CIS664: Natural Language Processing
SUID: 387781563
NetID: cchikkan

Homework 1 : Corpus Statistics and Python Programming

1. Data - Please check out the web site for more information about this dataset. Please describe the characteristics of this corpus briefly, e.g., who created this dataset, the format of .json file and the fields, the related policy (if any), etc. (no more than 200 words).

Solution:
The dataset we are analyzing for this assignment is the free dataset from news/message boards/blogs about the pandemic we are facing currently i.e. CoronaVirus which contains four months data. The time frame of the data is Dec/2019 - March/2020. Here, we are analyzing two json files, they are:

- 16119_webhose_2020_01_db21c91a1ab47385bb13773ed8238c31_0000002.json
- 16119_webhose_2020_02_db21c91a1ab47385bb13773ed8238c31_0000002.json

The dataset was extracted from https://ieee-dataport.org/open-access/free-dataset-newsmessage-boardsblogs-about-coronavirus-4-month-data-52m-posts. The citation author of the dataset we are analyzing is Rav Geva. This data set was extracted by the author using the following links:
Online discussions that mention “CoronaVirus”
Blog posts that mention “CoronaVirus”
News articles that mention “CoronaVirus”

The most common keywords we can find in this data set are : covid-19, coronavirus, news, blogs, discussions, web data.

The sample data taken from the json file is shown below:

- {
- "organizations":[],
- "uuid":"2b50b3f00e04fc17912154a7b88f3359db2b1ae8",
- "thread":{
- "social":{
- "gplus":{
- "shares":0
- },
- "pinterest":{
- "shares":1
- },
- "vk":{
- "shares":0
- },
- "linkedin":{

- "shares":0
- },

Each json file contains the data related to Coronavirus which includes the following:

- Facebook information represented as a key value pair such as the number of likes, shares and comments for the data.
- Title of the data.
- Date and time of which the data has been published in data-time format.
- Author name.
- The url from which the data was found published.
- Country name where data for published.
- Data related to covid19.

2. Data Pre-processing (20%)

You will code in Python to extract the content of the following fields and save it to a CSV file: "facebook":{…}, "title", "published", "replies_count", "author", "url", "country", "text".

Note: regarding the "text" field, you will decide how to process the words, i.e. decide on tokenization and whether to use all lower case, use or modify the stop word list, or lemmatization. Briefly state why you chose the processing options that you did.

Solution:
First step in Data Pre-processing is **data extraction.** Under data extraction, we first open the two json files and read the data using the json.loads() in python and then loop through the file content line by line.

After reading the json files, we convert it to csv file using csv_writer by specifying the column names and parsing the data in the json file.

CIS664: Natural Language Processing
SUID: 387781563
NetID: cchikkan

```
import csv
import json
import pandas as pd
import nltk
```

```
file1 = [json.loads(line) for line in open('16119_webhose_2020_01_db21c91a1ab47385bb13773ed8238c31_0000002.json', 'r')]
file2 = [json.loads(line) for line in open('16119_webhose_2020_02_db21c91a1ab47385bb13773ed8238c31_0000002.json', 'r')]
```

```
data_to_file = open('covid19.csv', 'w', newline='')
csv_writer = csv.writer(data_to_file)
csv_writer.writerow(["facebook","title","published","replies_count", "author", "url", "country", "text"]);
```

```
for i in range(0, len(file1)):
        data = data1[i]
        facebook = data['thread']['social']['facebook']
        title = data['title']
        published = data['thread']['published']
        replies_count = data['thread']['replies_count']
        author = data['author']
        url = data['url']
        country = data['thread']['country']
        text = data['text']
        csv_writer.writerow([facebook,title,published,replies_count,author,url,country,text])
for i in range(0, len(file2)):
        data = data2[i]
        facebook = data['thread']['social']['facebook']
        title = data['title']
        published = data['thread']['published']
        replies_count = data['thread']['replies_count']
        author = data['author']
        url = data['url']
        country = data['thread']['country']
        text = data['text']
        csv_writer.writerow([facebook,title,published,replies_count,author,url,country,text])
data_to_file.close()
```

Code for data extraction

The contents of the file are first stored in a string variable and then tokenized to get the words of the file in the python list. This tokenization can be achieved by the nltk.word tokenize which is provided by python library.

