# FAKE NEWS DETECTION USING NLP

**TEAM MEMBER**

**VAISHNAVI S**

**Phase 4 Submission Document**

**Project:** Fake news detection



**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.

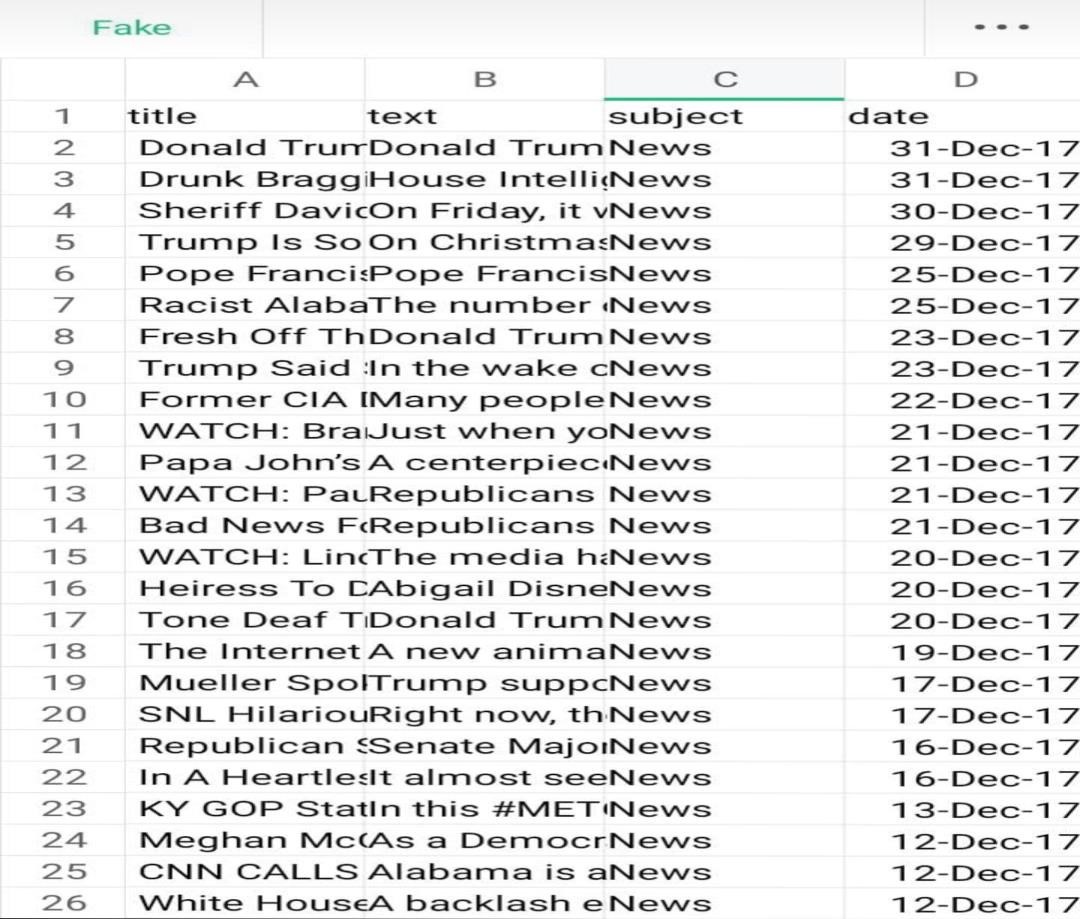
* **Impact of Fake News:** Fake news can have serious consequences, including influencing public opinion, swaying elections, causing panic, or harming individuals’ reputations. It can erode trust in journalism and democratic processes.

* **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.

**Data Source:**

A good data source for Fake news detection using NLP should be Accurate, Complete, Covering the geographic area of interest, Accessible.

Dataset Link:([https://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-newsdataset)](https://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-news-dataset)



**DATA COLLECTION AND PREPARATION :**

* Gather a diverse dataset of news articles or social media posts, including both real and fake examples. These articles should cover a wide range of topics and sources.

* Prepare the text data by removing stop words, punctuation, and converting text to lowercase. Tokenization and stemming or lemmatization may also be applied to standardize the text.

