Лабораторная работа 1

Сиденко Анастасия

10 сентября 2019 г.

Оглавление

1 Введение **2**
1.1 Постановка задачи 2

2 Аналитическая часть **3**
2.1 Расстояние Левенштейна и Дамерау-Левенштейна 3

3 Конструкторская часть **4**
3.1 Алгоритмы 4

4 Технологическая часть **5**
4.1 Листинг кода 5
4.2 Заготовленные тесты 6

5 Экспериментальная(исследовательская) часть **7**
5.1 Результаты тестов 7
5.2 Замеры времени (графики зависимости времени от длины слова) 7

6 Заключение **10**

Часть 1

Введение

1.1 Постановка задачи

Алгоритмы поиска расстояний:

1. Левенштейна
2. Дамерау-Левенштейна

Способы реализации:

1. матричный
2. рекурсивный

Часть 2

Аналитическая часть

2.1 Расстояние Левенштейна и Дамерау-Левенштейна

Расстояние Левенштейна или редакционное расстояние между строками двумя стрками - это минимальное количество редакторских операций, необходимых для преобразования первой строки во вторую.

Редакторские операции:

1. I – Вставка – штраф 1
2. D – Удаление – штраф 1
3. R – Замена – штраф 1
4. M – Совпадение – штраф 0

Расстояние Дамерау-Левенштейна - расширение, вводится операция транспозиции или перестановки 2 соседних символов (штраф 1).

Часть 3

Конструкторская часть

3.1 Алгоритмы

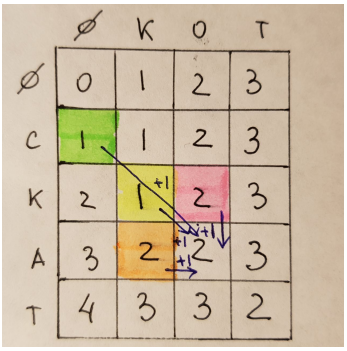
$$D(S_1[1...i], S_2[1...j]) = \begin{cases} D(S_1[1...i], S_2[1...j-1]+1) - Insert \\ D(S_1[1...i-1], S_2[1...j]+1) - Delete \\ D(S_1[1...i-1], S_2[1...j-1] + \begin{bmatrix} 0,\ if\ S_1[i] = S_2[j] - Match \\ 1,\ else - Replace \end{bmatrix} \end{cases}$$

Если используется модифицированное расстояние, то добавляется дополнительное условие:

$$D(S_1[1...i-2], S_2[1...j-2]+1,\ if\ S_1[i] = S_2[j-1]\ and\ S_1[i-1] = S_2[j]$$

Способы:

1. Матричный:
   При матричном способе для каждой следующей клетки выбирается минимальное расстояние в одном из укзанных направлений:



2. Рекурсивный:
   Раскрываем рекурсивно по формулам каждую из 4 операций.

Часть 4

Технологическая часть

4.1 Листинг кода

Расстояние Левинштейна, матричный способ:

```
def levinstein(s1, s2):
    n_matrix = len(s1) + 1
    m_matrix = len(s2) + 1
    matrix = [[0] * m_matrix for i in range(n_matrix)]

    for i in range(n_matrix):
        matrix[i][0] = i
    for j in range(m_matrix):
        matrix[0][j] = j

    for i in range(1, n_matrix):
        for j in range(1, m_matrix):
            matrix[i][j] = min(matrix[i - 1][j] + 1, matrix[i][j - 1] + 1,
            matrix[i - 1][j - 1] + (s1[i - 1] != s2[j - 1]))

    for i in range(n_matrix):
        for j in range(m_matrix):
            print(matrix[i][j], end = ' ')
        print()
    return matrix[n_matrix - 1][m_matrix - 1]
```

Расстояние Дамерау-Левинштейна, матричный способ:

```
def damerau_levinstein_matrix(s1, s2):
    n_matrix = len(s1) + 1
    m_matrix = len(s2) + 1
    matrix = [[0] * m_matrix for i in range(n_matrix)]

    for i in range(n_matrix):
        matrix[i][0] = i
    for j in range(m_matrix):
        matrix[0][j] = j

    for i in range(1, n_matrix):
        for j in range(1, m_matrix):
            if (i > 1 and s1[i - 1] == s2[j - 2] and s1[i - 2] == s2[j - 1]):
                matrix[i][j] = min(matrix[i - 1][j] + 1, matrix[i][j - 1] + 1,
                matrix[i - 1][j - 1] + (s1[i - 1] != s2[j - 1]), matrix[i - 2][j - 2] + 1)
            else:
                matrix[i][j] = min(matrix[i - 1][j] + 1, matrix[i][j - 1] + 1,
                matrix[i - 1][j - 1] + (s1[i - 1] != s2[j - 1]))

    for i in range(n_matrix):
```

```
21          for j in range(m_matrix):
22              print(matrix[i][j], end = ' ')
23          print()
24      return matrix[n_matrix - 1][m_matrix - 1]
```

