Team: Bakhtiyar Rakhimzhanov, Nazira Tukeyeva

Group: BD-1903

Github Link to the Repository: <https://github.com/nazirait/Information-Theory-Project>

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

Github Link of Bakhtiyar: <https://github.com/Godadoreu>

Github Link of Nazira: <https://github.com/nazirait>

**Assignment 2 Report**

|  |  |
| --- | --- |
| Task Part | Responsible team member |
| Part 1 Function (Reading txt file, splitting words, counting the characters) | Nazira |
| Part 2 Function (Counting the probability for one element, applying calculation to all the elements in the text) | Bakhtiyar |
| Assignment Report | Both |

**General Description:**

The program is written in Python. There are 5 functions, which are responsible for specific roles, we described it by steps, where you can find code and screenshots.

Step 1: Reading the txt file itself and splitting each string into from that file into a list.

*Source code:*

def reads(file):

'''Given function reads only txt file and returns the splitted words from txt'''

# Checking if the text is txt format

if file.endswith('.txt'):

# Open the text

f = open(file)

# reading the file

reading = f.read()

# getting the number of spaces

space\_num = reading.count(' ')

# Dividing the file for separate words

splits = reading.split()

# appending the spaces for the list by its frequency

splits.append(' ' \* space\_num)

# Closing the file

f.close()

# returning the list with words

return splits

else:

return 'It should be txt file'

work\_list = reads('q1\_file.txt')

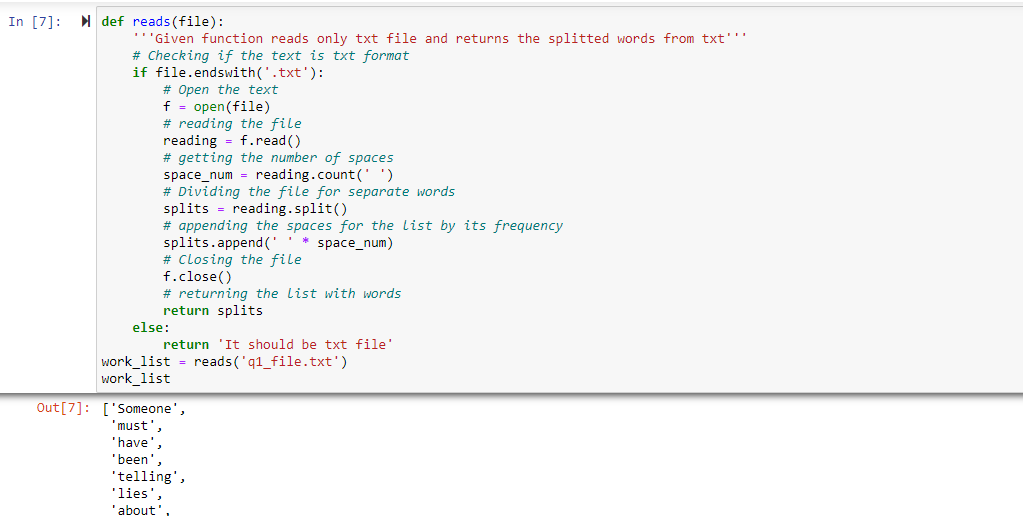


Fig.1

Step 2: Working with the list from Step 1 and getting the number of occurrences for each character in the text.

*Source code:*

def occurence(lists):

'''Given function works with lists and returns the dictionary with the frequency of the characters'''

# Checking if the variable is list format

if isinstance(lists,list):

# creating dict variable for dictionary

dictionary = {}

for x in lists:

# getting the separate characters from the words

character = list(x)

# counting the characters by their appearence

for char in character:

if char in dictionary:

dictionary[char]+=1

else:

dictionary[char]=1

# returning the dictionary variable

return dictionary

else:

return 'It should be list'

dicts = occurence(work\_list)

print(dicts)

