AI_phase3

```
# This Python 3 environment comes with many helpful analytics
libraries installed
# It is defined by the kaggle/python Docker image:
https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/"
directory
# For example, running this (by clicking run or pressing Shift+Enter)
will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/)
that gets preserved as output when you create a version using "Save &
Run All"
# You can also write temporary files to /kaggle/temp/, but they won't
be saved outside of the current session
```

```
/kaggle/input/masksforwordclouds/twitter_mask2.png
/kaggle/input/masksforwordclouds/twitter_mask4.jpg
/kaggle/input/masksforwordclouds/twitter_mask3.jpg
/kaggle/input/masksforwordclouds/book-logo-1.jpg
/kaggle/input/masksforwordclouds/twitter_mask.png
/kaggle/input/masksforwordclouds/wordcloud-man.png
/kaggle/input/masksforwordclouds/twitter_mask3.png
/kaggle/input/fake-and-real-news-dataset/True.csv
/kaggle/input/fake-and-real-news-dataset/Fake.csv
```

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import nltk
from sklearn.preprocessing import LabelBinarizer
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from wordcloud import WordCloud,STOPWORDS
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize,sent_tokenize

fake=pd.read_csv("../input/fake-and-real-news-dataset/Fake.csv")
true=pd.read_csv("../input/fake-and-real-news-dataset/True.csv")

fake.head()
```

```
                                                   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...

                                                 text subject  \
0  Donald Trump just couldn t wish all Americans ...    News
1  House Intelligence Committee Chairman Devin Nu...    News
2  On Friday, it was revealed that former Milwauk...    News
3  On Christmas day, Donald Trump announced that ...    News
4  Pope Francis used his annual Christmas Day mes...    News

                 date
0  December 31, 2017
1  December 31, 2017
2  December 30, 2017
3  December 29, 2017
4  December 25, 2017
```

```
true.head()
```

```
                                                   title  \
0  As U.S. budget fight looms, Republicans flip t...
1  U.S. military to accept transgender recruits o...
2  Senior U.S. Republican senator: 'Let Mr. Muell...
3  FBI Russia probe helped by Australian diplomat...
4  Trump wants Postal Service to charge 'much mor...

                                                text       subject  \
0  WASHINGTON (Reuters) - The head of a conservat...  politicsNews
1  WASHINGTON (Reuters) - Transgender people will...  politicsNews
2  WASHINGTON (Reuters) - The special counsel inv...  politicsNews
3  WASHINGTON (Reuters) - Trump campaign adviser ...  politicsNews
4  SEATTLE/WASHINGTON (Reuters) - President Donal...  politicsNews

                 date
0  December 31, 2017
1  December 29, 2017
2  December 31, 2017
3  December 30, 2017
4  December 29, 2017
```

```
true['target']=1
fake['target']=0

df=pd.concat([true,fake])

df.shape
```

```
(44898, 5)
```

```
df.head(2)
```

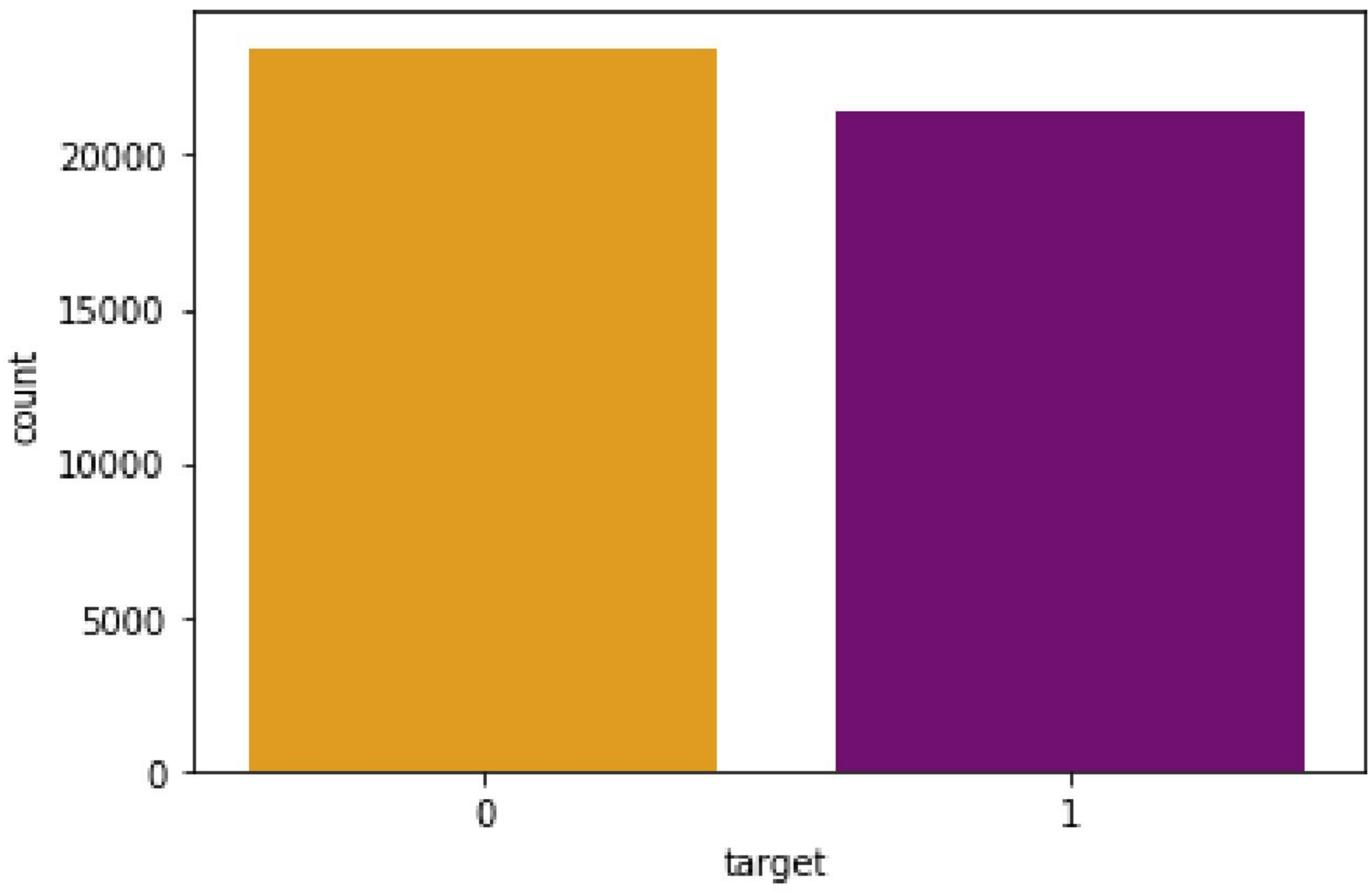
```
                                                title  \
0  As U.S. budget fight looms, Republicans flip t...
1  U.S. military to accept transgender recruits o...