Расстояние Дамерау-Левинштейна, рекурсивный способ:

```
1  def damerau_levinstein_recursive(s1, s2):
2      n = len(s1)
3      m = len(s2)
4      if (0 == n):
5          return m
6      if (0 == m):
7          return n
8
9      delete = damerau_levinstein_recursive(s1[:-1], s2) + 1
10     insert = damerau_levinstein_recursive(s1, s2[:-1]) + 1
11     replace = damerau_levinstein_recursive(s1[:-1], s2[:-1]) + (s1[-1] != s2[-1])
12     transposition = 1000
13     if (n > 1 and m > 1 and s1[-1] == s2[-2] and s1[-2] == s2[-1]):
14         transposition = damerau_levinstein_recursive(s1[:-2], s2[:-2]) + 1
15
16     return min(delete, insert, replace, transposition)
```

4.2 Заготовленные тесты

Строка1	Строка2	Ожидаемый результат
dessert	desert	1
cook	cooker	2
mother	money	3
woman	water	4
program	friend	6
house	girl	5
probelm	problem	1 or 2
head	ehda	2 or 3
bring	brought	4
happy	happy	0
minute	moment	5
person	eye	5
week	weeks	1
member	morning	6
death	health	2
education	question	4
room	moor	2
car	city	3
air	area	3
country	office	6

Часть 5

Экспериментальная(исследовательская) часть

5.1 Результаты тестов

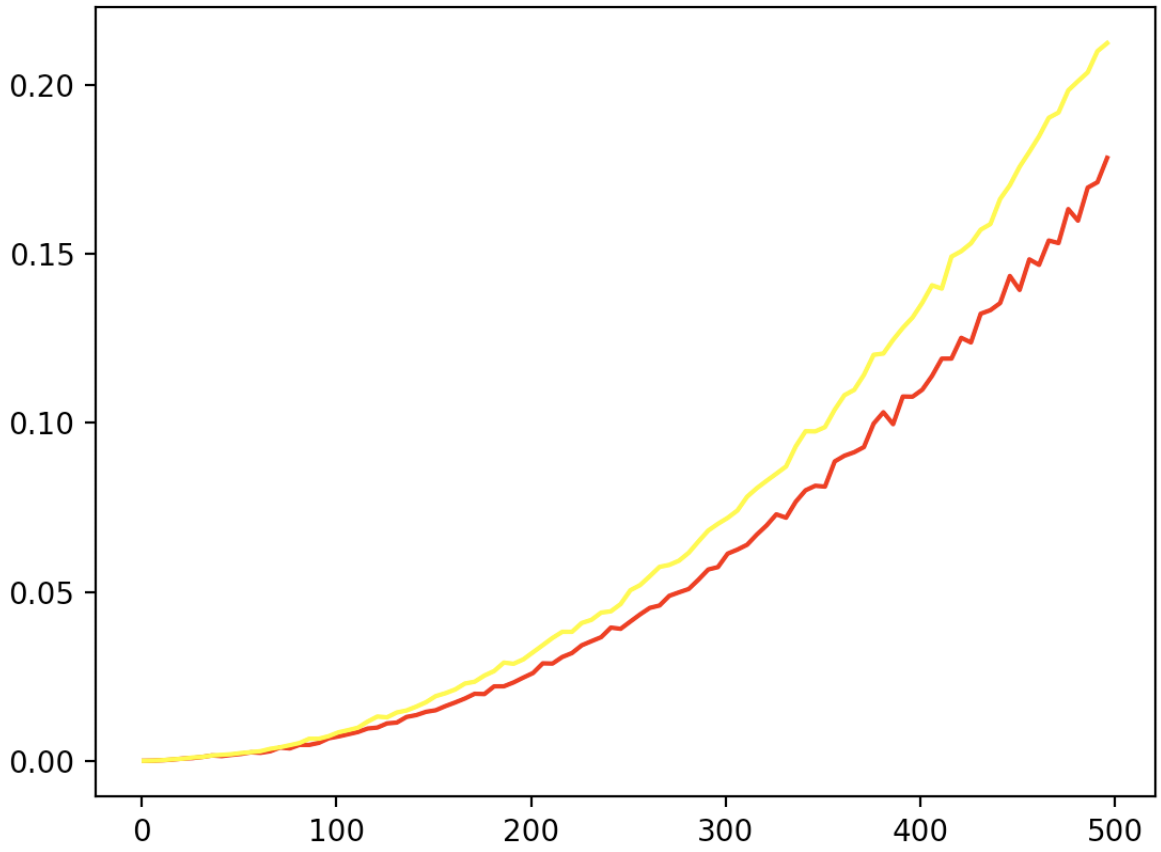
Строка1	Строка2	Л. матричный	Д.-Л. матричный	Д.-Л. рекурсивный
dessert	desert	1	1	1
cook	cooker	2	2	2
mother	money	3	3	3
woman	water	4	4	4
program	friend	6	6	6
house	girl	5	5	5
probelm	problem	2	1	1
head	ehda	3	2	2
bring	brought	4	4	4
happy	happy	0	0	0
minute	moment	5	5	5
person	eye	5	5	5
week	weeks	1	1	1
member	morning	6	6	6
death	health	2	2	2
education	question	4	4	4
room	moor	2	2	2
car	city	3	3	3
air	area	3	3	3
country	office	6	6	6

