FAKE NEWS DETECTION USING NLP

Project Name : Fake News Detection Using NLP

Phase 3 : Development part 1

Topic : 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.

FAKE NEWS DETECTION USING NLP

Introduction :

Fake news detection is the process of identifying and verifying the accuracy of news or information that is intentionally false, misleading, or fabricated. It has become a critical concern in today's digital age, where misinformation can spread rapidly through various media channels. Here's an introduction to the topic.

Definition of Fake News: Fake news encompasses various types of misinformation, including fabricated stories, manipulated images or videos, and misleading headlines. It can be spread through websites, social media, or traditional media outlets.

Motivations for Fake News: Fake news can be created for various reasons, such as political manipulation, financial gain, or simply for entertainment. It often seeks to exploit emotions,

biases, or controversy to gain attention and traction.

Impact of Fake News: Fake news can have serious consequences, including influencing public opinion, swaying elections, causing panic, or harming individuals' reputations. It can erode trust in journalism and democratic processes.

Challenges in Fake News Detection: Detecting fake news is a complex task due to its constantly evolving nature. Some challenges include the speed at which fake news spreads, the use of sophisticated techniques to make it appear legitimate, and the fine line between satire and actual misinformation.

Given data set :

Dataset Link: **https://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-news-dataset**

Fake

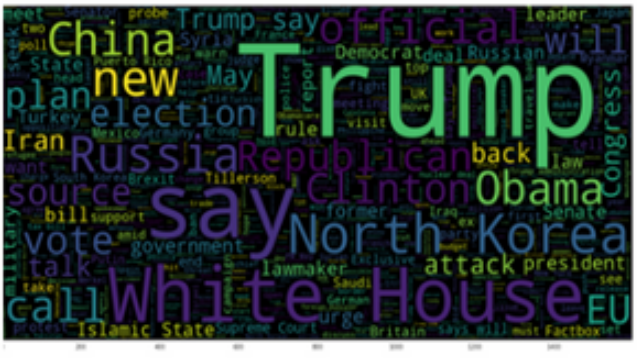
	A	B	C	D
1	title	text	subject	date
2	Donald Trum	Donald Trum	News	31-Dec-17
3	Drunk Bragg	House Intelli	News	31-Dec-17
4	Sheriff Davi	On Friday, it v	News	30-Dec-17
5	Trump Is So	On Christma	News	29-Dec-17
6	Pope Franci	Pope Francis	News	25-Dec-17
7	Racist Alaba	The number	News	25-Dec-17
8	Fresh Off Th	Donald Trum	News	23-Dec-17
9	Trump Said	In the wake c	News	23-Dec-17
10	Former CIA I	Many people	News	22-Dec-17
11	WATCH: Bra	Just when yo	News	21-Dec-17
12	Papa John's	A centerpiec	News	21-Dec-17
13	WATCH: Pa	Republicans	News	21-Dec-17
14	Bad News F	Republicans	News	21-Dec-17
15	WATCH: Lin	The media h	News	20-Dec-17
16	Heiress To D	Abigail Disne	News	20-Dec-17
17	Tone Deaf T	Donald Trum	News	20-Dec-17
18	The Internet	A new anima	News	19-Dec-17
19	Mueller Spo	Trump supp	News	17-Dec-17
20	SNL Hilariou	Right now, th	News	17-Dec-17
21	Republican	Senate Majo	News	16-Dec-17
22	In A Heartle	It almost see	News	16-Dec-17
23	KY GOP Stat	In this #MET	News	13-Dec-17
24	Meghan Mc	As a Democr	News	12-Dec-17
25	CNN CALLS	Alabama is a	News	12-Dec-17
26	White House	A backlash e	News	12-Dec-17

23503 Rows X 5 columns (False Dat)

21418 Rows X 5 columns (True Data)

size of the words represents their frequency. For plotting word cloud we have used word cloud python library.