**FEATURES EXTRACTION AND LABELLING:**

* Convert the textual content into numerical features that machine learning algorithms can understand. Common techniques include TF-IDF (Term Frequency-Inverse Document Frequency) and word embeddings like Word2Vec or GloVe.

* Annotate your dataset to indicate which articles are real and which are fake. This labeled data will be used for training and testing your model.

**MODEL SELECTION AND TRAINING:**

* Choose an appropriate NLP model or algorithm. Common choices include logistic regression, random forests, or more advanced methods like recurrent neural networks (RNNs) or transformer-based models like BERT.

* Use the labelled dataset to train your NLP model. The model learns to recognize patterns and features that distinguish real news from fake news.

**TRAINING AND EVALUATION:**

* Use the labelled dataset to train your NLP model. The model learns to recognize patterns and features that distinguish real news from fake news.

* Assess the performance of your model using metrics such as accuracy, precision, recall, and F1-score on a separate validation or test dataset. Fine-tune your model to improve its performance.

**FEATURE ENGINEERING AND BIAS DETECTION:**

* Experiment with different features or techniques, such as n-grams, to enhance your model’s ability to detect fake news.

* Be aware of potential biases in your dataset and model. Ensure that your model doesn’t unfairly label certain sources or topics as fake news.

**DEPLOYMENT :**

* Once satisfied with the model’s performance, deploy it to analyze and classify news articles or social media content in real-time.

* Continuously monitor your model’s performance and update it as needed to adapt to evolving fake news tactics.

**ETHICAL CONSIDERATION:**

* Be mindful of ethical considerations, such as privacy and freedom of speech, when developing and deploying fake news detection systems.

* Remember that fake detection is a challenging task, and achieving high accuracy can be difficult due to the evolving nature of fake news. It often requires ongoing research and adaptation to stay effective in identifying misinformation and disinformation online.

* It’s important to balance the detection of fake news with respect for free speech and privacy. Striking this balance can be challenging and requires careful consideration.

**CONTINUOUS EVOLUTION:**

* Fake news detection methods must continually adapt to new tactics used by purveyors of misinformation. Ongoing research and collaboration are crucial in this ever-changing landscape.

**TEXT ANALYSIS:**

* NLP techniques are used to analyze the content of news articles or social media posts. This includes sentiment analysis, identifying unusual language patterns, and examining the tone of the text.
* Creating meaningful features from the text data is crucial. Features might include word frequency, readability scores, or linguistic features that can help distinguish fake from real news.

**SUPERVISED LEARNING:**

* Most fake news detection models are trained using supervised learning. They learn from labeled datasets that contain examples of both fake and real news to make predictions on new, unlabeled data.

**ENSEMBLE METHODS:**

* Combining the predictions of multiple machine learning models can enhance accuracy. Techniques like Random Forests or Gradient Boosting are commonly used.

**PROGRAM:**

# FAKE NEWS DETECTION

**IMPORT LIBRARIES:**

**In[1]:**

Import numpy as np

Import pandas as pd

Import matplotlib.pyplot as plt

Import seaborn as sns

Import nltk

Import re

Import string

From sklearn.model\_selection import train\_test\_split

From sklearn.metrics import classification\_report

Import keras

From keras.preprocessing import text,sequence

From keras.models import Sequential

From keras.layers import Dense,Embedding,LSTM,Dropout

Import warnings

Warnings.filterwarnings(‘ignore’)

Import os

For dirname, \_, filenames in os.walk(‘/kaggle/input’):

For filename in filenames:

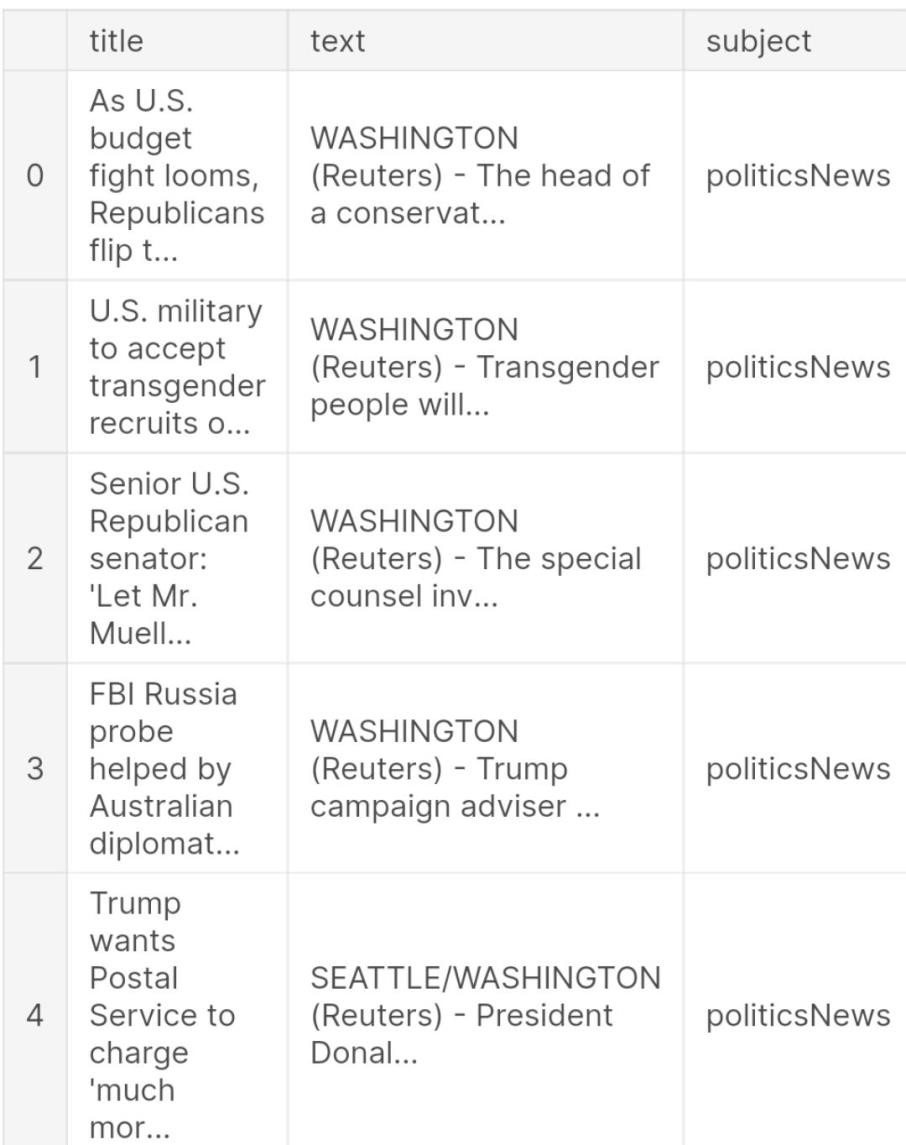
Print(os.path.join(dirname, filename))

**LOAD AND CHECK DATA:**

**In[2]:**

Real\_data = pd.read\_csv(‘/kaggle/input/fake-and-real-news-dataset/True.csv’)

Fake\_data = pd.read\_csv(‘/kaggle/input/fake-and-real-news-dataset/Fake.csv’)



