Assignment 5

1. Name : Vaibhav Bichave
2. class : BE 10 (IT)
3. Roll No.: 43209

Problem Statement :

Exploratory analysis on Twitter text data. Perform text pre-pocessing, Apply Zips and heaps law, Identify topics.

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
import re
```

In [2]:
```
df = pd.read_csv('climate_tweets.csv.xls')
df.head()
```

Out[2]:

	tweet
0	Global warming report urges governments to act...
1	Fighting poverty and global warming in Africa ...
2	Carbon offsets: How a Vatican forest failed to...
3	Carbon offsets: How a Vatican forest failed to...
4	URUGUAY: Tools Needed for Those Most Vulnerabl...

Exploratory Data Analysis

In [3]:
```
# shape of dataset
print('Shape of dataset = ',df.shape)
# shape of unique elems in dataset
print('Shape of dataset with unique tweets = ',df.tweet.unique().shape)
```

```
Shape of dataset =  (6090, 1)
Shape of dataset with unique tweets =  (5541,)
```

In [4]:
```
# make a new column to highlight retweets
df['is_retweet'] = df['tweet'].apply(lambda x: x[:2]=='RT')
df['is_retweet'].sum()  # number of retweets
```

Out[4]: 773

In [5]:
```
df.head()
```

Out[5]:

	tweet	is_retweet
0	Global warming report urges governments to act...	False
1	Fighting poverty and global warming in Africa ...	False
2	Carbon offsets: How a Vatican forest failed to...	False
3	Carbon offsets: How a Vatican forest failed to...	False
4	URUGUAY: Tools Needed for Those Most Vulnerabl...	False

In [6]:
```
# 10 most repeated tweets
df.groupby(['tweet']).size().reset_index(name='counts')\
  .sort_values('counts', ascending=False).head(10)
```

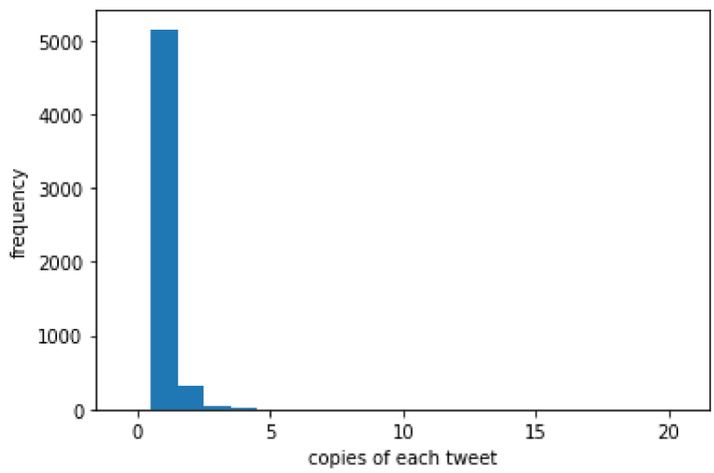
Out[6]:

	tweet	counts
3131	No matter if you believe in global warming or ...	20
4555	Take Action @change: Help Protect Wildlife Hab...	14
4027	RT @newtgingrich: Historic snow storm in washi...	9
1765	Fight Climate Change From All Fronts: [link]	8
1626	Earth's polar ice sheets vulnerable to even mo...	7
1941	Global Warming Dent's El Ni'o's Protective Shi...	7
1799	Foes of California's global warming law pour m...	6
1351	Coalition of the Tired of Waiting: Fighting Cl...	6
4271	SCIENCE: Scientists explore the evolution of c...	6
1040	Carbon offsets: How a Vatican forest failed to...	6

In [7]:
```
# number of times each tweet appears
counts = df.groupby(['tweet']).size().reset_index(name='counts').counts