CIS664: Natural Language Processing
SUID: 387781563
NetID: cchikkan

```
get_text = open("covid19.csv", 'r')
df = pd.read_csv("covid19.csv")
columns = df.text
print(len(columns))
```

```
106017
```

```
#tokenization
df['tokenized'] = df.apply(lambda row: nltk.word_tokenize(row['text']), axis=1)
print(df['tokenized'])
```

```
0         [Bengaluru, :, Isolation, wards, in, hospitals...
1         [The, government, making, sure, that, the, new...
2         [Apart, from, more, people, falling, sick, (, ...
3         [Asian, stock, markets, are, mostly, higher, o...
4         [Cash, flow, was, also, “, very, strong, ,, ”,...
                                ...
106012    [Corona, virus, is, the, most, effective, weap...
106013    [Hope, they, do, n't, bring, coronavirus, to, ...
106014    [Wall, Street, Ends, In, The, Red, As, Virus, ...
106015    [Since, mid-January, ,, the, name, on, the, li...
106016    [PH, shares, fall, back, to, 7,200, amid, ling...
Name: tokenized, Length: 106017, dtype: object
```

Code for tokenization

To get more meaningful tokens we convert all upper-case letters to their lower case. This is achieved by using lower() function. I converted the letters to lowers case to avoid duplication of words. For example, ‘are’ and ‘Are’ are same words which will be considered as two separate words if not in the same character case.

```
#tolowercase
df['lower_case_words'] = df['tokenized'].apply(lambda x: [word.lower() for word in x])
print(df['lower_case_words'])
```

```
0         [bengaluru, :, isolation, wards, in, hospitals...
1         [the, government, making, sure, that, the, new...
2         [apart, from, more, people, falling, sick, (, ...
3         [asian, stock, markets, are, mostly, higher, o...
4         [cash, flow, was, also, “, very, strong, ,, ”,...
                                ...
106012    [corona, virus, is, the, most, effective, weap...
106013    [hope, they, do, n't, bring, coronavirus, to, ...
106014    [wall, street, ends, in, the, red, as, virus, ...
106015    [since, mid-january, ,, the, name, on, the, li...
106016    [ph, shares, fall, back, to, 7,200, amid, ling...
Name: lower_case_words, Length: 106017, dtype: object
```

Code to convert letters to lowercase

CIS664: Natural Language Processing
SUID: 387781563
NetID: cchikkan

I have used alphafiler(), a built-in unction in python which helps to eliminate full stops, comma's, apostrophes etc. It retains tokens that only have alphabets giving more meaningful data.

```
#remove_punctuation,numeric
df['is_alpha'] = df['lower_case_words'].apply(lambda x: [word for word in x if word.isalpha()])
print(df['is_alpha'])
```

```
0         [bengaluru, isolation, wards, in, hospitals, a...
1         [the, government, making, sure, that, the, new...
2         [apart, from, more, people, falling, sick, as,...
3         [asian, stock, markets, are, mostly, higher, o...
4         [cash, flow, was, also, very, strong, at, more...
                                ...
106012    [corona, virus, is, the, most, effective, weap...
106013    [hope, they, do, bring, coronavirus, to, these...
106014    [wall, street, ends, in, the, red, as, virus, ...
106015    [since, the, name, on, the, lips, of, increasi...
106016    [ph, shares, fall, back, to, amid, lingering, ...
Name: is_alpha, Length: 106017, dtype: object
```

Code to remove alpha-numeric character

After tokenizing and cleaning the data, we need to remove the stop words which are the most commonly used words that have no relevance to the sentence and are redundant. So, in the next step, I have removed the stop words using nltk.stopwords()

```
#Stopwords
stop_words = nltk.corpus.stopwords.words('english')
df['stopwords_removed'] = df['is_alpha'].apply(lambda x: [word for word in x if word not in stop_words])
```

```
print(df['stopwords_removed'])
```

```
0         [bengaluru, isolation, wards, hospitals, acros...
1         [government, making, sure, new, coronavirus, m...
2         [apart, people, falling, sick, bad, fundamenta...
3         [asian, stock, markets, mostly, higher, friday...
4         [cash, flow, also, strong, us, billion, fourth...
                                ...
106012    [corona, virus, effective, weapon, created, xi...
106013    [hope, bring, coronavirus, poor, cambodian, pe...
106014    [wall, street, ends, red, virus, fears, spook,...
106015    [since, name, lips, increasing, numbers, tv, n...
106016    [ph, shares, fall, back, amid, lingering, jitt...
Name: stopwords_removed, Length: 106017, dtype: object
```