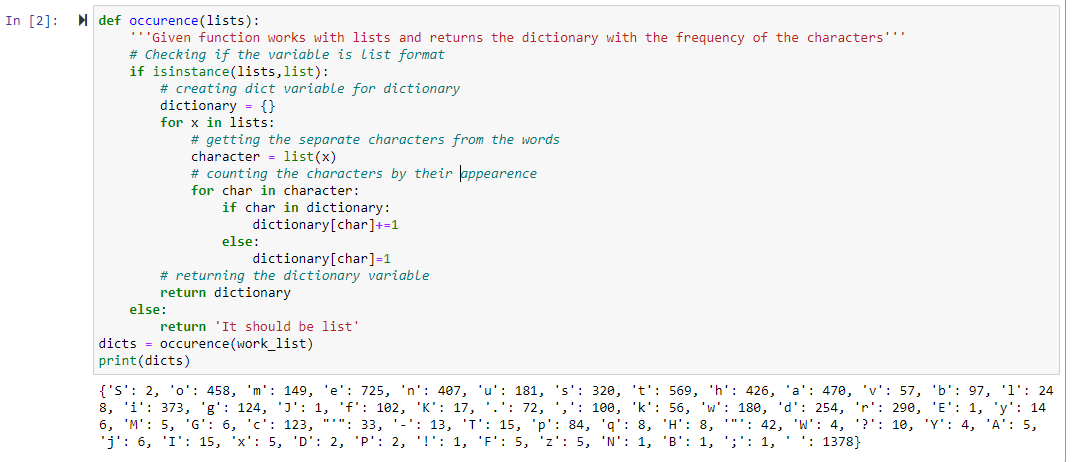


Fig.2

Step 3: Counting the probability of specific letter out of all symbols (in this example, it is ‘S’).

*Source code:*

def probability(dictionary,symbol = None):

if dictionary.get(symbol)!=None:

sums = sum(dictionary.values())

print(sums)

for key,value in dictionary.items():

if symbol==key:

prob = round(value/sums,4)

print(key,'-',prob)

else:

sums = sum(dictionary.values())

for key,value in dictionary.items():

prob = round(value/sums,4)

print(key,' - ',prob)

probabilities\_tuple = probability(dicts,'S')

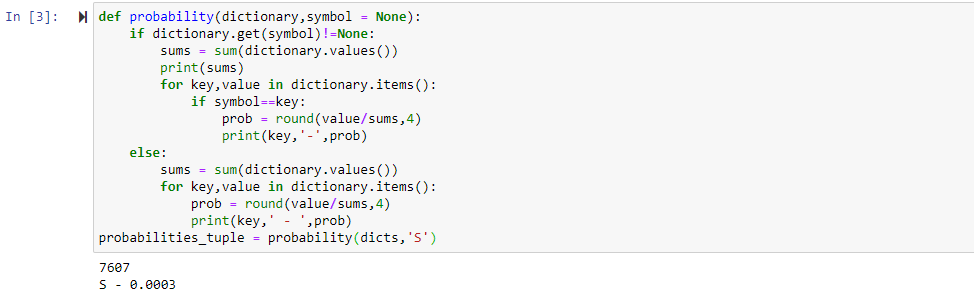


Fig.3

Step 4: Computing the probabilities for all characters out of all available symbols in the text.

*Source code:*

def probability(dictionary):

‘’’The given function returns the probability of characters which divides

the values of characters by the overall character sum’’’

# checking whether variable is dict format

lists = []

if isinstance(dictionary,dict):

# getting the overall frequency of characters

sums = sum(dictionary.values())

# working with keys and values of dictionary

for key,value in dictionary.items():

# getting the probability

prob = round(value/sums,4)

# appending the probabilities

lists.append((key,prob))

return lists

else:

return (‘It should be in dictionary format’)

probabilities\_list = probability(dicts)

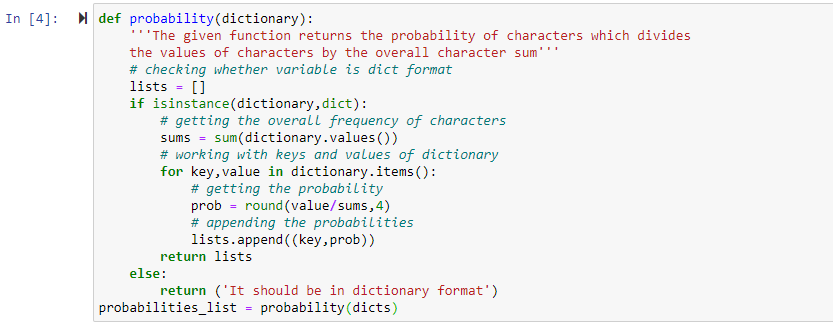


Fig.4

Step 5: Displaying the result sorted by the probability values in descending order.

*Source code:*

def sorted\_prob(lists):

# sorting the elements in descending order

desc = sorted(lists,key = lambda x: x[1],reverse = True)

# showing the result

for key,probability in desc:

print(key,'-',probability)

result = sorted\_prob(probabilities\_list)

result





Fig.5-6