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

Отчет по лабораторной работе №3
«Булев Индекс»
по курсу
«Информационный поиск»

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

Москва, 2019

Задание

Требуется построить поисковый индекс, пригодный для булева поиска, по подготовленному в ЛР1 корпусу документов.

Требования к индексу:

· Самостоятельно разработанный, бинарный формат представления данных. Формат необходимо описать в отчёте, в побайтовом представлении.

· Формат должен предполагать расширение, т.к. в следующих работах он будет меняться под требования новых лабораторных работ.

· Использование текстового представления или готовых баз данных не допускается.

· Кроме обратного индекса, должен быть создан «прямой» индекс, содержащий в себе как минимум заголовки документов и ссылки на них (понадобятся для выполнения ЛР4)

· Для термов должна быть как минимум понижена капитализация. В результатах работы должна быть указаны статистическая информация о корпусе:

В отчёте должно быть отмечено как минимум:

· Выбранное внутренне представление документов после токенизации.

· Выбранный метод сортировки, его достоинства и недостатки для задачи индексации.

Среди результатов и выводов работы нужно указать:

· Количество термов.

· Средняя длина терма. Сравнить со средней длинной токена, вычисленной в ЛР1 по курсу ОТЕЯ. Объяснить причину отличий.

· Скорость индексации: общую, в расчёте на один документ, на килобайт текста.

· Оптимальна ли работа индексации? Что можно ускорить? Каким образом? Чем она ограниченна? Что произойдёт, если объём входных данных увеличится в 10 раз, в 100 раз, в 1000 раз?

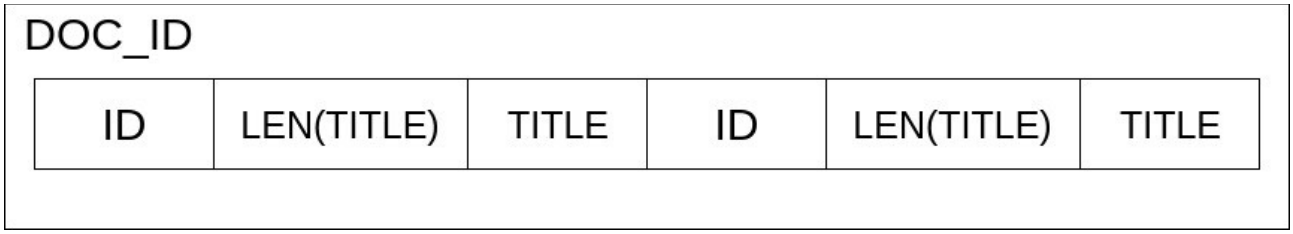
Выполнение лабораторной работы

Для выполнения лабораторной работы использовался язык Python с использованием библиотек pickle, struct, hashlib и структура defaultdict. Токены

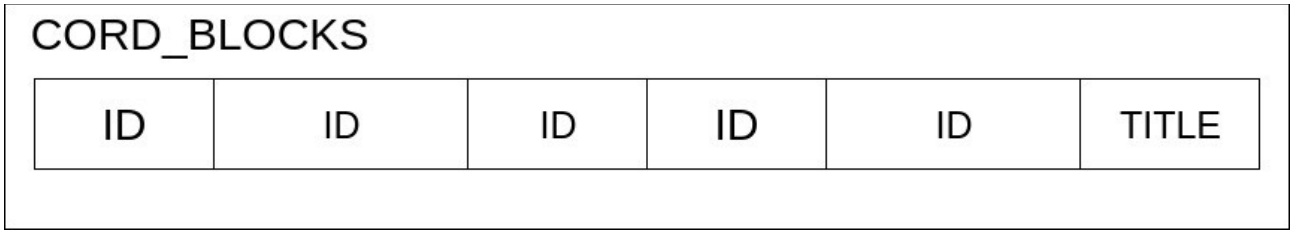
хешировались при помощи hashlib.sha1 с ограничением хеша для 8-ми байтной записи.

Представление файлов.

DOC_ID хранит в является прямым индексом. Хранит в себе id, len(title), title. Url я решил не хранить в целях экономии, так как для всех статей префикс одинаковый и его можно получить конкатенацией префикса и title.



CORD_BLOCKS — файл, в котором находятся номера последовательно записанные номера документов. По ним будет производиться поиск.



INVERT_INDEX — представляет из себя бинарный файл с сериализованным при помощи pickle словаря. Было решено хранить его так, так как pickle обеспечивает достаточно высокую степень сжатия и обеспечивает быструю загрузку файла в структуру. Словарь выглядит следующим образом.

{

hash(word): (offset_in_file, count_of_elements),

...

