IST 664 – Natural Language Processing
Homework 2

In this assignment, we worked on the texts related to COVID-19 to gain some understanding of the sentiment in these texts. We extracted those sentences that contain adjective phrases or exclamation marks.

Data Preprocessing

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']))
```

The raw text is split into sentences using a sentence segmenter, and each sentence is further subdivided into words using a word tokenizer. The following lines of code were used for the same:

```
#extracting individual sentences using sentence segmenter
import nltk
sentences = nltk.sent_tokenize(str1)
```

```
#tokenizing each sentence using word tokenizer
tokens = [nltk.word_tokenize(sent) for sent in sentences]
```

POS tagging

Next, each sentence is tagged with part-of-speech tags, which will be helpful in entity detection. POS tagging is the process of marking up a word in a corpus to a corresponding part of a speech tag, based on its context and definition. This task is not straightforward, as a particular word may have a different part of speech based on the context in which the word is used. This is implemented using the following line of code:

```
#POS tagging the tokens
tags = [nltk.pos_tag(tok) for tok in tokens]
```

For the analysis we also consider those tokens that contain exclamation marks. It is done using the following code.

```
#sentences with exclamation

excla =[ ]
for i in range(0,len(tokens)):
    if '!' in tokens[i][0]:
        excla.append(tokens[i])
```

Chunking

Chunking is a process of extracting phrases from unstructured text, which means analyzing a sentence to identify the constituents (Noun Groups, Verbs, verb groups, etc.). However, it does not specify their internal structure, nor their role in the main sentence. It works on top of POS tagging. It uses POS-tags as input and provides chunks as output.

We write a grammar for adjectives (JJ), to find the chunk structure for a given sentence, the RegexpParser chunker begins with a flat structure in which no tokens are chunked. The chunking rules are applied in turn, successively updating the chunk structure. Once all of the rules have been invoked, the resulting chunk structure is returned.

Defining a grammar and chunking is implemented using the code below.

```
#writing grammar for adjectives

grammar = r"""
    AP: {<JJ+>}
"""
cp = nltk.RegexpParser(grammar)
adj = cp.parse(tags[0])
```

Data Analysis

The aim to process the review text is to extract sentences that contain adjectives and exclamation marks. The analysis has been performed in three forms.

i) Average length of a sentence
ii) Top 50 adjectives
iii) Top 50 nouns
iv) Top 50 verbs

The average length of a sentence is obtained with the help of following block of code:

```
#average length of a sentence

sum1 = 0
for i in range(0, len(trees)):
    sum1 += len(trees[i])

sum1 / len(trees)
```

Average length of a sentence is calculated by dividing the total number of tokens by length of tree.

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

```
import itertools

all_tags = list(itertools.chain.from_iterable(tags))

cfd = nltk.ConditionalFreqDist(((tag, word) for (word, tag) in all_tags))
```

```
#top 50 adjectives

list(cfd['JJ'])[0:50]
```

Here we are considering all the tags that contain at least one adjective.

ConditionalFreqDist collects how frequently a word occurs in a text based on certain condition.

For listing out top 50 nouns the following code is used:

```
#top 50 nouns

list(cfd['NN'])[0:50]
```

For listing top 50 verbs the following code is used:

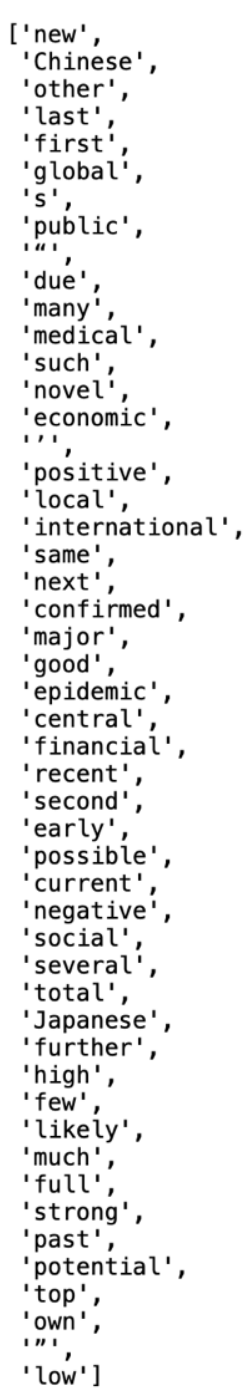
```
#top 50 verbs

