МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)
Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Отчет по лабораторной работе №1
«Токенизация»
по курсу
«Обработка текстов на естественном языке»

Группа: 80-106М
Выполнил: Демин И.А.
Преподаватель: Калинин А.Л.

Москва, 2019

Задание

Нужно реализовать процесс разбиения текстов документов на токены, который потом будет использоваться при индексации. Для этого потребуется выработать правила, по которым текст делится на токены. Необходимо описать их в отчёте, указать достоинства и недостатки выбранного метода. Привести примеры токенов, которые были выделены неудачно, объяснить, как можно было бы поправить правила, чтобы исправить найденные проблемы. В результатах выполнения работы нужно указать следующие статистические данные:

• Количество токенов.

• Среднюю длину токена.

Кроме того, нужно привести время выполнения программы, указать зависимость времени от объема входных данных. Указать скорость токенизации в расчете на килобайт входного текста. Является ли эта скорость оптимальной? Как её можно ускорить?

Выполнение лабораторной работы

Токенизацию было решено проводить при помощи языка Python. После поиска информации о ток, как это можно было сделать, была найдена библиотека nltk. С ее помощью было получено неплохое разбиение, однако обнаружены следующие проблемы:

1 - римские цифры

2 - транслит

3 - производные слова

4 - пустые слова - решено. проверка при создании Counter

5 - стоп-символы

6 - слова-сокращения

7 - тире, скобки

8 - перечисления

Первая версия программы.

```
def tokenize_me(file_text):
    words = nltk.word_tokenize(file_text)
```

```
    functors_pos = {'CONJ', 'ADV-PRO', 'CONJ', 'PART'}  # function words
    return Counter([word.lower() for word, pos in nltk.pos_tag(words, lang='rus')
        if pos not in functors_pos])
```

Данный код был полностью взят из примеров работы nltk. Однако он очень долго работает и имеет плохое качество токенизации.

Время работы: 0:04:04

Пример токенов для article-1:

```
{
    "(": 33,
    ")": 34,
    ",": 333,
    ".": 145,
    ".несмотря": 1,
    "000": 1,
    "1-е": 1,
    "1-м": 2,
    "10": 1,
    "123,2": 1,
    "14": 1,
    "1500": 1,
    "16": 1,
    ...
}
```

Во второй версии я попытался решить часть проблем, введением дополнительных проверок.

Вторая версия программы решила слудующие проблемы:

5 - пустые слова - решено. проверка при создании Counter

6 - символы '«': 36, '»': 36 - решено. доп проверка

8 - тире, скобки — решено.

```
def tokenize_me(file_text):
    tokens = nltk.word_tokenize(file_text)
    tokens = [i.lower() for i in tokens if ( i not in string.punctuation )]
    stop_words = stopwords.words('russian')
    ext_stop_words = ['—']
    stop_words.extend(ext_stop_words)
    tokens = Counter(
        [i.replace("«", "").replace("»", "")
        for i in tokens if (i not in stop_words and len(i.replace("«", "").replace("»", "")))]
    )
    return tokens
```

Данная версия работает быстрее. Так же стоит отметить, что был создан частотный словарь, так как его создание обеспечивает исключение повторяющихся слов. Создание списка требует дополнительных преобразований, а если попытаться воспользоваться set(), то у него нет доступного для записи метода сериализации.

Отдельно хотел упомянуть про встречающиеся цифры. Было решено не удалять их из токенов, так как при поиске по спортивной категории поиск по датам имеет место быть, а отделать цифры от дат очень сложно.

Пример токенов, полеченных с помощью первой версии программы находятся в папке data-tokens1, а второй — в data-tokens.

Пример токенов для article-1:


```
{
    ".несмотря": 1,
    "000": 1,
    "1-е": 1,
    "1-м": 2,
    "10": 1,
```

"123,2": 1,

"14": 1,

"1500": 1,

}

Статистические данные для работы финальной версии программы

Time: 0:00:56

Tokens count: 248162

Average length: 8

Вывод

В ходе выполнения лабораторной текст был разбит на токены. Я узнал о способах токенизации на языке Python, познакомился с библиотекой nltk.