IST 664 – Natural Language Processing
Homework 1

Abstract
Corpus is a collection of written text. For the purpose of this report we are using the corpus text obtained from news/message boards/blogs about Corona Virus. Analysis of this corpus can lead to interesting findings about the impact of COVID-19. There are various ways to analyze a text corpus. The problem here is to understand people's views and their concerns, and things they care about.

Data
The given data is the free dataset from news/message boards/blogs about CoronaVirus. This dataset has four months data – 5.2 M posts. The time frame of the data is December 2019 – March 2020. The posts are in English mentioning at least one of the following: "Covid" or CoronaVirus or "Corona Virus".

Data Preprocessing
We first read the data from both the JSON files and extract the required fields 'facebook', 'title', 'published', 'replies_count', 'author', 'url', 'country', 'text'. The required data is stored into a CSV file for further analysis.

For the purpose of corpus analysis, the text data present in text field would be the only one useful. For this reason, we extract the data and change it to a string. The following lines of code are used for the same:

```
#reading the data from csv

import pandas as pd

covid = pd.read_csv('test.csv')
```

```
#changing text data to string

str1 = ''.join(list(covid['text']))
```

Tokenization
The assembly language is a language of numbers unlike human languages which contains words. Thus, the only thing computer can do with the human language is counting. To initiate this process of counting the text is divided or is grouped into words. This process of dividing/grouping the text is known as tokenization.

There are plenty tokenizers provided by NLTK which are available for us to use. For tokenizing the text, we can use the RegexpTokenzier available in the NLTK's tokenize library, which is a Regular Expression tokenizer. The text data contains symbols like *, :, etc. which are of no use

thus, we use RegularExpression tokenizer to consider only the words and numbers in the text. The following lines of code were used for the same:

```
#tokenizing using RegexTokenizer by consideringing only alphabets and numbers

import nltk

tokenizer = nltk.RegexpTokenizer('\w+')
r_tokens = tokenizer.tokenize(str1)
```

On evaluating the tokens, we see that the tokenizer is considering same words written in different cases as different tokens. For example, “The” and “the” are being considered as different tokens. Thus, to resolve this problem we can change all of the tokens to lowercases. These will affect the frequencies of the tokens and also further analysis of those tokens. This is implemented using the following line of code:

```
#changing all the tokens to lower case

tokens = [word.lower( ) for word in r_tokens]
```

Removing stop words

In computing, stop words are words which are filtered out before processing of natural language data. Stop words are generally the most common words in a language; there is no single universal list of stop words used by all natural language processing tools, and indeed not all tools use such a list. Some tools avoid removing stop words to support phrase search. (Reference Wikipedia)

To get a better insight of the reviews I have removed all the stop words from tokens predefined as stopwords in English language. Line of code to remove the stop words:

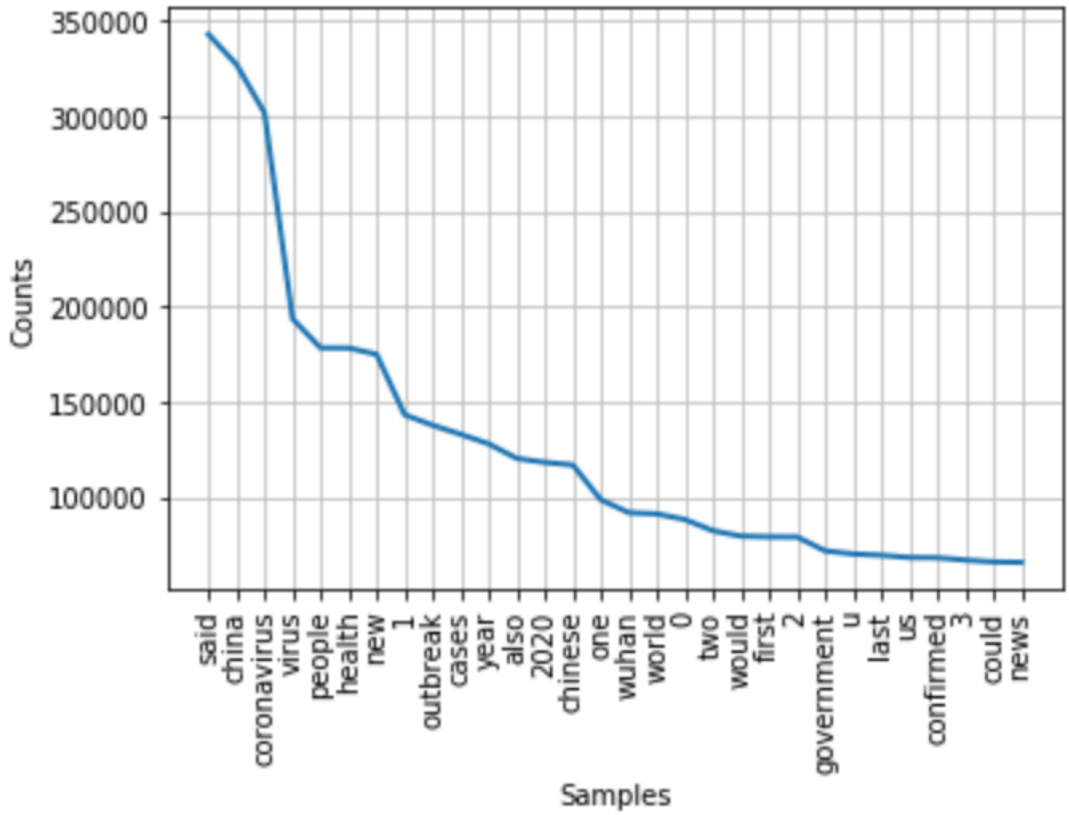
```
#removing stop words

from nltk.corpus import stopwords

stop_words = set(stopwords.words('english'))
covid_tokens = [token for token in tokens if token not in stop_words]
```

Data Analysis

We have a total of 106017 different data regarding Corona Virus from 149 countries. ‘said’, ‘china’ and ‘coronavirus’ are the most frequent words in the corpus.



