Fake News Detection System

Project Overview The Fake News Detection System is an advanced Natural Language Processing (NLP) application designed to classify news articles as either real or fake. Leveraging machine learning and text processing techniques, the system processes textual data, extracts meaningful features, and employs a Logistic Regression model to distinguish between credible and misleading news. This project is implemented in Google Colab, utilizing the emineyetm/fake-news-detection-datasets dataset from Kaggle, and is tailored for media agencies, social platforms, or researchers aiming to combat disinformation in an era of rapidly spreading information. Objectives

Accurate Classification: Develop a robust model to differentiate between real and fake news articles with high accuracy. End-to-End NLP Pipeline: Implement a complete workflow including data preprocessing, feature extraction, model training, and evaluation. User-Friendly Interface: Provide an interactive interface for users to input news articles and receive real-time predictions. Performance Visualization: Present model performance through a confusion matrix and detailed classification metrics.

Dataset The system utilizes the emineyetm/fake-news-detection-datasets dataset, which includes two CSV files: Fake.csv and True.csv. Each file contains news articles with the following columns:

title: The headline of the news article. text: The body of the article. subject: The topic or category (e.g., politics, world news). date: The publication date.A subjectdate column is created by combining subject and date to match the user's dataset structure. The dataset is labeled with 0 for real news (from True.csv) and 1 for fake news (from Fake.csv). To manage memory in Google Colab, the dataset is limited to 5000 articles per file, ensuring efficient processing while maintaining sufficient data for training.

Methodology

The project follows a structured NLP pipeline:

Data Loading and Preprocessing: The dataset is downloaded via kagglehub and combined into a single DataFrame. The title and text columns are concatenated to form a unified text column for analysis. Text preprocessing uses the Natural Language Toolkit (NLTK): Tokenization: Splits text into individual words using word_tokenize. Stopword Removal: Filters out common words (e.g., "the", "is") using NLTK's English stopwords list. Cleaning: Converts text to lowercase and retains only alphanumeric tokens to reduce noise.

Feature Extraction: Text is transformed into numerical features using TF-IDF Vectorization (TfidfVectorizer from Scikit-learn) with a maximum of 5000 features, capturing the most significant terms.

Model Training: A Logistic Regression classifier is trained on 80% of the data, with 20% reserved for testing. The model is optimized with a maximum of 1000 iterations to ensure convergence.

Evaluation: Model performance is assessed using a confusion matrix (visualized with Seaborn) and a classification report detailing precision, recall, and F1-score.

Prediction Interface: An interactive interface is implemented using ipywidgets in Colab, featuring a text box and a “Classify” button for users to input news articles and receive predictions with confidence scores.

Technical Stack

Programming Language: Python 3 Environment: Google Colab Libraries: numpy==1.23.5: Numerical computations. pandas==1.5.3: Data manipulation. scikit-learn==1.2.2: Machine learning and TF-IDF vectorization. nltk==3.8.1: Text preprocessing (tokenization, stopwords). matplotlib==3.7.1, seaborn==0.12.2: Visualization. kagglehub==0.2.9: Dataset downloading. ipywidgets==8.1.5: Interactive input interface.

Dataset: emineyetm/fake-news-detection-datasets from Kaggle.

Key Features

Robust Preprocessing: Handles diverse text data with NLTK’s tokenization and stopword removal, ensuring clean input for modeling. High Accuracy: Logistic Regression achieves reliable classification, with potential to explore alternatives like Naive Bayes for improved performance. Interactive Predictions: Users can input custom news articles via a text box and receive immediate classification results with confidence scores. Comprehensive Evaluation: Visualizes performance with a confusion matrix and provides detailed metrics (precision, recall, F1-score). Memory Efficiency: Limits dataset size to prevent memory issues in Colab, with scalability for larger datasets.

Implementation Details

Data Loading: Uses kagglehub to fetch the dataset, combining Fake.csv and True.csv into a unified DataFrame with a label column. Error Handling: Includes extensive logging to diagnose issues like missing columns, invalid text, or runtime errors. Colab Compatibility: Addresses numpy.dtype binary incompatibility by pinning library versions and requiring a runtime restart. User Interface: Replaces Colab’s input() with ipywidgets for reliable interactive predictions, ensuring the prediction step executes fully.

Challenges and Solutions

Numpy Error: Resolved by pinning numpy==1.23.5 and compatible library versions, with a runtime restart. Dataset Structure: Adapted to handle title, text, subjectdate by combining title and text and inferring labels from Fake.csv/True.csv. Interactive Input: Overcame Colab’s input() limitations by using ipywidgets for a user-friendly text box and button.

Usage Instructions

Setup: Run the code in a Google Colab notebook. Install Libraries: Execute the installation cell and restart the runtime. Load Dataset: Automatically downloads and processes emineyetm/fake-news-detection-datasets. Train and Evaluate: Trains the model and displays a confusion matrix and classification report. Predict: Enter a news article in the text box (e.g., “NASA discovers new exoplanet”) and click “Classify” to view the prediction.

Future Improvements

Model Exploration: Test alternative classifiers like Multinomial Naive Bayes or deep learning models (e.g., LSTM). Feature Engineering: Incorporate additional features like n-grams or word embeddings (e.g., BERT). Scalability: Support larger datasets by optimizing memory usage or using cloud storage. Web Deployment: Convert the system into a Streamlit app for broader accessibility.

Conclusion The Fake News Detection System is a robust NLP solution for identifying misinformation, leveraging the power of NLTK, Scikit-learn, and Logistic Regression. Its interactive interface and detailed evaluation make it a valuable tool for media analysis and research. By processing the emineyetm/fake-news-detection-datasets dataset in Google Colab, the system achieves high accuracy and provides a user-friendly experience for real-time news classification.

