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 4 Report**

|  |  |
| --- | --- |
| Task Part | Responsible team member |
| Part 1 – Reading decoded text and checking for the similarity | Nazira |
| Part 2 – Decoding back the binary sequence into the original text | Bakhtiyar |
| Assignment Report | Both |

**General Description:**

**Shannon-Fano Algorithm | Decoding**

Step 1: Reading a sequence of bits

*Source code:*

def decoded\_text(texts):

'''The given function reads sequence of bits'''

# Checking if the text is txt format

f = open(texts)

# reading the file

reading = f.read()

# Closing the file

f.close()

# returning the words

return reading

decoder = decoded\_text('q1\_file.txt')

decoder

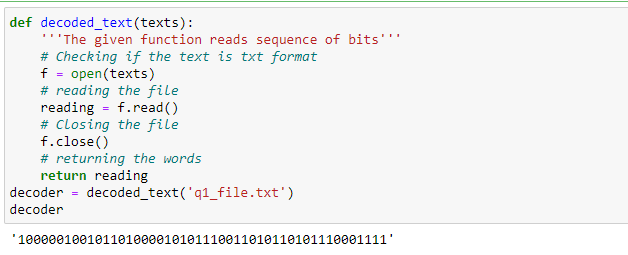


Fig. 1

Step 2: Decoding back the binary sequence into original text.

*Source code:*

def traversing(decoded\_text, dictionary):

'''The given function decodes the text to its initial state'''

# making the list of bits

list\_of\_bits = list(decoded\_text)

# assigning the variable

symbols = ''

appearance\_of\_word = []

# making the loop over list of bits

for x in list\_of\_bits:

# adding the symbols for string

symbols+=x

# if adding symbols matches with any value from the encoded dictionary

if symbols in dictionary.values():

# getting key, value from dictionary

for key,value in dictionary.items():

# if the symbols matches with value

if symbols==value:

# append the symbol for the list

appearance\_of\_word.append(key)

# nullifying the string

symbols = ''

# joining all the symbols together to construct the word

initial\_words = ''.join(appearance\_of\_word)

# return the strings

return initial\_words

decoding\_result = traversing(decoder,dicts)

decoding\_result

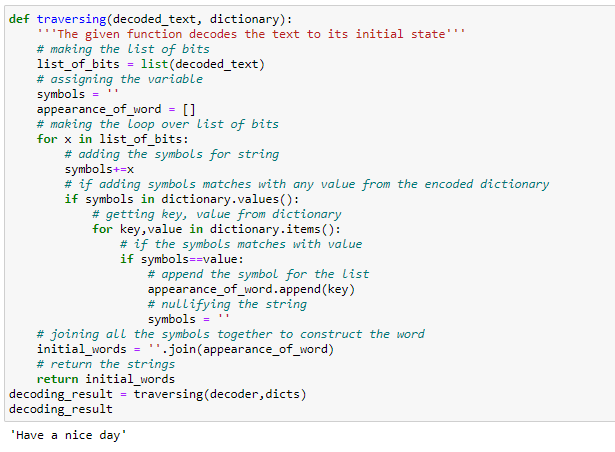


Fig. 2

Step 3: Checking whether the decoded version coincides with the original text.

*Source code:*

def checker(initial\_text,encoded\_text):

'''The given function checks the similarity of original text with its decoded version'''

f = open(initial\_text)

# reading the file

reading = f.read()

# Closing the file

f.close()

# printing the decoded and original text

print('Decoded text: ', encoded\_text)

print('Original text: ',reading)

# checking the similarity

if reading==encoded\_text:

# return positive if everything is okay

return 'The original text and decoded text match'

else:

# return negative results

return 'You had some problems'

checks = checker('new\_files.txt',decoding\_result)

checks

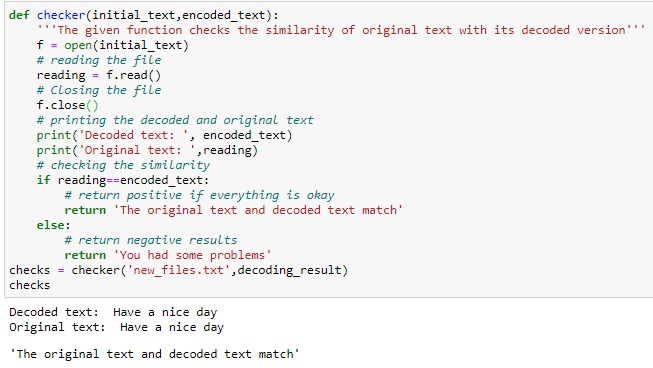


Fig. 3