NAAN MUDHALVAN

IBM: AI101

ARTIFICIAL INTELLIGENCE

PHASE 3

Fake News Detection Using NLP

PROJECT NO: 08

TEAM MEMBERS:

1. SIVA JEGADEESH C B
2. RAJKUMAR M
3. BABITH SARISH S
4. LOGESH S
5. SHARVESH

MENTOR

- Dr.SUDHAKAR T

MADRAS INSTITUTE OF TECHNOLOGY, ANNA UNIVERSITY, CHENNAI

Fake News Detection Using NLP

PHASE 3

Problem Statement: The fake news dataset is one of the classic text analytics datasets available on Kaggle. It consists of genuine and fake articles' titles and text from different authors. Our job is to create a modelwhich predicts whether a given news is real or fake.

In this crucial phase of our project, we embark on the foundational steps ofcreating a fake news detection model. Our primary objective is to load and preprocess the dataset, setting the stage for the subsequent stages of model development and training.

Detecting fake news is a pertinent challenge in the age of information overload, where misinformation can have far-reaching consequences. The effectiveness of our model hinges on the quality and appropriateness of thedata it is trained on, making dataset loading and preprocessing pivotal.

In the following sections, we will take a closer look at how we load the fakenews dataset, examine its structure, and implement preprocessing techniques to ensure that the textual data is in a format conducive to model training. These initial steps lay the groundwork for the developmentof a robust and accurate fake news detection model, which has the potential to contribute significantly to the critical task of distinguishing factfrom fiction in the digital age.

Data Source: We use a *fake news dataset* available on Kaggle. This dataset contains articles' titles and text, along with their correspondinglabels indicating whether the news is genuine or fake.

TOOLS:

Google Colab: Google Colab, a cloud-based Jupyter notebook environment, serves as our primary coding platform

Python and other libraries for natural language processing (NLP) andmachine learning.

IMPLEMENTATION STEPS:

1) **Import Necessary Libraries:**
   Start by importing the required Python libraries, such as pandas, numpy, scikit-learn, and natural language processing libraries like NLTK or spaCy.

   CODE:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import nltk
from nltk.corpus import stopwords
nltk.download('punkt')
nltk.download('stopwords')
```

2) **Load and Explore the Dataset:**
   Load the CSV files 'true.csv' and 'false.csv' using pandas and explorethe dataset to understand its structure.

   CODE:

```
# Load the dataset
true_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/fake-news-detective/dataset/True.csv')
false_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/fake-news-detective/dataset/Fake.csv')

# Add labels to indicate real and fake news
true_df['label'] = 1
false_df['label'] = 0

# Concatenate both datasets
data = pd.concat([true_df, false_df])
```

3) **Data Preprocessing:**
   Data preprocessing is essential for text data. Perform the following preprocessing steps:
   - **Lowercasing: Convert text to lowercase.**
   - **Tokenization: Split text into words or tokens.**
   - **Stopword Removal: Remove common words like 'and', 'the', etc.**

- **Text Vectorization: Convert text into numerical format (e.g., using TF-IDF or Count Vectorization).**

CODE:

```
# Lowercasing and tokenization
data['text'] = data['text'].str.lower()
data['title'] = data['title'].str.lower()
data['text'] = data['text'].apply(nltk.word_tokenize)
data['title'] = data['title'].apply(nltk.word_tokenize)

# Remove stopwords
stop_words = set(stopwords.words('english'))
data['text'] = data['text'].apply(lambda x: [word for word in x if word not in stop_words])
data['title'] = data['title'].apply(lambda x: [word for word in x if word not in stop_words])
```

4) Feature Extraction (TF-IDF):

Choose a text vectorization method. You can use either CountVectorization or TF-IDF Vectorization.

CODE:

```
tfidf_vectorizer = TfidfVectorizer(max_features=5000)
text_tfidf = tfidf_vectorizer.fit_transform(data['text'].apply(lambda x: ' '.join(x)))

title_tfidf = tfidf_vectorizer.transform(data['title'].apply(lambda x: ' '.join(x)))
```

5) Split the Data into Training and Testing Sets:

Split the dataset into training and testing sets to evaluate the model'sperformance.

CODE:

```
X = text_tfidf
y = data['label']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

The code for the dataset preprocessing and data splitting stages can be found in our GitHub repository, which streamlines the implementation process and ensures reproducibility. For access to this code, please refer to the following GitHub link: Fake New Detective.
https://github.com/vijaisuria/Fake-News-Detective

SAMPLE OUTPUT (Data Preprocessing and splitting):

```
# Display the shapes of training and testing data
print(f"X_train shape: {X_train.shape}")
print(f"X_test shape: {X_test.shape}")
print(f"y_train shape: {y_train.shape}")
print(f"y_test shape: {y_test.shape}")
```

```
X_train shape: (35918, 5000)
X_test shape: (8980, 5000)
y_train shape: (35918,)
y_test shape: (8980,)
```

Conclusion

In this section, we loaded the dataset, performed data preprocessing, andsplit the data into training and testing sets. The dataset is now ready for model building and evaluation.