VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Lab Digital Assignment 6

TEXT VISUALIZATION

B.Tech in Computer Science and Engineering **(CSE)**, **Winter** Semester **2020-21**

Name:	Swaranjana Nayak
Registration Number:	19BCE0977
Slot:	L55 + L56
Date:	29.04.2021

Aim:
To visualize the Climate Change Twitter Sentiment Analysis dataset.

Importing the dataset:

```
tsa = pd.read_csv("twitter_sentiment_data.csv")
tsa.head()
```

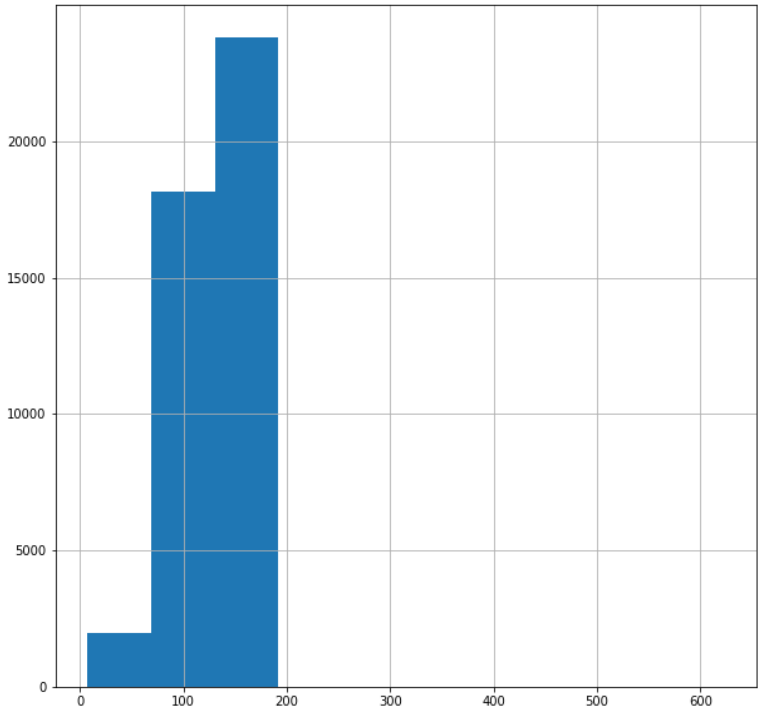
	sentiment	message	tweetid
0	-1	@tiniebeany climate change is an interesting h...	792927353886371840
1	1	RT @NatGeoChannel: Watch #BeforeTheFlood right...	793124211518832641
2	1	Fabulous! Leonardo #DiCaprio's film on #climat...	793124402388832256
3	1	RT @Mick_Fanning: Just watched this amazing do...	793124635873275904
4	2	RT @cnalive: Pranita Biswasi, a Lutheran from ...	793125156185137153

Visualizing distribution of length of tweets

Code:

```
fig = plt.figure(figsize=(10, 10))
tsa['message'].str.len().hist()
fig.savefig("hist-length-of-tweet.png", bbox_inches = 'tight')
```

Output:

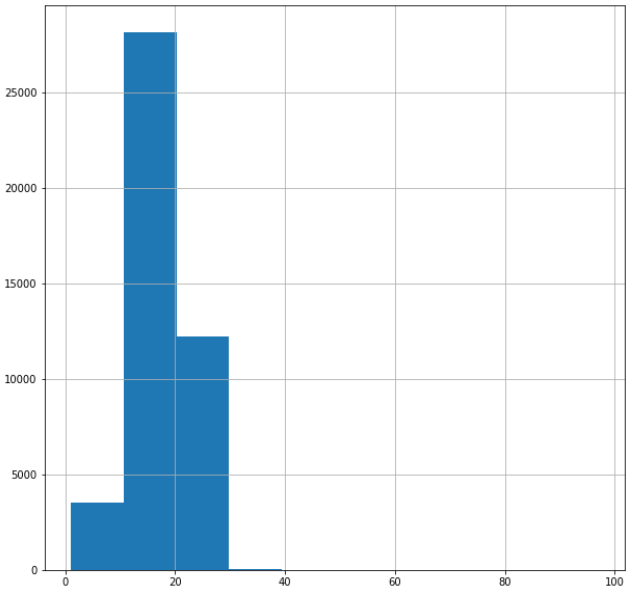


Visualizing distribution of the number of words in a tweet

Code:

```
fig = plt.figure(figsize=(10, 10))
tsa['message'].str.split().map(lambda x : len(x)).hist()
fig.savefig("hist-no-words-of-tweet.png", bbox_inches = 'tight')
```