CIS664: Natural Language Processing
SUID: 387781563
NetID: cchikkan

Lemmatization is the process of grouping together the different inflected forms of a word so they can be analyzed as a single item. It is similar to stemming but it brings context to the words. So, it links words with similar meaning to one word. NLTK provides us with WordNetLemmatizer() function which I have used in my code. Lemmatization offers better precision and it normalizes the words. For example, the word “studying” will become “Study” after applying lemmatization.

```
#lemmetaziation
lemmetizer = nltk.WordNetLemmatizer()
lemmatized_words = [lemmetizer.lemmatize(t) for t in stopped_text]
lemmatized_words[:30]
```

```
['bengaluru',
 'isolation',
 'ward',
 'hospital',
 'across',
 'karnataka',
 'helpline',
 'take',
 'call',
 'query',
 'ready',
 'prevent',
 'spread',
 'virus',
 'first',
 'case',
 'india',
 'reported',
 'kerala',
 'yesterday',
 'chief',
 'secretary',
 'state',
 'government',
 'thursday',
 'held',
 'meeting',
 'additional',
 'chief',
 'secretary']
```

Code to implement lemmatization

CIS664: Natural Language Processing
SUID: 387781563
NetID: cchikkan

3. Data Analysis:

To get a rough understanding of what these texts were about, you will also perform the following three tasks on the pre-processed data:

• list the top 50 words by frequency
• list the top 50 bigrams by frequencies, and
• list the top 50 bigrams by their Mutual Information scores (using min frequency 5)

Solution:

After performing the above pre-processing techniques, the frequency and normalized frequency can be found out by using FreqDist().
Here we are displaying top 50 words by frequencies

```
# list the top 50 words by frequency
from nltk import FreqDist
fdist = FreqDist(lemmatized_words)
lemmetized_text_topKeys = fdist.most_common(50)
```

```
print(lemmetized_text_topKeys)
```

```
[('said', 341836), ('china', 320330), ('coronavirus', 296230), ('virus', 200704), ('people', 177961), ('health', 1754
88), ('new', 174300), ('case', 167723), ('outbreak', 142623), ('year', 120429), ('also', 120408), ('chinese', 11478
7), ('one', 96585), ('country', 94624), ('day', 91480), ('wuhan', 91011), ('world', 89887), ('company', 85632), ('wou
ld', 80589), ('week', 80187), ('time', 80187), ('market', 76452), ('government', 75976), ('first', 75636), ('two', 74
386), ('say', 72486), ('last', 69719), ('spread', 68402), ('million', 68353), ('disease', 67784), ('confirmed', 6773
6), ('u', 66846), ('could', 66658), ('official', 65943), ('number', 64840), ('news', 64468), ('death', 63549), ('cit
y', 63450), ('ship', 61488), ('text', 61401), ('global', 61195), ('reported', 60271), ('state', 60113), ('public', 60
062), ('passenger', 59543), ('february', 57921), ('hospital', 57269), ('patient', 56497), ('including', 53353), ('ma
y', 52876)]
```

Code to list the top 50 words by frequency

Next, to calculate the top 50 bigrams we use the function nltk.bigrams(). Top 50 bigrams can be found by applying function FreqDist() on these bigrams. By using the nltk.collocations.BigramAssocMeasures() we can find the raw frequency of the bigrams and this raw frequency can be given to the score_ngrams() function to find a bigram score which is calculated as the ratio of the count of bigram over the count of total bigrams.
Here we are displaying top 50 bigrams by frequencies