                                                text       subject  \
0  WASHINGTON (Reuters) - The head of a conservat...  politicsNews
1  WASHINGTON (Reuters) - Transgender people will...  politicsNews

                 date  target
0  December 31, 2017        1
1  December 29, 2017        1
```

```
temp = df.groupby('target').count()
['text'].reset_index().sort_values(by='text',ascending=False)
temp.style.background_gradient(cmap='Purples')
```

```
<pandas.io.formats.style.Styler at 0x7ff18dd07410>
```

```
sns.countplot(x='target',data=df,palette=['orange','purple'])
```

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




```
from plotly import graph_objs as go
fig = go.Figure(go.Funnelarea(
    text =temp.target,
```

```
    values = temp.text,
    title = {"position": "top center", "text": "Funnel-Chart of Target
Distribution"}
    ))
fig.show()

df.isnull().sum()
```

```
title      0
text       0
subject    0
date       0
target     0
dtype: int64
```

```
df.nunique()
```

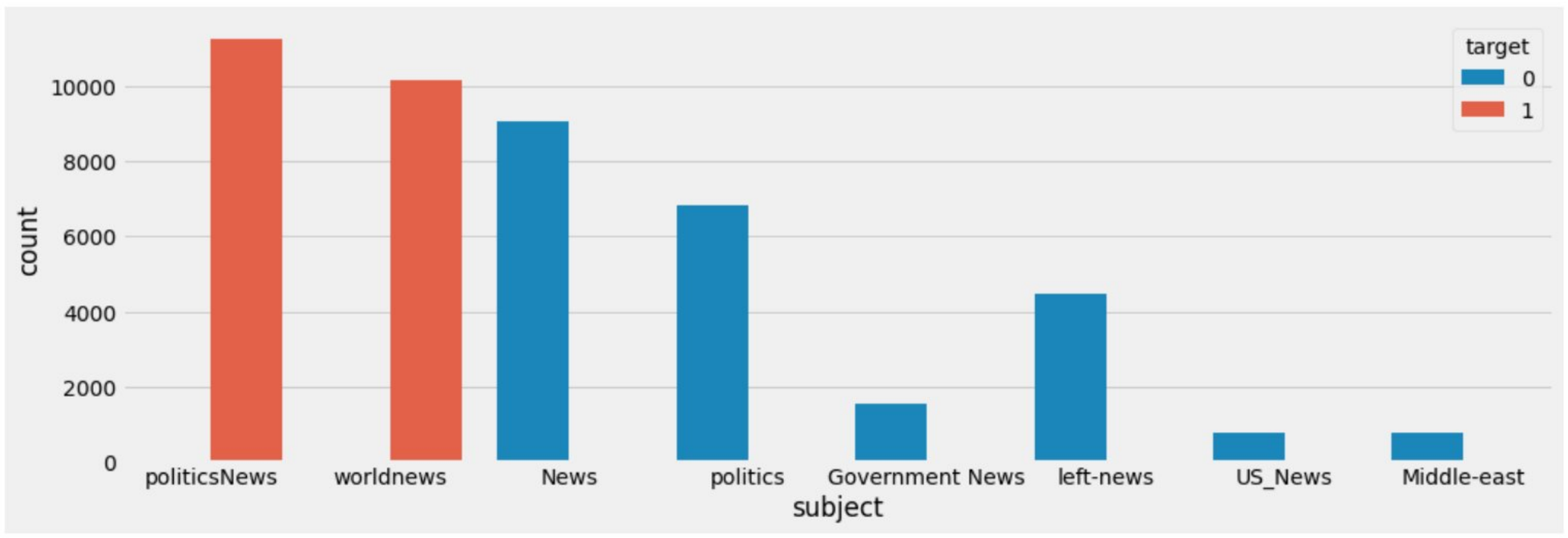
```
title      38729
text       38646
subject        8
date        2397
target         2
dtype: int64
```

```
df.dtypes
```

```
title      object
text       object
subject    object
date       object
target      int64
dtype: object
```

```
plt.style.use('fivethirtyeight')
plt.figure(figsize=(15,5))
sns.countplot(x='subject',data=df,hue='target')
```

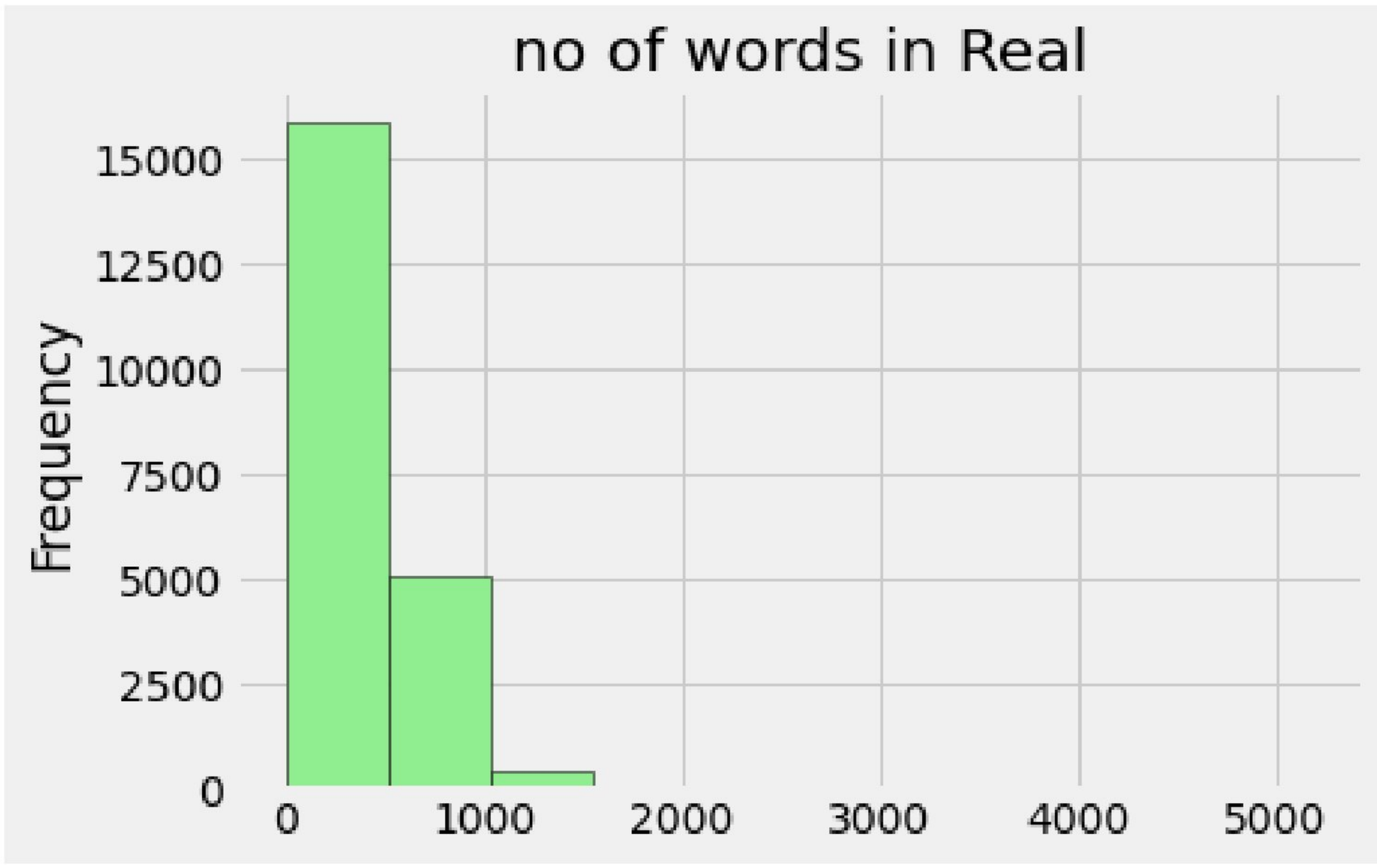
```
<AxesSubplot:xlabel='subject', ylabel='count'>
```

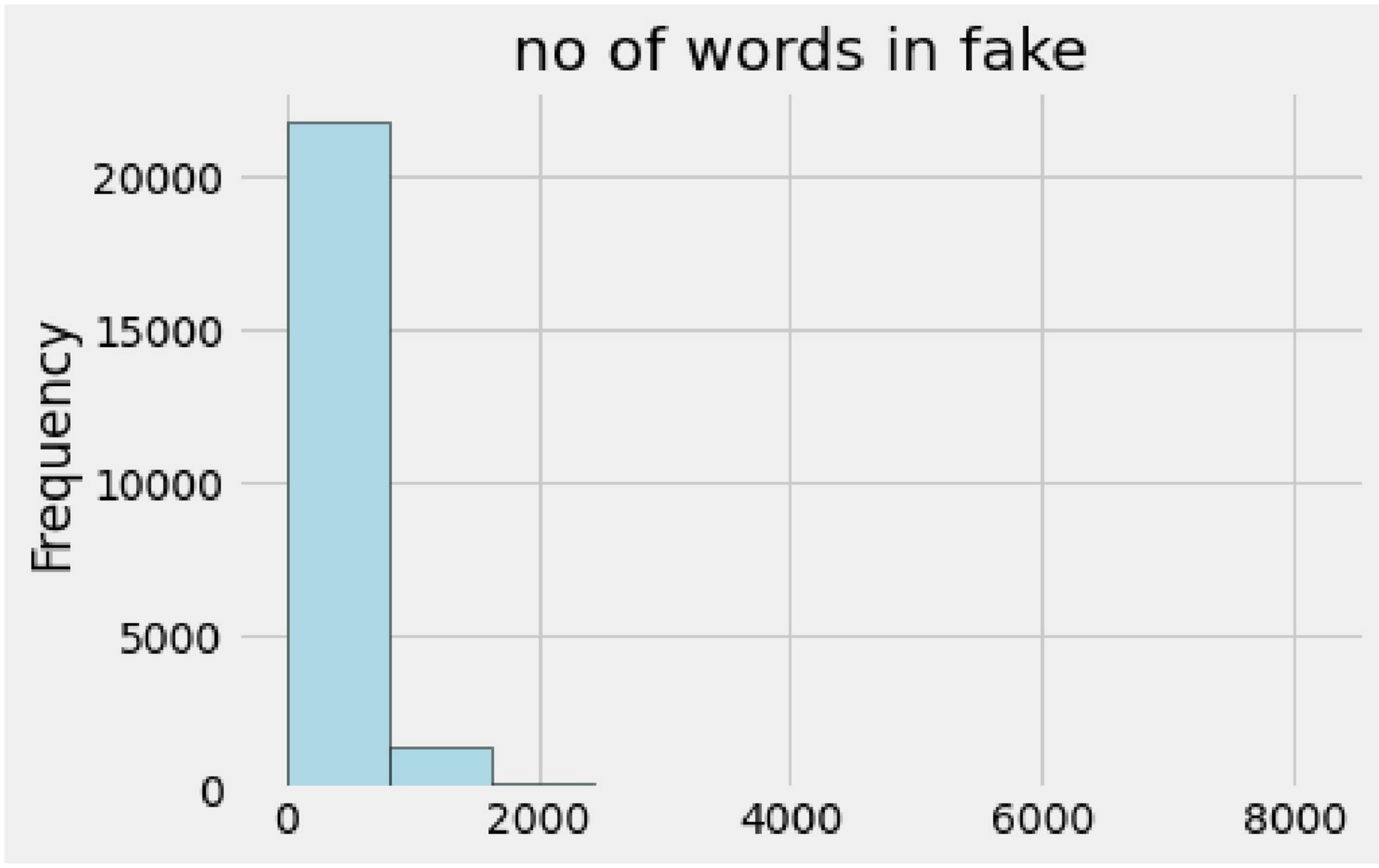