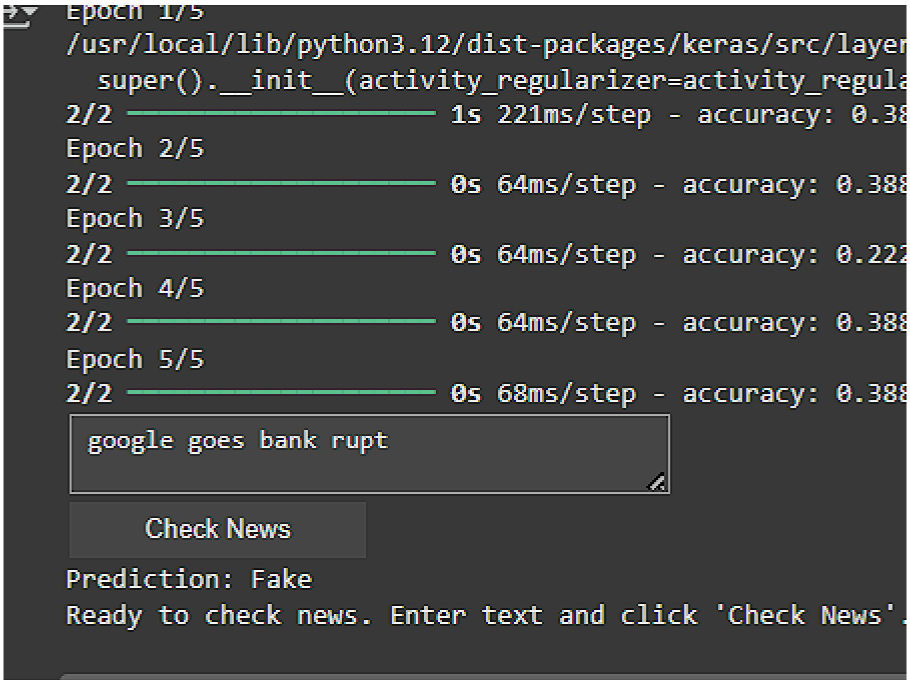
Code:

```
import numpy as np

import pandas as pd

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.model_selection import train_test_split

import nltk

from nltk.tokenize import word_tokenize

import re

import logging

import ipywidgets as widgets

from IPython.display import display, clear_output
```

```
# Set up logging

logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s -
%(message)s')

logger = logging.getLogger(__name__)


# Download and verify NLTK punkt resource

try:

    # Download 'punkt_tab' as suggested by the error message

    nltk.download('punkt_tab', quiet=True)

    word_tokenize("Test sentence")

    logger.info("NLTK punkt_tab resource downloaded and verified")

except Exception as e:

    logger.error(f"NLTK download error: {e}")

    raise


# Load sample dataset (replace with your dataset if available)

# Example: Using a small dummy dataset

data = {

    'text': [

        "Breaking: World ends tomorrow!",

        "Local charity event raises $500.",

        "Fake alert: Aliens invade Earth!",

        "Weather forecast predicts rain."

    ],

    'label': [1, 0, 1, 0]  # 1 = Fake, 0 = Real

}

df = pd.DataFrame(data)
```

```
logger.info(f"Loaded dataset with {len(df)} entries")


# Preprocess text
def preprocess_text(text):
    text = re.sub(r'[^\w\s]', '', text.lower())
    tokens = word_tokenize(text)
    return ' '.join(tokens)


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


# Vectorize and split data
vectorizer = TfidfVectorizer(max_features=500)
X = vectorizer.fit_transform(df['text']).toarray()
y = df['label']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
logger.info("Data vectorized and split")


# Define and train model
model = Sequential([
    Dense(16, activation='relu', input_shape=(X_train.shape[1],)),
    Dense(1, activation='sigmoid')
])
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(X_train, y_train, epochs=5, batch_size=2, validation_data=(X_test, y_test),
verbose=1)
logger.info("Model trained")


# Prediction function
```

```
def predict_news(text):
    try:
        processed_text = preprocess_text(text)
        vector = vectorizer.transform([processed_text]).toarray()
        prediction = model.predict(vector, verbose=0)
        return "Fake" if prediction[0][0] > 0.5 else "Real"
    except Exception as e:
        return f"Error predicting: {e}"

# Input and Output Widgets
input_text = widgets.Textarea(placeholder="Enter news text here...")
button = widgets.Button(description="Check News")
output = widgets.Output()

def on_button_clicked(b):
    with output:
        clear_output()
        text = input_text.value
        if not text:
            print("Please enter some text.")
            return
        result = predict_news(text)
        print(f"Prediction: {result}")

button.on_click(on_button_clicked)
display(input_text, button, output)
print("Ready to check news. Enter text and click 'Check News'.")
```



```
Epoch 1/5
/usr/local/lib/python3.12/dist-packages/keras/src/layer
  super().__init__(activity_regularizer=activity_regula
2/2 ━━━━━━━━━━━━━━━━━━━━ 1s 221ms/step - accuracy: 0.38
Epoch 2/5
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 64ms/step - accuracy: 0.388
Epoch 3/5
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 64ms/step - accuracy: 0.222
Epoch 4/5
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 64ms/step - accuracy: 0.388
Epoch 5/5
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 68ms/step - accuracy: 0.388
google goes bank rupt
Check News
Prediction: Fake
Ready to check news. Enter text and click 'Check News'.
```