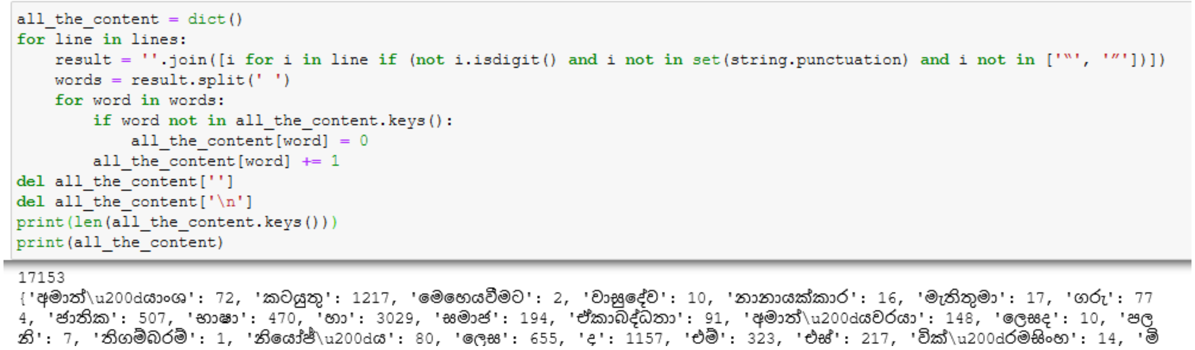
In20-S2-CS5616- Natural Language Processing

Assignment 1

Name	Jayasuriya D.P.
Index	209337M
Module	CS5616

Steps that carried out

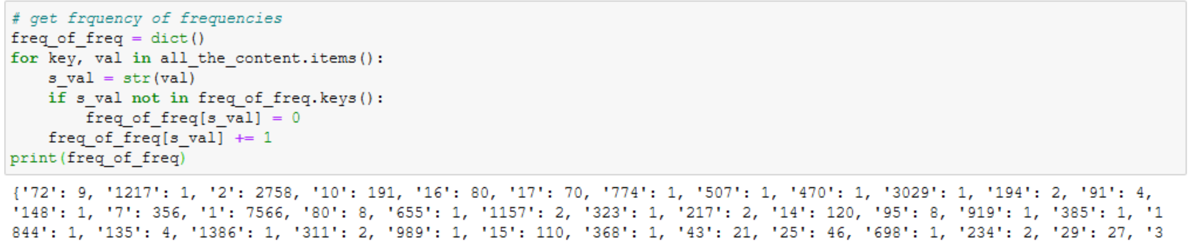
1. Calculate the frequencies of each word



```
all_the_content = dict()
for line in lines:
    result = ''.join([i for i in line if (not i.isdigit() and i not in set(string.punctuation) and i not in ['“', '”'])])
    words = result.split(' ')
    for word in words:
        if word not in all_the_content.keys():
            all_the_content[word] = 0
        all_the_content[word] += 1
del all_the_content['']
del all_the_content['\n']
print(len(all_the_content.keys()))
print(all_the_content)
```

```
17153
{'අමාත්\u200dයාංශ': 72, 'කටයුතු': 1217, 'මෙහෙයවීමට': 2, 'වාසුදේව': 10, 'නානායක්කාර': 16, 'මැතිතුමා': 17, 'ගරු': 77
4, 'ජාතික': 507, 'භාෂා': 470, 'හා': 3029, 'සමාජ': 194, 'ඒකාබද්ධතා': 91, 'අමාත්\u200dයවරයා': 148, 'ලෙසද': 10, 'පල
නි': 7, 'තිගම්බරම්': 1, 'නියෝජ්\u200dය': 80, 'ලෙස': 655, 'ද': 1157, 'එම්': 323, 'එස්': 217, 'වික්\u200dරමසිංහ': 14, 'මි
```

