Date	**18 october 2023**
project_id	**Proj_212171_Team1**
Project name	**Fake news detection using nlp**

Introduction 📖📖

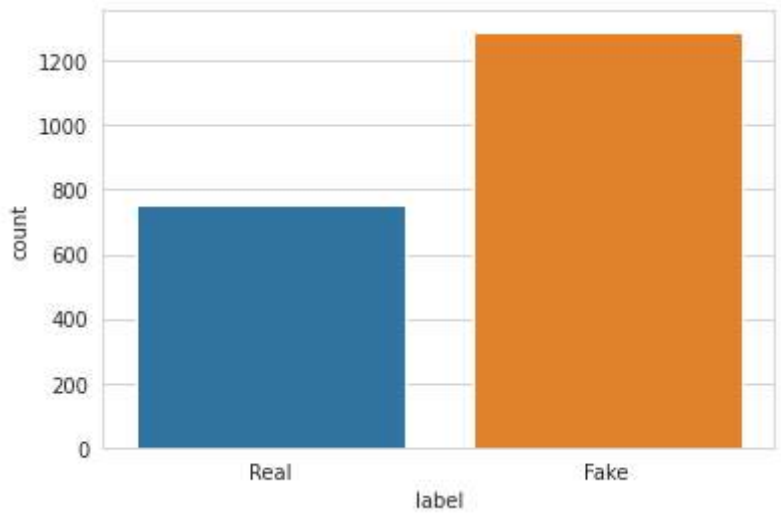
Now a day's Fake news is an increasingly common concern in our society. The term 'Fake news' is refers as disinformation/misleading information/misinformation/hoax/rumor. which are actually different varients of false information. It is spread through traditional media such as News paper or by online medium such as websites and social media. Misleading information is spread to harm the reputation of the person or an organization.

Import Libraries 📚

```
#import libraries

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import cufflinks as cf
import plotly.express as px
from math import pi
from plotly import __version__
%matplotlib inline
from plotly.offline import download_plotlyjs, init_notebook_mode, plot,
iplot
init_notebook_mode(connected=True)

from IPython.core.display import HTML
from nltk.corpus import stopwords
from wordcloud import WordCloud
from sklearn import metrics

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier,AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from pandas import DataFrame
from colorama import Fore, Back, Style
y_ = Fore.YELLOW
r_ = Fore.RED
g_ = Fore.GREEN
b_ = Fore.BLUE
m_ = Fore.MAGENTA

sr_ = Style.RESET_ALL
```