```
df['text']=df['text']+" "+df['title']

df.drop(['title','subject','date'],axis=1,inplace=True)
df.head()
```

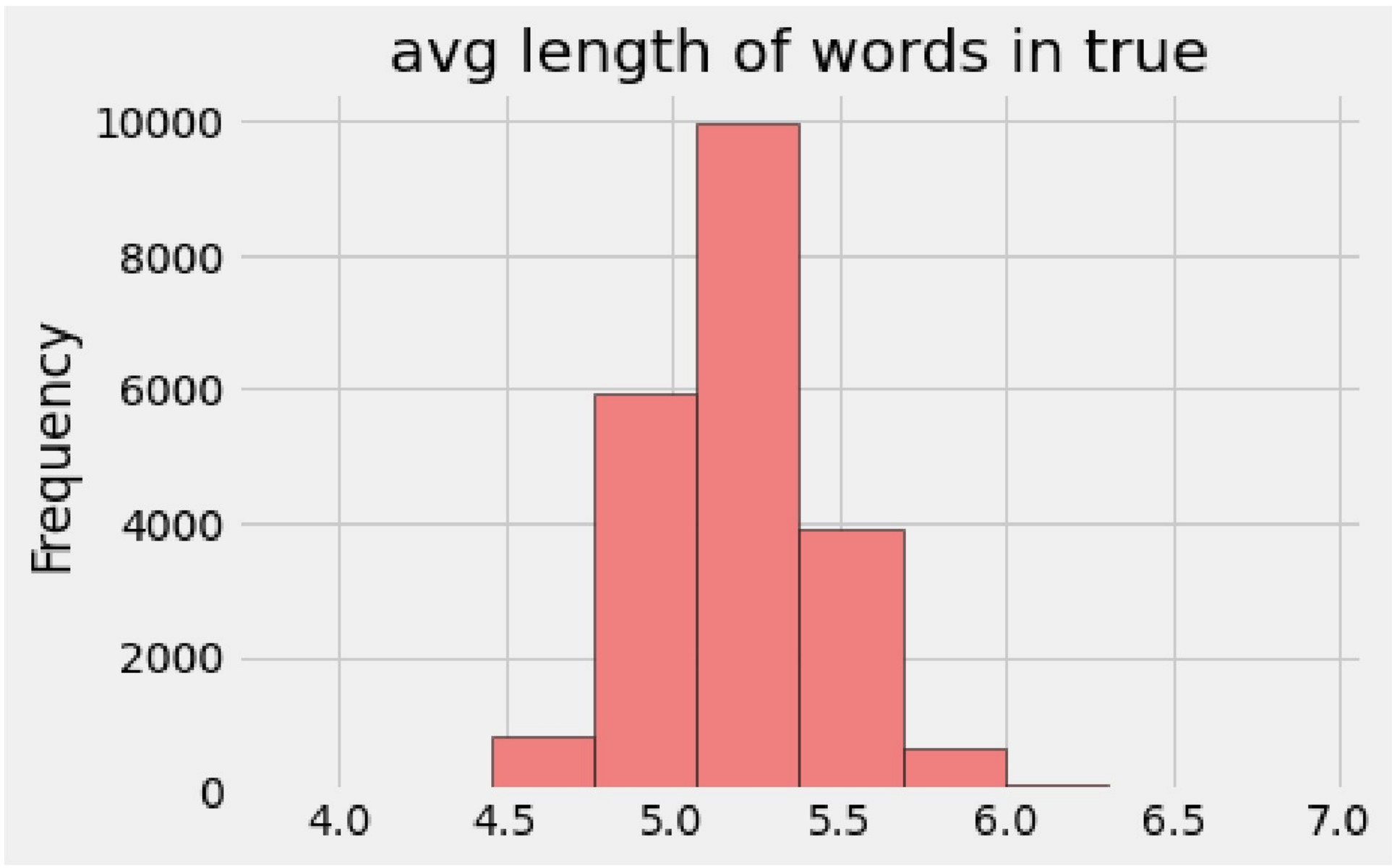
	text	target
0	WASHINGTON (Reuters) - The head of a conservat...	1
1	WASHINGTON (Reuters) - Transgender people will...	1
2	WASHINGTON (Reuters) - The special counsel inv...	1
3	WASHINGTON (Reuters) - Trump campaign adviser ...	1
4	SEATTLE/WASHINGTON (Reuters) - President Donal...	1

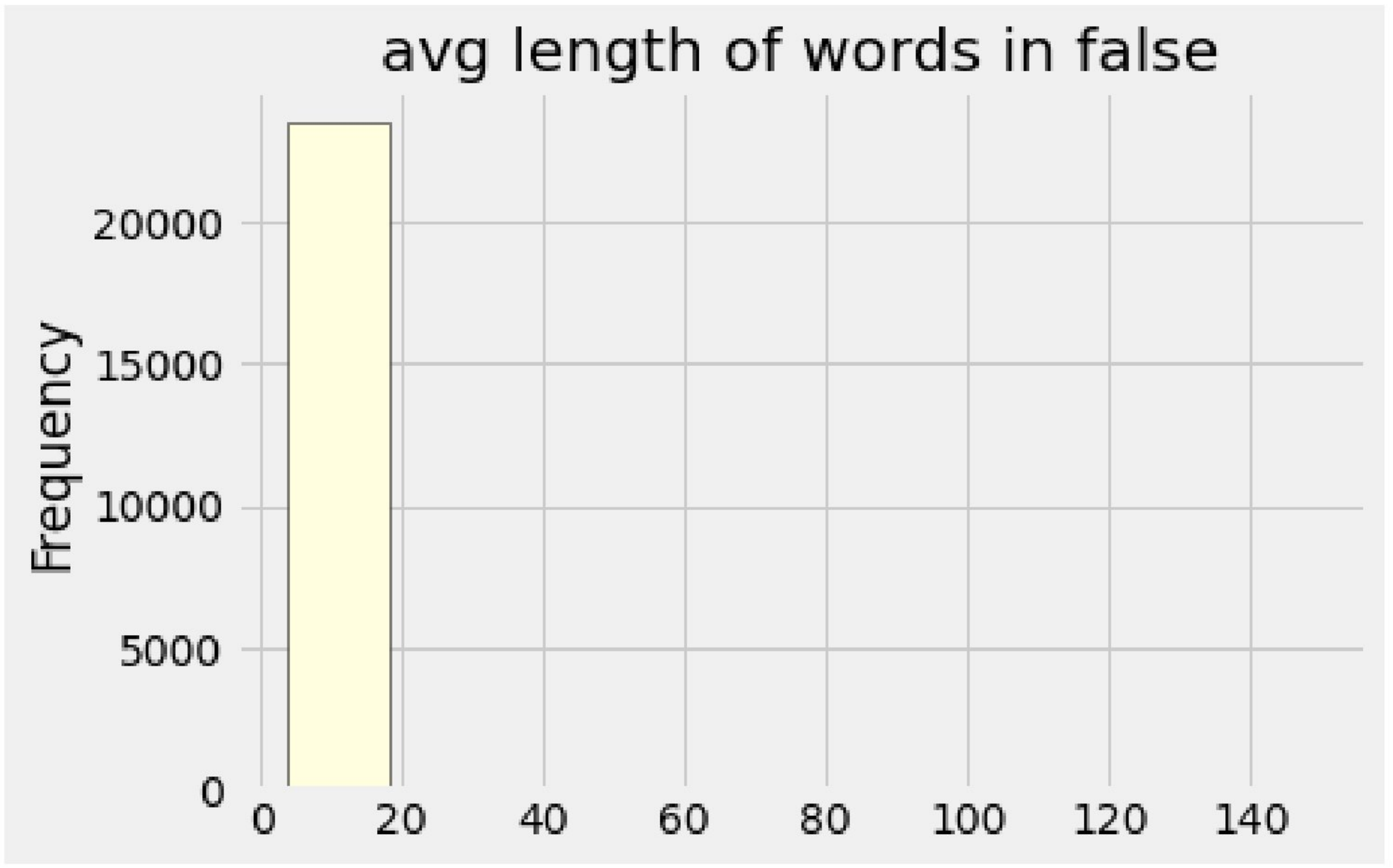
```
no_words=df[df['target']==1].text.str.split().map(lambda x:len(x))
no_words.plot(kind='hist',edgecolor='black',color='lightgreen',title='
no of words in Real')
plt.show()
no_words=df[df['target']==0].text.str.split().map(lambda x:len(x))
no_words.plot(kind='hist',edgecolor='black',color='lightblue',title='n
o of words in fake')
plt.show()
```





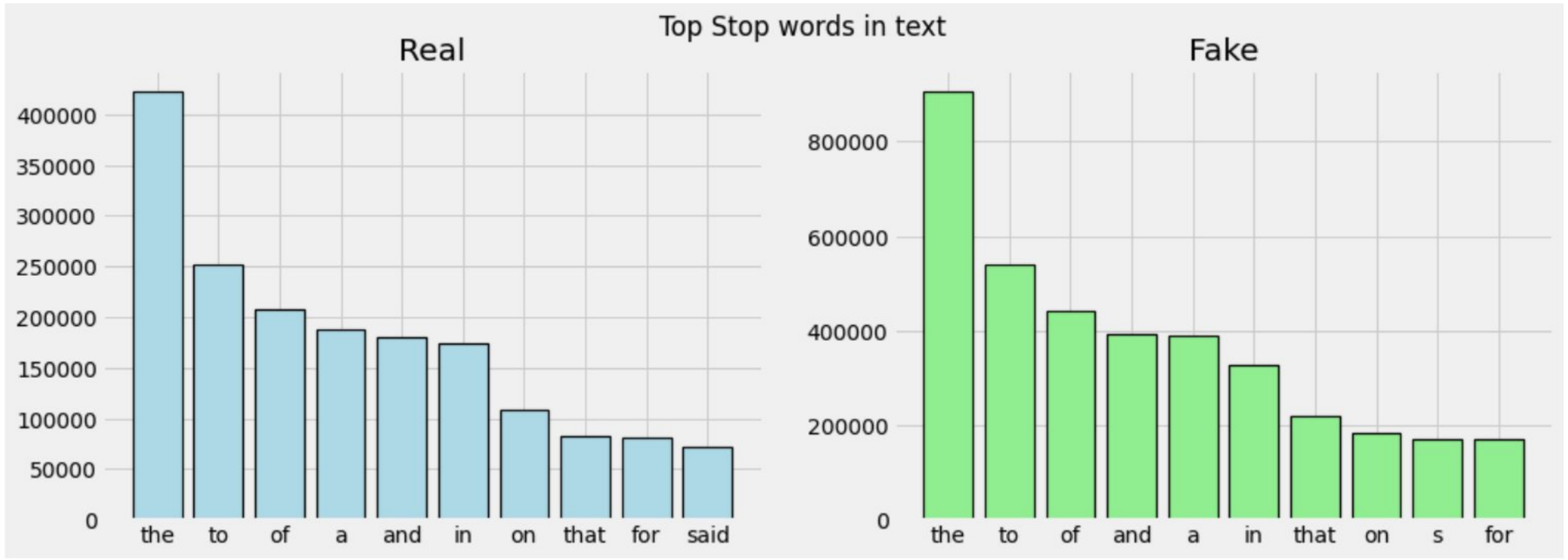

