Московский Авиационный Институт

(Национальный исследовательский Университет)

Факультет: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

**Лабораторная работа №3**

**по курсу «Криптография»**

|  |  |
| --- | --- |
| Студент: | Бердикин Т.А. |
| Группа: | М80-307Б-18 |
| Преподаватель: | Борисов А.В. |
| Оценка: |  |
| Дата: | 12.06.2021 |

Москва

2021

**Содержание**

1. Постановка задачи.
2. Общий метод и алгоритм решения.
3. Демонстрация работы программы.
4. Вывод.

**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

Как сравнивать: 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.

**Метод решения**

В качестве осмысленных текстов на естественном языке были взяты поэмы «Илиада» и «Одиссея» Гомера. Обе на английском языке. За текст отдельное спасибо Project Gutenberg – единственному удобному сайту, находящемуся в быстром доступе.

Ссылки:

<https://www.gutenberg.org/files/6130/6130-0.txt> – Илиада

<https://www.gutenberg.org/files/1727/1727-0.txt> – Одиссея

Текст из случайных слов генерируется из следующего словаря (примерно 25500 английских слов):

https://svnweb.freebsd.org/csrg/share/dict/words?revision=61569&view=co

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.

**Демонстрация работы программы**

**tim@tim-GL503VD:~/Desktop/crypto\_tim/3$ python lab3.py**

**Case #1: two meaningful texts in natural language.**

**Text length: 710821**

**Match: 0.062405303163524994**

**Case #2: meaningful text and text from random letters.**

**Text length: 1140036**

**Match: 0.03244932616162998**

**Case #3: meaningful text and text from random words.**

**Text length: 1140036**

**Match: 0.05636300149609324**

**Case #4: two texts from random letters .**

**Text length : 500000**

**Match : 0.031823590000000001**

**Case #5: two texts from random words .**

**Text length : 500000**

**Match : 0.06793058**

**Код программы**

import random

import string

import getopt

import os

import sys

import urllib.request

CNT\_RANDOM\_TEXTS = 10

LEN\_RANDOM\_TEXT = 10 \*\* 6

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

"""

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://svnweb.freebsd.org/csrg/share/dict/words?revision=61569&view=co'

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/files/6130/6130-0.txt'

url2 = 'https://www.gutenberg.org/files/1727/1727-0.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/files/6130/6130-0.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/files/1727/1727-0.txt'

try:

urlopen(url)

except:

pass

response = urllib.request.urlopen(url)

text1 = response.read().decode()

s = 0

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

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

s += match\_perc (text1, text2)

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()

**Выводы.**

Судя по результатам, выведенным программой, наилучшие совпадения получаются, если сравнить два осмысленных текста и два текста, созданных из случайных слов. Не секрет, что во всех мировых языках наблюдаются определённые закономерности и тенденции. Существует огромное количество лингвистических законов, и, возможно, среди них нашёлся бы тот, который смог бы объяснить данный результат. К сожалению, это совершенно другая профессиональная область, в которой я не разбираюсь даже на уровне дилетанта.

Также существует статистика употребления букв в языке, и логично, что в осмысленном тексте частотность служебных слов (предлоги, союзы и т.п.) будет примерно одинаковая, а значит эти моменты будут совпадать. Плюс добавим туда стандартные сочетания букв в окончаниях и суффиксах. Плюс частотность отдельных букв в словах. То есть буквы, условно говоря, находящиеся в центре клавиатуры, встречаются в речи значительно чаще, чем всякие Э, Щ, и т.п. Видимо поэтому тексты из осмысленных слов лучше совпадают.

Также можно отметить, что для сравнения мы взяли два текста от одного и того же автора. Уникальный авторский стиль и словарь из наиболее часто употребляемых автором слов, возможно, несколько нивелируется переводом, но всё же является одной из причин наибольшего количества совпадений в наших примерах

С текстами, созданными из случайных слов, всё ещё проще. random.choice() использует равномерное распределение в обоих случаях, так что логично, что получается довольно большое количество совпадений. Я также уверен, что на текстах большой длины (на несколько порядков больше символов) совпадений будет ещё больше.

Длину текста для корректных сравнений надо, конечно, увеличивать. По закону больших чисел, на определённой длине текста среднее значение совпадений станет равным математическому ожиданию этих совпадений. Мат. ожидание для осмысленного текста не посчитать, а вот для рандомно сгенерированного – вполне можно. Правда, значение мат. ожидания сильно отличалось от количества совпадений. Отчасти это объясняется пробелами, отчасти – недостаточно большой длиной текста. Поэтому количество символов следует увеличить многократно.