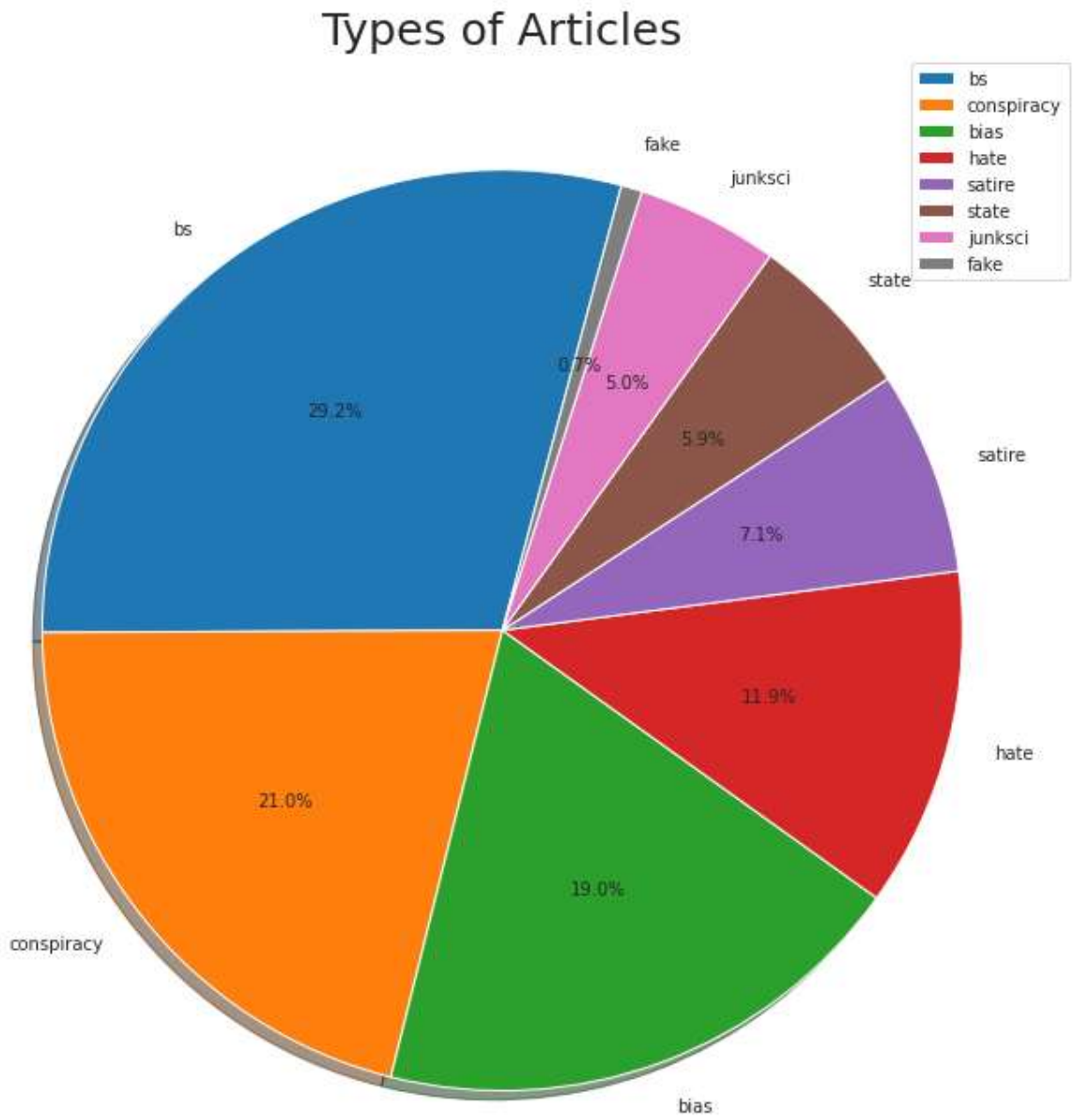
Using Matplotlib python library:

```
import seaborn as sns
sns.set_style("whitegrid")
sns.countplot(df['label']);
/opt/conda/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning:

Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional
argument will be `data`, and passing other arguments without an explicit keyword will result in
an error or misinterpretation.
```



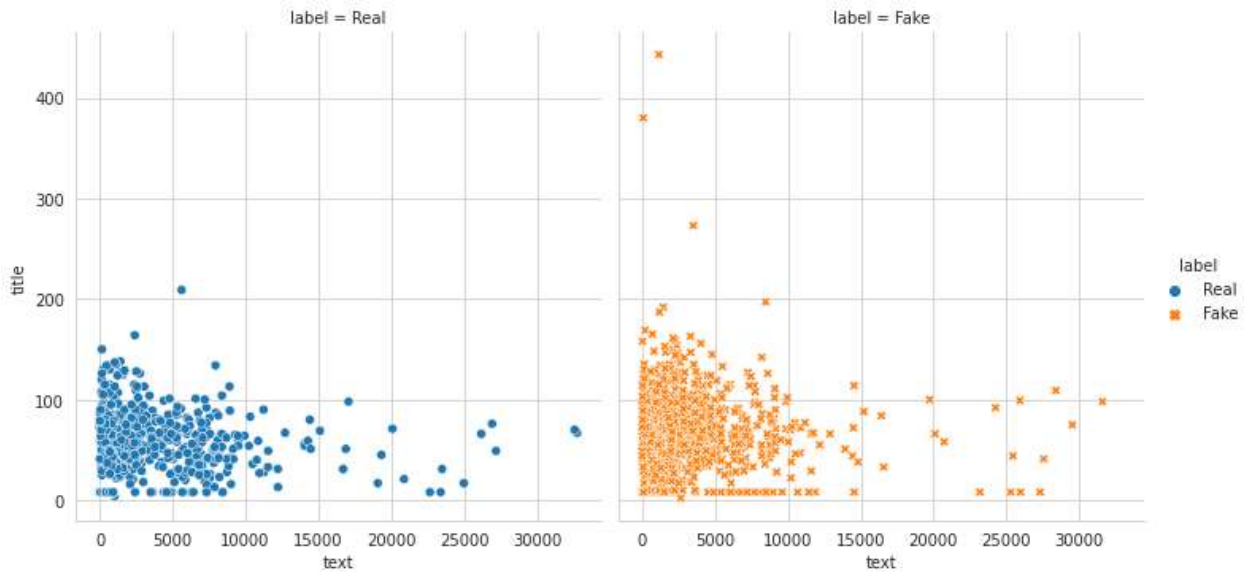
Distribution of Types of Artices:

```
df['type'].value_counts().plot.pie(figsize = (15,12), startangle = 75,autopct =
"%.1f%%",shadow=True)
plt.title('Types of Articles', fontsize = 25)
plt.axis('off')
plt.legend()
plt.show()
```




```
title_len=df['title'].apply(len)
text_len=df['text'].apply(len)

plt.figure(figsize = (15,10))
#sns.scatterplot(data=df, x=text_len,
y=title_len,hue='label',style="label",col='label')
```

