# Московский авиационный институт (национальный исследовательский университет)

## Факультет информационных технологий и прикладной математики

## Кафедра вычислительной математики и программирования


**Лабораторная работа №4 по курсу «Криптография»**


Студент: И. Т. Батяновский
Преподаватель: А. В. Борисов
Группа: М8О-307Б

Дата:
Оценка:
Подпись:

**Москва, 2020**

# Лабораторная работа №4

**Задача:** Необходимо 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.

# 1 Описание алгортма

Для данной работы я взял текст романов Л.Н. Толстого «*Война и Мир*» и Ф. М. Достоевского «*Братья Карамазовы*». Я решил написать программу на ЯП `Python`. Содержание этих романов и есть осмысленные тексты на естественном языке. Для этого я из строк файла(в котором хранится роман) составил одну большую строку.

Для генерации текста из слов мне понадобился список из уникальных слов. Должен отметить, что я пытаю огромное уважение к классикам, поэтому для создания этого списка я использовал структуру данных set, которая после прохода по всему роману хранит в себе только уникальные слова. Далее я обработал этот список слов, убрав от туда мешающие знаки пунктуации(т.к. текст на английском языке, то знак ‘ не был удален из-за сокращений he‘s she‘s и так далее). Далее использовал готовый метод `r`andom.choices(), который случайным образом брал слова из списка, тем самым создавая текст из случайных слов(знак пробела является отдельным словом).

Ситуация с текстом, состоящим из случайных символов, еще проще. Используется тот же метод(`r`andom.choices()), однако, на этот раз не из готовых слов, а из множества символов `ASCII`, включающий в себя заглавные, прописные, цифры и знаки пунктуации(включая пробел).

Все сгенерированные тексты представлены в виде строки, поэтому дальше они по символьно сравниваются для анализа.