list(cfd['VB'])[0:50]
```

Results

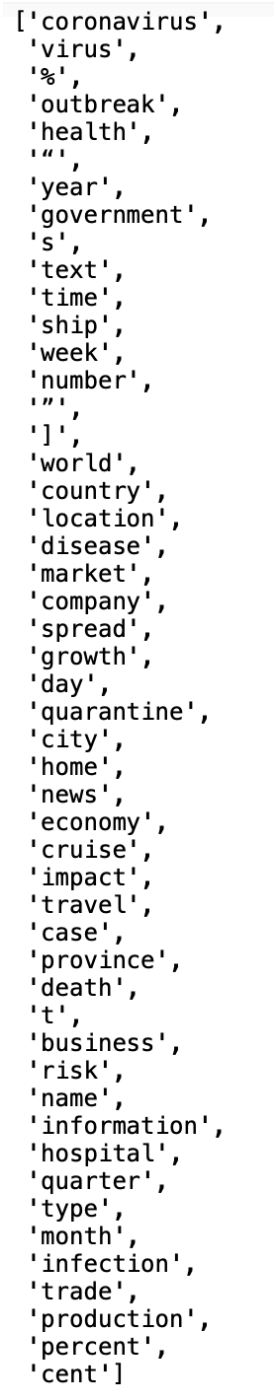
The average length of a sentence is 33.

The following are the results obtained for top 50 adjectives, nouns and verbs.



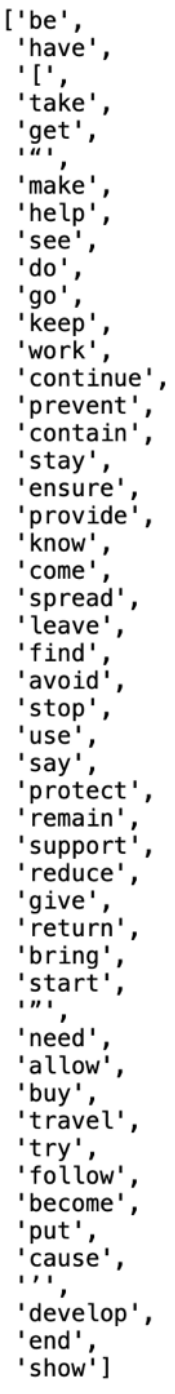
```
['new',
 'Chinese',
 'other',
 'last',
 'first',
 'global',
 's',
 'public',
 '“',
 'due',
 'many',
 'medical',
 'such',
 'novel',
 'economic',
 '’',
 'positive',
 'local',
 'international',
 'same',
 'next',
 'confirmed',
 'major',
 'good',
 'epidemic',
 'central',
 'financial',
 'recent',
 'second',
 'early',
 'possible',
 'current',
 'negative',
 'social',
 'several',
 'total',
 'Japanese',
 'further',
 'high',
 'few',
 'likely',
 'much',
 'full',
 'strong',
 'past',
 'potential',
 'top',
 'own',
 '”',
 'low']
```

Adjectives



```
['coronavirus',
 'virus',
 '%',
 'outbreak',
 'health',
 '“',
 'year',
 'government',
 's',
 'text',
 'time',
 'ship',
 'week',
 'number',
 '”',
 ']',
 'world',
 'country',
 'location',
 'disease',
 'market',
 'company',
 'spread',
 'growth',
 'day',
 'quarantine',
 'city',
 'home',
 'news',
 'economy',
 'cruise',
 'impact',
 'travel',
 'case',
 'province',
 'death',
 't',
 'business',
 'risk',
 'name',
 'information',
 'hospital',
 'quarter',
 'type',
 'month',
 'infection',
 'trade',
 'production',
 'percent',
 'cent']
```

Nouns



```
['be',
 'have',
 '[',
 'take',
 'get',
 '’',
 'make',
 'help',
 'see',
 'do',
 'go',
 'keep',
 'work',
 'continue',
 'prevent',
 'contain',
 'stay',
 'ensure',
 'provide',
 'know',
 'come',
 'spread',
 'leave',
 'find',
 'avoid',
 'stop',
 'use',
 'say',
 'protect',
 'remain',
 'support',
 'reduce',
 'give',
 'return',
 'bring',
 'start',
 '”',
 'need',
 'allow',
 'buy',
 'travel',
 'try',
 'follow',
 'become',
 'put',
 'cause',
 '’',
 'develop',
 'end',
 'show']
```

Verbs

As we can see in the above images most of the reviews contain the adjectives new, China, the nouns coronavirus, virus and the verbs be, have. From this it is evident that coronavirus is related to China. For further evaluation we can use SentiAnalyzer from NLTK on the tokens which does not require POS tagging.