**In[3]:** real\_data.head



**In[5]:** real\_data[‘target’] = 1 fake\_data[‘target’] = 0

**In[6]:** real\_data.tail()

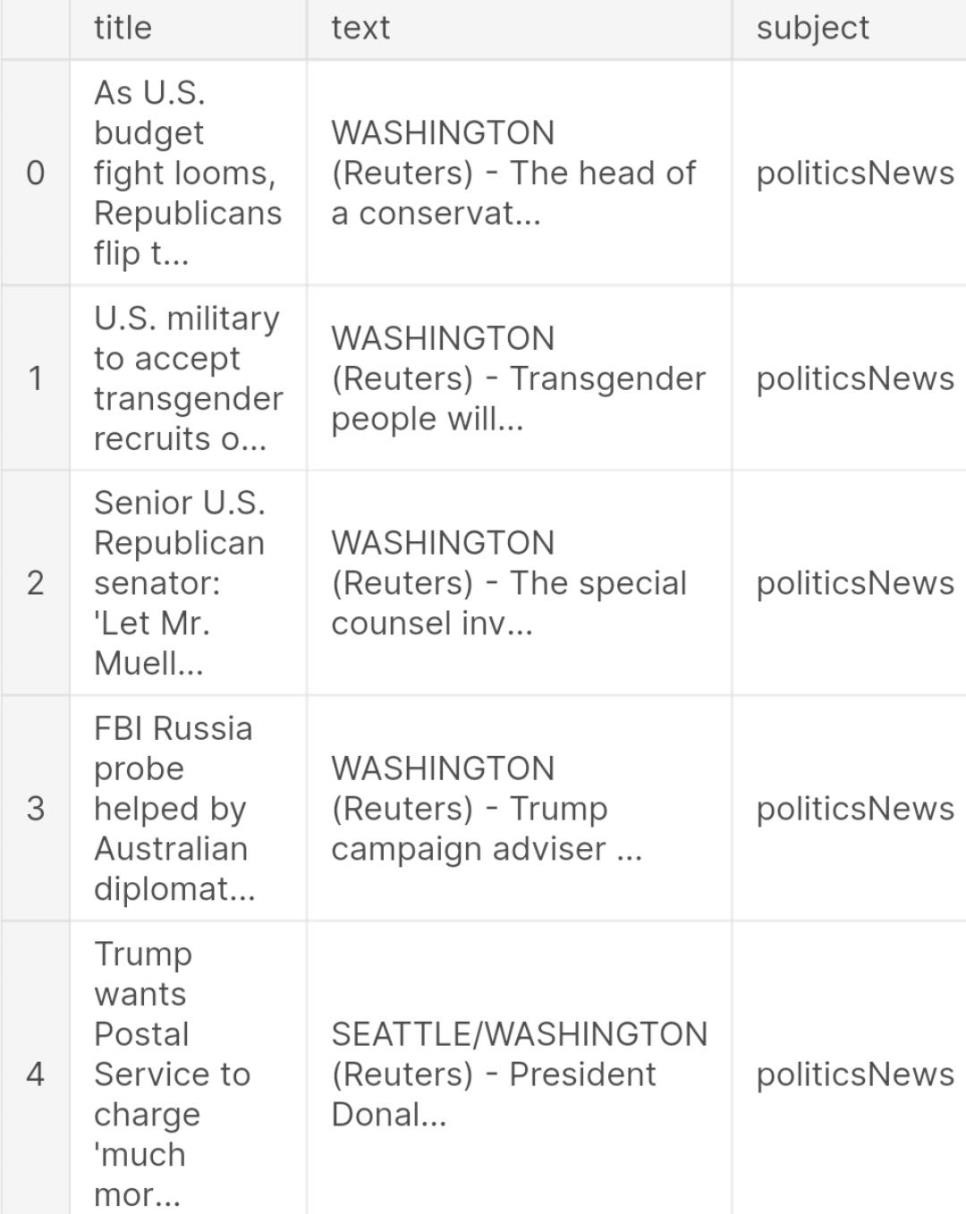
**Out[6]:**



**In[7]:**

Data = pd.concat([real\_data, fake\_data], ignore\_index=True, sort=False) Data.head()

**Out[7]:**



**In[8]:** data.isnull().sum()

**Out[8]:**

Title 0

Text 0

Subject 0

Date 0

Target 0

Dtype: int64

# VISUALIZATION

**Count of Fake and Real Data**

**In[9]:**

print(data[“target”].value\_counts()) fig, ax = plt.subplots(1,2, figsize=(19, 5))

G1 = sns.countplot(data.target,ax=ax[0],palette=”pastel”);

G1.set\_title(“Count of real and fake data”)

G1.set\_ylabel(“Count”)

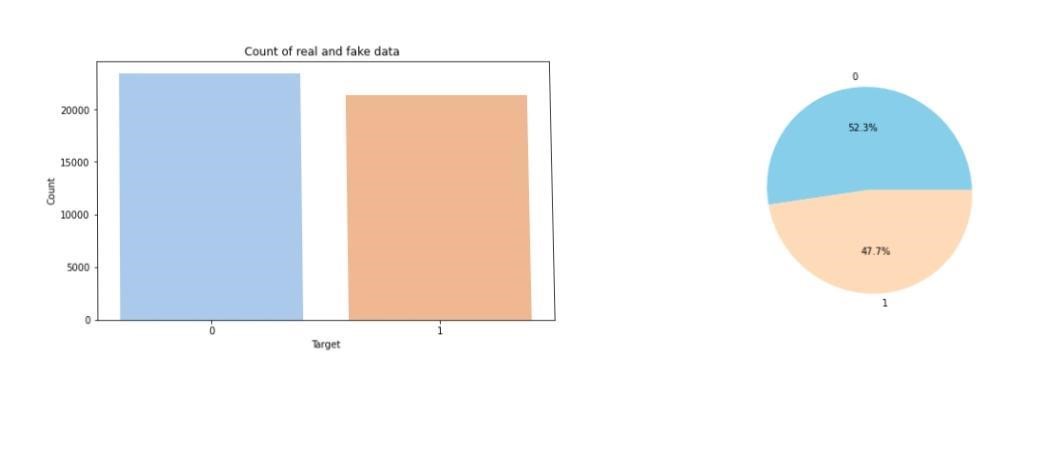
G1.set\_xlabel(“Target”)