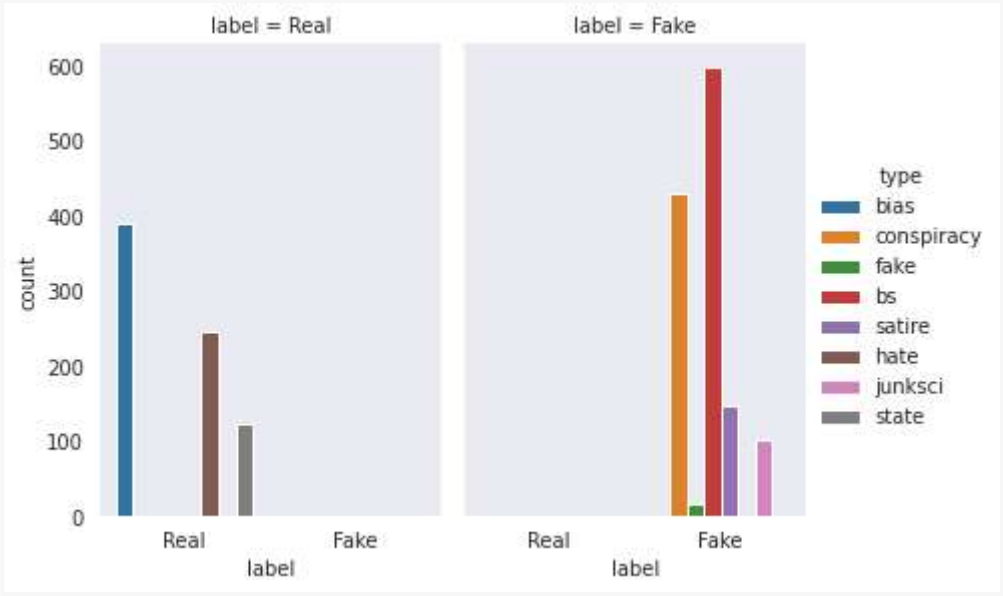
```
sns.relplot(
    data=df, x=text_len, y=title_len,
    col="label", hue="label", style="label",
    kind="scatter"
)
plt.show()
```



Code for

```
sns.stripplot(x="label", y="type", data=df,size=8,palette="Set1")
```


```
plt.figure(figsize = (8,10))
sns.set_style("dark")
#chart = sns.countplot(x = "label", hue = "type" , data = df , palette = 'muted')
#chart.set_xticklabels(chart.get_xticklabels(),rotation=30)

sns.catplot(x="label", hue="type", col="label",
                data=df, kind="count",
                height=4, aspect=.7);
```



Most Frequent word

In [16]:

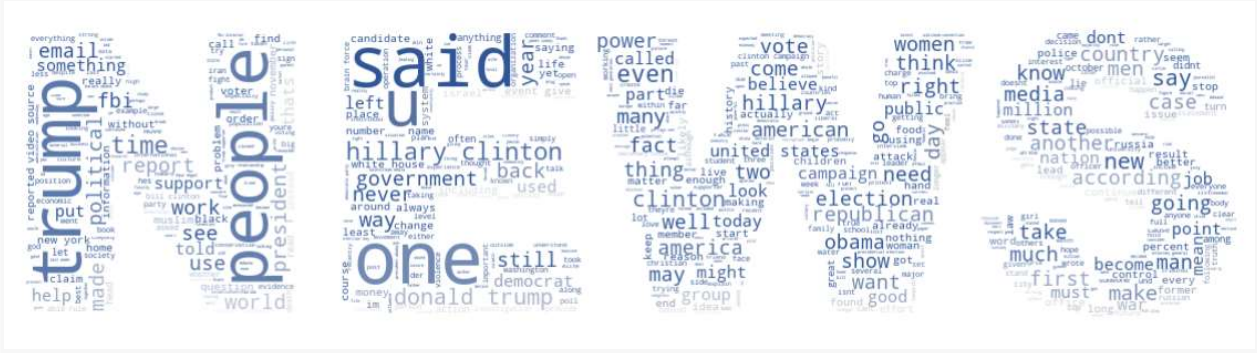
```
def get_top_n_words(corpus, n=None):
    vec = CountVectorizer().fit(corpus)
    bag_of_words = vec.transform(corpus)
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in
vec.vocabulary_.items()]
    words_freq =sorted(words_freq, key = lambda x: x[1],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

common_words = get_top_n_words(df['text_without_stopwords'], 20)
common_words_df = DataFrame (common_words,columns=['word','freq'])
fig = px.bar(common_words_df, x='word', y='freq',color='freq',
             labels={'Top 20 word & their frequency'}, height=400)

fig.show()
```

Word_cloud ☁

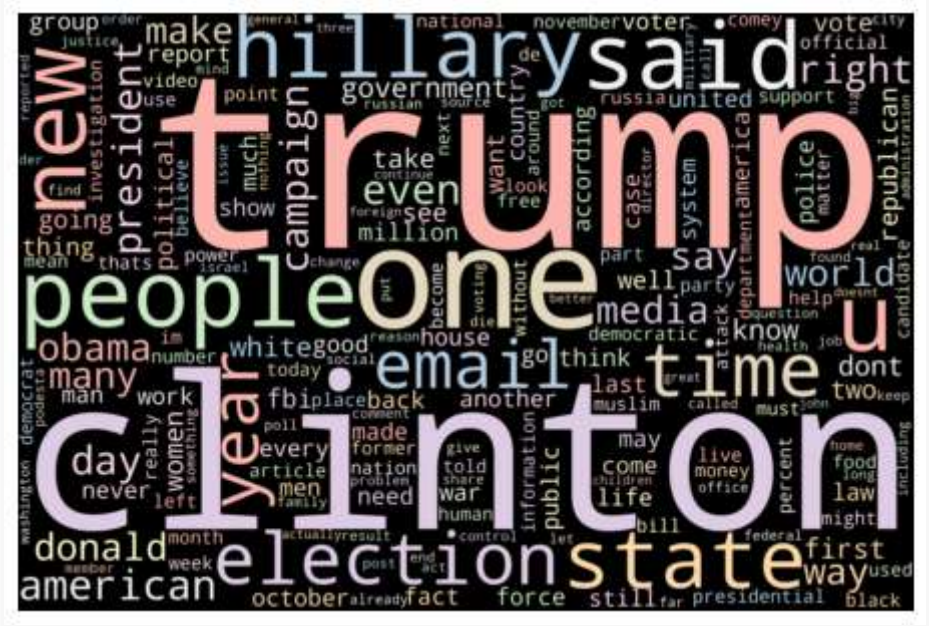
linkcode

```
# Generate a word cloud image
from PIL import Image
```

```
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator



stopwords = set(STOPWORDS)
mask = np.array(Image.open("../input/input-img/News_mask.PNG"))



wordcloud = WordCloud(stopwords=stopwords, background_color="white",
mode="RGBA", max_words=1000, mask=mask).generate('
'.join(df['text_without_stopwords']))

# create coloring from image
image_colors = ImageColorGenerator(mask)
plt.figure(figsize=[20,20])
plt.imshow(wordcloud.recolor(color_func=image_colors),
interpolation="bilinear")
plt.axis("off")

# store to file

plt.savefig("news.png", format="png")
```