CIS664: Natural Language Processing
SUID: 387781563
NetID: cchikkan

```
#list the top 50 bigrams by frequencies
from nltk.collocations import *
bigram_measures_plot = nltk.collocations.BigramAssocMeasures()
finder = BigramCollocationFinder.from_words(lemmatized_words)
finder.apply_freq_filter(5)
scored = finder.score_ngrams(bigram_measures_plot.raw_freq)
for bigramscore in scored[:50]:
    print(bigramscore)
```

```
(('coronavirus', 'outbreak'), 0.001467440730029942)
(('location', 'text'), 0.0014113983501381027)
(('hong', 'kong'), 0.0013410555008945522)
(('novel', 'coronavirus'), 0.0012370872237158984)
(('confirmed', 'case'), 0.0009699518795647968)
(('cruise', 'ship'), 0.000888368346078573)
(('public', 'health'), 0.0008444362735656597)
(('world', 'health'), 0.0008098445977013863)
(('new', 'coronavirus'), 0.0007389219997003343)
(('per', 'cent'), 0.0007223347435829221)
(('hubei', 'province'), 0.0007103210609969128)
(('health', 'organization'), 0.0006263219076877648)
(('mainland', 'china'), 0.0006245504531509538)
(('text', 'location'), 0.0006220060002708071)
(('united', 'state'), 0.0006088328201697943)
(('last', 'year'), 0.0005928253128098837)
(('new', 'case'), 0.0005921167309951593)
(('rating', 'green'), 0.0005652872468285488)
(('green', 'name'), 0.0005595219675178366)
(('threshold', 'type'), 0.0005537244799428187)
(('type', 'textpattern'), 0.0005537244799428187)
(('textpattern', 'score'), 0.0005456079973377937)
(('death', 'toll'), 0.0005308888205501094)
(('diamond', 'princess'), 0.0005248336668606463)
(('last', 'week'), 0.0005227723379450844)
(('new', 'virus'), 0.0004777745184445778887)
(('health', 'official'), 0.0004718510720778404)
(('tested', 'positive'), 0.0004703694919197839)
(('new', 'year'), 0.0004455047118758183)
(('coronavirus', 'case'), 0.00038643475513924726)
```

CIS664: Natural Language Processing
SUID: 387781563
NetID: cchikkan

```
(('score', 'rating'), 0.00037912347914186365)
(('new', 'york'), 0.0003727784510736496)
(('count', 'rule'), 0.0003603460610516668)
(('pattern', 'count'), 0.0003549350726483168)
(('social', 'medium'), 0.0003523584115038644)
(('case', 'coronavirus'), 0.0003460133834356504)
(('face', 'mask'), 0.00034485388592064685)
(('infectious', 'disease'), 0.000338863148759795)
(('disease', 'control'), 0.00033232487110574705)
(('two', 'week'), 0.00032836325459615153)
(('spread', 'virus'), 0.0003180888182826476)
(('health', 'emergency'), 0.0003107453340209583)
(('lunar', 'new'), 0.00030291872579468 42)
(('official', 'said'), 0.0003013083125794014)
(('supply', 'chain'), 0.0002990537340780056)
(('health', 'commission'), 0.00029692798863383235)
(('around', 'world'), 0.0002958651159117457)
(('associated', 'press'), 0.00029270870600979157)
(('killed', 'people'), 0.00028159685482434066)
(('prime', 'minister'), 0.00027686223997140934)
```

Code to list the top 50 bigrams by frequency

The top 50 bigrams by the Mutual Information Scores using minimum frequency of 5 can be found out by using the function score_ngrams() and passing nltk.collocations.BigramAssocMeasures() object as a parameter. This object uses pmi variable to calculate the mutual information score.
Here we are displaying top 50 bigrams by their Mutual Information scores