Output:

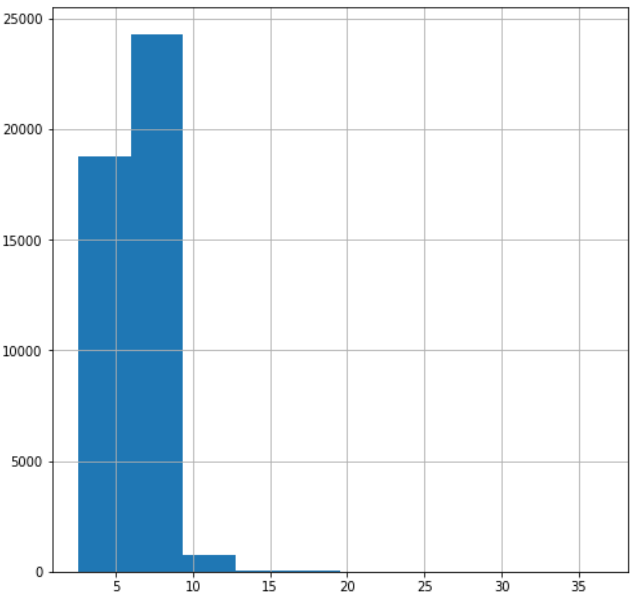


Visualization of the average length of a word in a tweet

Code:

```
fig = plt.figure(figsize=(8, 8))
obj = tsa['message'].str.split().apply(lambda x : [len(i) for i in x])
obj = obj.map(lambda x : np.mean(x)) #map is only for pandas objects
obj.hist()
fig.savefig("hist-avg-wordlen-of-tweet.png", bbox_inches = 'tight')
```

Output:

