Лабораторная работа № 4 по курсу криптографии

Выполнил студент группы М8О-306Б *Дубинин Артем*.

Условие

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

Парсинг

Нужно было взять два текста на естественном языке. Я выбрал для этой задачи две книги 1984 и Великий Гэтсби в оригинале. Я подумал, что задача подразумевала сравнение символов английского алфавита и нам не важен регистр букв, поэтому возьмем только символы английского алфавита и преобразуем их все в нижний регистр.

parser.py

```
import argparse


def parser(input_fn, output_fn):
    with open(input_fn, 'r') as input, open(output_fn, 'w') as output:
        for row in input:
            str = ''
            for ch in row:
                if ch.isalpha():
                    output.write(ch.lower())
```

```
def main():
    a_parser = argparse.ArgumentParser()
    a_parser.add_argument('--input', required=True,
    help='File for input')
    a_parser.add_argument('--output', required=True,
    help='File for output')
    args = a_parser.parse_args()
    parser(args.input, args.output)


if __name__ == "__main__":
    main()
```

Генерация случайных букв

Так как я решил сранивать буквы английского алфавита в нижнем регистре, мне нужно было взять псведо-рандомно символы, для этого я воспользовался функциями рандома в python3.

gen_symbols.py

```
import argparse
import random
import string


def create(input_fn, output_fn):
    with open(input_fn, 'r') as input, open(output_fn, 'w') as output:
        l = len(input.read())
        for _ in range(l):
            output.write(random.choice(string.ascii_lowercase))


def main():
    a_parser = argparse.ArgumentParser()
    a_parser.add_argument('--input', required=True,
    help='File to create a new file of the same size')
    a_parser.add_argument('--output', required=True,
    help='File for new file')
    args = a_parser.parse_args()

    create(args.input, args.output)
```

```
if __name__ == '__main__':
    main()
```

Генерация случайных слов

Чтобы решить задачу с генерацией слов, я решил взять банк английских слов с `https://github.com/dwyl/english-words`. Файл содержал 466k английских слов. Генерация происходил методом random.choice python'a.

gen_words.py

```
import argparse
import random


def isWordAlpha(word):
    for ch in word:
        if not ch.isalpha():
            return False
    return True


def create(input_fn, output_fn, dict_fn):
    with open(input_fn, 'r') as input, \
            open(output_fn, 'w') as output, \
            open(dict_fn, 'r') as dict_fl:
        words = []
        for word in dict_fl:
            if word[-1] == '\n':
                word = word[:-1]
            words.append(word)

        l = len(input.read())
        new_l = 0
        while True:
            word = random.choice(words)
            if not isWordAlpha(word):
                continue
            word = word.lower()
            if new_l < l:
                output.write(word)
                new_l += len(word)
            else:
                output.write(word[:l - new_l])
                break
```

```
def main():
    a_parser = argparse.ArgumentParser()
    a_parser.add_argument('--input', required=True,
    help='File to create a new file of the same size')
    a_parser.add_argument('--dict', required=True,
    help='File with words(dictionary)')
    a_parser.add_argument('--output', required=True,
    help='File for new file')
    args = a_parser.parse_args()

    create(args.input, args.output, args.dict)


if __name__ == '__main__':
    main()
```

Сравнение символов

Для начала приравняем тексты к одинаковому размеру, если это ещё не было сделано. Сравнивать символы я решил таким образом: будем записывать в массив кол-во вхождений каждой буквы английского алфавита текста. Например буква 'a' появляется в тексте 14 раз, следовательно в массиве будет записано следующее alpha['a'] = 14. Сделаем так для двух текстов, которые нужно сравнить. Далее пробегаем по всем буквам алфавита. И берем минимум из массивов, этот минимум будет количеством совпадений букв i в двух текстах. Для примера, буква 'x' встречается в text1 18 раз, а в text2 10 раз, следовательно совпадают 10 символов буквы 'x' в двух текстах. Прибавляем это количество к общее количество совпадений. После того, как мы подсчитаем для всех букв кол-во совпадений, поделим это число на общее кол-во букв, это и будет ответ.

comp.py

```
import argparse


def cmp(input_fn_1, input_fn_2):
    with open(input_fn_1, 'r') as input1, open(input_fn_2, 'r') as input2:
        alpha1 = [0 for _ in range(26)]
        alpha2 = [0 for _ in range(26)]
        s1 = input1.read()
        s2 = input2.read()
        if len(s1) > len(s2):
            s1 = s1[:len(s2)]
        else:
```

```
            s2 = s2[:len(s1)]
        all_chars = len(s1)
        success = 0
        for ch in s1:
            alpha1[ord(ch) - ord('a')] += 1
        for ch in s2:
            alpha2[ord(ch) - ord('a')] += 1

        for i in range(26):
            success += min(alpha1[i], alpha2[i])

        return success / all_chars


def main():
    a_parser = argparse.ArgumentParser()
    a_parser.add_argument('--input1', required=True,
    help='File for input №1')
    a_parser.add_argument('--input2', required=True,
    help='File for input №2')
    args = a_parser.parse_args()
    print(cmp(args.input1, args.input2))


if __name__ == "__main__":
    main()
```

Результат работы программы

1. два осмысленных текста на естественном языке

```
art@mars:~/study/Cryptography/lab_4$ python3 comp.py --input1
books/clean_gatsby.txt  --input2 books/clean_1984.txt
0.9618477357869493
```

2. осмысленный текст и текст из случайных букв

```
art@mars:~/study/Cryptography/lab_4$ python3 comp.py --input1
books/clean_gatsby.txt  --input2 gen_files/symbols_for_gatsby.txt
0.6494192516889509
```

3. осмысленный текст и текст из случайных слов

```
art@mars:~/study/Cryptography/lab_4$ python3 comp.py --input1
books/clean_gatsby.txt  --input2 gen_files/words_for_gatsby.txt
0.8842500431054752
```

4. два текста из случайных букв

```
art@mars:~/study/Cryptography/lab_4$ python3 comp.py --input1
gen_files/symbols_for_gatsby.txt  --input2 gen_files/symbols2.txt
0.9933695941815446
```

5. два текста из случайных слов

```
art@mars:~/study/Cryptography/lab_4$ python3 comp.py --input1
gen_files/words_for_gatsby.txt  --input2 gen_files/words2.txt
0.9944589870997069
```

Выводы

Сравнивая символы двух осмысленных текстов, я сделал вывод, что в среднем каждый символ английского языка используется одинаково в текстах, где используются слова, поэтому такой и большой score. Сравнивая осмысленный текст и текст и случайных букв, score получился гораздо ниже, возможно потому, что в первом варианте использовались осмысленные слова. И в словах есть закономерность в буквах, в отличие от текста составленного из рандомных символов. Осмысленный текст и текст из случайных слов, вроде бы доказывает мою теорию, в словах определенный порядок букв, но почему score получился меньше, чем в случае двух осмысленных текстов, возможно потому, что в предложениях закладывают смысл и порядок слов закономерен, и потому что я взял два романа и сранивал их между собой. Возможно если бы я взял один технический текст и один роман, то score был бы меньше. Два текста из случайных букв показали хороший score, из-за того, что псевдо-рандомом выпадает одинаковое кол-во букв на текста одинакового размера. Так же и для двух текстов из случайных слов, только вместо букв, в среднем одинаковые слова.