Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет

по заданию №3
«Расстояние Левенштейна»

по дисциплине «Компьютерная лингвистика»

Автор: Лакиза Александр Николаевич

Факультет: ИКТ

Группа: К3242

Преподаватель: Чернышева Анастасия Владимировна

Санкт-Петербург 2021

Цель работы: ознакомиться с таким алгоритмом, применяемом в обработке языка, как Расстояние Левенштейна

Ссылка на исходный код и json файл с корпусом: https://github.com/alexanderlakiza/cs224
Вся работа была сделана в файлах **task3.py** и **task3_ui.py**. JSON-файлы, которые были созданы в процессе работы **corpus_as_dict.json**, **corpus_as_dict_of_norms.json**, **each_token_appearance.json**, **unique_norms_in_corpus.json**.

Ход Работы: Для начала посмотрим файл **task3.py**. Мне необходимо создать словарь, в котором для каждого токена из корпуса будет подставлено значение с id документов, в котором этот токен встречается. Для этого сначала надо предобработать текст.

```
for i in range(len(texts)):
    texts[i] = re.sub(r'\n', ' ', texts[i])  # Удаление символа переноса строки

    while len(re.findall(r'\([^()]*\)', texts[i])) != 0:  # Удаление скобок и пробелов
        texts[i] = re.sub(r'\([^()]*\)', '', texts[i])
        texts[i] = re.sub(r'\s+(?=(?:[,.?!:;…]))', '', texts[i])
        texts[i] = re.sub(r'\s+', ' ', texts[i])
    # Заменяем диакритические знаки
    texts[i] = del_diacritics(texts[i])
```

Для этого я использую методы, описанные в Задании 2. Затем я записал весь корпус (где документы отредактированы) в словарь *corpus_as_dict.json* (В этом словаре ключ = заголовок, значение = отредактированный текст документа).

Далее я подгружаю свой корпус, записанный в одну строку, сделанный в Задании 2. Также чищу его на всякий случай. И получаю список всех уникальных токенов, привожу их все к начальной форме с помощью pymorphy2. Записываю этот список в *unique_norms_in_corpus.json*.

```
def del_punct(tokens_list):
    punct = string.punctuation + "—" + "«" + "»"
    return [token.lower() for token in tokens_list if token not in punct]


tokens = word_tokenize(corpus)
tokens = del_punct(tokens)
normed_tokens = [morph.parse(i)[0].normal_form for i in tokens  # Список начальных форм всех токенов
                 if str.isalpha(morph.parse(i)[0].normal_form)]
    print(normed_tokens)
    unique_norm_tokens = list(set(normed_tokens))
    with open("unique_norms_in_corpus.json", "w") as f:
```

```
        # Запишем корпус в виде словаря в json, чтобы потом использовать готовый словарь
        # вместо использования pymorphy2
        json.dump(unique_norm_tokens, f, ensure_ascii=False)
```

Далее я записываю корпус в еще один словарь. В этом словаре заголовок документа = ключ, а значение = список токенов в начальной форме этого документа. Записываю в *corpus_as_dict_of_norms.json*

```
main_normed_dict = {}
md_keys = list(main_dict.keys())
md_values = list(main_dict.values())
for i in range(len(main_dict)):
    tokened_value = md_values[i]
    tokened_value = word_tokenize(tokened_value)
    tokened_value = del_punct(tokened_value)
    normed_tokens = [morph.parse(i)[0].normal_form for i in tokened_value
                     if str.isalpha(morph.parse(i)[0].normal_form)]
        main_normed_dict[md_keys[i]] = normed_tokens
    with open("corpus_as_dict_of_norms.json", "w") as f:
        # Запишем корпус в виде словаря в json, чтобы потом использовать готовый словарь
        # вместо использования pymorphy2
        json.dump(main_normed_dict, f, ensure_ascii=False)
```

И последнее, что я делаю в файле [task3.py](): Я создаю необходимый нам словарь, в котором для каждого токена, использованного в тексте в разных формах слова, ставится в значение словарь, в котором ключ = количество использований токена, а значение = заголовки документов, в которых появляется токен.

```
with open("unique_norms_in_corpus.json") as f:
    unique_norm_tokens = json.load(f)

with open("corpus_as_dict_of_norms.json") as f:
    main_normed_dict = json.load(f)

word_usage = {}
for word in unique_norm_tokens:
    counter = 0
        docs_include_word = {}
        list_docs_include_word = []
        for i in range(len(main_normed_dict)):
```

```
            if word in list(main_normed_dict.values())[i]:
                # print('Номер документа:', i)
                c = list(main_normed_dict.values())[i].count(word)
                counter += c
                list_docs_include_word.append(list(main_normed_dict.keys())[i])
        docs_include_word[counter] = list_docs_include_word
        word_usage[word] = docs_include_word
    with open("each_token_appearance.json", "w") as f:
        # Запишем корпус в виде словаря в json, чтобы потом использовать готовый словарь
        json.dump(word_usage, f, ensure_ascii=False)
```

Записываем этот словарь в *each_token_appearance.json*.

Стоит отметить, что в исходном коде на гитхабе многие этапы будут закомментированы. Это было сделано для того, чтобы ускорить процесс работы – вместо того, чтобы каждый раз пробегаться по всем тысячам токенов и ждать ~2 минуты, я записывал промежуточные этапы в json файлы. Поэтому их и получилось 4 новых.

Переходим к task3_ui.py.

