Московский авиационный институт (национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №4 по курсу «Криптография»

Студент: М. А. Бронников
Преподаватель: А. В. Борисов
Группа: М8О-307Б

Дата:
Оценка:
Подпись:

Москва, 2020

Вариант №4

Задача:

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

Мне было интересно выполнить эту аналитическую работу относительно русского языка. В качестве языка программирования я выбрал **Python**, поскольку он позволяет крайне легко работать со строками. По ходу работы я генерировал файлы с текстами, которые можно нати в директории `data/`. Сам исходный код можно найти в файле `Comparator.ipynb`.

По заданию необходимо проанализировать появление именно букв в нескольких текста, поэтому текстами в этой работе являются последовательности из 36 букв кириллицы вместе с пробелом в качестве разделителя. Для сравнения в качестве осознанного текста я взял текстовый файл поэмы «Русслан и Людмила» *А. С. Пушкина*, которую можно найти в файле `poema.txt`, после чего произвел токенезацию текста(разделение его на слова) и преобразовал все слова к нижнему регистру для чистоты эксперимента.
Код который отвечает за токенезацию:

```
def split_words(a_text):
    cur_word = ''
    prev_is_alpha = False

    for letter in a_text:
        if letter.isdigit():
            continue
        if (letter.isalpha() and prev_is_alpha):
            cur_word += letter
        elif (letter.isalpha() and not prev_is_alpha):
            if cur_word: yield cur_word
            cur_word = letter
            prev_is_alpha = not prev_is_alpha
        else:
            if cur_word: yield cur_word
            cur_word = ''
            prev_is_alpha = False
    if cur_word: yield cur_word

poema_words = list(map(lambda x: x.lower(), split_words(poema)))
```

Далее из полученного списка слов я получил два равных по величине текста, которые и являются моими осознанными текстами в этой работе.

```
poema_text = " ".join(poema_words)
text_len = len(poema_text) // 2
human_text1 = poema_text[:text_len]
human_text2 = poema_text[text_len:2*text_len]
```

Далее я сгенерировал случайные тексты слов и букв равной длины, где в качестве

алфавита слов выстает набор всех слов поэмы, полученный при токенезации.
За генерацию отвечает следующие функции:

```
def generate_random_chars(alphabet, lenght):
    ans = ""
    max_idx = len(alphabet) - 1
    for _ in range(lenght):
        ans += alphabet[random.randint(0, max_idx)]
    return ans


def generate_random_words(base, lenght):
    gen_len = 0
    ans = ""
    while gen_len < lenght:
        possible_words = list(filter(lambda x: len(x) <= lenght - gen_len, base))
        idx = random.randint(0, len(possible_words)-1)
        ans += possible_words[idx]
        gen_len += len(possible_words[idx])
        if gen_len < lenght:
            ans += " "
            gen_len += 1
    return ans
```

Далее я сравнил полученные тексты. Для сравнения я просто обычным циклом прошел по двум текстам, инкрементируя счетчик совпадений при равенстве букв на одинаковых позициях. После чего в качестве результата брал частное полученного количества совпадений и длины текста.
Код, возвращающий долю совпадений в двух равных по длине текстов:

```
def compare_texts(text1, text2):
    if len(text1) != len(text2):
        raise ValueError
    lenght = len(text1)
    equals = 0
    for i in range(lenght):
        if text1[i] == text2[i]:
            equals += 1
    return equals / lenght
```

После этого оставалось лишь сравнить полученные тексты. Если сравнивались тексты разной природы, бралось среднее значение искомой доли совпадений.

2 Результаты

Для каждого из пяти пунктов задания я получил следующую таблицу результатов при сравнении:

Доля совпадений при сравнении	
Два осмысленных текста на естественном языке	Доля: 6.38%
Осмысленный текст и текст из случайных букв	Доля: 2.69%
Осмысленный текст и текст из случайных слов	Доля: 5.92%
Два текста из случайных букв	Доля: 2.71%
Два текста из случайных слов	Доля: 5.65%

Как можно заметить, результат заметно отличается в зависимости от природы сравниваемого текста.

