**Московский авиационный институт**

**(Национальный исследовательский университет)**

Факультет прикладной математики и физики

Кафедра вычислительной математики и программирования

**Лабораторная работа № 4**

по курсу «Криптография»

Студент: Аксенов А. Е.

Группа: М80-306б-19

Преподаватель: Борисов А. В.

Оценка:

Москва, 2022

**Содержание**

1. Постановка задачи.

2. Общий метод и алгоритм решения.

3. Код программы.

4. Демонстрация работы.

5. Выводы.

1. **Постановка задачи**

Сравнить:

1. Два осмысленных текста на естественном языке.

2. Осмысленный текст и текст из случайных букв.

3. Осмысленный текст и текст из случайных слов.

4. Два текста из случайных букв.

5. Два текста из случайных слов.

Считать процент совпадения букв в сравниваемых текстах – получить дробное значение от 0 до 1 как результат деления количества совпадений на общее число букв. Расписать подробно в отчёте алгоритм сравнения и приложить сравниваемые тексты в отчёте хотя бы для одного запуска по всем пяти подпунктам. Осознать какие значения получаются в этих пяти подпунктах. Привести свои соображения о том почему так происходит.

Длина сравниваемых текстов должна совпадать. Привести соображения о том какой длины текста должно быть достаточно для корректного сравнения.

1. **Общий метод и алгоритм решения**

В качестве осмысленных текстов на естественном языке были взяты «Моё африканское путешествие» У. Черчилля и «Знак четырёх» А. Конан Дойла, оба английском языке.

Ссылки:

<https://www.gutenberg.org/cache/epub/43035/pg43035.txt>

<https://www.gutenberg.org/cache/epub/2097/pg2097.txt>

Текст из случайных слов генерируется из следующего словаря (466 тысяч английских слов):

<https://raw.githubusercontent.com/dwyl/english-words/master/words_alpha.txt>

Текст из случайных букв генерируется из букв английского алфавита в обоих регистрах и состоит из слов длиной от 3 до 10 знаков.

Алгоритм сравнения: параллельно обходим оба текста, сравниваем знаки на одинаковых позициях. Если знаки совпадают, то увеличиваем счётчик совпавших символов на 1. Сравнение регистрозависимое.

1. **Код программы**

import random

import string

import getopt

import os

import sys

import urllib.request

CNT\_RANDOM\_TEXTS = 10

LEN\_RANDOM\_TEXT = 2 \* 10 \*\* 5

CASES = 5

USAGE = """

Syntax: main.py [--cases=#]

Flags:

cases=#

Numbers of cases to use. By default all cases are used.

1 -- two meaningful texts in natural language

2 -- meaningful text and text from random letters

3 -- meaningful text and text from random words

4 -- two texts from random letters

5 -- two texts from random words

Example:

--cases=1,3

"""

def count\_common\_letters(text1, text2):

cnt = 0

for char1, char2 in zip(text1, text2):

if char1 == char2:

cnt += 1

return cnt

def match\_perc(text1, text2):

return count\_common\_letters(text1, text2) / len(text1)

def gen\_random\_letters(n):

text = ''

while len(text) < n:

len\_word = random.randint(3, 10)

word = ''.join(random.choice(string.ascii\_letters) for \_ in range(len\_word))

text += ' ' + word

rem = len(text) - n

if rem != 0:

text = text[:-rem]

return text

def gen\_random\_words(n):

url = 'https://raw.githubusercontent.com/dwyl/english-words/master/words\_alpha.txt'

response = urllib.request.urlopen(url)

words = response.read().decode()

words = words.splitlines()

text = ''

while len(text) < n:

text += ' ' + random.choice(words)

rem = len(text) - n

if rem != 0:

text = text[:-rem]

return text

def case1():

print("Case #1: two meaningful texts in natural language.")

url = 'https://www.gutenberg.org/cache/epub/43035/pg43035.txt'

url2 = 'https://www.gutenberg.org/cache/epub/2097/pg2097.txt'

response = urllib.request.urlopen(url)

text1 = response.read().decode()

response = urllib.request.urlopen(url2)

text2 = response.read().decode()

min\_len = min(len(text1), len(text2))

text1 = text1[:min\_len]

text2 = text2[:min\_len]

print("Text length: {0}".format(min\_len))

print("Match: {0}".format(match\_perc(text1, text2)))

def case2():

print("Case #2: meaningful text and text from random letters.")

url = 'https://www.gutenberg.org/cache/epub/43035/pg43035.txt'

response = urllib.request.urlopen(url)

text1 = response.read().decode()

s = 0

for i in range(CNT\_RANDOM\_TEXTS):

text2 = gen\_random\_letters(len(text1))

with open('./tests/case2\_text\_{0}'.format(i + 1), 'w') as f:

f.write(text2)

s += match\_perc(text1, text2)

s /= CNT\_RANDOM\_TEXTS

print("Text length: {0}".format(len(text1)))

print("Match: {0}".format(s))

def case3():

print("Case #3: meaningful text and text from random words.")

url = 'https://www.gutenberg.org/cache/epub/43035/pg43035.txt'

response = urllib.request.urlopen(url)

text1 = response.read().decode()

s = 0

for i in range(CNT\_RANDOM\_TEXTS):

text2 = gen\_random\_words(len(text1))

with open('./tests/case3\_text\_{0}'.format(i + 1), 'w') as f:

f.write(text2)

s += match\_perc(text1, text2)

s /= CNT\_RANDOM\_TEXTS

print("Text length: {0}".format(len(text1)))

print("Match: {0}".format(s))

def case4():

print("Case #4: two texts from random letters.")