The aim to clean and process the review text is to analyze it and derive meaningful insights from it. The analysis has been performed in three forms:

i) listing out the most commonly occurring words
ii) listing out the most commonly occurring bigrams and
iii) listing out bigrams by their mutual information score

in descending order, occurring at-least a specific number of times.

The top 50 words with highest frequencies are obtained with the help of following block of code:

```
#top 50 words by frequency

freq_dist = nltk.FreqDist(covid_tokens)
most_common_50 = freq_dist.most_common(50)

for w in most_common_50:
    print(w)
```

The FreqDist function generates a list of tuples as (word, frequency) and the most_common(50) lists out the 50 most frequent tokens.

The 50 most frequent bigrams can be obtained using the following block of code:

```
#top 50 bigrams by frequencies

bigrams = list(nltk.bigrams(tokens))
bigram_freq_dist = nltk.FreqDist(bigrams)
bigram_most_common_50 = bigram_freq_dist.most_common(50)

for w in bigram_most_common_50:
    print(w)
```

The FreqDist function generates a list of tuples as (word1_of_bigram, word2_of_bigram, frequency) and the most_common(50) lists out the 50 most frequent tokens.

For listing out bigrams according to their mutual information score in descending order the following code is used:

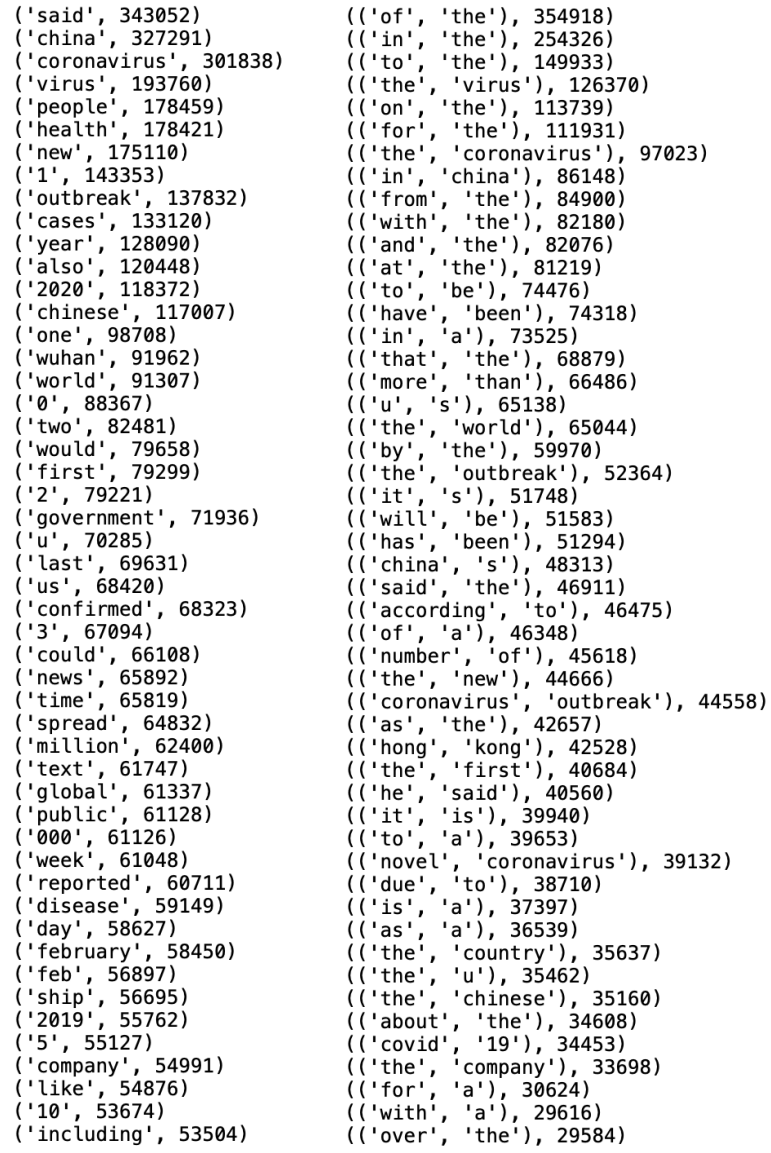
```
#top 50 bigrams by their mutual information scores (using min frequency 5)

bigram_measures = nltk.collocations.BigramAssocMeasures()
finder = nltk.BigramCollocationFinder.from_words(tokens)
finder.apply_freq_filter(5)
scored = finder.score_ngrams(bigram_measures.pmi)

for bscore in scored[:50]:
    print(bscore)
```

Results

The following are the results obtained for top 50 tokens according to their frequencies, top 50 bigrams according to their frequencies and top 50 bigrams according to their mutual information score with minimum occurrence as 5.



Top 50 tokens | Top 50 bigrams | Top 50 bigrams (pmi)

As we can see in the above images most of the reviews are about china, coronavirus, health, outbreak. Also, from bigrams we can detect that most of the people spoke about virus and its origin. Furthermore, we can also see that the corpus doesn't follow the Zipf's law as frequency of china is not twice the frequency of coronavirus. For further evaluation I suggest performing sentiment analysis on tokens and bigrams and collectively determine people's view on COVID-19 and its spread. We should tokenize each post separately and generate a sentiment for each post as positive, neutral or negative.