tokenize the text data

tokenizer = Tokenizer(num_words=5000)

tokenizer.fit_on_texts(train_data['Sentence1'])


X_train = tokenizer.texts_to_sequences(train_data['Sentence1'])

X_train = pad_sequences(X_train, maxlen=100)


X_test = tokenizer.texts_to_sequences(test_data['Sentence1'])

X_test = pad_sequences(X_test, maxlen=100)


# Define the LSTM model architecture

model = Sequential()

model.add(Embedding(5000, 128, input_length=100))

model.add(LSTM(128))

model.add(Dense(1, activation='sigmoid'))


# Compile the model

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


# Train the model

model.fit(X_train, train_data['Class'], batch_size=64, epochs=10, validation_data=(X_test, test_data['Class']))
```

```
# Evaluate the model

loss, accuracy = model.evaluate(X_test, test_data['Class'])


###Example:When there is paraphrase detection

new_text = ["A barometer and a built-in scale are shown in this room .", "In this room are presented a barometer and a built-in scale ."]

new_text = tokenizer.texts_to_sequences(new_text)

new_text = pad_sequences(new_text, maxlen=100)

prediction = model.predict(new_text)

print(prediction)


###Example2: When there is no paraphrase detection

new_text = ["How do I file for bankruptcy?", "Can you file for bankruptcy twice?"]

new_text = tokenizer.texts_to_sequences(new_text)

new_text = pad_sequences(new_text, maxlen=100)

prediction = model.predict(new_text)

print(prediction)


#Logistic Regression(LR) Model

data = pd.read_csv('train1.csv')

# Split data into training and testing sets

train_data, test_data, train_labels, test_labels = train_test_split(data['Sentence1'].astype(str) + ' ' + data['Sentence2'].astype(str),

                                                    data['Class'],

                                                    test_size=0.2,

                                                    random_state=42)


# Vectorize the text data

vectorizer = CountVectorizer(stop_words='english')

train_vectors = vectorizer.fit_transform(train_data)

test_vectors = vectorizer.transform(test_data)
```

```
# Train logistic regression model
model = LogisticRegression(max_iter=1000)
model.fit(train_vectors, train_labels)

# Make predictions on test data
pred_labels = model.predict(test_vectors)

# Evaluate model accuracy
accuracy = accuracy_score(test_labels, pred_labels)
print("Model accuracy:", accuracy)

#Naive Bayes Model
# Split the dataset into training and testing sets
train_set = data.sample(frac=0.8, random_state=42)
test_set = data.drop(train_set.index)

# Create a CountVectorizer object to convert text into numerical vectors
vectorizer = CountVectorizer()

# Convert the training and testing sets into numerical vectors
X_train = vectorizer.fit_transform(train_set["Sentence1"])
X_test = vectorizer.transform(test_set["Sentence1"])
y_train = train_set["Class"]
y_test = test_set["Class"]

# Train the Naive Bayes classifier on the training set
clf = MultinomialNB()
clf.fit(X_train, y_train)

# Test the classifier on the testing set
```

```
y_pred = clf.predict(X_test)



# Calculate the accuracy of the classifier
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

##Comparision:
# Data for the bar graph
algorithms = ['LSTM', 'Logistic Regression', 'Naive Bayes']
accuracy = [0.95, 0.53, 0.54]  # Accuracy scores for each algorithm

# Plotting the bar graph
fig, ax = plt.subplots()
ax.bar(algorithms, accuracy, color=['b', 'g', 'r'])
ax.set_ylim([0, 1])  # Setting the y-axis limit to be between 0 and 1
ax.set_xlabel('Algorithms')
ax.set_ylabel('Accuracy')
ax.set_title('Accuracy of Three Machine Learning Algorithms')
# Displaying the graph
plt.show()
```

DATASET LINK:

https://github.com/wasiahmad/paraphrase_identification/tree/master/dataset