FAKE NEWS DETECTION USING NLP

TECHNOLOGY: ARTIFICIAL INTELLIGENCE

NAME: M.VIJAY SASAN
NM ID: au311121205701

PHASE 3 - PROJECT SUBMISSION

GUIDELINES:

Phase 3: Development Part 1

In this part you will begin building your project by loading and preprocessing the dataset.

Begin building the fake news detection model by loading and preprocessing the dataset.

Load the fake news dataset and preprocess the textual data.

ABSTRACT:

In this project, we took the approach of building a fake news detection model by first preparing the dataset. We began by importing the required libraries, which included Pandas for data manipulation and NLTK for natural language processing tasks. We ensured access to essential NLTK resources by downloading 'punkt' for tokenization and 'stopwords' for common English stop words. Next, we loaded the dataset from a CSV file, creating a Pandas DataFrame for easy data handling. Our primary focus was on preprocessing the textual data within the dataset. This involved converting all text to lowercase to maintain consistency, tokenizing the text into individual words, removing common English stop words to eliminate less informative words, and applying stemming to reduce words to their root forms. The preprocessed tokens were then concatenated back into text format,

ready for further analysis. The approach ensures that text data is transformed into a more manageable and standardized format, making it suitable for subsequent steps, such as machine learning model development.

Importing Libraries

- import pandas as pd: This line imports the Pandas library and aliases it as 'pd'. Pandas is a data manipulation and analysis library used for handling data in tabular form, like CSV files.
- import nltk: This line imports the Natural Language Toolkit (NLTK), a library for working with human language data.

Downloading Necessary NLTK Resources

- nltk.download('punkt'): This command downloads the 'punkt' resource for NLTK. The 'punkt' resource includes data files used for tokenization, which is the process of splitting text into individual words or tokens.
- nltk.download('stopwords'): This command downloads the 'stopwords' resource, which contains a list of common English stop words. Stop words are words like 'and,' 'the,' 'is,' etc., which are often removed from text data during preprocessing because they are considered less informative.

Loading the Dataset

- df = pd.read_csv('Fake.csv'): This line reads the data from a CSV file named 'Fake.csv' using the Pandas library and stores it in a DataFrame called 'df.' A DataFrame is a tabular data structure used for data analysis. This line loads your dataset into memory so you can work with it.

Data Preprocessing for Textual Data

The following lines perform various text preprocessing steps on the 'text' column of the DataFrame:

- df['text'] = df['text'].str.lower(): This line converts all the text in the 'text' column to lowercase. This is a common preprocessing step to ensure that text data is treated consistently and doesn't differentiate between "word" and "Word," for example.
- df['text'] = df['text'].apply(word_tokenize): This line tokenizes the text, meaning it splits the text into individual words (tokens). The word_tokenize function from NLTK is used to perform this task.
- stop_words = set(stopwords.words('english')): Here, a set of common English stop words is created. The stopwords module from NLTK provides a list of these words.
- df['text'] = df['text'].apply(lambda words: [word for word in words if word not in stop_words]): This line removes common stop words from the tokenized text data. It uses a lambda function to filter out words that are present in the set of stop words. This step is important because stop words often do not provide valuable information for natural language processing tasks.
- stemmer = PorterStemmer(): The Porter Stemmer is an algorithm for reducing words to their root form. In this case, it's used for stemming the words in the text data. Stemming helps in standardizing words and reducing inflected words to their base or root form.
- df['text'] = df['text'].apply(lambda words: [stemmer.stem(word) for word in words]): This line applies stemming to the tokenized words, reducing them to their root form.
- df['text'] = df['text'].apply(lambda words: ' '.join(words)): After tokenization, stop word removal, and stemming, this line concatenates the tokens back into a single string, separated by spaces. This prepares the text data for further analysis or machine learning.

Displaying the Preprocessed Data

- print(df): Finally, this line prints the preprocessed DataFrame, which includes the 'text' column containing the preprocessed textual data.