# define bins for histogram
my_bins = np.arange(0,counts.max()+2, 1)-0.5

# plot histogram of tweet counts
plt.figure()
plt.hist(counts, bins = my_bins)
plt.xlabels = np.arange(1,counts.max()+1, 1)
plt.xlabel('copies of each tweet')
plt.ylabel('frequency')
plt.show()
```



Extracting substrings with regex

In [8]:
```
def find_retweeted(tweet):
    '''This function will extract the twitter handles of retweed people'''
    return re.findall('(?<=RT\s)(@[A-Za-z]+[A-Za-z0-9-_]+)', tweet)

def find_mentioned(tweet):
    '''This function will extract the twitter handles of people mentioned in
    return re.findall('(?<!RT\s)(@[A-Za-z]+[A-Za-z0-9-_]+)', tweet)

def find_hashtags(tweet):
    '''This function will extract hashtags'''
    return re.findall('(#[A-Za-z]+[A-Za-z0-9-_]+)', tweet)
```

In [9]:
```
# make new columns for retweeted usernames, mentioned usernames and hashtags
df['retweeted'] = df.tweet.apply(find_retweeted)
df['mentioned'] = df.tweet.apply(find_mentioned)
df['hashtags'] = df.tweet.apply(find_hashtags)
```

In [10]:
```
df
```

Out[10]:

	tweet	is_retweet	retweeted	mentioned	hashtags
0	Global warming report urges governments to act...	False	[]	[]	[]
1	Fighting poverty and global warming in Africa ...	False	[]	[]	[]
2	Carbon offsets: How a Vatican forest failed to...	False	[]	[]	[]
3	Carbon offsets: How a Vatican forest failed to...	False	[]	[]	[]
4	URUGUAY: Tools Needed for Those Most Vulnerabl...	False	[]	[]	[]
...	...	...	...	...	...
6085	@bloodless_coup "The phrase 'global warming' s...	False	[]	[@bloodless_coup]	[#p2, #tcot]
6086	Virginia to Investigate Global Warming Scienti...	False	[]	[]	[]
6087	Global warming you tube parody you will enjoy ...	False	[]	[]	[#IPCC, #ocra]
6088	One-Eyed Golfer: Don't dare tell me about glob...	False	[]	[]	[]
6089	man made global warming a hair brained theory ...	False	[]	[]	[#tcot, #p2, #climategate]

6090 rows × 5 columns

Hashtag Analysis

In [11]:
```
# take the rows from the hashtag columns where there are actually hashtags
hashtags_list_df = df.loc[
                      df.hashtags.apply(
                          lambda hashtags_list: hashtags_list !=[]
                      ),['hashtags']]
```

In [12]:
```
hashtags_list_df
```

Out[12]:

	hashtags
12	[#Climate, #population]
16	[#EarthDay]
26	[#ac]
31	[#tcot]
36	[#tornadocot, #ocra, #sgp, #gop, #ucot, #tlot,...
...	...
6076	[#liberalFascism, #News, #tcot]
6083	[#climate]
6085	[#p2, #tcot]
6087	[#IPCC, #ocra]
6089	[#tcot, #p2, #climategate]

1129 rows × 1 columns

In [13]:
```
# create dataframe where each use of hashtag gets its own row
flattened_hashtags_df = pd.DataFrame(
    [hashtag for hashtags_list in hashtags_list_df.hashtags
    for hashtag in hashtags_list],
    columns=['hashtag'])
```

In [14]:
```
flattened_hashtags_df
```

Out[14]:

	hashtag
0	#Climate
1	#population
2	#EarthDay
3	#ac
4	#tcot
...	...
2062	#IPCC
2063	#ocra
2064	#tcot
2065	#p2
2066	#climategate

2067 rows × 1 columns

In [15]:
```
# number of unique hashtags
flattened_hashtags_df['hashtag'].unique().size
```

Out[15]: 477

In [16]:
```
# count of appearances of each hashtag
popular_hashtags = flattened_hashtags_df.groupby('hashtag').size()\
                                        .reset_index(name='counts')\
                                        .sort_values('counts', ascending=Fals
                                        .reset_index(drop=True)
```

In [17]:
```
popular_hashtags
```

Out[17]:

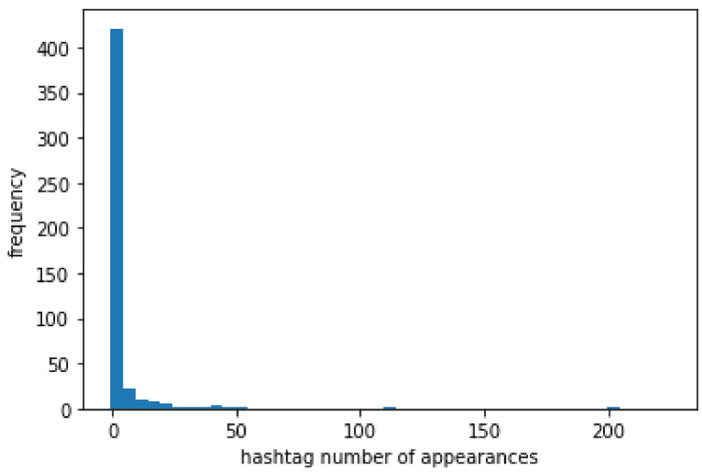
	hashtag	counts
0	#tcot	227
1	#climate	202
2	#p2	112
3	#green	50
4	#climatechange	47
...	...	...
472	#home	1
473	#hoth	1
474	#houston	1
475	#humanrights	1
476	#digg	1

477 rows × 2 columns

In [19]:
```
# number of times each hashtag appears
counts = flattened_hashtags_df.groupby(['hashtag']).size()\
                              .reset_index(name='counts')\
                              .counts