CIS664: Natural Language Processing
SUID: 387781563
NetID: cchikkan

```
# list the top 50 bigrams by their Mutual Information scores
finder2 = BigramCollocationFinder.from_words(lemmatized_words)
finder2.apply_freq_filter(5)
scored = finder2.score_ngrams(bigram_measures_plot.pmi)
for bigramscore in scored[:50]:
    print(bigramscore)
```

```
(('aline', 'oyamada'), 22.56606574782782)
(('andreo', 'calonzo'), 22.56606574782782)
(('artha', 'ardhana'), 22.56606574782782)
(('asaduddin', 'owaisi'), 22.56606574782782)
(('babulal', 'marandi'), 22.56606574782782)
(('bichigt', 'bulgan'), 22.56606574782782)
(('campeões', 'clássica'), 22.56606574782782)
(('celestino', 'gallares'), 22.56606574782782)
(('chiwuike', 'onyeanu'), 22.56606574782782)
(('clássica', 'franquia'), 22.56606574782782)
(('combizym', 'hirudoid'), 22.56606574782782)
(('diretoras', 'esnobadas'), 22.56606574782782)
(('ekstra', 'bladet'), 22.56606574782782)
(('embedvideo', 'pagetitle'), 22.56606574782782)
(('eren', 'sengezer'), 22.56606574782782)
(('ezzeldin', 'bahader'), 22.56606574782782)
(('filou', 'oostende'), 22.56606574782782)
(('foodgrain', 'prodution'), 22.56606574782782)
(('galtung', 'døsvig'), 22.56606574782782)
(('ghanti', 'bajao'), 22.56606574782782)
(('hatidze', 'muratova'), 22.56606574782782)
(('heikki', 'kovalainen'), 22.56606574782782)
(('intramuscularly', 'intradermally'), 22.56606574782782)
(('intravascular', 'coagulopathy'), 22.56606574782782)
(('jeoffrey', 'lambujon'), 22.56606574782782)
(('jirasat', 'wittaya'), 22.56606574782782)
(('joventut', 'badalona'), 22.56606574782782)
(('jìnpíng', '习近平'), 22.56606574782782)
(('kalagayang', 'epidemiko'), 22.56606574782782)
(('kalidou', 'koulibaly'), 22.56606574782782)
(('kalyeena', 'makortoff'), 22.56606574782782)
(('karasyov', 'cannonball'), 22.56606574782782)
```

CIS664: Natural Language Processing
SUID: 387781563
NetID: cchikkan

```
(('kktc', 'cumhurbaşkani'), 22.56606574782782)
(('krister', 'magnusson'), 22.56606574782782)
(('labto', 'ofdlr'), 22.56606574782782)
(('lagendijk', 'cgco'), 22.56606574782782)
(('lavoropiu', 'fortitudo'), 22.56606574782782)
(('leptopilina', 'boulardi'), 22.56606574782782)
(('magagandang', 'mapuntahan'), 22.56606574782782)
(('movicol', 'combizym'), 22.56606574782782)
(('myeonguet', 'walehet'), 22.56606574782782)
(('narumon', 'pinyosinwat'), 22.56606574782782)
(('natnicha', 'chuwiruch'), 22.56606574782782)
(('nayla', 'razzouk'), 22.56606574782782)
(('nenad', 'lalovic'), 22.56606574782782)
(('nguessan', 'myeonguet'), 22.56606574782782)
(('ofdlr', 'owedi'), 22.56606574782782)
(('ofri', 'ilany'), 22.56606574782782)
(('ognen', 'stojanovski'), 22.56606574782782)
(('omotayo', 'okusanya'), 22.56606574782782)
```

code to list the top 50 bigrams by their mutual information score

I have also extracted the summary of the data in graphical representation of the average length of text and the countries. For graphical representation, I have used python's seaborn, learn.feature_extraction.text and collections modules.

