

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 естественного языка»

Группа: М80-106М

Выполнил: Забелин М. К.

Преподаватель: Калинин А.Л.

Москва, 2019

Лабораторная работа №1. Токенизация

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 объёма входных данных. Указать скорость токенизации в расчёте на килобайт входного текста. Является ли эта скорость оптимальной? Как её можно ускорить?

На языке Python есть очень известная библиотека для решения проблем в области обработки естественного языка – `nltk`. В ней есть функция `word_tokenize`. Однако мне не понравилось, как она справляется с разбиением текста на слова, так как скобки и знаки препинания она воспринимает в качестве полноценных слов. Например, для статьи “100 крупнейших авиакатастроф в США” самыми популярными токенами оказались:

“.”, “,”, “в”, “(”, “)”

Проанализировав нежелательные токены, я включил их все в один список, и перед записью в файл проверяю текущий токен на вхождение в этот список, и пишу в файл только если его нет в списке. Никаких потерь от того, что знаков препинания в виде отдельных токенов, нет, так как вряд ли кому-то в голову придет мысль искать их. Также было решено не писать повторяющиеся токены в файл (до этого одни и те же предлоги встречались много раз). Поэтому было решено хранить в одном экземпляры токены. Для этого вместо списка использую `set`. Это позволило ускорить процесс в 1.5 раза.

Время работы чистой `nltk`: 460240.139 ms

Время работы `nltk` с моей валидацей токенов и использованием `set`: 433539.292 ms.

Средняя длина токена без валидации: 7.382708744555641 символа.

Средняя длина токена с валидацией: 7.521757931058175 символа.

Кол-во токенов без валидации: 5115846

Кол-во токенов с валидацией: 5006772

Соответственно, токены всех документов можно посмотреть в корне проекта в папках `bad_tokens` (с токенами в виде отдельных символов) и `good_tokens` с токенами без повторений и без нежелательных токенов.

Вывод

В ходе выполнения лабораторной я познакомился с библиотекой `nltk` для Python и выделил ее слабые стороны.

Ссылка на Git репозиторий

<https://github.com/Mikhail-Z/MAI/tree/master/sem10/Informational%20Search/nlp1>