Fake news



True news

Text pre-processing

After analyzing the data, we move towards text pre-processing before building machine learning models. The text pre-processing consists of the following steps:

Step 1: Lower casing

Step 2: Stop word removal

Step 3: Special character removal

Train Test Split

In this step, we split the data into train and test set in the ratio of 75:25 i.e., 75% of the data used in training the model and rest 25% used for testing the model. The code for splitting data is shown below.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,test_size=0.25, random_state=0)
```

Necessary steps to follow :

1. Importing Libraries and Datasets
2. Data Preprocessing
3. Preprocessing and analysis of News column
4. Converting text into Vectors
5. Model training, Evaluation, and Prediction

1.Importing Libraries and Datasets

The libraries used are :

- Pandas: For importing the dataset.
- Seaborn/Matplotlib: For data visualization.

```
import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt
```

Let's import the downloaded dataset.

```
data = pd.read_csv('News.csv',index_col=0)

data.head()
```

OUTPUT:

	title	text	subject	date	class
0	Donald Trump Sends Out Embarrassing New Year'...	Donald Trump just couldn t wish all Americans ...	News	December 31, 2017	0
1	Drunk Bragging Trump Staffer Started Russian ...	House Intelligence Committee Chairman Devin Nu...	News	December 31, 2017	0
2	Sheriff David Clarke Becomes An Internet Joke...	On Friday, it was revealed that former Milwauk...	News	December 30, 2017	0
3	Trump Is So Obsessed He Even Has Obama's Name...	On Christmas day, Donald Trump announced that ...	News	December 29, 2017	0
4	Pope Francis Just Called Out Donald Trump Dur...	Pope Francis used his annual Christmas Day mes...	News	December 25, 2017	0

2.Data preprocessing

The shape of the dataset can be found by the below code

```
data.shape
```

OUTPUT:

(44919, 5)

As the title, subject and date column will not going to be helpful in identification of the news. So, we can drop these column.

```
data = data.drop(["title", "subject","date"], axis = 1)
```

Now, we have to check if there is any null value (we will drop those rows)

```
data.isnull().sum()
```

Output:

text 0

class 0

So there is no null value.

Now we have to shuffle the dataset to prevent the model to get bias. After that we will reset the index and then drop it. Because index column is not useful to us.

```
# Shuffling

data = data.sample(frac=1)

data.reset_index(inplace=True)

data.drop(["index"], axis=1, inplace=True)
```

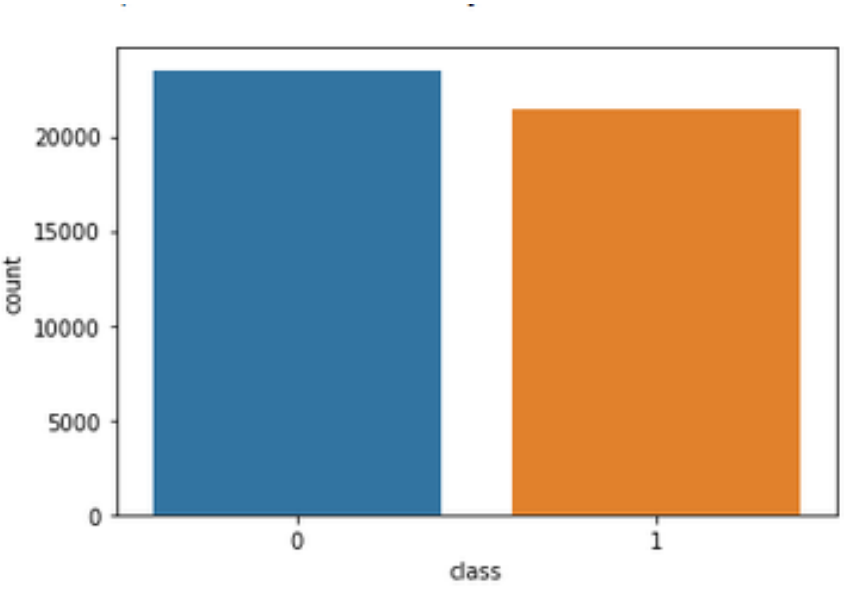
Now Let's explore the unique values in the each category using below code.

```
sns.countplot(data=data,

        x='class',

        order=data['class'].value_counts().index)
```

Output:



3.Preprocessing and analysis of News column:

Firstly we will remove all the stopwords, punctuations and any irrelevant spaces from the text. For that NLTK Library is required and some of it's module need to be downloaded.