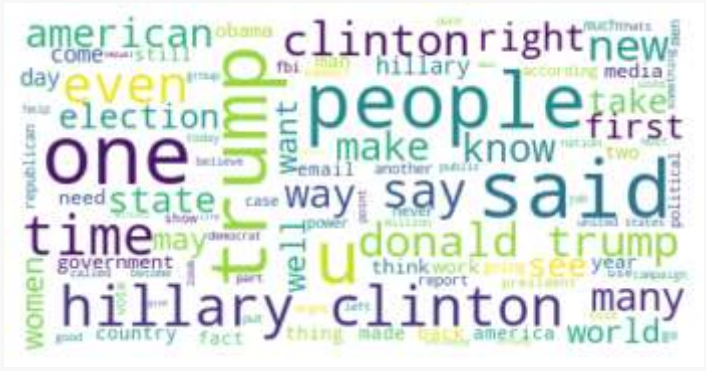


```
# Generate word cloud
wordcloud = WordCloud(width = 3000, height = 2000, random_state=1,
background_color='black', colormap='Pastel1', collocations=False,
stopwords = STOPWORDS).generate(' '.join(df['text_without_stopwords']))

plt.figure(figsize=[8,10])
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
```



 # lower max_font_size, change the maximum number of word and lighten the background:

```
wordcloud = WordCloud(max_font_size=50, max_words=100,
background_color="white").generate(' '.join(df['text_without_stopwords']))
plt.figure()
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()
```



Code for generate word cloud:

```
text = (' '.join(df['text_without_stopwords']))
wordcloud = WordCloud().generate(text)
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.figure()
plt.show()
```

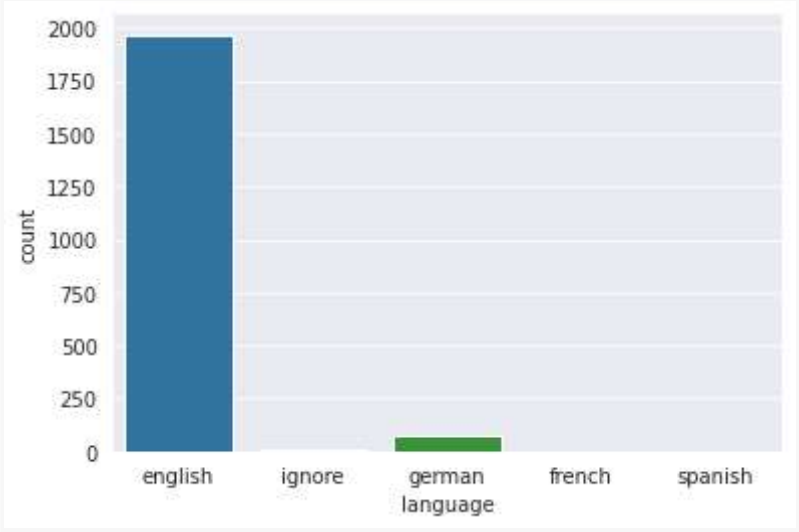
Different Languages of Articles

In [21]:

```
import seaborn as sns
sns.set_style("darkgrid")
sns.countplot(df['language'])
```

```
/opt/conda/lib/python3.7/site-packages/seaborn/_decorators.py:43:
FutureWarning:

Pass the following variable as a keyword arg: x. From version 0.12, the
only valid positional argument will be `data`, and passing other arguments
without an explicit keyword will result in an error or misinterpretation.
```

```
<AxesSubplot:xlabel='language', ylabel='count'>
```



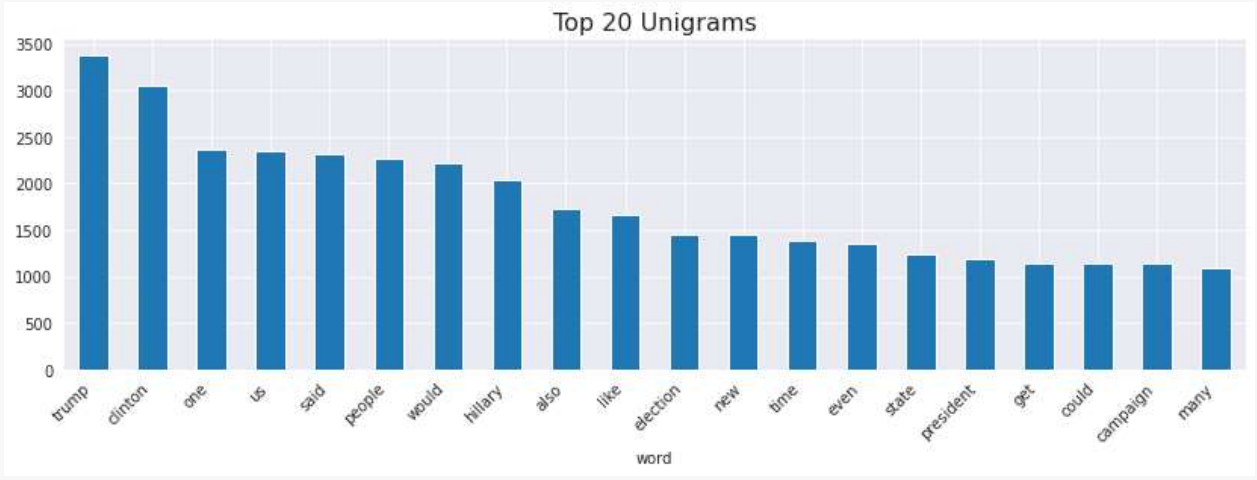
Unigram & Bigram

Linkcode

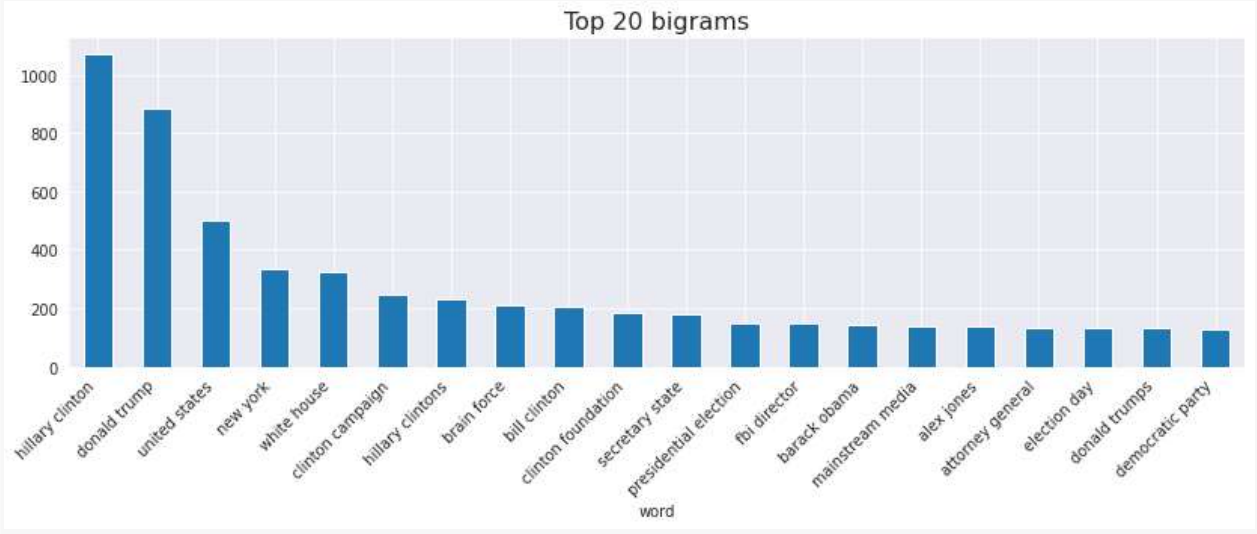
```
def get_top_n_words(corpus, n=None):

    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]


def get_top_n_bigram(corpus, n=None):
    vec = CountVectorizer(ngram_range=(2, 2)).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]


def get_top_n_trigram(corpus, n=None):
    vec = CountVectorizer(ngram_range=(3, 3)).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]

fig = plt.subplots(figsize=(14,4))

common_words = get_top_n_words(df['text_without_stopwords'], 20)

df2 = DataFrame (common_words,columns=['word','count'])

chart =
df2.groupby('word').sum()['count'].sort_values(ascending=False).plot(kind
='bar')

chart.set_xticklabels(chart.get_xticklabels(),rotation=45,
horizontalalignment='right')
chart.set_title("Top 20 Unigrams",size=16)
```

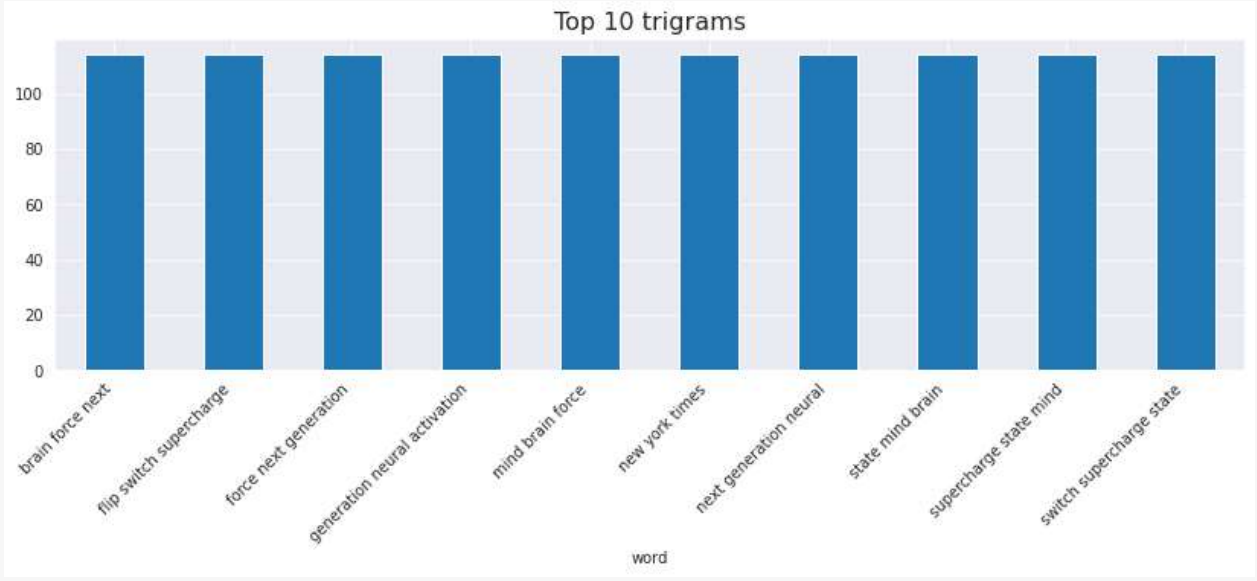



```
fig = plt.subplots(figsize=(14,4))

common_words = get_top_n_bigram(df['text_without_stopwords'], 20)

df3 = pd.DataFrame(common_words, columns = ['word' ,'count'])

chart =
df3.groupby('word').sum()['count'].sort_values(ascending=False).plot(kind=
'bar')

chart.set_xticklabels(chart.get_xticklabels(),rotation=45,
horizontalalignment='right')
chart.set_title("Top 20 bigrams",size=16)
```



Code for top 10 trigrams

```
g = plt.subplots(figsize=(14,4))

common_words = get_top_n_trigram(df['text_without_stopwords'], 10)

df4 = pd.DataFrame(common_words, columns = ['word' ,'count'])

chart =
df4.groupby('word').sum()['count'].sort_values(ascending=False).plot(kind=
'bar')

chart.set_xticklabels(chart.get_xticklabels(),rotation=45,
horizontalalignment='right')
chart.set_title("Top 10 trigrams",size=16)
```


Top 10 trigrams
100
80
60
40
20
0
brain force next
flip switch supercharge
force next generation
generation neural activation
mind brain force
new york times
next generation neural
state mind brain
supercharge state mind
switch supercharge state
word