## 2 Реализация алгоритма

```
1
2  import string
3  import random
4
5
6  def textParser(file):
7      wordSet = set()
8      origText = list()
9      for line in file:
10         for word in line.split(sep="\n"):
11             origText.append(word)
12         for word in line.split():
13             wordSet.add(word)
14     return wordSet, origText
15
16
17 def getRigOfPunctuation(wordList):
18     newList = []
19     for word in wordList:
20         newList.append(word.translate(str.maketrans('', '', string.punctuation)))
21     return newList
22
23
24 def randomSymbolsGen(N, lang):
25     return ''.join(random.choices(finalLang, k=N))
26
27
28 def fromListToStr(wordList):
29     newStr = str()
30     for word in wordList:
31         newStr += word
32     return newStr
33
34
35 def randomWordsGen(N, wordList):
36     return random.choices(wordList, k=N)
37
38
39 def countTheDifferences(textA, textB):
40     counter = 0
41     for i in range(len(textA)):
42         if textA[i] == textB[i]:
43             counter += 1
44     return counter
45
46
47 string.punctuation += '”' + '“' # этихсимволовнетпоумолчанию
```

```
48
49  random.seed(11)
50
51  WAPF = open("WarAndPeaceFull", encoding='utf-8', mode="r") # reading txt
52  TBKF = open("The Brothers Karamazov", encoding='utf-8', mode="r") # reading txt
53
54  charCounter = 0
55  allWordsSetWAPF, origWAPF = textParser(WAPF) # taking unique set of words and orig
        text
56  allWordsSetTBKF, origTBKF = textParser(TBKF) # taking unique set of words and orig
        text
57
58  allWordsSetWAPF = sorted(allWordsSetWAPF) # sorting set
59  allWordsSetTBKF = sorted(allWordsSetTBKF) # sorting set
60
61  allWordsSetWAPF = getRigOfPunctuation(allWordsSetWAPF) + [" "] # getting rid of
        punctuation
62  allWordsSetTBKF = getRigOfPunctuation(allWordsSetTBKF) + [" "] # getting rid of
        punctuation
63
64  # print(string.punctuation) # checking punctuation symbols
65
66  WAPF.close() # closing file
67  TBKF.close() # closing file
68
69  origWAPF = fromListToStr(origWAPF) # convert list to str
70  origTBKF = fromListToStr(origTBKF) # convert list to str
71
72  setup = set()
73  for symb in origWAPF[:]:
74      setup.add(symb)
75
76  finalLang = str()
77  for symbol in setup:
78      finalLang += symbol
79
80  rWordsWAPF = fromListToStr(randomWordsGen(100000, allWordsSetWAPF))
81  rWordsTBKF = fromListToStr(randomWordsGen(100000, allWordsSetTBKF))
82  rSymbolsWAPF = randomSymbolsGen(1000000, finalLang)
83  rSymbolsTBKF = randomSymbolsGen(1000000, finalLang)
84  croppedOriginalTextWAPF = origWAPF
85  croppedOriginalTextTBKF = origTBKF
86
87
88  # Нижепроцессможнозаменитьфункцией, ноуменяуженетсилсегодняэтододелывать, авременинет
89  numberOfSymbols = 1000
90  print("-" * 100)
91  print("Для текстов, размеркоторыхравен", numberOfSymbols, "символов")
92  print("1)", countTheDifferences(origWAPF[:numberOfSymbols], origTBKF[:numberOfSymbols
```

```
    ]))
print("2)", countTheDifferences(origWAPF[:numberOfSymbols], rSymbolsWAPF[:
    numberOfSymbols]))
print("3)", countTheDifferences(origWAPF[:numberOfSymbols], rWordsWAPF[:
    numberOfSymbols]))
print("4)", countTheDifferences(rSymbolsTBKF[:numberOfSymbols], rSymbolsWAPF[:
    numberOfSymbols]))
print("5)", countTheDifferences(rWordsTBKF[:numberOfSymbols], rWordsWAPF[:
    numberOfSymbols]))
print("-" * 100)

numberOfSymbols = 10000
print("-" * 100)
print("Для текстов, размеркоторыхравен", numberOfSymbols, "символов")
print("1)", countTheDifferences(origWAPF[:numberOfSymbols], origTBKF[:numberOfSymbols
    ]))
print("2)", countTheDifferences(origWAPF[:numberOfSymbols], rSymbolsWAPF[:
    numberOfSymbols]))
print("3)", countTheDifferences(origWAPF[:numberOfSymbols], rWordsWAPF[:
    numberOfSymbols]))
print("4)", countTheDifferences(rSymbolsTBKF[:numberOfSymbols], rSymbolsWAPF[:
    numberOfSymbols]))
print("5)", countTheDifferences(rWordsTBKF[:numberOfSymbols], rWordsWAPF[:
    numberOfSymbols]))

numberOfSymbols = 100000
print("-" * 100)
print("Для текстов, размеркоторыхравен", numberOfSymbols, "символов")
print("1)", countTheDifferences(origWAPF[:numberOfSymbols], origTBKF[:numberOfSymbols
    ]))
print("2)", countTheDifferences(origWAPF[:numberOfSymbols], rSymbolsWAPF[:
    numberOfSymbols]))
print("3)", countTheDifferences(origWAPF[:numberOfSymbols], rWordsWAPF[:
    numberOfSymbols]))
print("4)", countTheDifferences(rSymbolsTBKF[:numberOfSymbols], rSymbolsWAPF[:
    numberOfSymbols]))
print("5)", countTheDifferences(rWordsTBKF[:numberOfSymbols], rWordsWAPF[:
    numberOfSymbols]))
print("-" * 100)

numberOfSymbols = 500000
print("-" * 100)
print("Для текстов, размеркоторыхравен", numberOfSymbols, "символов")
print("1)", countTheDifferences(origWAPF[:numberOfSymbols], origTBKF[:numberOfSymbols
    ]))
print("2)", countTheDifferences(origWAPF[:numberOfSymbols], rSymbolsWAPF[:
    numberOfSymbols]))
print("3)", countTheDifferences(origWAPF[:numberOfSymbols], rWordsWAPF[:
    numberOfSymbols]))
```

```
124 print("4)", countTheDifferences(rSymbolsTBKF[:numberOfSymbols], rSymbolsWAPF[:
        numberOfSymbols]))
125 print("5)", countTheDifferences(rWordsTBKF[:numberOfSymbols], rWordsWAPF[:
        numberOfSymbols]))
126 print("-" * 100)
127
128 print(rWordsWAPF[:60])
129 print(rSymbolsWAPF[:60])
130 print(origWAPF[:60])
```

Программа выводит количество совпавших символом из соотношений описанных в разделе «Описание алгоритма».

В конце программа выводит часть сгенерированного текста из слов, букв соответственно и оригинального текста.

# 3 Результат работы программы

C:\Anaconda\envs\Cryptography4\python.exe
C:/Users/Ivan/PycharmProjects/Cryptography4/textGen.py
———————————————————————————————-
Для текстов, размер которых равен 1000 символов
1) 73
2) 13
3) 2
4) 8
5) 55
———————————————————————————————-
———————————————————————————————-
Для текстов, размер которых равен 10000 символов
1) 595
2) 102
3) 190
4) 112
5) 551
———————————————————————————————-
Для текстов, размер которых равен 100000 символов
1) 6644
2) 970
3) 4483
4) 974
5) 5417
———————————————————————————————-
———————————————————————————————-
Для текстов, размер которых равен 500000 символов
1) 32615
2) 4811
3) 23009
4) 4870
5) 27111
———————————————————————————————-
goinglikewaltzhackimpotencemedalsassignationinconceivableobl
dfrkvUq6yhm SMr4Ga1D?lQjsnMSzghJpwUO,”y0!a5HFcu4t7VOaL8EksKN
By Leo Tolstoy/TolstoiWAR AND PEACE CONTENTS BOOK ONE:

Process finished with exit code 0

# 4 Анализ результатов

| Процент совпадений при длине текста: | 1000 | 10000 | 100000 | 500000 |
|---|---|---|---|---|
| Два осмысленных текста на естественном языке | 7,3% | 5,95% | 6,644% | 6,523% |
| Осмысленный текст и текст из случайных букв | 0,8% | 0,87% | 0,945% | 0,9682% |
| Осмысленный текст и текст из случайных слов | 0,2% | 1,9% | 4,483% | 4,6018% |
| Два текста из случайных букв | 0,8% | 1,12% | 0,974% | 0,973% |
| Два текста из случайных слов | 5,5% | 5,51% | 5,417% | 5,4222% |

Рассмотрим случай: два текста из случайных букв. Пусть событие A - появление символа из алфавита $\Omega$ в первом тексте. Тогда P(A) - вероятность события A. Аналогично, событие B - появление символа из алфавита $\Omega$ во втором тексте. Тогда P(B) - вероятность события B. Будем считать, что события A и B независимы, так как, метод генерации текста из случайных символов работает подобным образом.

Тогда событие C - совпадение двух взятых символов из текстов. Тогда P(C) - вероятность события C. Найдем P(C).

Так как события A и B независимы, то

$P(C) = |\Omega|P(A)P(B)$, где $|\Omega|$ - мощность алфавита $\Omega$

P(A) = P(B), из-за одинакового алфавита и случайного генератора последовательностей символов. Алфавит в моей лобораторной состоит из 103 уникальных символов (включая запятые, пробелы и так далее). Следовательно, вероятность P(A) равна:

$P(A) = \frac{1}{103}$
$P(C) = |\Omega|P(A)P(B)$
$P(C) = 103 * \frac{1}{103}\frac{1}{103} \approx 0{,}972\%$

Результат очень к тому, что получили на практике. Можно заметить, что это также близко к результатам в графе «Осмысленный текст и текст из случайных букв».

Остальные варианты сравнения легче показать на практике, что и сделано в данной лабораторной. Связано это с тем, что распределены символы в осмысленных текстах и текстах с случайными словам неравномерно.

# 5 Выводы

Эта лабораторная заставила меня задуматься над стратегией в расшифровке сообщений. Благодаря этой работе я понял, что наиболее выгодно пытаться расшифровать один осмысленный текст другим, так как вероятность совпадения отдельных символов наибольшая.

С другой стороны это задание показало мне, что алгоритмы шифрования должны уметь защищать данную уязвимость (потому как даже на моем тексте удалось добиться 6,523% успеха совпадения, что на мой взгляд очень много и при усердной работе можно добиться расшифровки).

И такие методы защиты мне уже известны, к примеру: шифрование последующих символом в зависимости от предыдущих. Этот метод мешает взломщику напрямую поставить в соответствие одному символу другой, что осложняет процесс взлома.