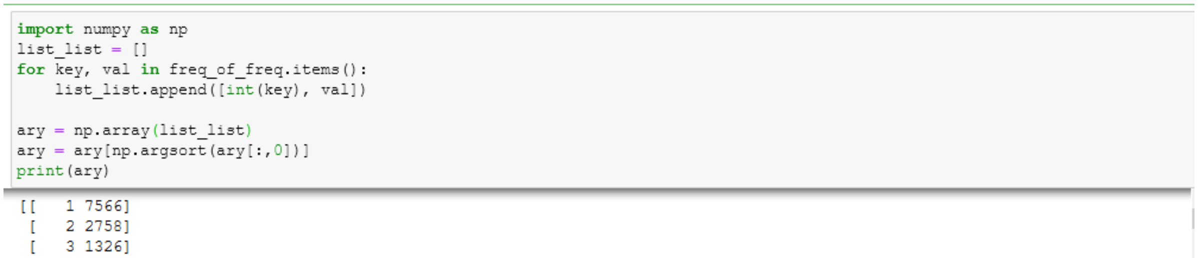
2. Calculate the frequency of frequencies of each word



```
# get frquency of frequencies
freq_of_freq = dict()
for key, val in all_the_content.items():
    s_val = str(val)
    if s_val not in freq_of_freq.keys():
        freq_of_freq[s_val] = 0
    freq_of_freq[s_val] += 1
print(freq_of_freq)
```

```
{'72': 9, '1217': 1, '2': 2758, '10': 191, '16': 80, '17': 70, '774': 1, '507': 1, '470': 1, '3029': 1, '194': 2, '91': 4,
'148': 1, '7': 356, '1': 7566, '80': 8, '655': 1, '1157': 2, '323': 1, '217': 2, '14': 120, '95': 8, '919': 1, '385': 1, '1
844': 1, '135': 4, '1386': 1, '311': 2, '989': 1, '15': 110, '368': 1, '43': 21, '25': 46, '698': 1, '234': 2, '29': 27, '3
```

3. Using the frequencies, you calculate in 1, rank the words as shown in the example in lecture slides

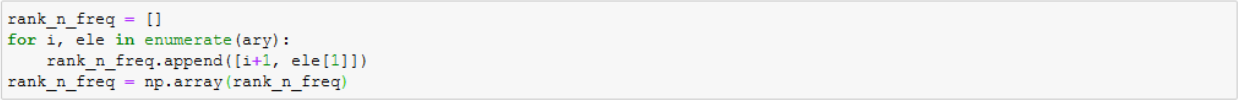


```
import numpy as np
list_list = []
for key, val in freq_of_freq.items():
    list_list.append([int(key), val])

ary = np.array(list_list)
ary = ary[np.argsort(ary[:,0])]
print(ary)
```

```
[[   1 7566]
 [   2 2758]
 [   3 1326]
```

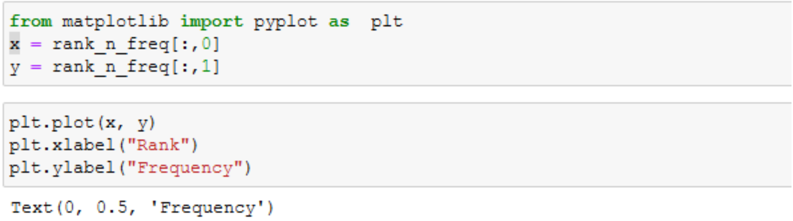
4. plot the rank vs frequency graph.

- Calculate Rank and Frequency



```
rank_n_freq = []
for i, ele in enumerate(ary):
    rank_n_freq.append([i+1, ele[1]])
rank_n_freq = np.array(rank_n_freq)
```

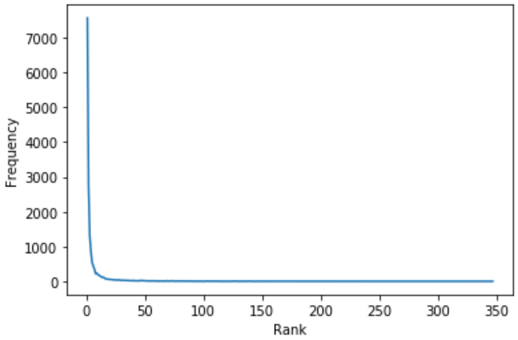
- Plot the graph



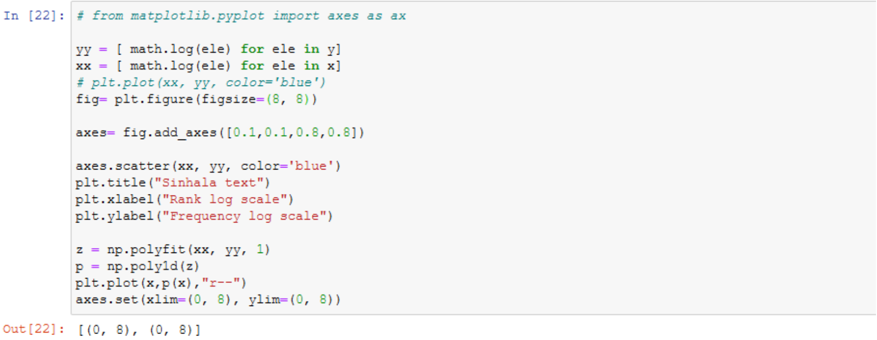
```
from matplotlib import pyplot as  plt
x = rank_n_freq[:,0]
y = rank_n_freq[:,1]
```

```
plt.plot(x, y)
plt.xlabel("Rank")
plt.ylabel("Frequency")
```

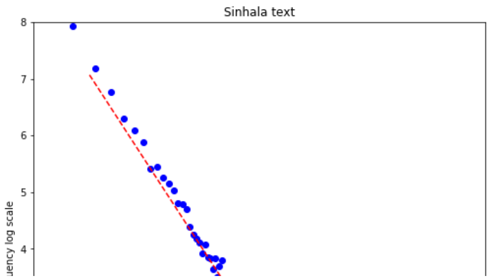
```
Text(0, 0.5, 'Frequency')
```



- Log scale graph



```
In [22]: # from matplotlib.pyplot import axes as ax

         yy = [ math.log(ele) for ele in y]
         xx = [ math.log(ele) for ele in x]
         # plt.plot(xx, yy, color='blue')
         fig= plt.figure(figsize=(8, 8))

         axes= fig.add_axes([0.1,0.1,0.8,0.8])

         axes.scatter(xx, yy, color='blue')
         plt.title("Sinhala text")
         plt.xlabel("Rank log scale")
         plt.ylabel("Frequency log scale")

         z = np.polyfit(xx, yy, 1)
         p = np.poly1d(z)
         plt.plot(x,p(x),"r--")
         axes.set(xlim=(0, 8), ylim=(0, 8))

Out[22]: [(0, 8), (0, 8)]
```