# define bins for histogram
my_bins = np.arange(0,counts.max()+2, 5)-0.5

# plot histogram of tweet counts
plt.figure()
plt.hist(counts, bins = my_bins)
plt.xlabels = np.arange(1,counts.max()+1, 1)
plt.xlabel('hashtag number of appearances')
plt.ylabel('frequency')
plt.show()
```



Vectorization

In [20]:
```
# taking hashtags which appear at least this amount of times
min_appearance = 10
# find popular hashtags - make into python set for efficiency
popular_hashtags_set = set(popular_hashtags[
                           popular_hashtags.counts>=min_appearance
                           ]['hashtag'])
```

In [21]:
```
# make a new column with only the popular hashtags
hashtags_list_df['popular_hashtags'] = hashtags_list_df.hashtags.apply(
            lambda hashtag_list: [hashtag for hashtag in hashtag_list
                                  if hashtag in popular_hashtags_set])
# drop rows without popular hashtag
popular_hashtags_list_df = hashtags_list_df.loc[
            hashtags_list_df.popular_hashtags.apply(lambda hashtag_list: hash
```

In [22]:
```
# make new dataframe
hashtag_vector_df = popular_hashtags_list_df.loc[:, ['popular_hashtags']]

for hashtag in popular_hashtags_set:
    # make columns to encode presence of hashtags
