Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №3 по курсу**

**«Криптография»**

Студент: Кузьмичев Александр Николаевич

Группа: М80 – 306Б-18

Преподаватель: Борисов А. В.

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2021

**Задание**

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.

**Решение**

Задача была решена над латинским алфавитом. Я использовал два художественных произведения – “Мартин Иден” Джека Лондона и “Великий Гэтсби” Фитцджеральда в качестве осмысленных текстов на естественном языке. Для составления текста из случайных слов был использован словарь объемом около 25 000 слов. Функция words\_generator составляет текст на его основе.

Для составления текста из случайных букв используется функция letters\_generator. Она генерирует слова длинной от 3 до 10 букв. Параметры длины слова легко изменяются в коде.

Для сравнения текста реализована функция cnt\_common\_letters. Она сравнивает символы, стоящие на одинаковой позиции в двух текстах, и, если они совпадают, увеличивает на единицу счетчик совпадений. Стоит отметить, что сравнение регистрозависимое.

Ссылки:

Martin Eden (первый осмысленный текст) - <https://www.gutenberg.org/files/1056/1056-0.txt>

The Great Gatsby (второй осмысленный текст) – https://www.gutenberg.org/files/64317/64317-0.txt

Словарь - <https://svnweb.freebsd.org/csrg/share/dict/words?view=co&content-type=text/plain>

**Примеры работы программы**

*Case #1: Concious text vs concious text in natural language.*

*Text length: 296704*

*Match: 0.061778742450388266*

*Case #2: Concious text vs random letters text.*

*Text length: 804567*

*Match: 0.03427918370999557*

*Case #3: Concious text vs random words text.*

*Text length: 804567*

*Match: 0.05908892609316565*

*Case #4: Random letters text vs random letters text.*

*Text length: 1000000*

*Match: 0.032283599999999996*

*Case #5: Random words text vs random words text.*

*Text length: 1000000*

*Match: 0.06031479999999999*

**Выводы**

Проанализировав результаты нескольких запусков, можно с уверенностью сказать, что наилучшие результаты (больше всего совпадений) получаются при сравнении двух осмысленных текстов на естественном языке (*0.061778742450388266*). Наихудшие же результаты получаются при сравнении двух текстов, составленных из случайных букв разных регистров (*0.032283599999999996*).

Результаты вполне логичны и объяснимы. Так как предложения в любом языке строятся по определенным правилам, то совпадений в текстах с осмысленными предложениями будет больше, чем совпадений в текстах со случайными словами. Минимальное количество совпадений в двух текстах, составленных из рандомных букв, можно объяснить как минимум тем, что они составляются из букв разных регистров. Соответственно, вероятность того, что буквы совпадут, падает.

Для корректного сравнения требуется длинный текст. Это можно аргументировать тем, что, по законам больших чисел, начиная с какой-то длинны среднее значение совпадений станет равным математическому ожиданию совпадений.

**Листинг кода**

import random

import string

import getopt

import os

import sys

import requests

import urllib.request

CNT\_RT = 10

LEN\_RT = 10 \*\* 6

CASES = 5

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

    cnt = 0

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

        if char1 == char2:

            cnt += 1

    return cnt

def match(text1, text2):

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

def letters\_generator(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 words\_generator(n):

    url = 'http://svnweb.freebsd.org/csrg/share/dict/words?view=co&content-type=text/plain'

    hdrs = {'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_10\_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/39.0.2171.95 Safari/537.36'}

    req = urllib.request.Request(url, headers=hdrs)

    response = urllib.request.urlopen(req)

    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: Concious text vs concious text in natural language.")

    url = 'https://www.gutenberg.org/files/1056/1056-0.txt'

    url2 = 'https://www.gutenberg.org/files/64317/64317-0.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(text1, text2)))

def case2():

    print("Case #2: Concious text vs random letters text.")

    url = 'https://www.gutenberg.org/files/1056/1056-0.txt'

    response = urllib.request.urlopen(url)

    text1 = response.read().decode()

    s = 0

    for i in range(CNT\_RT):

        text2 = letters\_generator(len(text1))

        s += match(text1, text2)

    s /= CNT\_RT

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

    print("Match: {0}".format(s))

def case3():

    print("Case #3: Concious text vs random words text.")

    url = 'https://www.gutenberg.org/files/1056/1056-0.txt'

    response = urllib.request.urlopen(url)

    text1 = response.read().decode()

    s = 0

    for i in range(CNT\_RT):

        text2 = words\_generator(len(text1))

        s += match(text1, text2)

    s /= CNT\_RT

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

    print("Match: {0}".format(s))

def case4():

    print("Case #4: Random letters text vs random letters text.")

    s = 0

    for i in range(CNT\_RT):

        text1 = letters\_generator(LEN\_RT)

        text2 = letters\_generator(LEN\_RT)

        s += match(text1, text2)

    s /= CNT\_RT

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

    print("Match: {0}".format(s))

def case5():

    print("Case #5: Random words text vs random words text.")

    s = 0

    for i in range(CNT\_RT):

        text1 = words\_generator(LEN\_RT)

        text2 = words\_generator(LEN\_RT)

        s += match(text1, text2)

    s /= CNT\_RT

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

    print("Match: {0}".format(s))

def error\_control(message):

    if message:

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

    else:

        sys.exit(1)

def arguments(args):

    try:

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

    except getopt.GetoptError:

        error\_control('Invalid arguments.')

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

    for (opt, val) in opts:

        if opt == '--help':

            error\_control(None)

        elif opt == '--cases':

            try:

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

            except ValueError:

                error\_control('Cases must be comma separated list.')

            for i in cases:

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

                    error\_control('Incorrect cases')

    return cases

if \_\_name\_\_ == '\_\_main\_\_':

    cases = arguments(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()