Natural language Processing - Homework 1

MARCH 8

Syracuse University
Authored by: Kritika Chugh
SUID: 882046659

Corpus Statistics and Python Programming

Data

We received 2 data files in json format. They were huge files and the final csv generated with filtered column from them was huge too. Approximately, ~34 MB! Note that these data sets are available on https://ieee-dataport.org/ for free. (reference below)

These data sets represent the discussions online that mention “corona virus”. It could be blog posts, news articles etc. The posts are in English language hence the corpus in question would be analyzing English language only. One of the beauties of JSON file is that its easily processed as dictionary by languages such as python. And with libraries such as NLTK, using Pandas it is easier to process such structures than ever before.

Below is the basic one row from the json file. This JSON file has nested structure at places. **Example**: Thread {Social {Facebook {likes: , shares : , comments: } , LinkedIn , Pinterest ….}} . These places have been extracted into csv very meticulously.


```
{
"organizations":[],
"uuid":"2b50b3f00e04fc17912154a7b88f3359db2b1ae8",
"thread":{
"social":{
"gplus":{
"shares":0
},
"pinterest":{
"shares":1
},
"vk":{
"shares":0
},
"linkedin":{
"shares":0
},
"facebook":{
"likes":19,
"shares":63,
"comments":7
},
"stumbledupon":{
"shares":0
}
},
"site_full":"www.foxnews.com",
```

In [127]:
```
#interpretation of results
feed = pd.read_csv('out_final.csv')
feed.head(3)                        #printing csv to show how many columns exist here
```

Out[127]:

	social.facebook.comments	social.facebook.shares	social.facebook.likes	title	published	replies_count	
0	2	1	2	Nigeria To Spend N620m On Coronavirus Preventi...	2020-02-08T21:05:00.000+02:00	0	ht
1	0	0	0	After White House Meeting, Oil Execs Put in Ca...	2020-02-07T21:46:00.000+02:00	0	ht
2	0	0	0	Doctor killed by coronavirus 'a symbol of	2020-02-07T02:00:00.000+02:00	0	ht

Data Processing

For the report I extracted the following fields into the csv file:

"Facebook": {…}, "title", "published", "replies_count", "author", "url", "country", "text". Facebook had three more nested entries / attributes namely: "likes", "shares" and "comments". That collectively gave me 10 columns to begin with in the final csv file. The csv file is created by the 2 datasets that are extracted from the 2 json containing the data.

```
filtered_columns = ['title', 'published', ]
df1 = pd.read_json('16119_webhose_2020_01_db21c91a1ab47385bb13773ed8238c31_0000002.json', lines=True)
df2 = pd.read_json('16119_webhose_2020_02_db21c91a1ab47385bb13773ed8238c31_0000002.json', lines=True)
listdf = df1.thread.values.flatten()
listdf2 = df2.thread.values.flatten()
#print (listdf)
df1_new = pd.json_normalize(listdf, max_level=2)
df2_new = pd.json_normalize(listdf2,max_level=2)
#print(df1_new)
#"facebook":{…}, "title", "published", "replies_count", "author", "url", "country", "text".
df_csv_filter = df1_new.filter(items=['social.facebook.comments','social.facebook.shares','social.faceb
df_csv_filter2 = df2_new.filter(items=['social.facebook.comments','social.facebook.shares','social.face
#print(df1_new.columns)
df_csv_filter['author'] = df1['author']
df_csv_filter['text'] = df1['text']
df_csv_filter2['author'] = df2['author']
df_csv_filter2['text'] = df2['text']

df_csv_filter_combined = pd.concat([df_csv_filter2,df_csv_filter])
#print(df_csv_filter.head())
df_csv_filter_combined.to_csv('out_final.csv', index=False)
```

From above:

df1 and df2 – data sets from the 2 JSON files.

df_csv_filter and df_csv_filter2 – datasets containing filtered columns for the csv.

out_final.csv – final combined csv file.

```
In [86]: print(df_csv_filter.shape)
         print(df_csv_filter2.shape)
         print(df_csv_filter_combined.shape)

         (10956, 10)
         (95061, 10)
         (106017, 10)
```

In [87]:
```
#tokenization
import nltk
tokens = nltk.word_tokenize(textcol)
print(len(tokens))
```

```
1008434
```

This is done to separate each word from the text in the form a list. This helps to further process the list into stemming or lemmatization.

In [88]:
```
#convert all to lowercase
lowercase_words = [w.lower( ) for w in tokens]
lowercase_words[:30]
print(len(lowercase_words))
```

```
1008434
```

In [89]:
```
#remove all non-alphabetic characters
revised_words = [w for w in lowercase_words if w.isalpha()]
#revised_words[:30]
print(len(revised_words))
```

```
748264
```

The goal here is to remove all the non-alphabetic characters and to be able to understand the meaning of the text better, punctuation marks enable a language, but references could be understood without them too. This helps to remove the nuances from the text.

In [91]:
```
#remove all stop words a, an, the
stopwords = nltk.corpus.stopwords.words('english')
stopped_words = [w for w in revised_words if w not in stopwords]
stopped_words[:30]
print(len(stopped_words))
```

```
512195
```

In [92]:
```
#Lemmatization
wnl = nltk.WordNetLemmatizer()
lemmatized_words = [wnl.lemmatize(t) for t in stopped_words]
lemmatized_words[:100]
print(len(lemmatized_words))
```

```
512195
```

The goal was to make the inflections of verbs, nouns and sometimes adjectives reduced. I also wanted to show a sincere word count for my data analysis and unnecessary inflection(plural) could affect that. That's why I used this method.

Since “is”, “an”, “a”, “the” etc.. are known as stop words and they form the most skeleton structure of a sentence, we want to remove those if we want to understand the usage of other words relevant to “corona virus” for this corpus.

Data Analysis

Following snippets shows the following details: (inline)

- 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)

In [144]:
```
#top 50 words by frequency
from nltk import FreqDist
fdist = FreqDist(lemmatized_words)
items = fdist.most_common(50)
for item in items:
    print (item)
```

```
('coronavirus', 12799)
('china', 8765)
('new', 5012)
('virus', 3590)
('health', 3585)
('case', 3297)
('chinese', 3253)
('february', 2880)
('news', 2802)
('world', 2633)
('death', 2463)
('people', 2236)
('outbreak', 2181)
('feb', 2104)
('reuters', 2092)
('say', 1897)
```

In [116]:
```
#top 50 bigrams by frequencies
from nltk.collocations import *
bigram_measures = nltk.collocations.BigramAssocMeasures()
finder = BigramCollocationFinder.from_words(lemmatized_words)
finder.apply_freq_filter(5)
scored = finder.score_ngrams(bigram_measures.raw_freq)
for bscore in scored[:50]:
    print(bscore)
```

```
(('death', 'toll'), 0.002543952986655473)
(('cruise', 'ship'), 0.002256952918322123)
(('coronavirus', 'outbreak'), 0.0022452386298187216)
(('hong', 'kong'), 0.0019719052314060076)
(('world', 'health'), 0.001772762326848173)
(('novel', 'coronavirus'), 0.0016146194320522456)
(('new', 'coronavirus'), 0.0015970479992971427)
(('coronavirus', 'case'), 0.0014369527230839816)
(('hour', 'ago'), 0.0012241431486055116)
(('health', 'organization'), 0.0010777145423129862)
(('coronavirus', 'death'), 0.0010581907281406496)
(('corona', 'virus'), 0.000966428801530667)
(('virus', 'case'), 0.0009410478431066293)
(('new', 'virus'), 0.0009312859360204609)
(('china', 'coronavirus'), 0.000919571647517059)
```

In [145]:
```
bigramPMITable = pd.DataFrame(list(finder.score_ngrams(bigram_measures.pmi)),
                              columns=['bigram','PMI']).sort_values(by='PMI', ascending=False)

print (bigramPMITable[:50])
```

```
                       bigram        PMI
0           (billie, eilish)  16.644406
6              (minxin, pei)  16.644406
9             (questo, sito)  16.644406
8         (patricia, heaton)  16.644406
7            (ngio, spinout)  16.644406
1          (bottega, veneta)  16.644406
5             (leila, fadel)  16.644406
2            (brize, norton)  16.644406
4         (kylie, maclellan)  16.644406
3     (colecții, libertatea)  16.644406
11      (hunkering, cramped)  16.381371
10          (adrielly, eger)  16.381371
12           (estee, lauder)  16.381371
13            (hanna, ziady)  16.381371
14       (kidambi, srikanth)  16.381371
15          (tora, agarwala)  16.381371
21          (minka, klaudia)  16.158979
25            (đçặț, üńčńđő)  16.158979
```

```
In [118]: #frequencyDistribution
          from nltk.probability import FreqDist
          fdist = FreqDist(tokens)
          print(fdist)

          <FreqDist with 60642 samples and 1008434 outcomes>
          <FreqDist with 12771 samples and 12771 outcomes>
```

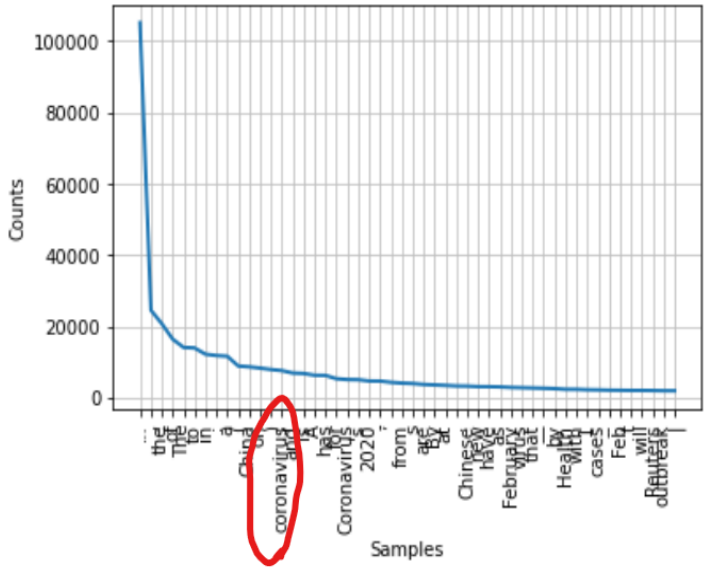
```
In [115]: fdist.most_common(50)

Out[115]: [('...', 105126),
           (',', 24513),
           ('the', 20709),
           ('of', 16372),
           ('The', 14063),
           ('to', 13924),
           ('in', 12170),
           (':', 11800),
           ('a', 11606),
           ('(', 8872),
           ('China', 8636),
           ('on', 8273),
           (')', 7885),
           ('coronavirus', 7572),
           ('and', 6893),
           ('is', 6761),
           ('A', 6270),
           ('has', 6230),
           ('for', 5292),
           ('Coronavirus', 5061),
```

Now it is important to note in the above snippet notice that frequency distribution is achieved on the tokenized text that is not filtered for STOP words yet. So it is populated with all such skeleton words. And hence, it is important to filter the sentence if we need to understand the corpus for what it is. The next word which is of focus is Coronavirus and is very pushed down the list because of the STOP words. To understand this take a look at the frequency distribution plot for the same.

Frequency Distribution plot:

```
In [114]: #frequencyDistribution - Plot
          import matplotlib.pyplot as plt
          fdist.plot(50,cumulative=False)
          plt.show()
```



Now, lets do some analysis on the filtered results in the csv file: **out_final.csv**

Below is a snippet which tries to show a part of the column structure with data:

```
In [127]: #interpretation of results
          feed = pd.read_csv('out_final.csv')
          feed.head(3)                    #printing csv to show how many columns exist here
```

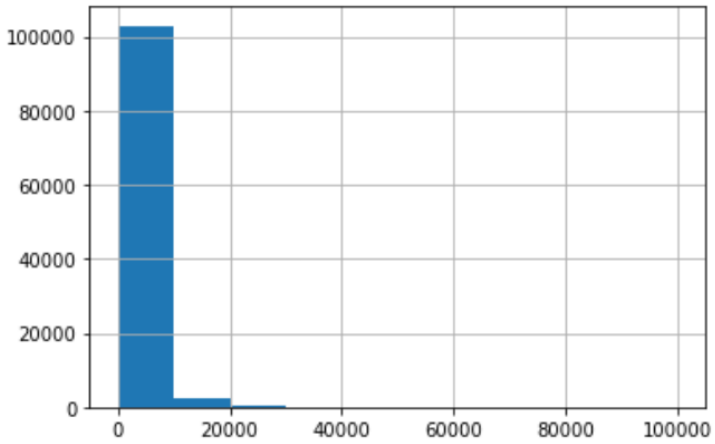
Out[127]:

ebook.shares	social.facebook.likes	title	published	replies_count	url	country	author	text
1	2	Nigeria To Spend N620m On Coronavirus Preventi...	2020-02-08T21:05:00.000+02:00	0	https://www.informationng.com/2020/02/nigeria-...	US	Olayemi Oladotun	Nigeria To Spend N620m On Coronavirus Preventi...
0	0	After White House Meeting, Oil Execs Put in Ca...	2020-02-07T21:46:00.000+02:00	0	https://my.gvtc.com/news/read/category/news/ar...	US	Newser — Newser Editors	The situation has apparently worsened for six ...
0	0	Doctor killed by coronavirus 'a symbol of tran...	2020-02-07T02:00:00.000+02:00	0	https://uk.news.yahoo.com/coronavirus-quaranti...	GB	Sky video	Doctor killed by coronavirus 'a symbol of tran...

Analyzing the text column from the csv that is derived by the 2 JSON data sets. The words range from 0 to 20 thousand and maximum in between that is 1000.

```
In [134]: # number of characters present in each sentence of the text in feed.
          # sentennce level analysis
          feed['text'].str.len().hist()
```

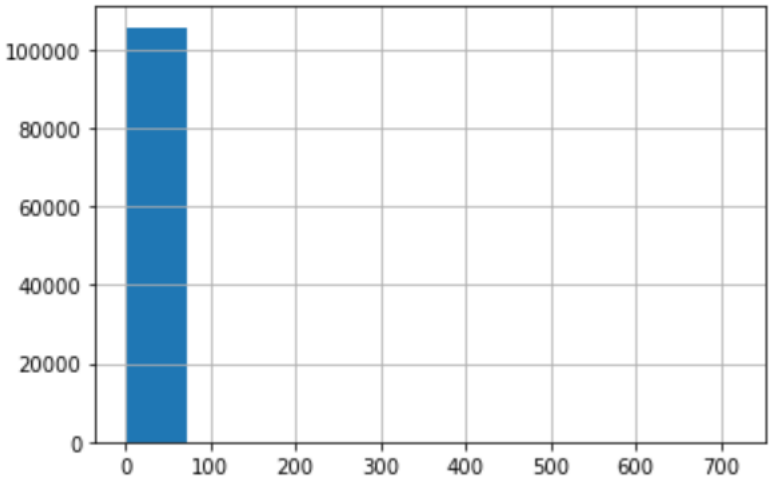
```
Out[134]: <matplotlib.axes._subplots.AxesSubplot at 0x11b64ab28b0>
```



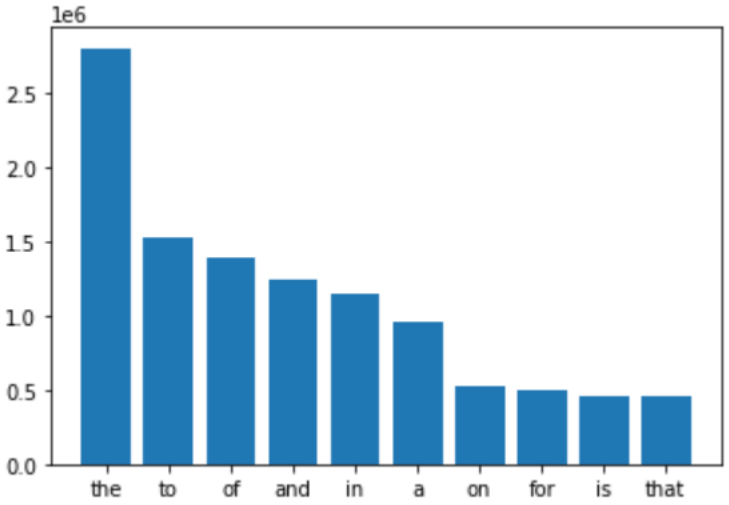
Word level analysis: The average word length ranges from 0 to 70. From the below graph.

```
In [130]: # word level analysis
          feed['text'].str.split().\
             apply(lambda x : [len(i) for i in x]). \
             map(lambda x: np.mean(x)).hist()
```

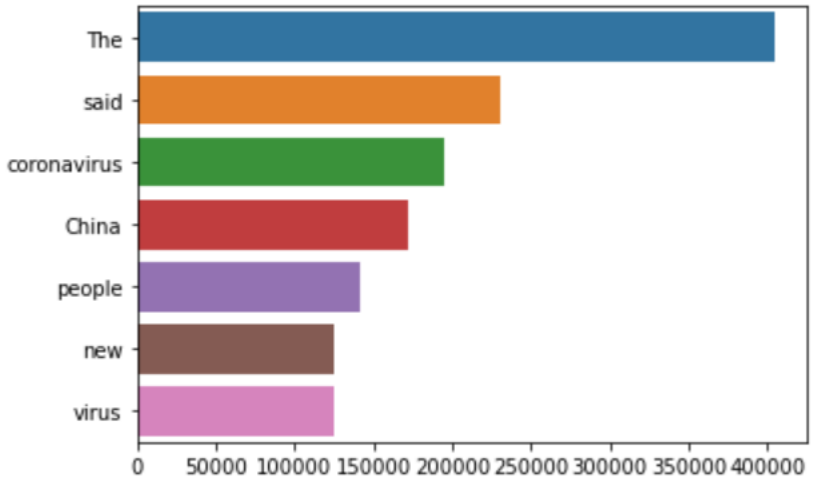
```
Out[130]: <matplotlib.axes._subplots.AxesSubplot at 0x11d45e26850>
```



Top STOP word: The top stop words are definitely “the” , “to” , “of” and so on.. I have analyzed top ten Stop words here as shown below in the histogram.



Top NON-STOP word: Its amazing to notice that corona virus and china is in the upper list of the NON-STOP words. The graph is very revealing in its essence of the corpus it represents.



Interpretation of the Results

The corpus is about corona virus and the results in the analysis above shows that how many times the word – “China”, “virus” , “ship” , “death” etc.. have been repeated. We also understood how important it is to handle the stop words that clutter the corpus mostly.

Future work, includes the sentiment analysis and the finding of distribution of the topic around each word (repeated more often) in the corpus. I am particularly interested in learning about **Latent Dirichlet Allocation** .

References

Ran Geva, March 31, 2020, "free dataset from news/message boards/blogs about CoronaVirus (4 month of data - 5.2M posts)", IEEE Dataport, doi: https://dx.doi.org/10.21227/kc4v-q323