    hashtag_vector_df['{}'.format(hashtag)] = hashtag_vector_df.popular_hasht
        lambda hashtag_list: int(hashtag in hashtag_list))
```

In [23]:
```
hashtag_vector_df
```

Out[23]:

	popular_hashtags	#Green	#GlobalWarming	#SaveTerra	#IPCC	#EarthDay	#globalwarming	
12	[#Climate]	0	0	0	0	0	0	
16	[#EarthDay]	0	0	0	0	1	0	
31	[#tcot]	0	0	0	0	0	0	
36	[#ocra, #sgp, #gop, #tlot, #p2]	0	0	0	0	0	0	
39	[#tcot, #p2]	0	0	0	0	0	0	
...	...	...	...	...	...	...	...	...
6076	[#News, #tcot]	0	0	0	0	0	0	
6083	[#climate]	0	0	0	0	0	0	
6085	[#p2, #tcot]	0	0	0	0	0	0	
6087	[#IPCC, #ocra]	0	0	0	1	0	0	
6089	[#tcot, #p2, #climategate]	0	0	0	0	0	0	

786 rows × 36 columns

In [24]:
```
hashtag_matrix = hashtag_vector_df.drop('popular_hashtags', axis=1)
```

In [25]:
```
hashtag_matrix
```

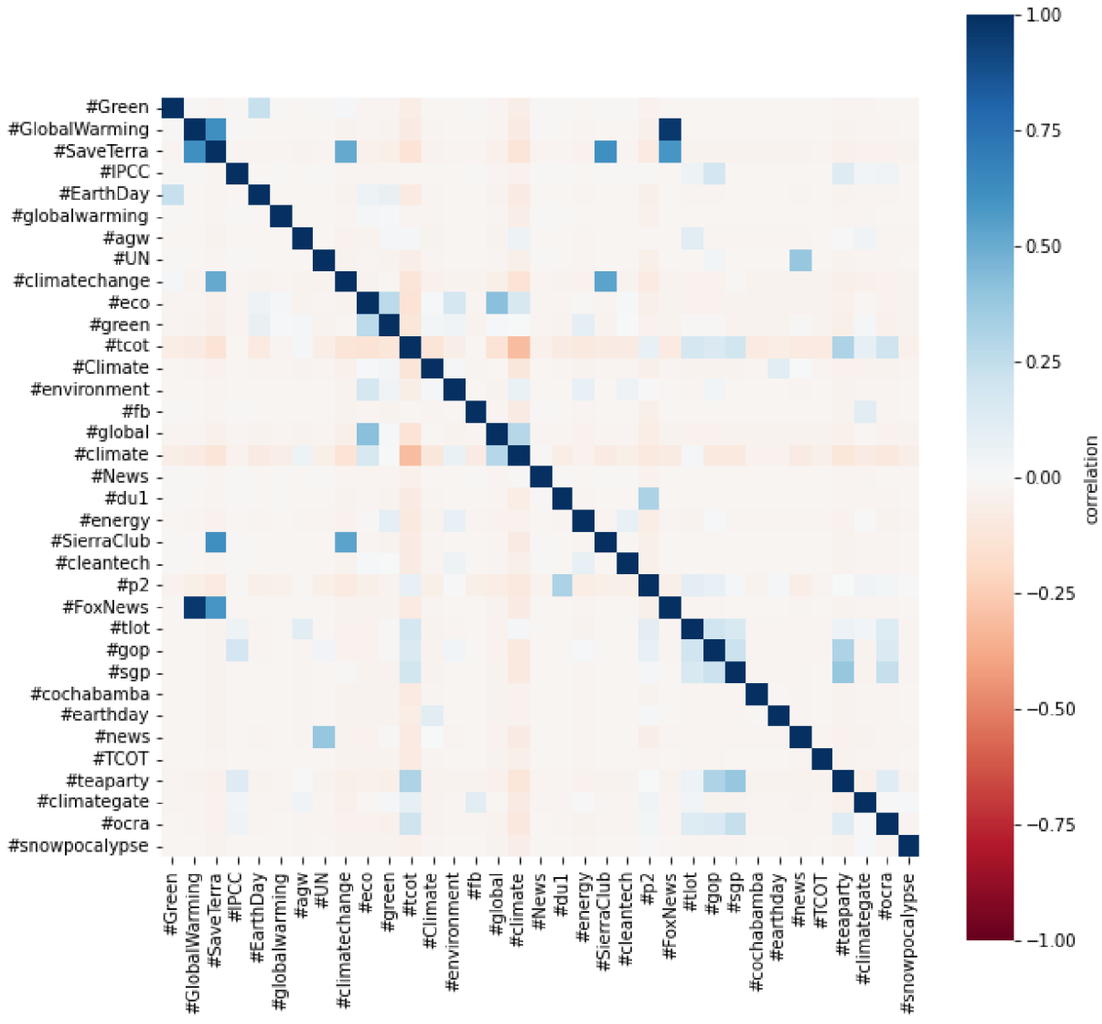
Out[25]:

	#Green	#GlobalWarming	#SaveTerra	#IPCC	#EarthDay	#globalwarming	#agw	#UN	#clima
12	0	0	0	0	0	0	0	0	
16	0	0	0	0	1	0	0	0	
31	0	0	0	0	0	0	0	0	
36	0	0	0	0	0	0	0	0	
39	0	0	0	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	
6076	0	0	0	0	0	0	0	0	
6083	0	0	0	0	0	0	0	0	
6085	0	0	0	0	0	0	0	0	
6087	0	0	0	1	0	0	0	0	
6089	0	0	0	0	0	0	0	0	

786 rows × 35 columns

In [26]:
```
# calculate the correlation matrix
correlations = hashtag_matrix.corr()