3 Анализ результатов

В данной работе мы напрямую увидели как уловие принадлежности буквы к слову из языка и самого слова к тесту естественного языка меняет вероятность появления совпадений в последовательностях. В этом эксперименте мы получили частоты $W_n(A) = \frac{m}{n}$ наступления события A совпадений двух взятых в последовательностях слов, при этом частота является приближенной оценкой вероятности $\lim\limits_{n\to\infty} W_n = P(A)$ при достаточно больших длинах текстов, поэтому чем больше размеры взятых для анализа текстов, тем точнее полученный результат. Посмотрим почему же мы получили такой результат!

Пусть Ω - алфавит языка, который в нашем случае состоит из 37 символов. Вероятность события $A = \sum\limits_{i=1}^{|\Omega|}(\{c_i = T_1[j]\} \cup \{c_i = T_2[j]\})$ совпадения букв можно записать как сумму событий в которых конкретно взятая буква алфавита будет выбрана случайным образом из обоих текстов T_1 и T_2, что эквивалентно тому, что в равных по длине больших текстах на позиции j окажутся одинаковые буквы алфавита, при условии что тектсы - случайные последоваельности-элементы своего языка. Эти события несовместны, причём будем считать, что тексты генерируются независимо друг от друга, тогда справедливо $P(A) = \sum\limits_{i=1}^{|\Omega|} P(\{c_i = T_1[j]\})P(\{c_i = T_2[j]\})$.

Рассмотрим простейший случай, когда сравниваются тексты T_1 и T_2 из случайных букв:

$$P(\{c_i = T_1[j]\}) = P(\{c_i = T_2[j]\}) = \frac{1}{37},\ \forall j \in N,\ \forall i : 1 \leq i \leq 37$$

.

Следовательно:

$$P(A) = \sum_{i=1}^{|\Omega|} P(\{c_i = T_1[j]\})P(\{c_i = T_2[j]\}) = 37\frac{1}{37}\frac{1}{37} = \frac{1}{37} \approx 2.7\%$$

Как видим, результат крайне близок к полученному. Также заметно, что сравнение осмысленного текста T_1 с текстом случайных букв T_2 дает приблизительно такой же результат. Это нетрудно доказать:

$$P(A) = \sum_{i=1}^{|\Omega|} P(\{c_i = T_1[j]\})P(\{c_i = T_2[j]\}) = \frac{1}{37}\sum_{i=1}^{37} P(\{c_i = T_1[j]\}) = 1 \cdot \frac{1}{37} \approx 2.7\%$$

Что касается остальных результатов, то в других сравнениях играет свою роль тот фактор, что появление букв в словах, а слов в текстах естественного языка не распределено равномерно, что не позволяет выносиить константный множитель за скобки. Вероятности появления букв в текстах носит условный характер, что непосредственно отражается на результате. Если попытаться грубо описать полученный результат, то в полученной сумме произведений для каждого символа получается, что меньшие вероятности перемножаются с меньшими, а большие с большими, что в сумме дает значение больше того, что мы получили для равномерного распределении. Этот результат проще показать экспериментальным способом, что я и сделал в этой лабораторной, вместо попытки вывести это аналитически.

4 Выводы

Благодаря четвёртой лабораторной работе по курсу «Криптография», я узнал о различии в вероятностях появления букв в последовательностях естественного языка и убедился в этом эксперементальным способом.

Эта работа была одной из самых простых для выполнения, однако полученные результаты заставили меня задуматься и разобраться в природе распределения вероятностей появления сиимволов в естественном языке.

Эта лабораторная работа наглядно показала мне, что при грубом взломе попытка подбора ключа случайными последовательностями является слишком примитивной, поскольку ключи, которые необходимо подбирать, зачастую придумываются людьми, а значит в них с большой долей вероятности заложены закономерности естественных языков, что можно использовать для кратного увеличения шансов успешного подбора. Теперь я задумался о том, что, наверное, мне стоит придумать себе более стойкий пароль в соц.сетях или выбрать случайную последовательность, сгенерированную машиной.