```
from tqdm import tqdm

import re

import nltk

nltk.download('punkt')

nltk.download('stopwords')

from nltk.corpus import stopwords

from nltk.tokenize import word_tokenize

from nltk.stem.porter import PorterStemmer

from wordcloud import WordCloud
```

Once we have all the required modules, we can create a function name preprocess text. This function will preprocess all the data given as input.

```
def preprocess_text(text_data):
```

```
    preprocessed_text = []

        for sentence in tqdm(text_data):

            sentence = re.sub(r'[^\w\s]', '', sentence)

            preprocessed_text.append(' '.join(token.lower()

                for token in str(sentence).split()

                if token not in stopwords.words('english')))

return preprocessed_text
```

To implement the function in all the news in the text column, run the below command.

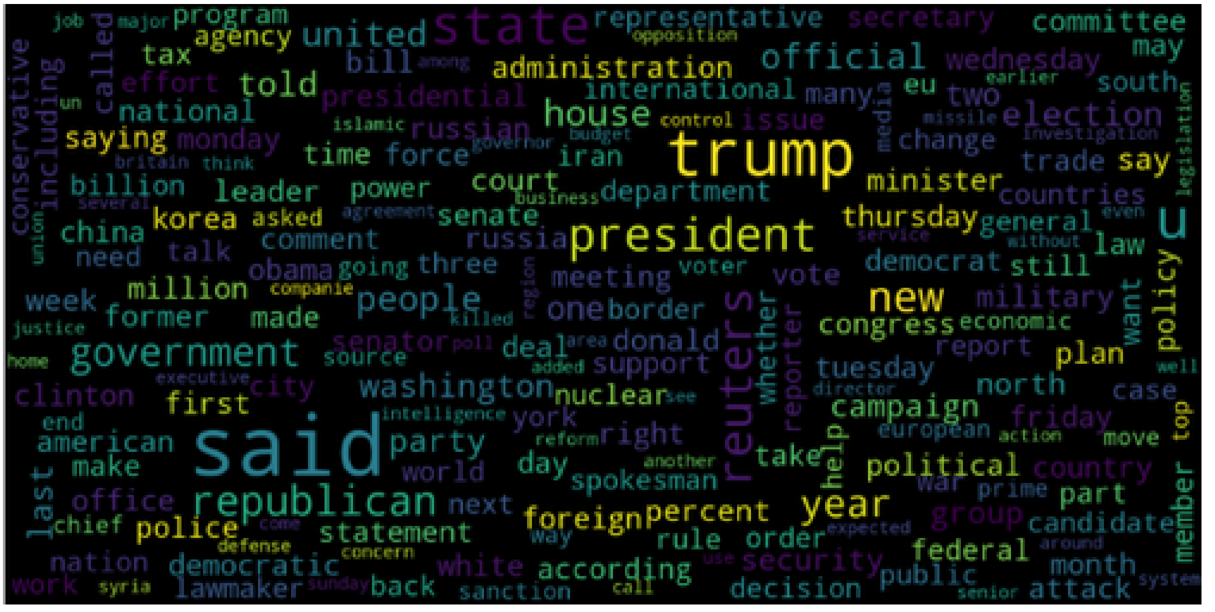
```
preprocessed_review = preprocess_text(data['text'].values)

data['text'] = preprocessed_review
```

Let's visualize the WordCloud for fake and real news separately.

```
# Real
consolidated = ' '.join(
    word for word in data['text'][data['class'] == 1].astype(str))
wordCloud = WordCloud(width=1600,
                      height=800,
                      random_state=21,
                      max_font_size=110,
                      collocations=False)
plt.figure(figsize=(15, 10))
plt.imshow(wordCloud.generate(consolidated), interpolation='bilinear')
plt.axis('off')
plt.show()
```

Output :