```
avg_len_word=df[df['target']==1].text.str.split().map(lambda
x:np.mean([len(word) for word in x]))
avg_len_word.plot(kind='hist',edgecolor='black',color='lightcoral',tit
le='avg length of words in true')
plt.show()
avg_len_word=df[df['target']==0].text.str.split().map(lambda
x:np.mean([len(word) for word in x]))
avg_len_word.plot(kind='hist',edgecolor='black',color='lightyellow',ti
tle='avg length of words in false')
plt.show()
```





First way

```
# creating sample words
def create_words(target):
    words = []
    for x in df[df['target']==target]['text'].str.split():
        for i in x:
            words.append(i)
    return words

from collections import defaultdict
def analyze_stopwords(data,fun,target):
    values_list=[]
    dic=defaultdict(int)
    for i in range(0,len(target)):
        corpus=fun(target[i])
        for word in corpus:
            dic[word]+=1
        top=sorted(dic.items(),key=lambda x:x[1],reverse=True)[:10]
        x_items,y_items=zip(*top)
        values_list.append(x_items)
        values_list.append(y_items)
    fig,(ax1,ax2) = plt.subplots(1,2,figsize=(15,5))

ax1.bar(values_list[0],values_list[1],color="lightblue",edgecolor='bla
ck', linewidth=1.2)
    ax1.set_title("Real")


ax2.bar(values_list[2],values_list[3],color="lightgreen",edgecolor='bl
ack', linewidth=1.2)
    ax2.set_title("Fake")

    plt.suptitle("Top Stop words in text")
    plt.show()

analyze_stopwords(df,create_words,[1,0])
```



Second way

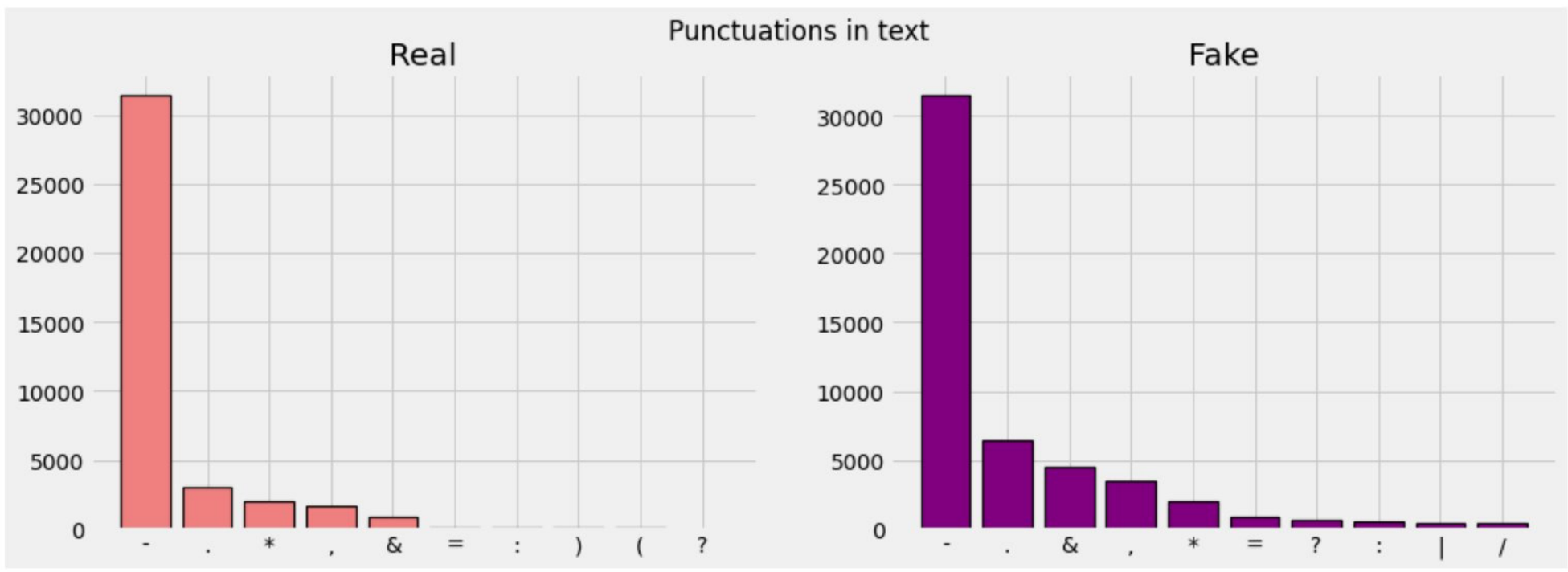
```
from collections import Counter
df['temp_list']=df['text'].apply(lambda x: str(x).split())
top=Counter([word for li in df['temp_list'] for word in li])
temp_1=pd.DataFrame(top.most_common(20))
temp_1.columns=["most_common_words","frequency"]
temp_1.style.background_gradient(cmap='Blues')
```

```
<pandas.io.formats.style.Styler at 0x7ff0ec4a6090>
```

```
import plotly.express as pe
import plotly.figure_factory as ff
fig = pe.bar(temp_1, x="frequency", y="most_common_words",
title='Commmon Words in Text', orientation='h',
             width=700, height=700,color='most_common_words')
fig.show()
```

```
import string
punctuation_list=list(string.punctuation)
value_list=[]
def most_occuring(dataset,fun,target):
    d=defaultdict(int)
    for j in range(0,len(target)):
        words=fun(target[j])
        for i in words:
            if i in punctuation_list:
                d[i]+=1
        top=sorted(d.items(),key=lambda x: x[1],reverse=True)[:10]
        x_items,y_counts=zip(*top)
        value_list.append(x_items)
        value_list.append(y_counts)
    fig,(ax1,ax2) = plt.subplots(1,2,figsize=(15,5))

ax1.bar(value_list[0],value_list[1],color="lightcoral",edgecolor='blac
```

```
k', linewidth=1.2)
    ax1.set_title("Real")


ax2.bar(value_list[2],value_list[3],color="purple",edgecolor='black',
linewidth=1.2)
    ax2.set_title("Fake")

    plt.suptitle("Punctuations in text")
    plt.show()

most_occuring(df,create_words,[1,0])
```




```
import re
import string
from nltk.corpus import stopwords

def clean_text(text):
    """Process text function.
    Input:
        tweet: a string containing a tweet
    Output:
        tweets_clean: a list of words containing the processed tweet
    """
    lemmatizer = WordNetLemmatizer()
    stopwords_english = stopwords.words('english')
    text= re.sub('\[[^]]*\]', '', text)
    # remove stock market tickers like $GE
    text = re.sub(r'\$\w*', '', text)
    #removal of html tags
    review =re.sub(r'<.*?>',' ',text)
    # remove old style retweet text "RT"
    text = re.sub(r'^RT[\s]+', '', text)
    # remove hyperlinks
    text = re.sub(r'https?:\/\/.*[\r\n]*', '', text)
```

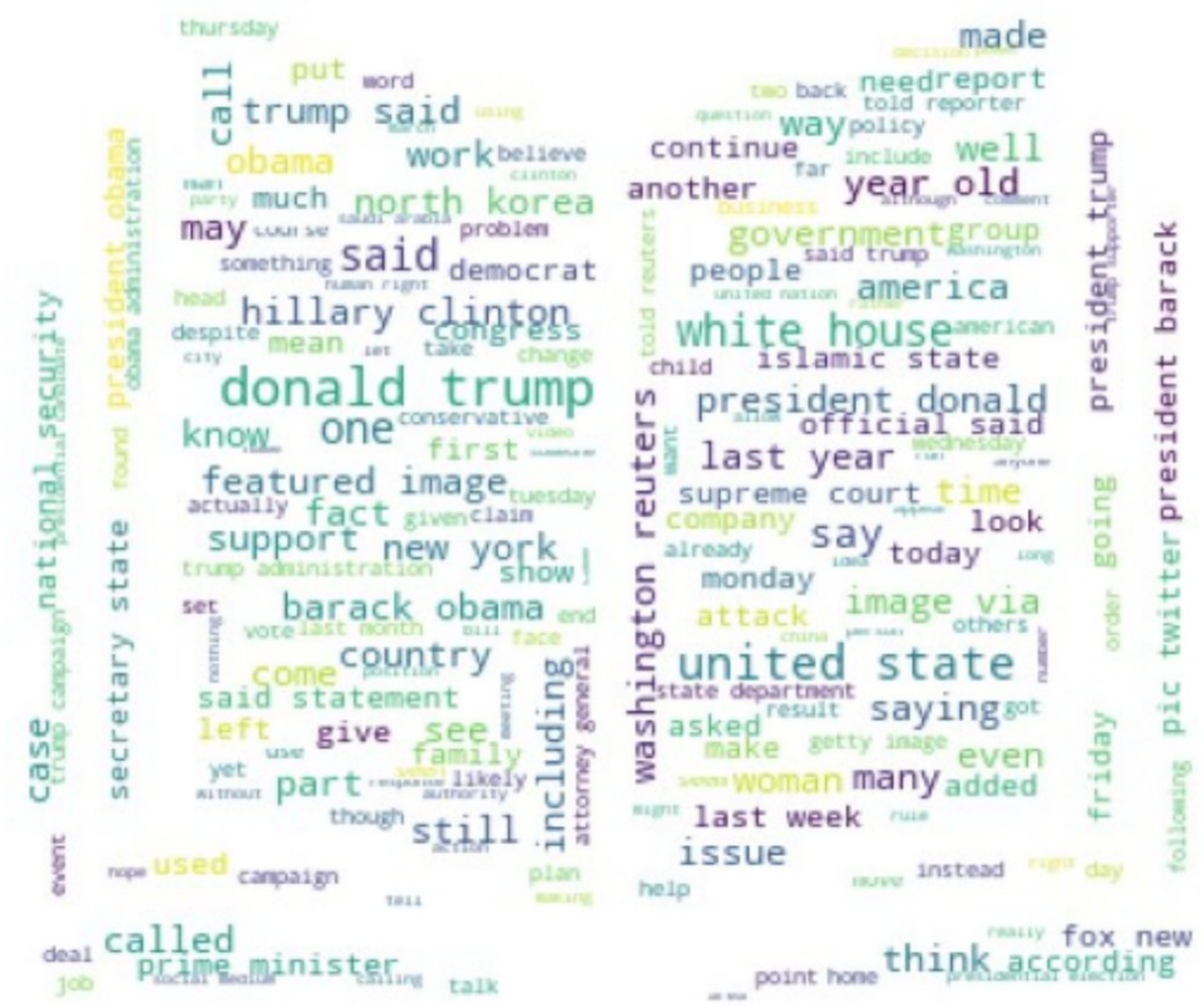
```
    # remove hashtags
    # only removing the hash # sign from the word
    text = re.sub(r'#', '', text)
    text = re.sub("["
                           u"\U0001F600-\U0001F64F"  # removal of
emoticons
                           u"\U0001F300-\U0001F5FF"  # symbols &
pictographs
                           u"\U0001F680-\U0001F6FF"  # transport & map
symbols
                           u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
                           u"\U00002702-\U000027B0"
                           u"\U000024C2-\U0001F251"
                           "]+",' ',text)
    text = re.sub('[^a-zA-Z]',' ',text)
    text = text.lower()
    text_tokens =word_tokenize(text)

    text_clean = []
    for word in  text_tokens:
        if (word not in stopwords_english and  # remove stopwords
                word not in string.punctuation):  # remove punctuation
            lem_word =lemmatizer.lemmatize(word)  # lemmitiging word
            text_clean.append(lem_word)
    text_mod=[i for i in text_clean if len(i)>2]
    text_clean=' '.join(text_mod)
    return  text_clean

df['clean_text']=df['text'].apply(lambda x: clean_text(x))

df['clean_text'][:2]
```

```
0    washington reuters head conservative republica...
1    washington reuters transgender people allowed ...
Name: clean_text, dtype: object
```

```
# wordcloud
from PIL import Image
book_mask = np.array(Image.open('../input/masksforwordclouds/book-
logo-1.jpg'))

wc = WordCloud(
    background_color='white',
    max_words=200,
    mask=book_mask,
)
wc.generate(' '.join(text for text in df.loc[:, 'clean_text']))
plt.figure(figsize=(18,10))
plt.title('Top words',
          fontdict={'size': 28,  'verticalalignment': 'bottom'})
```

```
plt.imshow(wc)
plt.axis("off")
plt.show()
```

Top words



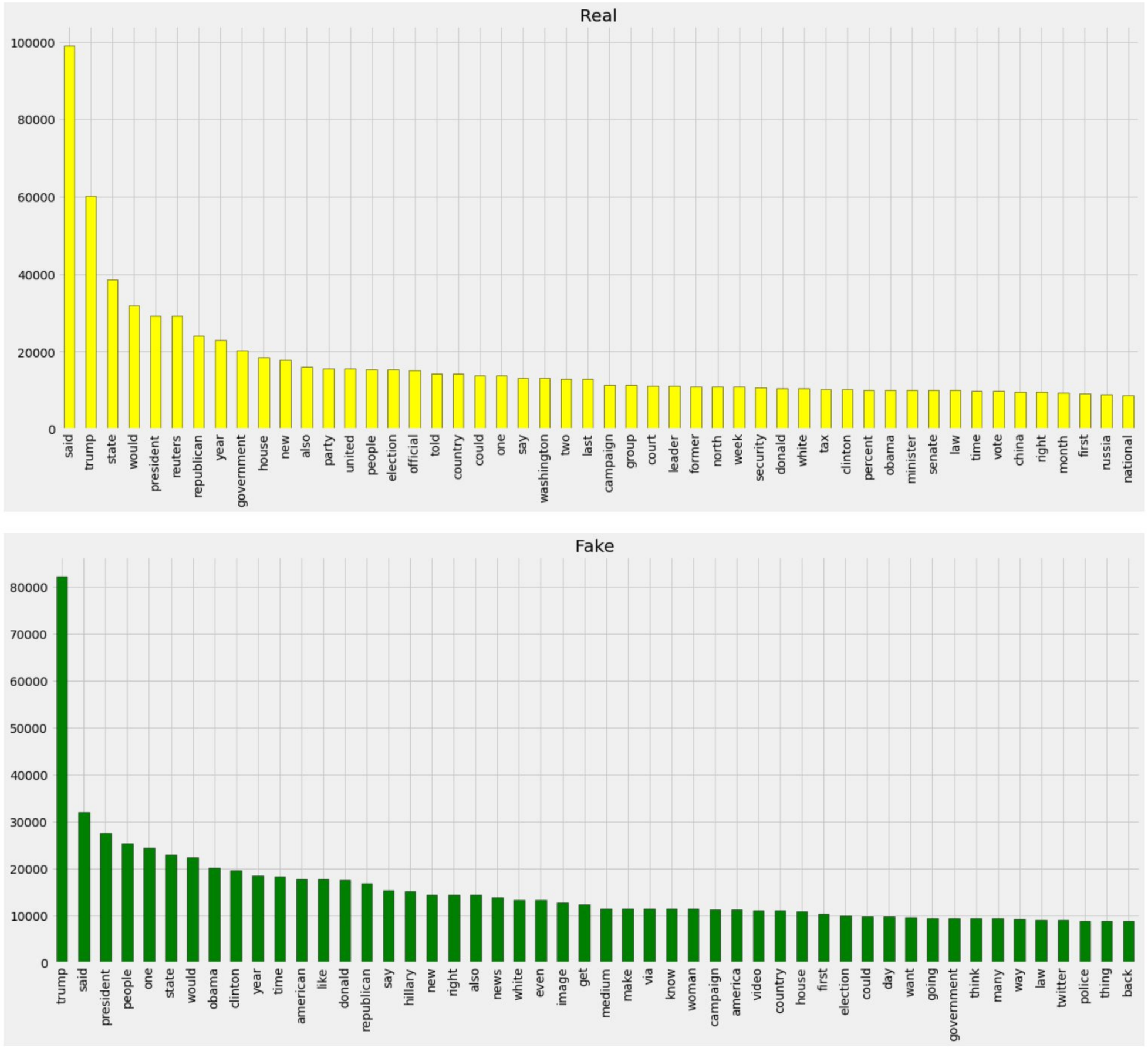

```
df['clean_temp']=df['clean_text'].apply(lambda x: str(x).split())
top=Counter([word for li in df['clean_temp'] for word in li])
temp_2=pd.DataFrame(top.most_common(20))
temp_2.columns=["common_words",'frequency']
temp_2.style.background_gradient(cmap='Blues')
```

```
<pandas.io.formats.style.Styler at 0x7ff0ec4b4850>
```

```
top.most_common(20)
```

```
[('trump', 142609),
 ('said', 131174),
 ('state', 61511),
 ('president', 56889),
 ('would', 54309),
```

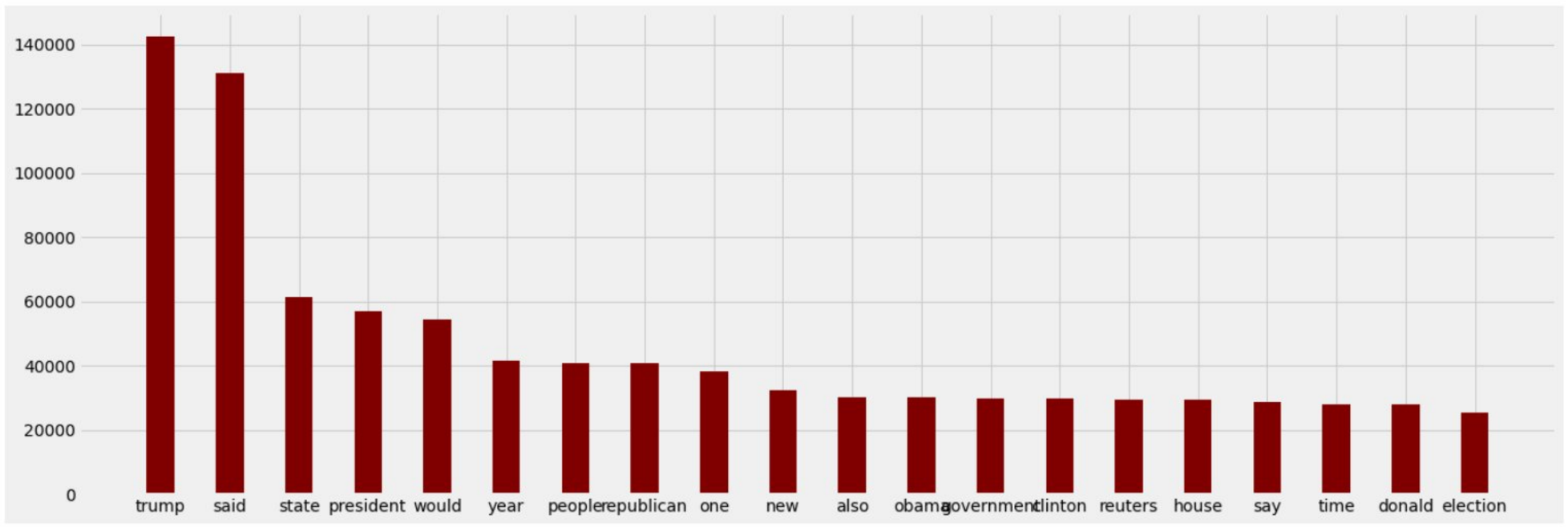
```
 ('year', 41475),
 ('people', 40894),
 ('republican', 40778),
 ('one', 38265),
 ('new', 32225),
 ('also', 30336),
 ('obama', 30225),
 ('government', 29878),
 ('clinton', 29872),
 ('reuters', 29546),
 ('house', 29505),
 ('say', 28649),
 ('time', 28163),
 ('donald', 28038),
 ('election', 25506)]

fig = pe.treemap(temp_2, path=['common_words'],
values='frequency',title='Tree of Most Common Words')
fig.show()

# analyze top 50 words in Real and false texts
data_1=df[df['target']==1]
pd.Series(' '.join([i for i in
data_1.clean_text]).split()).value_counts()
[:50].plot(kind='bar',figsize=(20,8),color='yellow'

,edgecolor='black',title='Real')
plt.show()

data_0=df[df['target']==0]
pd.Series(' '.join([i for i in
data_0.clean_text]).split()).value_counts()
[:50].plot(kind='bar',figsize=(20,8),color='green'

,edgecolor='black',title='Fake')
plt.show()
```




```
data=' '.join([sentance for sentance in df['clean_text']])

import nltk
from nltk.util import ngrams

# Function to generate n-grams from sentences.
def extract_ngrams(data, num):
    n_grams = ngrams(nltk.word_tokenize(data), num)
    return [ ' '.join(grams) for grams in n_grams]

unigrams=extract_ngrams(data, 1)
bigrams= extract_ngrams(data, 2)
```

```
trigrams= extract_ngrams(data, 3)
fourgrams=extract_ngrams(data, 4)

freq_uni = nltk.FreqDist(unigrams)
freq_bi = nltk.FreqDist(bigrams)
freq_tri = nltk.FreqDist(trigrams)
freq_four = nltk.FreqDist(fourgrams)

# top 20 uigrams
top_20_uni=freq_uni.most_common(20)
top_20_uni_words,top_20_uni_freq=list(zip(*top_20_uni))
plt.figure(figsize=(20,7))
plt.bar(top_20_uni_words, top_20_uni_freq, color ='maroon',
        width = 0.4)
```

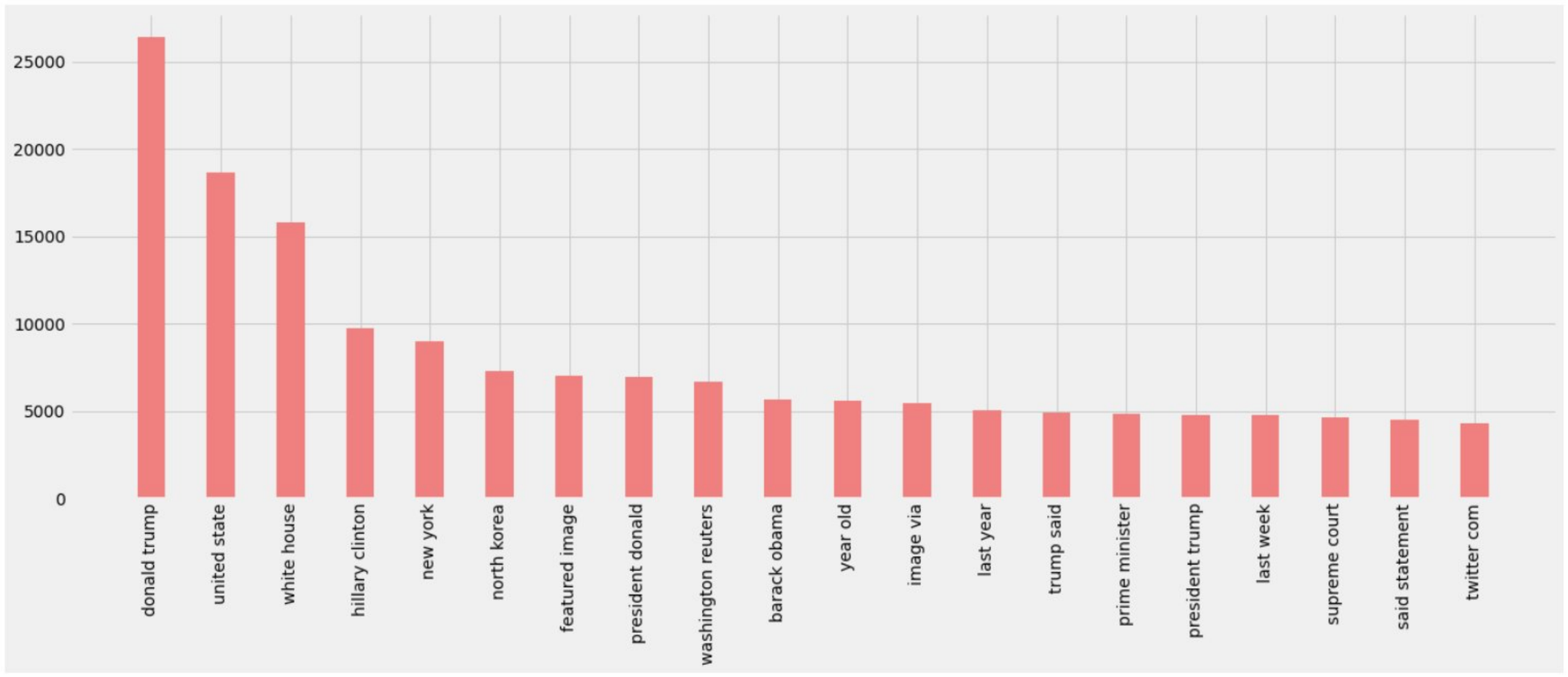
```
<BarContainer object of 20 artists>
```




```
top_20_bi=freq_bi.most_common(20)
top_20_bi_words,top_20_bi_freq=list(zip(*top_20_bi))
plt.figure(figsize=(20,7))
plt.bar(top_20_bi_words, top_20_bi_freq, color ='lightcoral',
        width = 0.4)
plt.xticks(rotation=90)
```

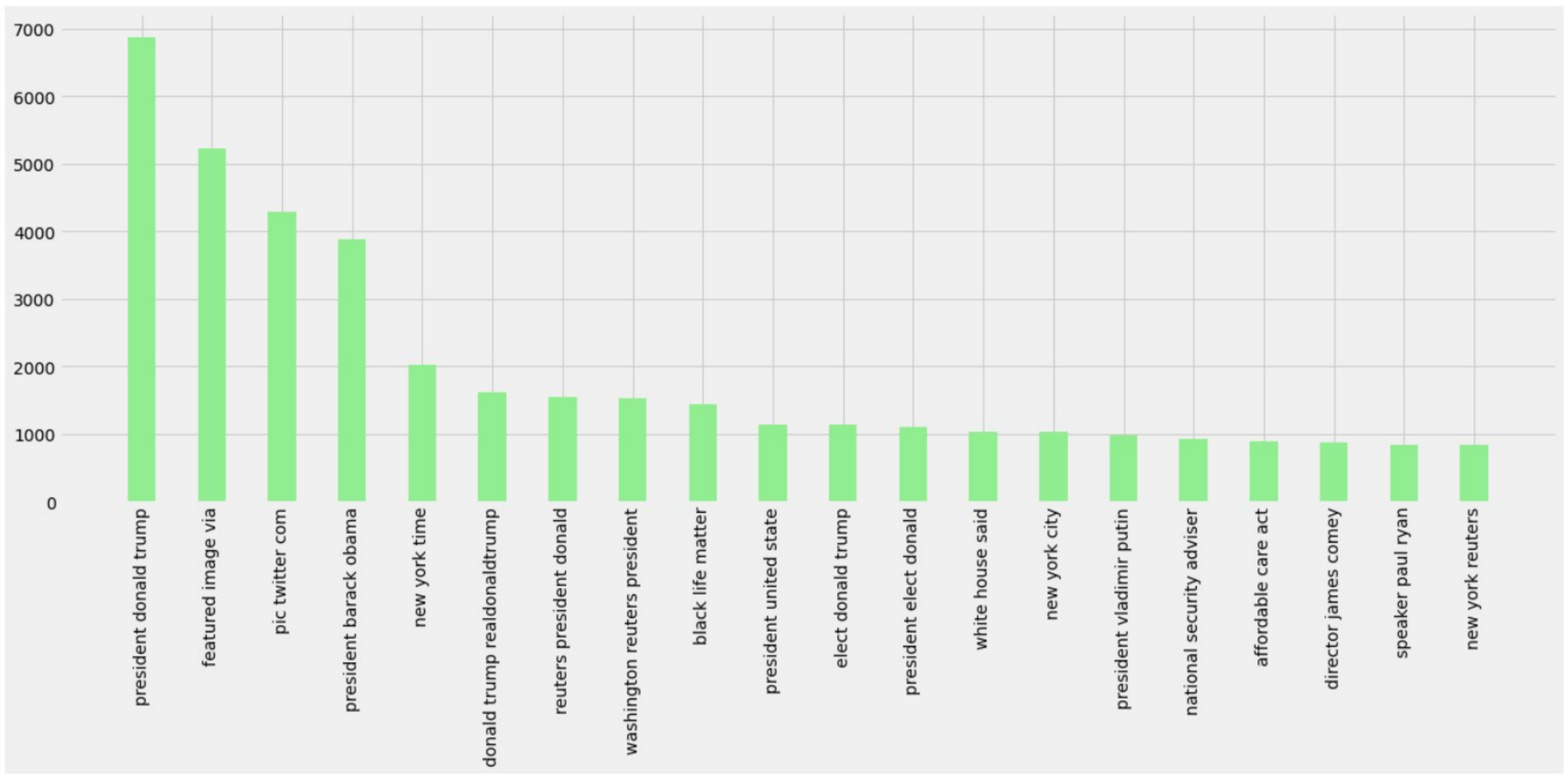
```
([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,
19],
 [Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
```

```
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, '')])
```




```
top_20_tri=freq_tri.most_common(20)
top_20_tri_words,top_20_tri_freq=list(zip(*top_20_tri))
plt.figure(figsize=(20,7))
plt.bar(top_20_tri_words, top_20_tri_freq, color ='lightgreen',
        width = 0.4)
plt.xticks(rotation=90)

([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,
19],
 [Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
```

```
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, '')])
```




```
top_20_four=freq_four.most_common(20)
top_20_four_words,top_20_four_freq=list(zip(*top_20_four))
plt.figure(figsize=(20,7))
plt.bar(top_20_four_words, top_20_four_freq, color ='blue',
        width = 0.4)
plt.xticks(rotation=90)

([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,
19],
 [Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, ''),
 Text(0, 0, '')])
```

```
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, '')])
```