This code segment demonstrates the entire process of importing necessary libraries, downloading NLTK resources, loading a dataset from a CSV file, and performing various text preprocessing steps to prepare the text data for analysis or modeling. The final preprocessed data is displayed for examination.

CODE:

```
import pandas as pd
import nltk

nltk.download('punkt')
nltk.download('stopwords') from
nltk.corpus import stopwords from
nltk.tokenize import word_tokenize from
nltk.stem import PorterStemmer

# Load the dataset from the CSV file
df = pd.read_csv('Fake.csv')

# Data preprocessing for textual data

# Lowercase the text
df['text'] = df['text'].str.lower()

# Tokenization (splitting text into words)
df['text'] = df['text'].apply(word_tokenize)
```

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

# Stemming (reducing words to their root form)
stemmer = PorterStemmer()
df['text'] = df['text'].apply(lambda words: [stemmer.stem(word) for word in words])
# Concatenate the tokens back into text df['text'] = df['text'].apply(lambda words: ' '.join(words))

# Display the preprocessed data
print(df)
```

OUTPUT:

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
                                                   title  \
0       Donald Trump Sends Out Embarrassing New Year'...
1       Drunk Bragging Trump Staffer Started Russian ...
2       Sheriff David Clarke Becomes An Internet Joke...
3       Trump Is So Obsessed He Even Has Obama's Name...
4       Pope Francis Just Called Out Donald Trump Dur...
...                                                   ...
23476  McPain: John McCain Furious That Iran Treated ...
23477  JUSTICE? Yahoo Settles E-mail Privacy Class-ac...
23478  Sunnistan: US and Allied 'Safe Zone' Plan to T...
23479  How to Blow $700 Million: Al Jazeera America F...
23480  10 U.S. Navy Sailors Held by Iranian Military ...

                                                   text      subject  \
0      donald trump wish american happi new year leav...         News
1      hous intellig committe chairman devin nune go ...         News
2      friday , reveal former milwauke sheriff david ...         News
3      christma day , donald trump announc would back...         News
4      pope franci use annual christma day messag reb...         News
...                                                  ...          ...
23476  21st centuri wire say 21wire report earlier we...  Middle-east
23477  21st centuri wire say familiar theme . whenev ...  Middle-east
23478  patrick henningsen 21st centuri wirerememb oba...  Middle-east
23479  21st centuri wire say al jazeera america go hi...  Middle-east
23480  21st centuri wire say 21wire predict new year ...  Middle-east

                   date
0      December 31, 2017
1      December 31, 2017
2      December 30, 2017
3      December 29, 2017
4      December 25, 2017
...                 ...
23476   January 16, 2016
23477   January 16, 2016
23478   January 15, 2016
23479   January 14, 2016
23480   January 12, 2016

[23481 rows x 4 columns]
```

INFERENCE:

- **Column Structure**: The DataFrame consists of four columns: 'title', 'text',

'subject', and 'date'. The 'title' and 'text' columns contain the textual content of the news articles, while 'subject' provides information about the subject or category of the article, and 'date' contains the publication date.

- **Textual Data Preprocessing**: The 'text' column has undergone several preprocessing steps. It has been converted to lowercase, tokenized into individual words, had common English stop words removed, and words have been stemmed to their root forms.
- **Cleaned Text**: The text in the 'text' column now contains words in lowercase, is tokenized into a list of words, and has had common stop words like 'the' or 'and' removed. Additionally, words have been stemmed to their root forms, simplifying the text data.
- **Data Size**: The dataset contains a total of 23,481 rows, each corresponding to a news article. This suggests a significant amount of data for potential analysis or model development.
- **Publication Dates**: The 'date' column contains publication dates for the articles, ranging from 2016 to the end of 2017.

In summary, the preprocessing steps have successfully cleaned and prepared the textual data for further analysis or machine learning tasks. The text data has been standardized and made more suitable for text-based tasks, such as fake news detection or text classification.