```
Fake
consolidated = ' '.join(
    word for word in data['text'][data['class'] == 0].astype(str))
wordCloud = WordCloud(width=1600,
                      height=800,
                      random_state=21,
                      max_font_size=110,
                      collocations=False)
plt.figure(figsize=(15, 10))
plt.imshow(wordCloud.generate(consolidated), interpolation='bilinear')
plt.axis('off')
plt.show()
```

<u>Output :</u>

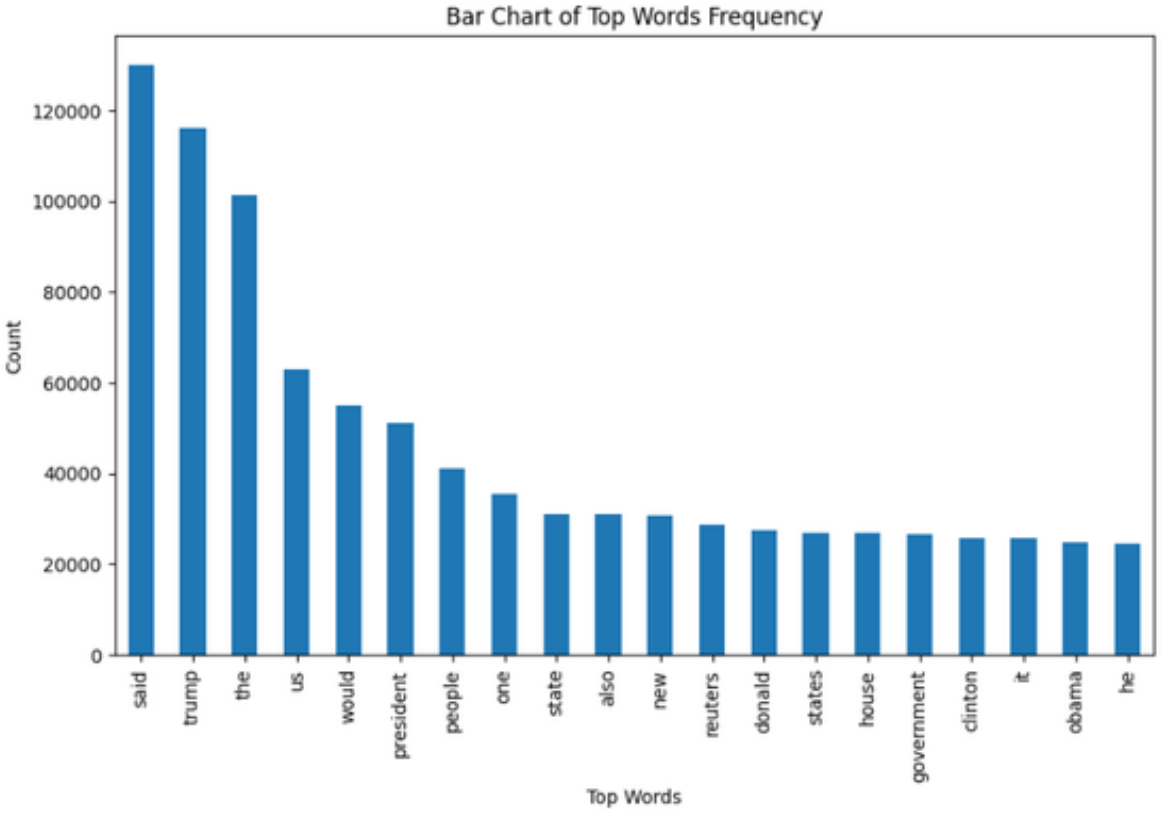
Now, Let’s plot the bargraph of the top 20 most frequent words.

```
from sklearn.feature_extraction.text import CountVectorizer
def get_top_n_words(corpus, n=None):
    vec = CountVectorizer().fit(corpus)
    bag_of_words = vec.transform(corpus)
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx])
                  for word, idx in vec.vocabulary_.items()]
    words_freq = sorted(words_freq, key=lambda x: x[1],
                        reverse=True)
    return words_freq[:n]



common_words = get_top_n_words(data['text'], 20)
df1 = pd.DataFrame(common_words, columns=['Review', 'count'])
  df1.groupby('Review').sum()['count'].sort_values(ascending=False).plot(
    kind='bar',
    figsize=(10, 6),
    xlabel="Top Words",
    ylabel="Count",
    title="Bar Chart of Top Words Frequency"
)
```

Output :



d.Converting text into Vectors:

Before converting the data into vectors, split it into train and test.