Тесты пройдены

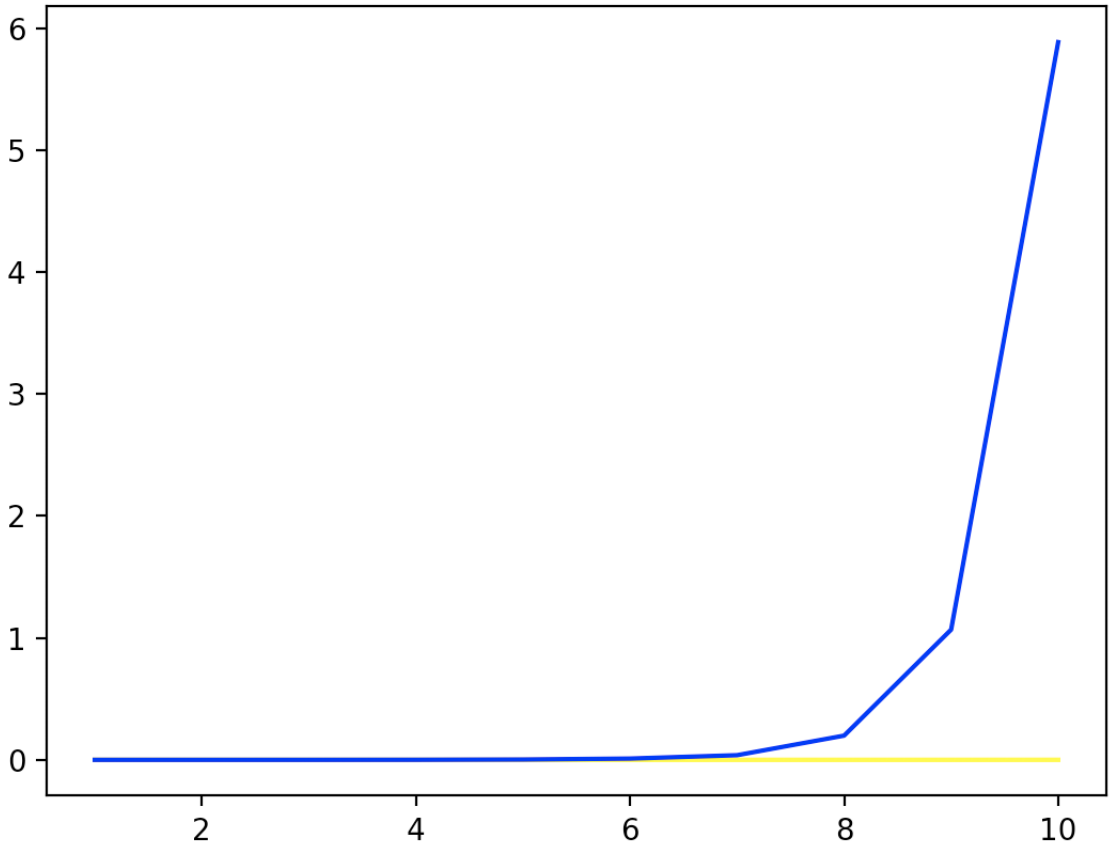
5.2 Замеры времени (графики зависимости времени от длины слова)

Графики:

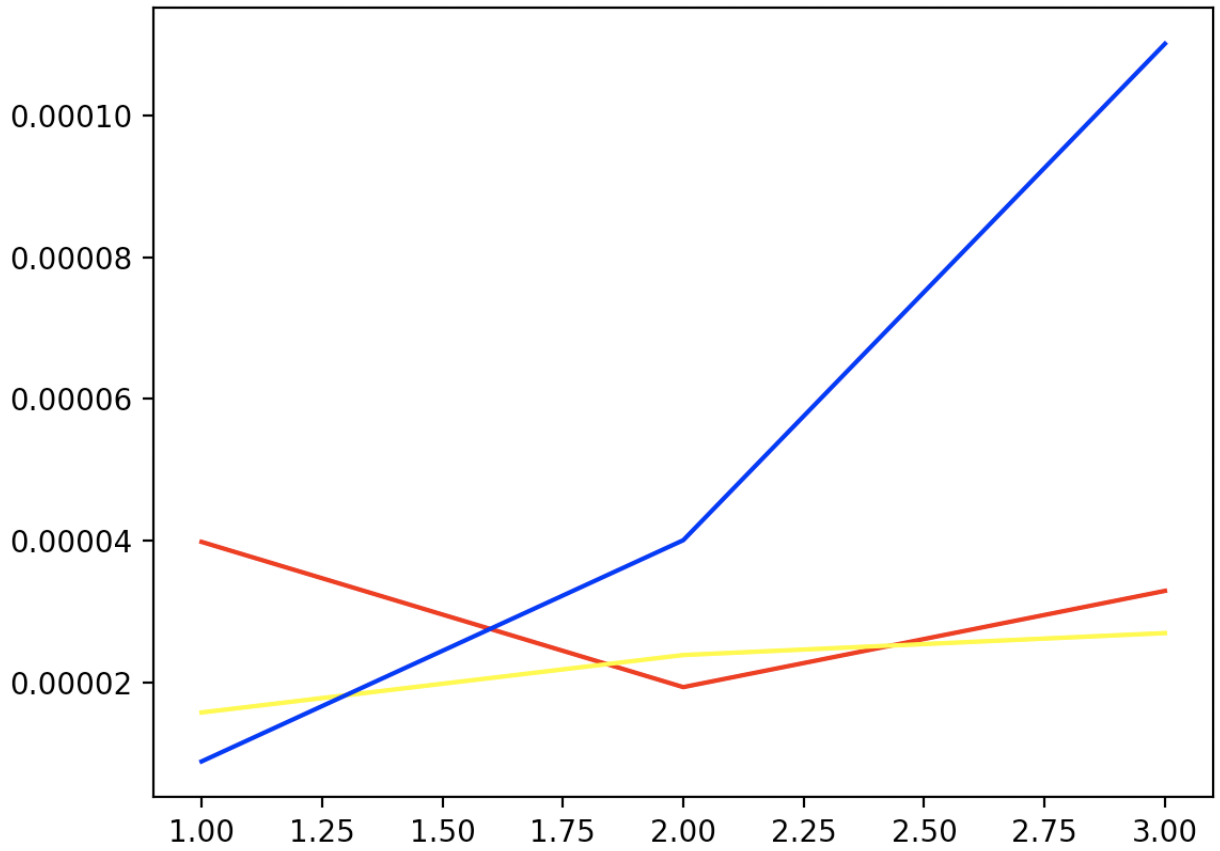
1. Левенштейн матричный – **красный**
2. Дамерау-Левенштейн матричный – **желтый**
3. Дамерау-Левенштейн рекурсивный – **синий**



Левенштейн и Дамерау-Левенштейн матричный



Дамерау-Левенштейн матричный и рекурсивный



Все 3 способа

Часть 6

Заключение

В данной лабораторной работе было реализовано 3 алгоритма:

1. нахождение расстояния Левенштейна матричным способом
2. нахождение расстояния Дамерау-Левенштейна матричным способом
3. нахождение расстояния Дамерау-Левенштейна рекурсивным способом

Применяются данные алгоритмы:

1. компьютерная, машинная лингвистика
2. биоинформатика

При сравнении данных алгоритмов были получены следующие результаты:

1. Рекурсивный алгоритм является самым медленным, гораздо быстрее использовать алгоритмы матричные.
2. Дамерау-Левенштейна проигрывает обычному Левенштейну только на очень больших длинах слов, но цена ошибки, в некоторых случаях, у него меньше.