import re

import numpy as np

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.linear\_model import LogisticRegression

from nltk.corpus import stopwords

import nltk

import pandas as pd

nltk.download('stopwords')

def clean\_text(text):

    text = text.lower()

    text = re.sub(r'<[^>]+>|https?://\S+|www\.\S+|[^\w\s]|\d+', ' ', text)

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

    stops = set(stopwords.words('english')) - {'not', 'no', 'never'}

    words = [word for word in text.split() if word not in stops]

    return ' '.join(words)

def load\_and\_prepare\_data():

    df = pd.read\_csv('/content/Test.csv')

    df['cleaned\_text'] = df['text'].apply(clean\_text)

    reviews = df['cleaned\_text'].tolist()

    sentiments = df['label'].tolist()

    neutral\_reviews = [

        "It was okay, nothing special.",

        "Average movie with some flaws.",

        "Not great but not terrible.",

        "Mediocre at best.",

        "Had potential but fell short."

    ]

    neutral\_sentiments = [2] \* len(neutral\_reviews)

    reviews.extend([clean\_text(review) for review in neutral\_reviews])

    sentiments.extend(neutral\_sentiments)

    return reviews, sentiments

def train\_model(reviews, sentiments):

    vectorizer = TfidfVectorizer(ngram\_range=(1, 2), max\_features=5000)

    X = vectorizer.fit\_transform(reviews)

    y = np.array(sentiments)

    model = LogisticRegression(class\_weight='balanced', max\_iter=1000)

    model.fit(X, y)

    return vectorizer, model

def predict\_sentiment(text, vectorizer, model):

    cleaned = clean\_text(text)

    vec = vectorizer.transform([cleaned])

    proba = model.predict\_proba(vec)[0]

    pred = model.predict(vec)[0]

    labels = ["NEGATIVE", "POSITIVE", "NEUTRAL"]

    return labels[pred], max(proba)

def analyze\_review(vectorizer, model):

    print("\n  Movie Review Sentiment Analyzer  \n")

    while True:

        review = input("Enter a review (or 'quit'): ").strip()

        if review.lower() == 'quit':

            break

        if len(review) < 10:

            print("Please enter a longer review.")

            continue

        sentiment, confidence = predict\_sentiment(review, vectorizer, model)

        color = "\033[92m" if sentiment == "POSITIVE" else "\033[91m" if sentiment == "NEGATIVE" else "\033[93m"

        print(f"\n{color}{sentiment}\033[0m (Confidence: {confidence:.0%})")

if \_\_name\_\_ == "\_\_main\_\_":

    reviews, sentiments = load\_and\_prepare\_data()

    vectorizer, model = train\_model(reviews, sentiments)

    analyze\_review(vectorizer, model)