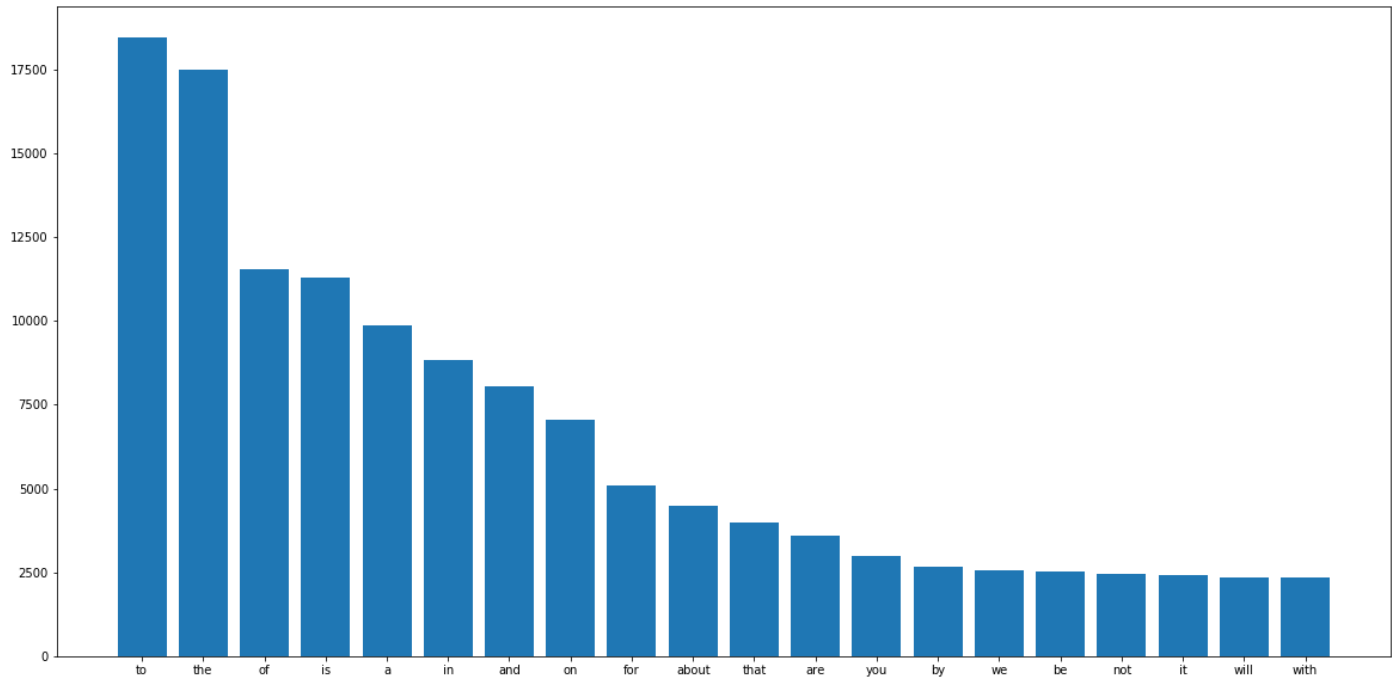


Frequency of stopwords in the tweets

Code:

```
import nltk
# nltk.download('stopwords')
from nltk.corpus import stopwords
stop = set(stopwords.words('english'))

corpus = []
twt = tsa['message'].str.split()
twt = twt.values.tolist()
corpus = [word for i in twt for word in i]

from collections import defaultdict
dic = defaultdict(int)

for word in corpus:
    if word in stop:
        dic[word] = dic[word] + 1

fig = plt.figure(figsize = (20, 10))
top = sorted(dic.items(), key=lambda x:x[1],reverse=True)[:20]
x,y = zip(*top)
plt.bar(x,y)
fig.savefig("top-stopwords-bar.png", bbox_inches="tight")
```

Output:

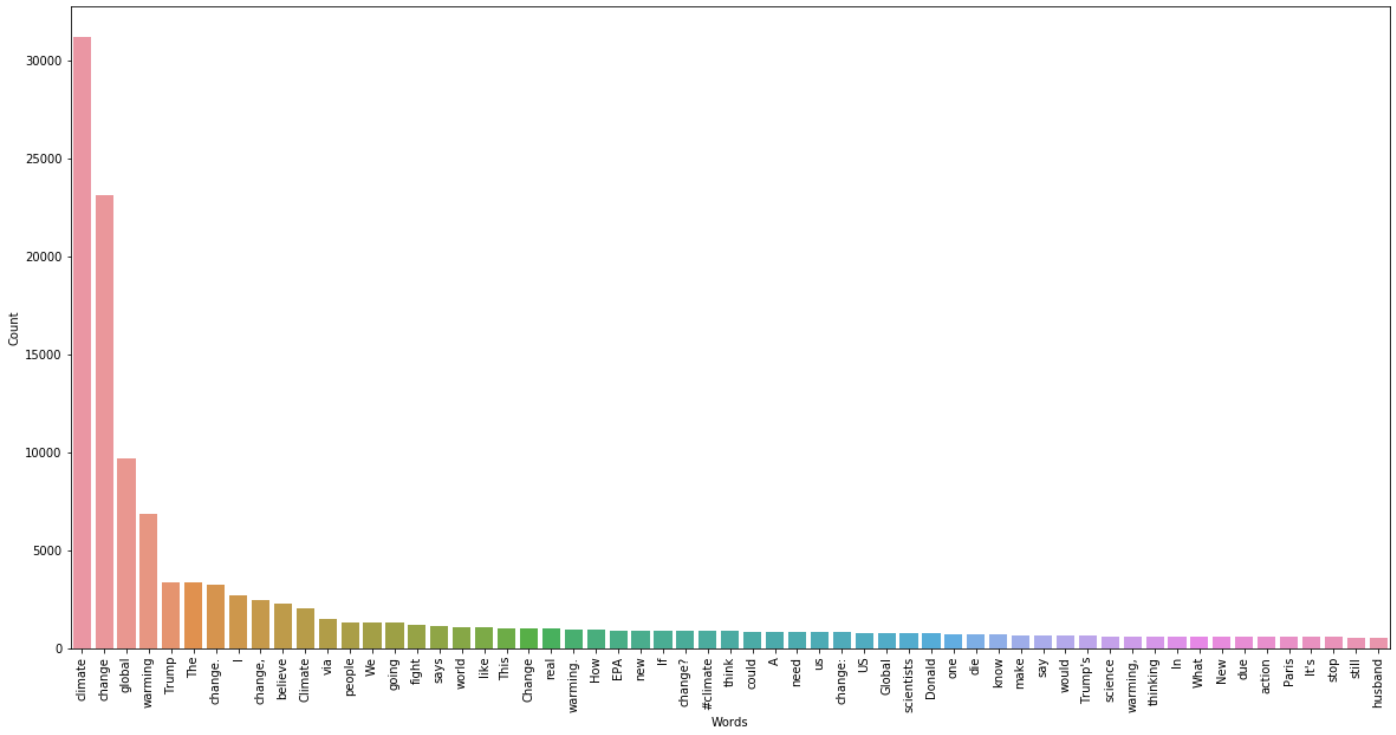


Frequency of words other than stopwords

Code:

```
import collections
from collections import Counter
import re
def username_filter(word):
     if(re.search("^@", word) == None and re.search("^((http|https)\:\/\/)",
word) == None
         and re.search("^&amp;$", word) == None and word != "RT" and word !=
"|" and word != "-"):
         return True
    else:
         return False
filtered_corpus = filter(username_filter, corpus)
filtered_corpus = list(filtered_corpus)
counter = Counter(filtered_corpus)
most = counter.most_common()
x, y= [], []
for word,count in most[:125]:
    if (word not in stop):
         x.append(word)
         y.append(count)
fig = plt.figure(figsize = (20, 10))
plt.xticks(rotation = 90)
plt.xlabel("Words")
plt.ylabel("Count")
sns.barplot(x = x, y = y)
fig.savefig("freq-words-barplot.png", bbox_inches = "tight")
```

Output:

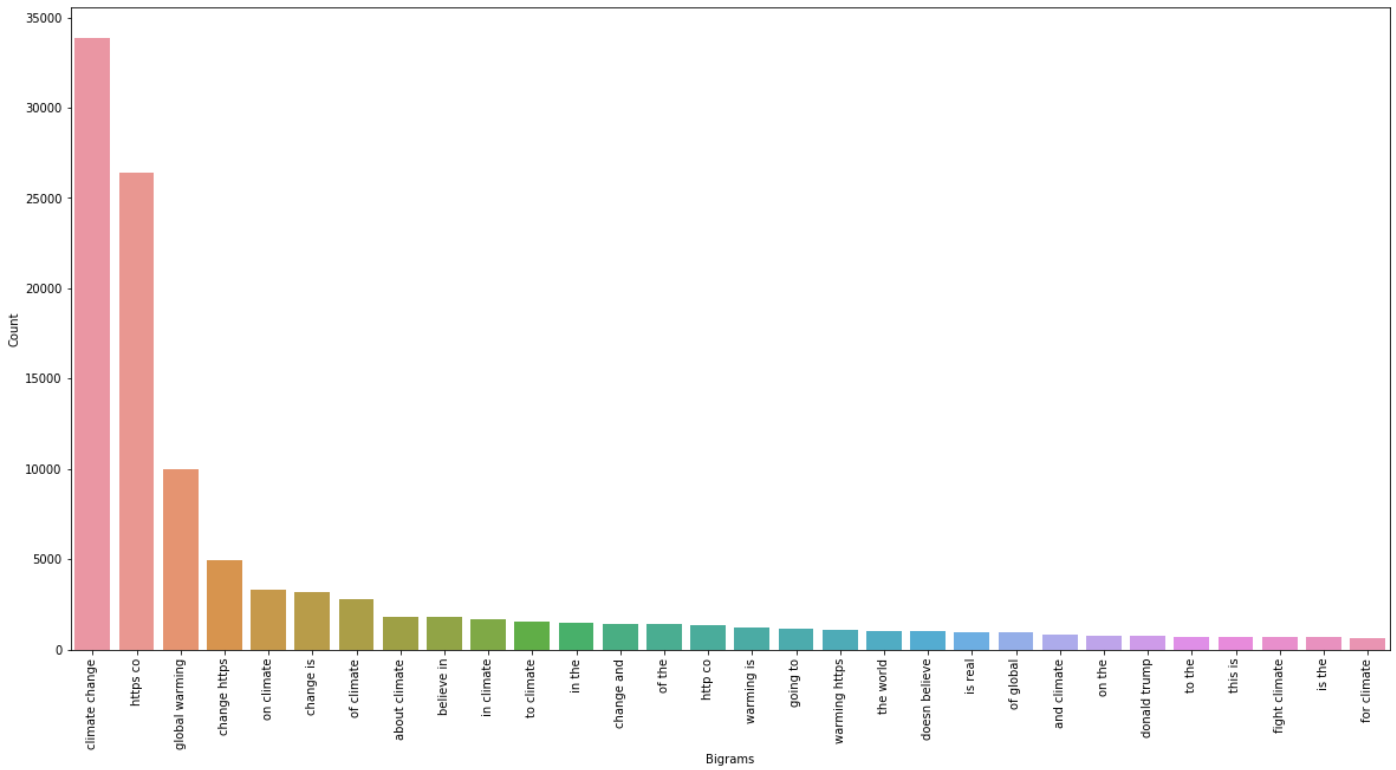


Frequency of bigrams

Code:

```
from sklearn.feature_extraction.text import CountVectorizer
def get_top_ngram(corpus, n=None):
    vec = CountVectorizer(ngram_range=(n, n)).fit(corpus)
    bow = vec.transform(corpus)
    sum_words = bow.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[:140]

top_n_bigrams=get_top_ngram(tsa['message'],2)[:30]
print(top_n_bigrams)
x,y=map(list,zip(*top_n_bigrams))
fig = plt.figure(figsize = (20, 10))
plt.xticks(rotation = 90)
plt.xlabel("Bigrams")
plt.ylabel("Count")
sns.barplot(x = x, y = y)
fig.savefig("freq-bigrams-barplot.png", bbox_inches = "tight")
```

Output:



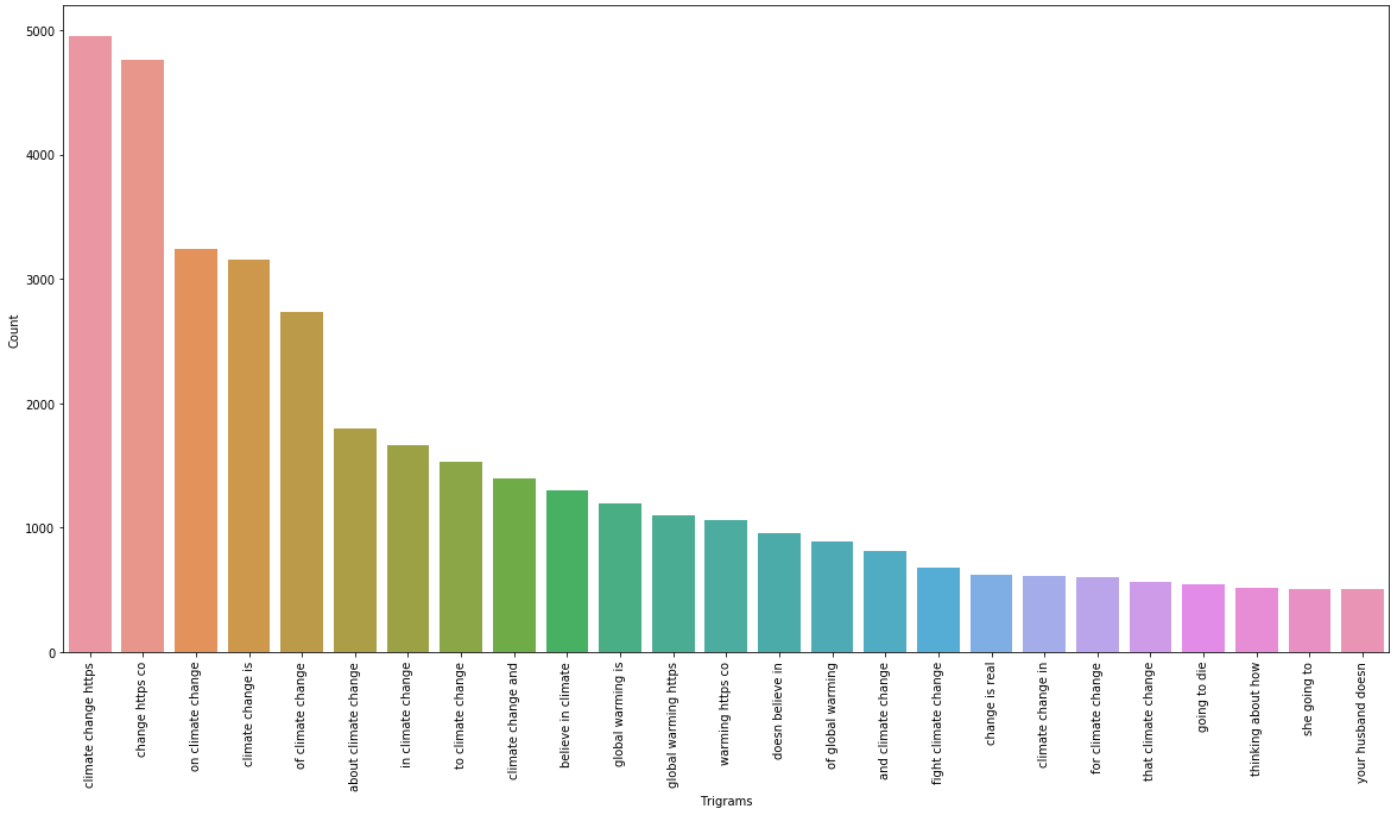
Frequency of trigrams

Code:

```
top_n_trigrams=get_top_ngram(tsa['message'],3)[:25]
print(top_n_trigrams)
x,y=map(list,zip(*top_n_trigrams))
fig = plt.figure(figsize = (20, 10))
plt.xticks(rotation = 90)
plt.xlabel("Trigrams")
plt.ylabel("Count")
sns.barplot(x = x, y = y)
fig.savefig("freq-trigrams-barplot.png", bbox_inches = "tight")
```

Output:

```
[('climate change https', 4948), ('change https co', 4756), ('on climate change', 3236), ('climate change is', 3159), ('of climate change', 2731), ('about climate change', 1802), ('in climate change', 1661), ('to climate change', 1534), ('climate change and', 1396), ('believe in climate', 1298), ('global warming is', 1197), ('global warming https', 1100), ('warming https co', 1061), ('doesn believe in', 955), ('of global warming', 890), ('and climate change', 812), ('fight climate change', 678), ('change is real', 626), ('climate change in', 612), ('for climate change', 606), ('that climate change', 562), ('going to die', 547), ('thinking about how', 512), ('she going to', 511), ('your husband doesn', 511)]
```



Preprocessing and Topic Modelling

Code:

```
import nltk
import gensim
from nltk.stem import PorterStemmer
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize
from nltk import punkt
from nltk import wordnet
# nltk.download('punkt')
# nltk.download('wordnet')

def preprocess_tweets(df):
    corpus = []
    stem = PorterStemmer()
    lem = wordnet.WordNetLemmatizer()
    for tweet in df['message']:
          words = [w for w in word_tokenize(tweet) if ((w.lower() not in
stop) and username_filter(w))]
        words = [lem.lemmatize(w) for w in words if len(w) > 2]
        corpus.append(words)
    return corpus

corpus = preprocess_tweets(tsa)

dic = gensim.corpora.Dictionary(corpus)
bow_corpus = [dic.doc2bow(doc) for doc in corpus]

lda_model = gensim.models.LdaMulticore(bow_corpus, num_topics = 4,
id2word = dic, passes = 10, workers = 2)
lda_model.show_topics()
```

Output:

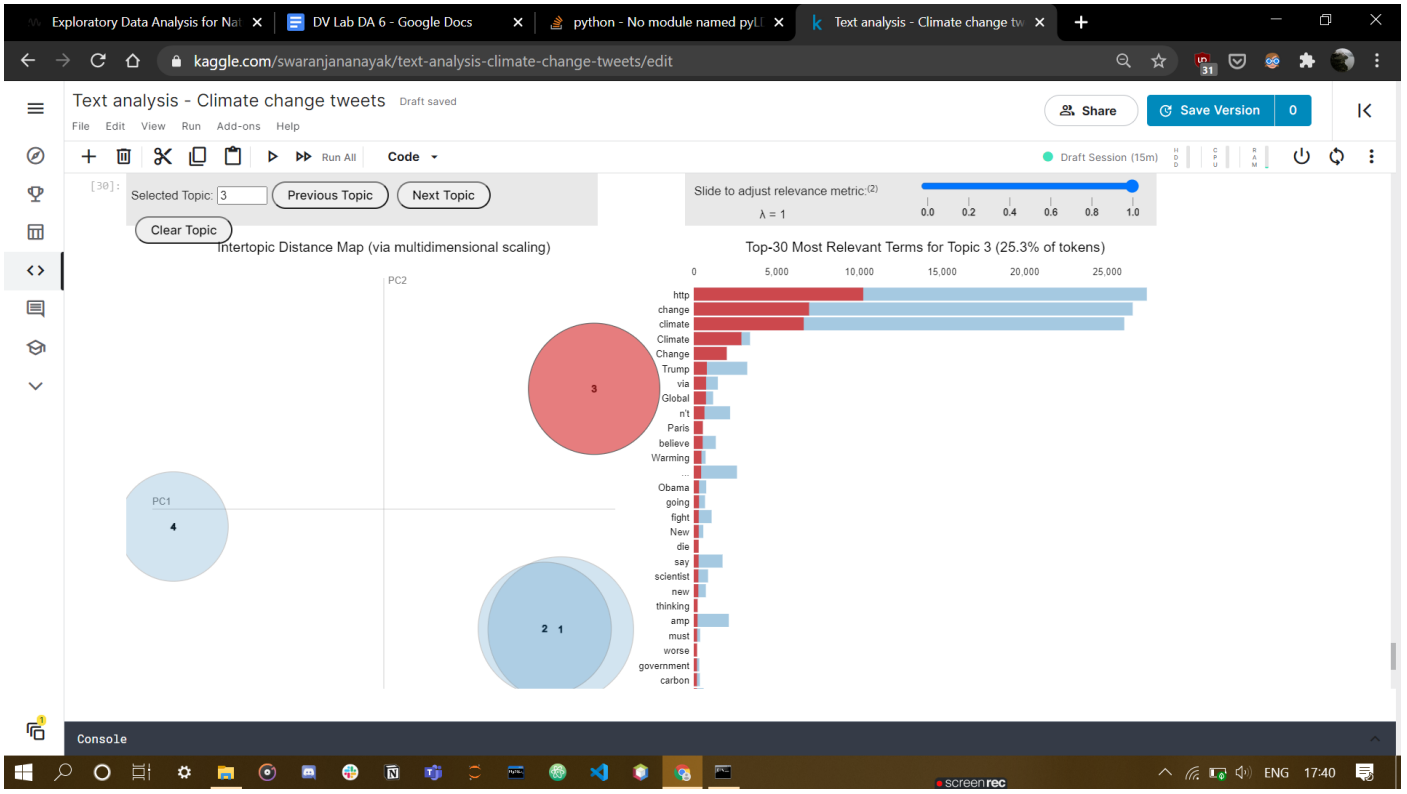
```
[(0,
    '0.072*"change" + 0.071*"climate" + 0.032*"http" + 0.007*"amp" +
0.007*"n\'t" + 0.006*"people" + 0.004*"real" + 0.004*"believe" +
0.004*"world" + 0.004*"say"'),
 (1,
    '0.076*"http" + 0.075*"change" + 0.074*"climate" + 0.014*"Trump" +
0.006*"via" + 0.005*"fight" + 0.004*"say" + 0.004*"EPA" + 0.003*"n\'t" +
0.003*"Paris"'),
 (2,
    '0.074*"global" + 0.072*"warming" + 0.032*"http" + 0.011*"..." +
0.005*"n\'t" + 0.005*"real" + 0.004*"weather" + 0.004*"Global" + 0.003*"like"
+ 0.003*"year"'),
 (3,
    '0.060*"http" + 0.035*"Climate" + 0.024*"Change" + 0.024*"change" +
0.022*"climate" + 0.009*"Global" + 0.008*"Warming" + 0.004*"New" +
0.004*"via" + 0.003*"tackle"')]
```

Visualizing results of LDA

Code:

```
import pyLDAvis
from pyLDAvis import gensim_models

pyLDAvis.enable_notebook()
vis = gensim_models.prepare(lda_model, bow_corpus, dic)
vis
```

Output:



Wordcloud visualization

Code:

```
from wordcloud import WordCloud, STOPWORDS
stopwords = set(STOPWORDS)

def show_wordcloud(data):
    wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        max_words=100,
        max_font_size=30,
        scale=3,
        random_state=1)

    wordcloud=wordcloud.generate(str(data))
```

```
    fig = plt.figure(1, figsize=(12, 12))
    plt.axis('off')

    plt.imshow(wordcloud)
    plt.show()
    fig.savefig("wordcloud.png", bbox_inches = "tight")

show_wordcloud(corpus)
```

Output: