МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики

Мегафакультет трансляционных информационных технологий

Факультет информационных технологий и программирования

Лабораторная работа №4

По дисциплине «Введение в цифровую культуру и программирование»

Исправление ошибок

Выполнил студент группы №M3109
Бабурин Тимур Сергеевич

Проверил:
Хлопотов Максим Валерьевич

САНКТ-ПЕТЕРБУРГ

2019

РАСЧЕТЫ

2. Первичные расчёты

2.1. Количество словоформ - 1549

2.2. Количество разных словоформ - 763

2.3. Количество словоформ, совпадающих со словарем, - 750

3. Количество словоформ, отсутствующих в словаре – 8

4. Расчеты после поиска и исправления ошибок:

4.1. Количество словоформ в исправленном тексте -1551

4.2. Количество разных словоформ - 758

4.3. Количество словоформ, совпадающих со словарем, - 758

5. Потенциальные ошибки:

1) внурти - внутри - 2
2) росположена - расположена -1
3) нутри - внутри - 1
4) 31пара - 31 пара - 1
5) певричную - первичную - 2
6) мото-нейроны - мотонейроны - 1
7) внутрениими - внутренними - 1
8) тогдакогда - тогда когда - 1

КОД ПРОГРАММЫ

```
# Данная функция эмулирует вставку пробела для проверки наличия слов в словаре.
def splitter(original_word, dictionary_words):
    for i in range(len(original_word)):
        left = original_word[:i]
        right = original_word[i:]
        if (left in dictionary_words) & (right in dictionary_words):
            return [left, right]

    return 0

# Данная функция подстчитывает количество словоформ в тексте, словаре и различных
словоформ в тексте.
def word_forms(words_in_text, dictionary_words):
    forms = []
    counter = 0
    print('\nСловоформы: ', len(words_in_text), '\n')
    for i in words_in_text:
        if i not in forms:
            forms.append(i)
        if i in dictionary_words:
            counter += 1
    print('Различные словоформы: ', len(forms), '\n')
    print('Количество словоформ в словаре:', counter, '\n')
    return len(forms)

# Удаляет знаки препинания.
def punctuation_replacer(orig_str):
    replaced_str = orig_str.replace(".", "").replace(",", "").replace("?",
"").replace("!", "").replace("(", "") \
        .replace(")", "").replace("»", "").replace("«", "").replace(";",
"").replace(":", "")
    return replaced_str

# Подсчитывает расстояние Левенштейна.
def levenstein_distance(word1, word2):
    n, m = len(word1), len(word2)
    if n > m:
        word1, word2 = word2, word1
        n, m = m, n
    current = range(n + 1)
    for i in range(1, m + 1):
        previous, current = current, [i] + [0] * n
        for j in range(1, n + 1):
            add = previous[j] + 1
            delete = current[j - 1] + 1
            change = previous[j - 1]
            if word1[j - 1] != word2[i - 1]:
                change += 1
            current[j] = min(add, delete, change)

    return current[n]

# Открытие файлов на чтение.
dict_filename = '/Users/tima_/Desktop/Education/Digital Culture/lab4/dict.txt'
text_filename = "/Users/tima_/Desktop/Education/Digital Culture/lab4/brain.txt"
with open(text_filename, "r") as original_text:
    original_words = punctuation_replacer(original_text.read().lower()).split()
print(original_words)

dictionary = {}

for line in open(dict_filename, "r"):
    data = line.split()
```

```
    dictionary[data[0]] = data[1]
keys = list(dictionary.keys())
word_forms(original_words, keys)
errors = []
for word in original_words:
    if word not in keys:
        errors.append(word)

print(errors, '\nКоличество ошибок в тексте: ', len(errors))
corrections = []
split_corrections = []
for error in errors:
    min_distance = 3
    correct = {}
    splitted = splitter(error, keys)
    quantity = 0
    if splitted != 0:
        correct[error] = splitted
        split_corrections.append(correct)
        print(correct, "Расстояние: ", 1)
        continue
    for right_word in keys:
        if levenstein_distance(error, right_word) < min_distance:
            correct[error] = right_word
            min_distance = levenstein_distance(error, right_word)
            quantity = int(dictionary[right_word])
        else:
            if (levenstein_distance(error, right_word) == min_distance) & (quantity >
int(dictionary[right_word])):
                correct[error] = right_word
                min_distance = levenstein_distance(error, right_word)
                quantity = int(dictionary[right_word])

    if min_distance < 3:
        print(correct, 'Расстояние:', min_distance)
        corrections.append(correct)
corrections_list = []
for i in corrections:
    corrections_list.append(list(i.keys())[0])
for i in split_corrections:
    corrections_list.append(list(i.keys())[0])
for i in corrections_list:
    errors.remove(i)

if errors:
    print(errors)
    for i in errors:
        min_distance = 1000
        for j in keys:
            if levenstein_distance(i, j) < min_distance:
                min_distance = levenstein_distance(i, j)
        print('Слово не исправлено: ', i, ", расстояние: ", min_distance)
else:
    print("Все ошибки исправлены")

with open(text_filename, "r") as original_text:
    text = original_text.read()
for i in corrections:
    if list(i.keys())[0] in text:
        text = text.replace(list(i.keys())[0], i[list(i.keys())[0]])
    if i[list(i.keys())[0]].replace(list(i.keys())[0], i[list(i.keys())[0]]) in text:
        text = text.replace(i[list(i.keys())[0]].replace(list(i.keys())[0],
i[list(i.keys())[0]]), i[list(i.keys())[0]])

for i in split_corrections:
```

```
    punctuation = ".,();:»«!?"
    pattern = i[list(i.keys())[0]][0] + i[list(i.keys())[0]][1]
    if pattern in text:
        text = text.replace(pattern, i[list(i.keys())[0]][0] + " " +
i[list(i.keys())[0]][1])
    for char in punctuation:
        pattern = i[list(i.keys())[0]][0] + char + i[list(i.keys())[0]][1]
        if pattern in text:
            text = text.replace(pattern, i[list(i.keys())[0]][0] + char + " " +
i[list(i.keys())[0]][1])

original_words = punctuation_replacer(text.lower()).split()
word_forms(original_words, keys)
print(text)
```