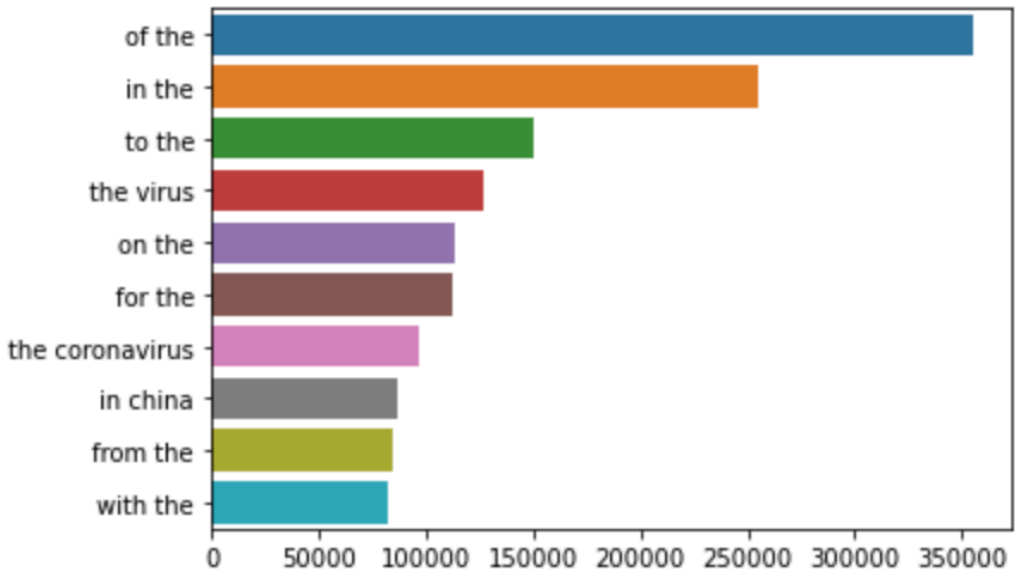
I have plotted the bar graph of top ngrams data which gave me the graphical representation of the top words used in the data i.e. in "text" column of the data.

CIS664: Natural Language Processing
SUID: 387781563
NetID: cchikkan

```
#Plotting_graphs
import seaborn as sns
import numpy as np
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import CountVectorizer
from collections import  Counter

def plot_top_ngrams_barchart(text, n=2):
    stop=set(stopwords.words('english'))

    new= text.str.split()
    new=new.values.tolist()
    corpus=[word for i in new for word in i]

    def _get_top_ngram(corpus, n=None):
        vec = CountVectorizer(ngram_range=(n, n)).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[:10]

    top_n_bigrams=_get_top_ngram(text,n)[:10]
    x,y=map(list,zip(*top_n_bigrams))
    sns.barplot(x=y,y=x)
```

```
plot_top_ngrams_barchart(df['text'], 2)
```

Code to plot top ngrams of “text” column



Graphical representation of top ngrams of “text” column

CIS664: Natural Language Processing
SUID: 387781563
NetID: cchikkan

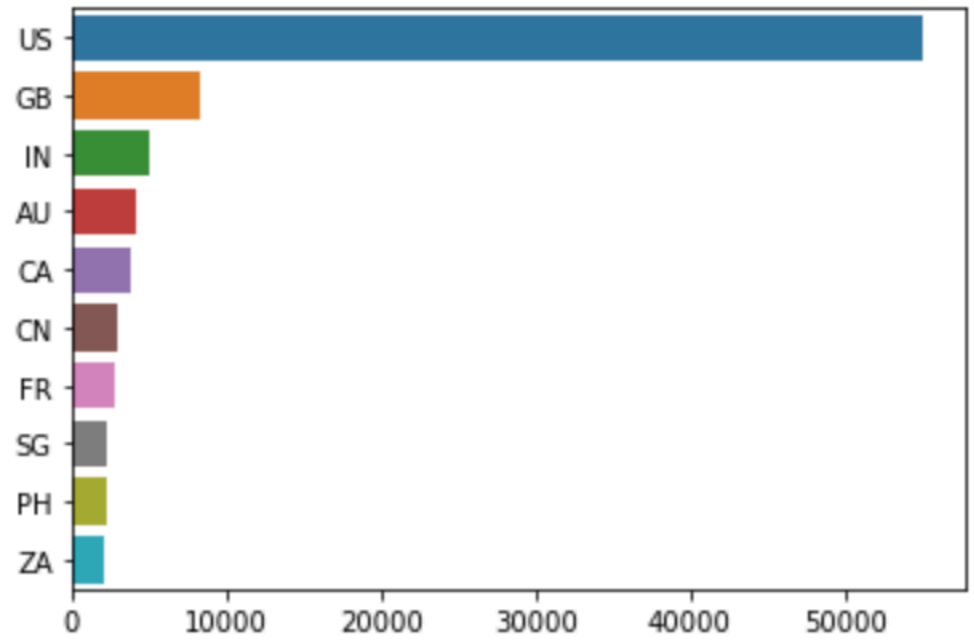
To get the graphical representation of number of countries, I have again used the seaborn and collections modules in python and have plotted bar graph for non stop words.

```
import seaborn as sns
from nltk.corpus import stopwords
from collections import  Counter

def plot_top_non_stopwords_barchart(text):
    stop=set(stopwords.words('english'))
    new= text.str.split()
    new=new.values.tolist()
    corpus=[word for i in new for word in i]

    counter=Counter(corpus)
    most=counter.most_common()
    x, y=[], []
    for word,count in most[:10]:
        if (word not in stop):
            x.append(word)
            y.append(count)

    sns.barplot(x=y,y=x)
```

```
df['country'] = df['country'].astype(str)
```

```
plot_top_non_stopwords_barchart(df['country'])
```