G2 =

plt.pie(data[“target”].value\_counts().values,explode=[0,0],labels=data.target.value\_counts().inde x, autopct=’%1.1f%%’,colors=[‘SkyBlue’,’PeachPuff’]) fig.show()

0 1 21417

Name: target, dtype: int64



**Distribution of The Subject According to Real and Fake Data**

**In[9]:**

print(data.subject.value\_counts()) plt.figure(figsize=(10, 5))

ax = sns.countplot(x=”subject”, hue=’target’, data=data, palette=”pastel”) plt.title(“Distribution of The Subject According to Real and Fake Data”)

politicsNews 11272 worldnews 10145 News 9050

Politics 6841 Left-news 4459

Government News 1570

US\_News 783

Middle-east 778

Name: subject, dtype: int64

**Out[10]:**

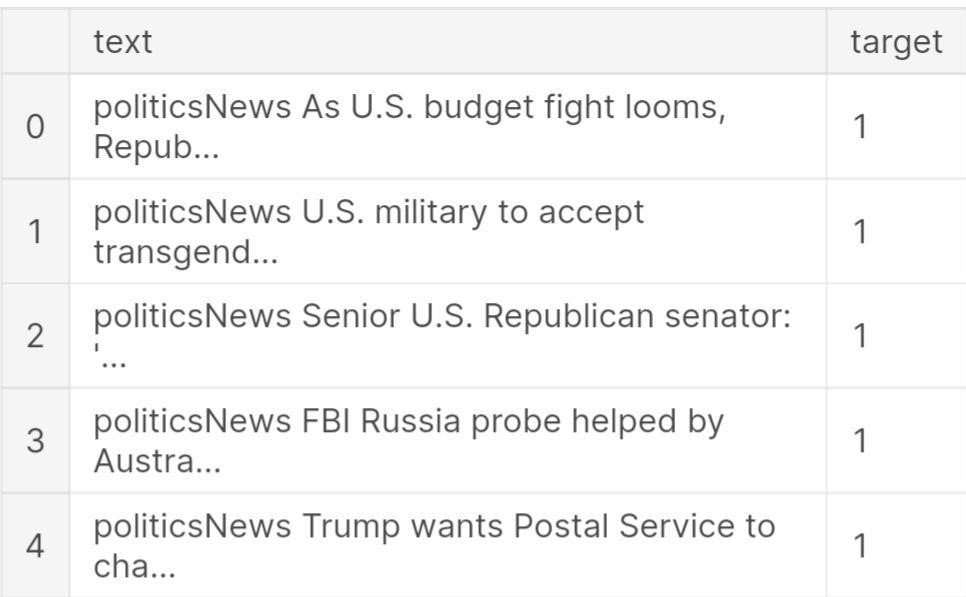
Text(0.5, 1.0, ‘Distribution of The Subject According to Real and Fake Data’)



**DATA CLEANING**

**In[11]:**

data[‘text’]= data[‘subject’] + “ “ + data[‘title’] + “ “ + data[‘text’] del data[‘title’] del data[‘subject’] del data[‘date’] data.head() **Out[11]:**



**Int[12]:** from wordcloud import WordCloud,STOPWORDS

plt.figure(figsize = (15,15))

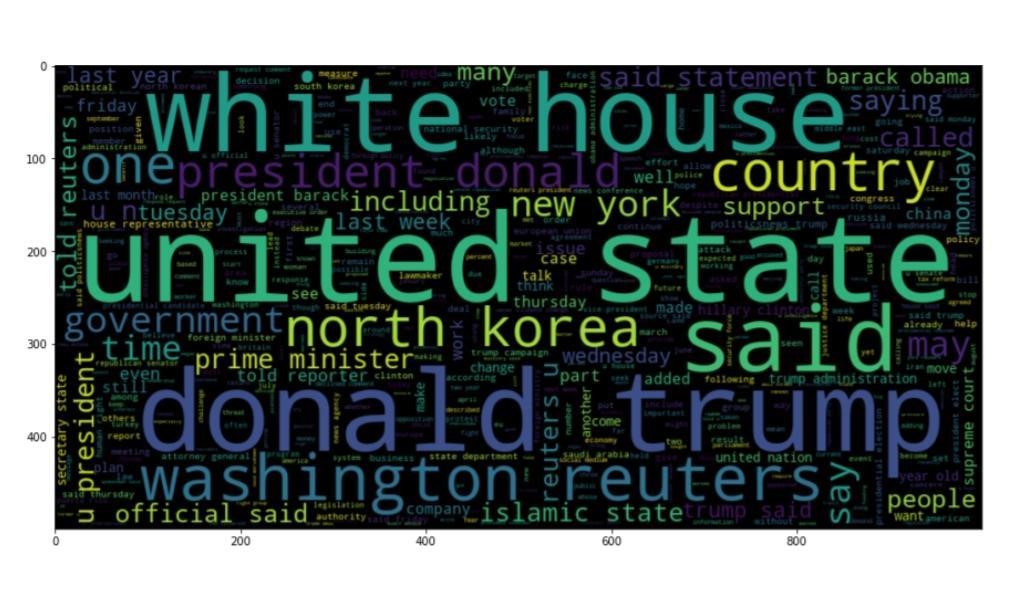
Wc = WordCloud(max\_words = 500 , width = 1000 , height = 500 , stopwords =

STOPWORDS).generate(“ “.join(data[data.target == 1].text))

Plt.imshow(wc , interpolation = ‘bilinear’)

**Out[12]:**

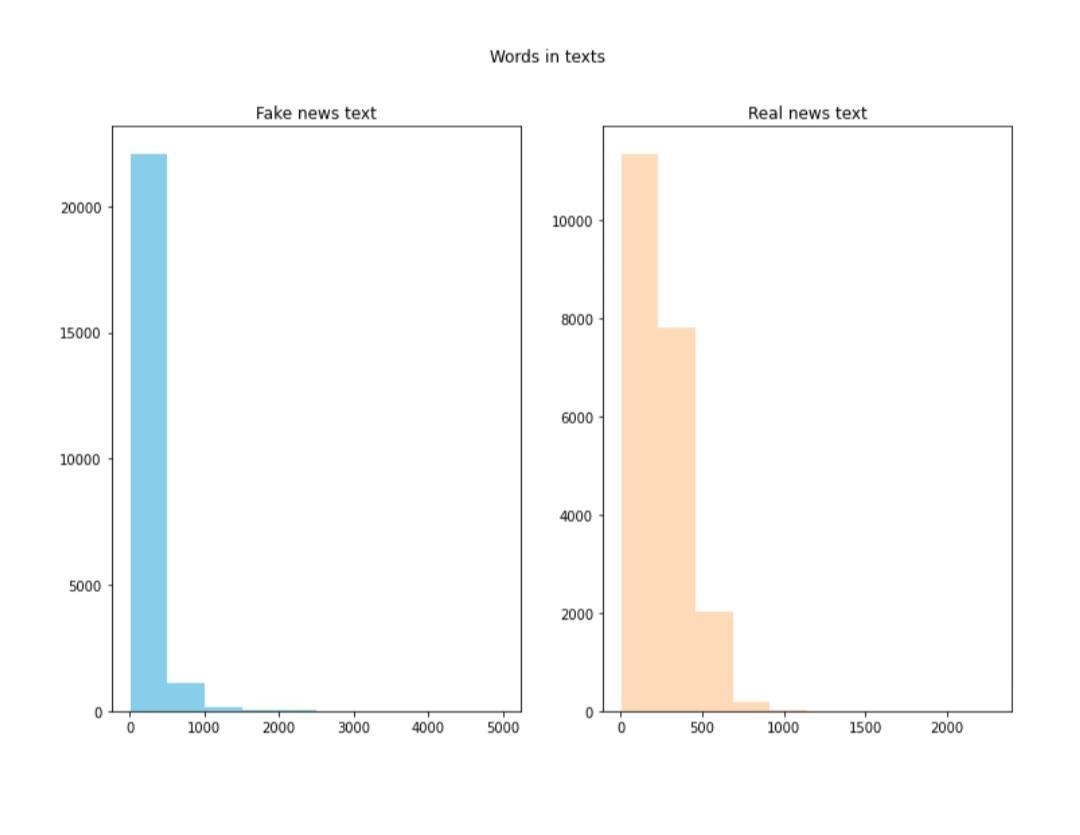
<matplotlib.image.AxesImage at 0x7f6934fd2750>



**Int[13]:**

Number of words in each text

fig,(ax1,ax2)=plt.subplots(1,2,figsize=(12,8)) text\_len=data[data[‘target’]==0][‘text’].str.split().map(lambda x: len(x)) ax1.hist(text\_len,color=’SkyBlue’) ax1.set\_title(‘Fake news text’) text\_len=data[data[‘target’]==1][‘text’].str.split().map(lambda x: len(x)) ax2.hist(text\_len,color=’PeachPuff’) ax2.set\_title(‘Real news text’) fig.suptitle(‘Words in texts’) plt.show()



The number of words seems to be a bit different. 500 words are most common in real news category while around 250 words are most common in fake news category.

**N-Gram Analysis**

**Int[14]:**

Texts = ‘ ‘.join(data[‘text’]

**Int[15]:**

String = texts.split(“ “)

**Int[16]:**

def draw\_n\_gram(string,i):

N\_gram = (pd.Series(nltk.ngrams(string, i)).value\_counts())[:15]

N\_gram\_df=pd.DataFrame(n\_gram)

N\_gram\_df = n\_gram\_df.reset\_index()

N\_gram\_df = n\_gram\_df.rename(columns={“index”: “word”, 0: “count”})

Print(n\_gram\_df.head())

Plt.figure(figsize = (16,9))

Return sns.barplot(x=’count’,y=’word’, data=n\_gram\_df)

**Unigram Analysis Int[17]:**

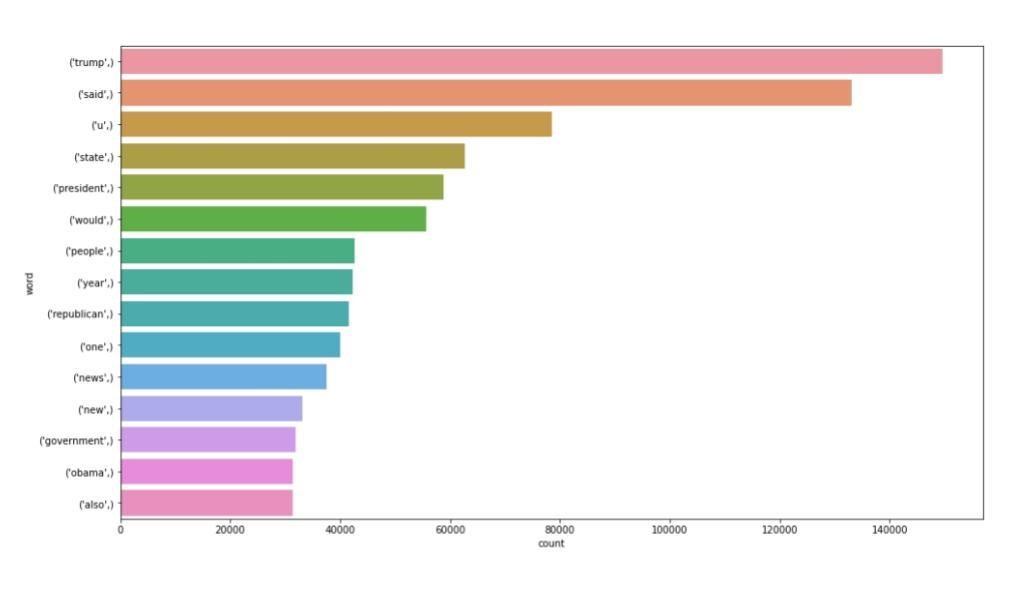
Draw\_n\_gram(string )

word count

1. (trump,) 149603
2. (said,) 133030
3. (u,) 78516
4. (state,) 62726 4 (president,) 58790

**Out[17]:**

<AxesSubplot:xlabel=’count’, ylabel=’word ’>

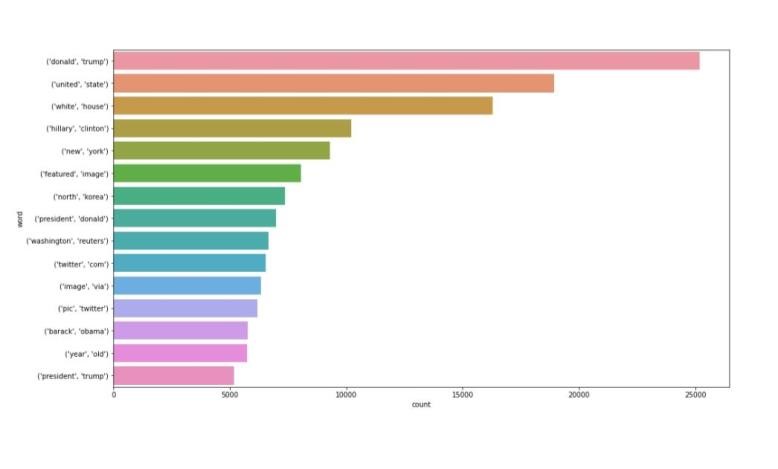


**Bigram Analysis Int[18]:**

Draw\_n\_gram(string,2)

1. (donald, trump) 25203
2. (united, state) 18943
3. (white, house) 16296
4. (hillary, clinton) 10217
5. (new, york) 9305

**Out[18]:**



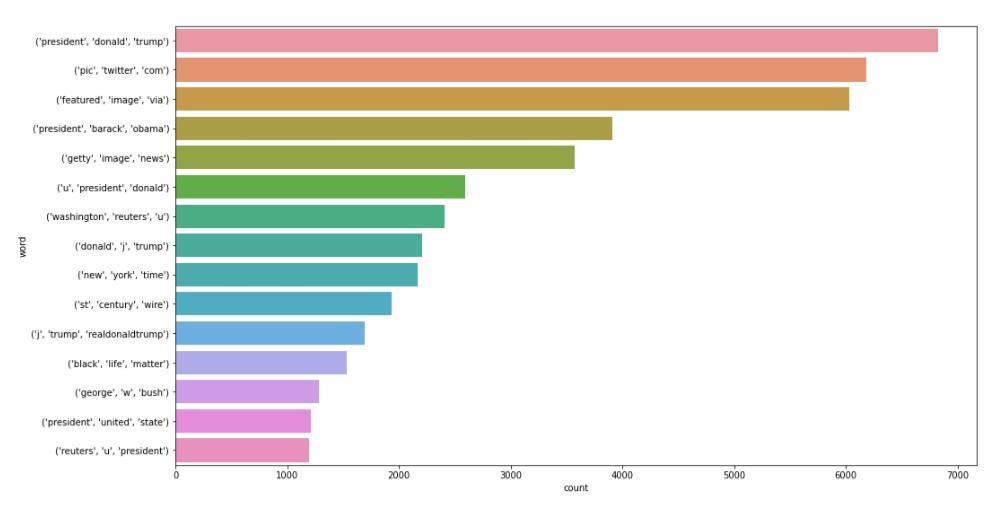
<AxesSubplot:xlabel=’count’, ylabel=’word’>

**Trigram Analysis Int[19]:**

Draw\_n\_gram(string,3)

**Out[19]**

<AxesSubplot:xlabel=’count’, ylabel=’word’>



**Train Test Split**

**Int[20]:**

X\_train, X\_test, y\_train, y\_test = train\_test\_split(data[‘text’], data[‘target’], random\_state=0)

**Tokenizing**

Tokenizing Text -> Repsesenting each word by a number

Mapping of orginal word to number is preserved in word\_index property of tokenizer

**CONCLUSION AND FUTURE WORK(Phase4):**

**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-checkers are crucial in building more robust and accurate fake news detection systems to promote trustworthy information in the digital age.