5. What can you comment on the graphs?

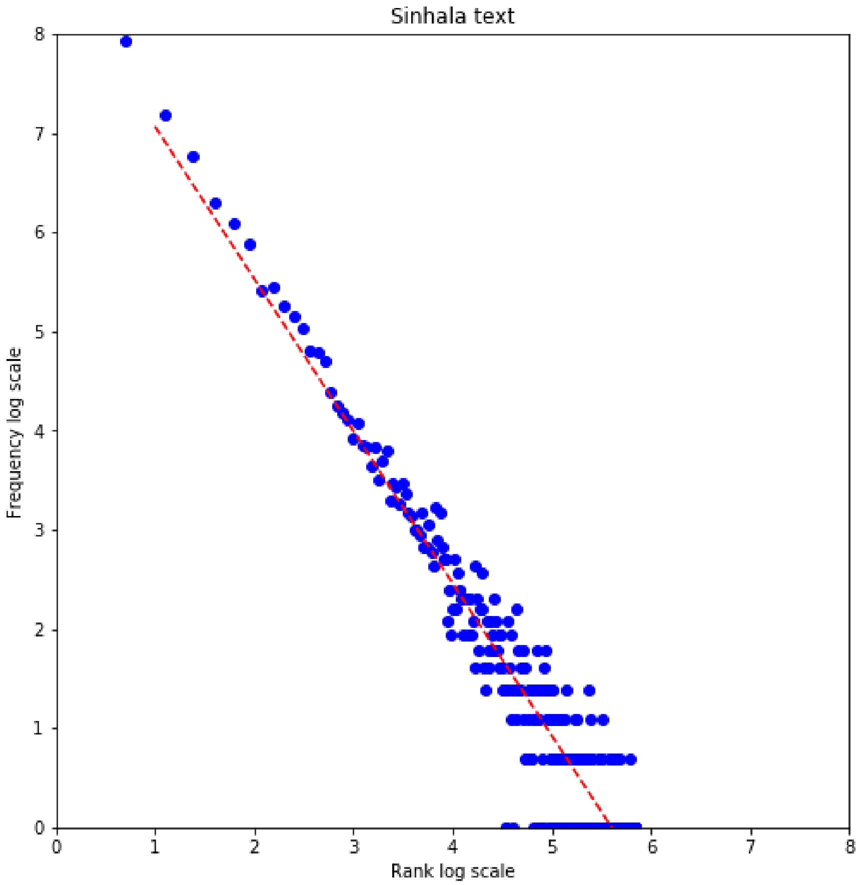
Zipf’s Law

This law states that given some word corpus a small number of words are used all the time, while the vast majority are used very rarely.

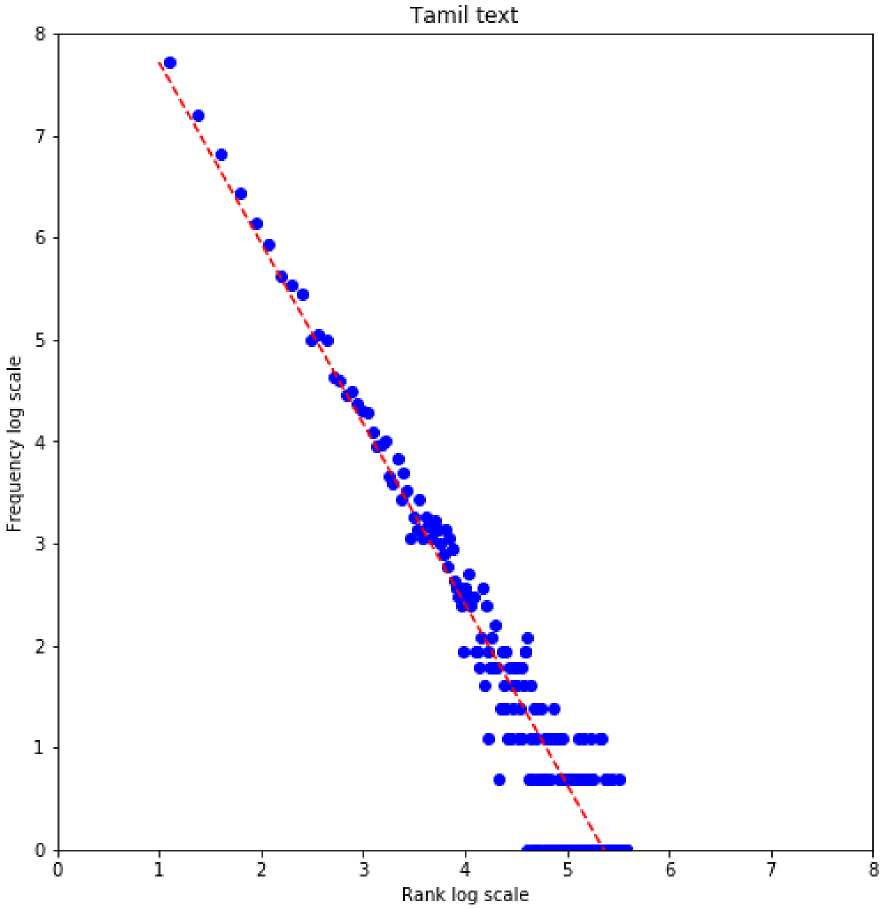
However, there is an interesting fact that in a given corpus of natural language utterance, the frequency of any word is inversely proportional to its rank in the frequency table. Following results exactly show this phenomenon.

Results

Sinhala text graph



Tamil text graph



This law can be written as follows

The rth most frequent word has a frequency f(r) that scales according to

$$f(r) \propto \frac{1}{r^{\alpha}}$$

Where alpha ~ 1,

This seems very strange behavior, but this was observed for many other statistics that follow an exponential distribution.