# plot the correlation matrix
plt.figure(figsize=(10,10))
sns.heatmap(correlations,
    cmap='RdBu',
    vmin=-1,
    vmax=1,
    square = True,
    cbar_kws={'label':'correlation'})
plt.show()
```



From the plot above we can see that there are fairly strong correlations between:

1. SaveTerra and SierraClub
2. GloablWarming and FoxNews

We can also see a fairly strong negative correlation between:

1. tcot and climate

Topic Modelling

Cleaning the unstructured text data

In [27]:
```
!pip install nltk
import nltk
nltk.download('stopwords')
from nltk.tokenize import RegexpTokenizer
from nltk.corpus import stopwords
```

```
Requirement already satisfied: nltk in c:\users\admin\appdata\local\programs
\python\python38\lib\site-packages (3.6.5)
Requirement already satisfied: joblib in c:\users\admin\appdata\local\progra
ms\python\python38\lib\site-packages (from nltk) (1.1.0)
Requirement already satisfied: click in c:\users\admin\appdata\local\program
s\python\python38\lib\site-packages (from nltk) (8.0.3)
Requirement already satisfied: tqdm in c:\users\admin\appdata\local\programs
\python\python38\lib\site-packages (from nltk) (4.62.3)
Requirement already satisfied: regex>=2021.8.3 in c:\users\admin\appdata\loc
al\programs\python\python38\lib\site-packages (from nltk) (2021.11.10)
Requirement already satisfied: colorama in c:\users\admin\appdata\local\prog
rams\python\python38\lib\site-packages (from click->nltk) (0.4.4)

[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Admin\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [28]:
```
def remove_links(tweet):
    '''Takes a string and removes web links from it'''
    tweet = re.sub(r'http\S+', '', tweet) # remove http links
    tweet = re.sub(r'bit.ly/\S+', '', tweet) # rempve bitly links
    tweet = tweet.strip('[link]') # remove [links]
    return tweet

def remove_users(tweet):
    '''Takes a string and removes retweet and @user information'''
    tweet = re.sub('(RT\s@[A-Za-z]+[A-Za-z0-9-_]+)', '', tweet) # remove retw
    tweet = re.sub('(@[A-Za-z]+[A-Za-z0-9-_]+)', '', tweet) # remove tweeted
    return tweet
```

In [29]:
```
my_stopwords = nltk.corpus.stopwords.words('english')
word_rooter = nltk.stem.snowball.PorterStemmer(ignore_stopwords=False).stem
my_punctuation = '!"$%&\'()*+,-./:;<=>?[\\]^_`{|}~•@'

# cleaning master function
def clean_tweet(tweet, bigrams=False):
    tweet = remove_users(tweet)
    tweet = remove_links(tweet)
    tweet = tweet.lower() # lower case
    tweet = re.sub('['+my_punctuation + ']+', ' ', tweet) # strip punctuation
    tweet = re.sub('\s+', ' ', tweet) #remove double spacing
    tweet = re.sub('([0-9]+)', '', tweet) # remove numbers
    tweet_token_list = [word for word in tweet.split(' ')
                            if word not in my_stopwords] # remove stopwords

    tweet_token_list = [word_rooter(word) if '#' not in word else word
                        for word in tweet_token_list] # apply word rooter
    if bigrams:
        tweet_token_list = tweet_token_list+[tweet_token_list[i]+'_'+tweet_to
                                             for i in range(len(tweet_token_li
    tweet = ' '.join(tweet_token_list)
    return tweet
```