Code for plotting bar graph for “country” column



Graphical representation of “country” column

CIS664: Natural Language Processing
SUID: 387781563
NetID: cchikkan

4. Interpretation of Result:

Solution:

I used the two json files for our analysis.

- 16119_webhose_2020_01_db21c91a1ab47385bb13773ed8238c31_0000002.json
- 16119_webhose_2020_02_db21c91a1ab47385bb13773ed8238c31_0000002.json

At first, I have processed the data and removed all the non-informative data. I decided to remove lower-case letter in order to avoid the duplication of the data. Then I have took out all the alpha-numeric characters, punctuation, and numerical value to make the data ready for the analysis. Hence, there won't be any common words like is, am, and, the etc. which makes no sense for the analysis. The result can be evaluated, by looking at the top list of words which are mainly used for dataset that we have used for our analysis. After removing all the redundancy, I calculated the Bigrams. Bigrams are the occurrence of two words together. In general, they are two-word frequency. For example, "tested positive", "coronavirus outbreak" "public health" etc.

We calculate the bigram probability to predict how likely it is that the second word follows the first.

We calculate the bigrams by using the probability formula.

$$P(W_i \mid Wi\text{-}1) = Count(W_{i-1}, W_i)/Count(W_{i-1})$$

Let's take the example of "coronavirus outbreak", we calculated the bigram frequency of the "coronavirus outbreak" by using the above formula. We divide the number of occurrences "coronavirus" and "outbreak" together by the total number of occurrences of the "coronavirus". There are other bigrams like "coronavirus case", "coronavirus pandemic" Among them "coronavirus outbreak" are used together more than the other bigrams.

For calculation of mutual information score we used the formula.

$$PMI\ (X,Y) = \log2\ (P(X,Y) / P(X)\ P(Y)\ )$$

P(X) is estimated by the number of observations of x in corpus

P(Y) is estimated by the number of observations of y in corpus

P(X,Y) is the number of times that X is followed by Y in corpus which are normalized by the size of the corpus. The Mutual Information Scores is measure of strength of the word X and word Y.

This data analysis helps me figure out that the data set we were analyzing was based on coronavirus by going through the list of bigrams we have generated. Also, this analysis will give an idea of the thought process of people regarding the issue. There are certain words which give positive review and certain words for negative review. By having the list of tokens in the data, it helps us get the proper review of the people.

CIS664: Natural Language Processing
SUID: 387781563
NetID: cchikkan

Additional Analysis suggestion:
In the result data what we have obtained after performing the analysis, there are few steps I think if we follow will help us better analyze the data such as mentioned below:

1. In the top 50 bigrams what we have obtained by their mutual information scores, there are data included in languages other than English. If we can sort out the data based on the language i.e. if we can obtain the data separately in English, Chinese, Japanese etc. it will be helpful to analyze the data in a better way. So I would recommend the language based analysis for future work.

2. As the data we are analyzing is based on Coronavirus, we would expect to see the tokens mostly and closely related to coronavirus. But while fetching the top 50 bigrams by frequency, I noticed data that are irrelevant to coronavirus such as “lunar new”, “diamond princess”, “cruise ship”. As these bigrams what we have obtained makes no sense to the analysis what we are performing, we can have the analysis that will filter out the irrelevant words in a more précised way that will help fetch the better data analysis.