Я чутка видоизменил функцию check_spelling(text), которая была нам продемонстрирована на паре, добавив в нее пару условий.

```
def check_spelling(text):
    domain = "https://speller.yandex.net/services/spellservice.json"
    words = text.split()
    if len(words) == 1:
        request = requests.get(domain + "/checkText?text=" + words[0])
        if requests:
            if not request.json():
                for i in range(len(words[0])):
                    if ord(words[0][i]) < 1040:
                        return 'В слове "{}" присутствуют буквы не русского языка'.format(words[0])
                    return 'В слове "{}" нет ошибки'.format(words[0])
                else:
                    return request.json()[0]["word"], request.json()[0]["s"]
```

```
            else:
                return None
        elif len(words) > 1:
            words = "+".join(words)
            request = requests.get(domain + "/checkText?text=" +
words)
            if requests:
                if not request.json():
                    words = words.split("+")
                    for word in words:
                        for i in range(len(word)):
                            if ord(word[i]) < 1040:
                                return 'В слове "{}"
присутствуют буквы не русского языка'.format(word)
                    return 'В словах {} нет
ошибок'.format(words)
                else:
                    response = [(i["word"], i["s"]) for i in
request.json()]
                    return response
            else:
                return None
        return None
```

Однако в дальнейшем понял, что мне не понадобится эта функция, так как заключается немного в другом. Я решил написать функцию, в которой мы используем Расстояние Левенштейна, следующим образом: пользователь вводит слово в начальной, если оно есть среди токенов, то программа без задней мысли выводит сколько раз встречается этот токен, и в каких джокументах. Если же токен не встречается (в введённом слове допущена ошибка или этого токена просто нет в документах), то программа пробегается по всем токенам и выбирает тот, у которого Расстояние Левенштейна с введённым словом наименьшее.

```
if __name__ == "__main__":
    print('Данная программа покажет вам названия документов, в
которых встречается введенное вами слово\n\n'
          'Введите ваше слово в начальной форме: ')
    word = input()
    if word in list(word_usage.keys()):
        show(word)
    else:
        print('Слово "{}" в документах не
наблюдается\n'.format(word))
```

```
        print('Возможно вы имели в виду слово "{}"\n'.format(fix_flaw(word)))
            fixed_word = fix_flaw(word)
            show(fixed_word)
```

Функция show(text):

```
with open("each_token_appearance.json") as f:
    word_usage = json.load(f)


def show(word):
    dict_of_docs = word_usage[word]
    number_of_mentions = list(dict_of_docs.keys())[0]
    print('Слово "{}" встречается {} раз в документах: {}'.format(word, number_of_mentions,
                                        dict_of_docs[number_of_mentions]))
```

Функция fix_flaw(text):

```
def fix_flaw(word):
    # options = check_spelling(word)[1]
    lev_distances = {}
    for i in list(word_usage.keys()):
        lev_distances[i] = lev.distance(word, i)
    """
    Реализация исправления ошибки через Яндекс спелл чекер
    for i in options:
        lev_distances[i] = lev.distance(word, i)
     """
        minimum = list(lev_distances.values()).index(min(list(lev_distances.values())))
        best_option = list(lev_distances.keys())[minimum]
        return best_option
```

Как видно из кода, в кавычках находится реализация выбора правильного варианта введенного пользователем слова через Яндекс спелл чекер. Я использовал готовую реализацию Расстояния Левенштейна из библиотеки python-Levenshtein.

Примеры работы программы:

```
alexanderlakiza@alela-KPL-W0X:~/pyfiles/cs224$ workon cs224
(cs224) alexanderlakiza@alela-KPL-W0X:~/pyfiles/cs224$ python task3_ui.py
Данная программа покажет вам названия документов, в которых встречается введенное вами слово

Введите ваше слово в начальной форме:
корабль
Слово "корабль" встречается 58 раз в документах: ['Балласт на надводных плавучих средствах', '
Бимс', 'Бортовая качка', 'Ватерлиния', 'Верфь', 'Водоизмещение', 'Вьюшка', 'Капитан судна', 'К
аюта', 'Килевая качка', 'Киль', 'Лавировка', 'Лоцман', 'Марсовая площадка', 'Морские термины',
 'Нактоуз', 'Непотопляемость', 'Осадка', 'Палуба', 'Полубак', 'Рулевой', 'Седловатость', 'Снач
ала женщины и дети', 'Спасательная шлюпка', 'Судовой руль', 'Сухой док', 'Те-Солент', 'Титаник
 Белфаст', 'Титаник-Белфаст', 'Титаник. Легенда продолжается', 'Тоннаж', 'Трюм', 'Форштевень',
 'Швартов', 'Швартовка', 'Шлюпбалка', 'Шпангоут', 'Штормтрап', 'Штраус-парк', 'Якорь']
(cs224) alexanderlakiza@alela-KPL-W0X:~/pyfiles/cs224$ python task3_ui.py
Данная программа покажет вам названия документов, в которых встречается введенное вами слово

Введите ваше слово в начальной форме:
кайтх
Слово "кайтх" в документах не наблюдается

Возможно вы имели в виду слово "уайт"

Слово "уайт" встречается 12 раз в документах: ['Джозеф Брюс Исмей', 'Номадик', 'Сначала женщин
ы и дети', 'Те-Солент', 'Уайт Стар Лайн']
(cs224) alexanderlakiza@alela-KPL-W0X:~/pyfiles/cs224$ python task3_ui.py
Данная программа покажет вам названия документов, в которых встречается введенное вами слово

Введите ваше слово в начальной форме:
аккомпанемент
Слово "аккомпанемент" в документах не наблюдается

Возможно вы имели в виду слово "компонент"

Слово "компонент" встречается 12 раз в документах: ['Бинокль', 'Дымогарные трубы', 'Жаровые тр
убы', 'Латунь', 'Минеральная вода', 'Мороженое', 'Сахар', 'Тонзиллит острый', 'Электрическое п
оле']
(cs224) alexanderlakiza@alela-KPL-W0X:~/pyfiles/cs224$
```