In [30]:
```
df['clean_tweet'] = df.tweet.apply(clean_tweet)
```

In [31]:
```
df
```

Out[31]:

	tweet	is_retweet	retweeted	mentioned	hashtags	clean_tweet
0	Global warming report urges governments to act...	False	[]	[]	[]	global warm report urg govern act brussel belg...
1	Fighting poverty and global warming in Africa ...	False	[]	[]	[]	fight poverti global warm africa
2	Carbon offsets: How a Vatican forest failed to...	False	[]	[]	[]	carbon offset vatican forest fail reduc global...
3	Carbon offsets: How a Vatican forest failed to...	False	[]	[]	[]	carbon offset vatican forest fail reduc global...
4	URUGUAY: Tools Needed for Those Most Vulnerabl...	False	[]	[]	[]	uruguay tool need vulner climat chang
...	...	...	...	...	...	...
6085	@bloodless_coup "The phrase 'global warming' s...	False	[]	[@bloodless_coup]	[#p2, #tcot]	phrase global warm abandon favor climat chang...
6086	Virginia to Investigate Global Warming Scienti...	False	[]	[]	[]	virginia investig global warm scientist mann
6087	Global warming you tube parody you will enjoy ...	False	[]	[]	[#IPCC, #ocra]	global warm tube parodi enjoy #ipcc #ocra
6088	One-Eyed Golfer: Don't dare tell me about glob...	False	[]	[]	[]	one eye golfer dare tell global warm twenti fi...
6089	man made global warming a hair brained theory ...	False	[]	[]	[#tcot, #p2, #climategate]	man made global warm hair brain theori scient...

6090 rows × 6 columns

Applying Topic Modelling

In [32]:
```
from sklearn.feature_extraction.text import CountVectorizer

# the vectorizer object will be used to transform text to vector form
vectorizer = CountVectorizer(max_df=0.9, min_df=25, token_pattern='\w+|\$[\d\
```

In [33]:
```
from sklearn.decomposition import LatentDirichletAllocation

# chosen arbitrarily
number_of_topics = 10

model = LatentDirichletAllocation(n_components=number_of_topics, random_state
```

In [34]:
```
model.fit(tf)
```

Out[34]:
```
LatentDirichletAllocation(random_state=0)
```

In [35]:
```
def display_topics(model, feature_names, no_top_words):
    topic_dict = {}
    for topic_idx, topic in enumerate(model.components_):
        topic_dict["Topic %d words" % (topic_idx)]= ['{}'.format(feature_name
                        for i in topic.argsort()[:-no_top_words - 1:-1]]
        topic_dict["Topic %d weights" % (topic_idx)]= ['{:.1f}'.format(topic[
                        for i in topic.argsort()[:-no_top_words - 1:-1]]
    return pd.DataFrame(topic_dict)
```

In [40]:
```
topic_df = display_topics(model, tf_feature_names, 10)
topic_df
```

Out[40]:

	Topic 0 words	Topic 0 weights	Topic 1 words	Topic 1 weights	Topic 2 words	Topic 2 weights	Topic 3 words	Topic 3 weights	Topic 4 words	Topic 4 weights	Topic 5 words
0	climat	1220.2	global	666.5	global	1147.2	global	473.1	climat	422.0	globa
1	chang	1184.5	warm	658.1	warm	1102.1	warm	450.7	chang	401.8	warm
2	via	257.9	snow	160.5	scientist	150.2	believ	101.3	legisl	123.2	gore
3	scienc	112.9	#tcot	121.6	say	87.1	california	87.1	us	105.1	snow
4	news	79.5	like	99.0	scienc	71.7	blame	82.1	via	60.5	a
5	day	77.8	blizzard	90.9	debat	66.6	law	78.6	say	55.9	grea
6	earth	68.2	dc	86.5	man	61.1	report	77.8	video	55.9	colc
7	trial	68.1	think	83.3	made	54.0	save	61.8	place	54.1	cal
8	clinic	68.1	due	80.1	show	51.9	money	56.1	good	52.6	#tco
9	carbon	64.5	make	70.4	water	50.6	live	49.3	human	48.2	one

In [38]:
```
topic_df.shape
```

Out[38]:
```
(10, 20)
```

Thus, 10 topics were identified with 10 words related to each topic.