```
from sklearn.model_selection import train_test_split

from sklearn.metrics import accuracy_score

from sklearn.linear_model import LogisticRegression


x_train, x_test, y_train, y_test = train_test_split(data['text'],
                                                    data['class'],
                                                    test_size=0.25)
```

Now we can convert the training data into vectors using TfidfVectorizer.

```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorization = TfidfVectorizer()

x_train = vectorization.fit_transform(x_train)

x_test = vectorization.transform(x_test)
```

E.Model training, Evaluation, and Prediction:

- Now, the dataset is ready to train the model.
- For training we will use Logistic Regression and evaluate the prediction accuracy using accuracy_score.

```
from sklearn.linear_model import LogisticRegression



model = LogisticRegression()

model.fit(x_train, y_train)



# testing the model

print(accuracy_score(y_train, model.predict(x_train)))

print(accuracy_score(y_test, model.predict(x_test)))
```

Output :

0.993766511324171

0.9893143365983972

Let's train with Decision Tree Classifier.

```
from sklearn.tree import DecisionTreeClassifier


model = DecisionTreeClassifier()
model.fit(x_train, y_train)


# testing the model
print(accuracy_score(y_train, model.predict(x_train)))
print(accuracy_score(y_test, model.predict(x_test)))
```

Output :

```
0.9999703167205913
0.0.9999703167205913
0.9951914514692787
```

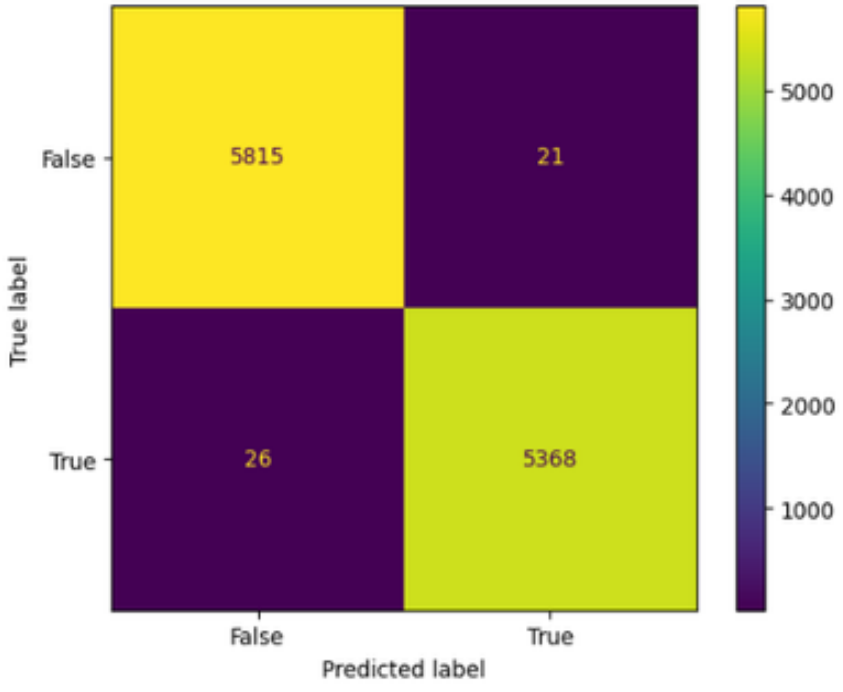
The confusion matrix for Decision Tree Classifier can be implemented with the code below.

```
# Confusion matrix of Results from Decision Tree classification
from sklearn import metrics
cm = metrics.confusion_matrix(y_test, model.predict(x_test))


cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix=cm,
                                            display_labels=[False, True])


cm_display.plot()
plt.show()
```

Output :



Conclusion:

Decision Tree Classifier and Logistic regression are performing well.

Project Conclusion:

In conclusion, fake news detection using Natural Language Processing (NLP) is a vital and evolving field in the fight against misinformation. NLP techniques have shown promise in identifying and flagging potentially deceptive content by analyzing linguistic patterns, sources, and context. However, it is essential to acknowledge that no single method is foolproof, and ongoing research and development are necessary to stay ahead of increasingly sophisticated fake news tactics. Collaborative efforts between researchers, technology companies, and fact-checkersare crucial in building more robust and accurate fake news detection systems to promote trustworthy information in the digital age.