s = 0

for i in range(CNT\_RANDOM\_TEXTS):

text1 = gen\_random\_letters(LEN\_RANDOM\_TEXT)

with open('./tests/case4\_text1\_{0}'.format(i + 1), 'w') as f:

f.write(text1)

text2 = gen\_random\_letters(LEN\_RANDOM\_TEXT)

with open('./tests/case4\_text2\_{0}'.format(i + 1), 'w') as f:

f.write(text2)

s += match\_perc(text1, text2)

s /= CNT\_RANDOM\_TEXTS

print("Text length: {0}".format(LEN\_RANDOM\_TEXT))

print("Match: {0}".format(s))

def case5():

print("Case #5: two texts from random words.")

s = 0

for i in range(CNT\_RANDOM\_TEXTS):

text1 = gen\_random\_words(LEN\_RANDOM\_TEXT)

with open('./tests/case5\_text1\_{0}'.format(i + 1), 'w') as f:

f.write(text1)

text2 = gen\_random\_words(LEN\_RANDOM\_TEXT)

with open('./tests/case5\_text2\_{0}'.format(i + 1), 'w') as f:

f.write(text2)

s += match\_perc(text1, text2)

s /= CNT\_RANDOM\_TEXTS

print("Text length: {0}".format(LEN\_RANDOM\_TEXT))

print("Match: {0}".format(s))

def print\_usage(message):

print(USAGE)

if message:

sys.exit('\nFATAL ERROR: ' + message)

else:

sys.exit(1)

def parse\_args(args):

try:

opts, args = getopt.getopt(args, '', ['help', 'cases='])

except getopt.GetoptError:

print\_usage('Invalid arguments.')

cases = [i for i in range(1, CASES + 1)]

for (opt, val) in opts:

if opt == '--help':

print\_usage(None)

elif opt == '--cases':

try:

cases = set(map(int, val.split(',')))

except ValueError:

print\_usage('Cases must be comma separated list.')

for i in cases:

if i not in range(1, CASES + 1):

print\_usage('Incorrect cases')

return cases

if \_\_name\_\_ == '\_\_main\_\_':

cases = parse\_args(sys.argv[1:])

for i in cases:

if i == 1:

case1()

elif i == 2:

case2()

elif i == 3:

case3()

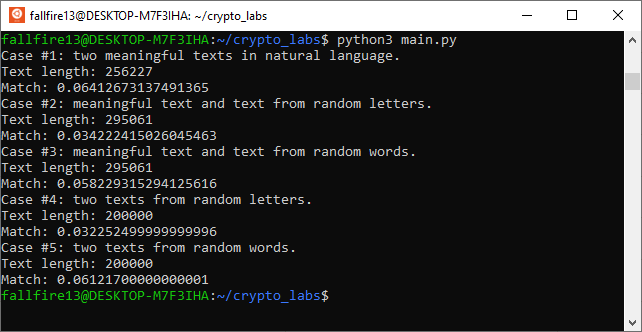
elif i == 4:

case4()

elif i == 5:

case5()

1. **Демонстрация работы**



1. **Выводы**

Как видно из результатов, наилучшие совпадения получаются путём сравнения двух осмысленных текстов и двух текстов из случайных слов. Худшие совпадения у осмысленного текста с текстом из случайных букв и у двух текстов из случайных букв. Думаю, полученные результаты можно было бы объяснить какими-то лингвистическими законами построения языка, но у меня нет достаточных знаний в этой области. Например, для букв английского языка характерна некоторая частотность, которая будет соблюдаться в осмысленных текстах, и которая не соблюдается в генерируемых из букв текстах. Эмпирически кажется, что размер слогов в осмысленных словах совпадает чаще, чем в случайных, и 5 букв, передающих гласные звуки в английском языке, будут совпадать чаще. Высокое совпадение тестов из случайных слов можно объяснить тем, что эти тексты были составлены по одному словарю. В текстах из случайных букв нет никаких ограничений на использование букв в верхнем регистре не на первой позиции в слове, что сильно снижает количество совпадений с осмысленным тестом.

Также кажется, что если в качестве осмысленных текстов взять разные произведения одного и того же автора, то совпадений будет больше, потому что у каждого автора можно выделить характерный для него словарь.

Что касается достаточной длины текста для корректного сравнения, начиная с какой-то достаточно большой длины, по закону больших чисел, среднее значение совпадений станет равным мат. ожиданию совпадений. Мат. ожидание количества совпадений для осмысленного текста определить сложно, потому что непонятно, какое там распределение. Поэтому рассмотрим два текста из случайных букв.

Если речь идёт о сгенерированном тексте, то *random.choice* использует равномерное распределение. Вероятность выбора любого знака (26 букв в обоих регистрах и пробел). Пусть случайная величина – индикатор совпадения знаков в *k*-ой позиции, т.е. , если знаки на *k*-ой позиции совпали и , если не совпали. Вероятность совпадения двух знаков , несовпадения - . Получаем распределение:

Математическое ожидание равно *E*(*Ik*) = . Случайная величина – число совпадений знаков – равна сумме индикаторов совпадения по всем позициям: , где – длина текста (в нашем тесте 200000). Переходя к ожиданию:

Как видно из нашего теста, получившееся количество совпадений - 3225, что плохо соотносится с 71. Думаю, это связано с тем, что пробелы в сгенерированых текстах встречаются чаще, чем другие знаки, и длина текста слишком маленькая. После того, как я убрал пробелы, количество совпадений стало равным 2096. Но это всё равно много, по-хорошему надо увеличивать длину текста.