**1.a>** 122/5

**1.b>** No

**1.c>** 16.9

**1.d>** 3207

**1.e>** 134

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**2.> Code:**

import os

import pandas as pd

import re

cor = pd.read\_csv('phy\_corpus.txt', sep='\n', header=None)[0]

#speed = '\w+[.]\*\w+\s[m][/][s]'

speed = '[0-9]+[.]\*[0-9]\*\s[m][/][s][^2]'

distance = '[0-9]+[.]\*[0-9]\*\s[m][^/s2]'

accelaration = '[0-9]+[.]\*[0-9]\*\s[m][/][s][2]'

time = '[0-9]+[.][0-9]\*\s[s]+'

speed\_value = []

distance\_value = []

accelaration\_value = []

time\_value = []

bim = []

spd = lambda s: re.findall(speed,s)

dist = lambda s: re.findall(distance,s)

acc = lambda s: re.findall(accelaration,s)

t = lambda s: re.findall(time,s)

def find(val):

if(val):

return 1

else:

return 0

def element(doc):

temp = [0,0,0,0]

if(spd(doc)):

temp[0]=1

if(dist(doc)):

temp[1]=1

if(acc(doc)):

temp[2]=1

if(t(doc)):

temp[3]=1

return temp

for doc in cor:

bim.append(element(doc))

def printm():

print("Terms \t",end='')

for i in range (1,10):

print("D",i,'\t',end='')

print("\n\nSpeed \t",end='')

for spd in range(0,9):

print(bim[spd][0],'\t',end='')

print("\nDistance \t",end='')

for d in range(0,9):

print(bim[d][1],'\t',end='')

print("\nAccelaration\t",end='')

for ac in range(0,9):

print(bim[ac][2],'\t',end='')

print("\nTime \t",end='')

for ti in range(0,9):

print(bim[ti][3],'\t',end='')

printm()

**OUTPUT(for first nine problems):**

Terms D 1 D 2 D 3 D 4 D 5 D 6 D 7 D 8 D 9

Speed 0 0 0 1 0 1 1 1 1

Distance 0 1 0 0 1 0 1 0 0

Accelaration 1 0 0 0 1 0 0 1 0

Time 1 1 1 1 0 1 0 0 1

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**3.>**With Zipf’s Law we can see that frequency\*rank had no correlation with frequency of the word; first it increases rapidly and then decreases slowly, maintaining correlation at the middle terms while with MandelBrot’s approximation frequency\*(rank+B) has a correlation with the frequency of the word.

The code is just an extension of the demo code given at: <https://github.com/Ramaseshanr/anlp/blob/master/zipf.ipynb>

**Code:>**

import re

from operator import itemgetter

import nltk

import pandas as pd

import math

frequency = {}

words\_emma = nltk.Text(nltk.corpus.gutenberg.words('austen-emma.txt'))

for word in words\_emma:

count = frequency.get(word, 0)

frequency[word] = count + 1

#Zipf's law

rank = 1;

column\_header = ['Rank', 'Frequency', 'Frequency\*Rank']

tf\_row = []

row = []

df = pd.DataFrame(columns=column\_header)

pd\_cols = []

rows = []

for word, freq in reversed(sorted(frequency.items(), key=itemgetter(1))):

df.loc[word] = [rank,freq,rank\*freq]

rank = rank+1

print(df)

#Mandelbrot's Approximation

rank = 1;

column\_header = ['Rank', 'Frequency', 'Frequency\*Rank+\u03B2']

tf\_row = []

row = []

df = pd.DataFrame(columns=column\_header)

pd\_cols = []

rows = []

for word, freq in reversed(sorted(frequency.items(), key=itemgetter(1))):

df.loc[word] = [rank,freq,(rank+2.7)\*freq]

rank = rank+1

print(df)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**4.>** For the chosen corpus Austin-emma text, value of k is coming out to be 21 giving very close approximation on the unique number of words.

**Code:**

import re

from operator import itemgetter

import nltk

import math

frequency = {}

tokens = nltk.Text(nltk.corpus.gutenberg.words('austen-emma.txt'))

words = []

for word in tokens:

x= word.lower()

words.append(x)

stop\_words = nltk.corpus.stopwords.words('English')

words\_ns=[]

for word in words:

if word not in stop\_words:

words\_ns.append(word)

uw = len(set(words\_ns))

print("Total number of tokens in the corpus: ",len(tokens))

m=21\*pow(len(tokens),0.48)

print("Unique number of words according to heaps law:",m)

print("Number of unique words: ", uw)

**Output:**

Total number of tokens in the corpus: 192427

Unique number of words according to heaps law: 7222.157896962308

Number of unique words: 7213