}

Работа скрипта

Проверка проводилась на тестовом множестве из 3х тысяч статей. Для них получены следующие результаты для 3х тысяч элементов:

komp@komp-G5:~/stud/search_MAI/l5_index$ python l5.py

expected_size: 114291

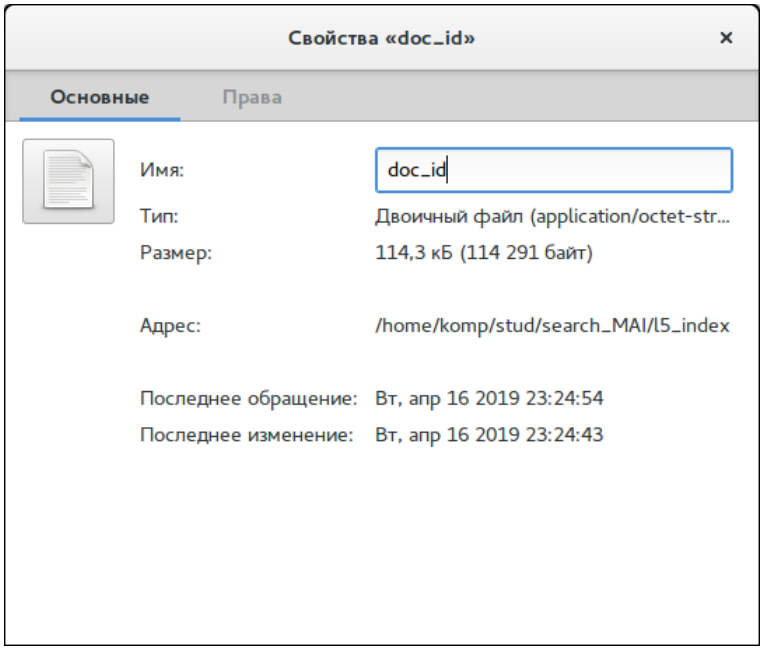
create_doc_id_files works 0:00:00.014318

expected_size: 16516008

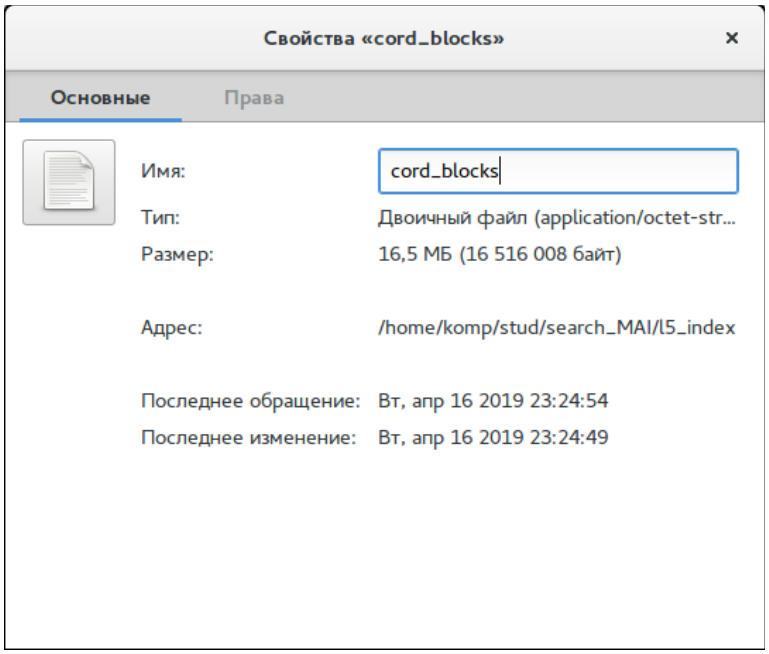
create_raw_invert_index works 0:00:05.811640

save_obj works 0:00:00.028661

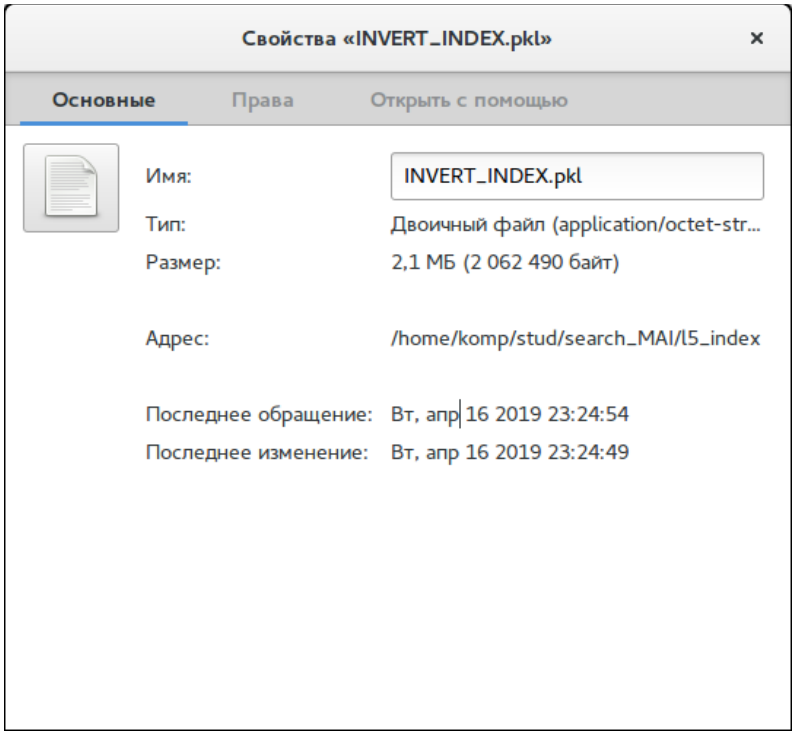
Ожидаемый размер, который должен занимать файл с doc_id - 114291 байт



С координатными блоками - 16516008



Файл, хранящий инвертированный индекс.



Статистические данные.

Средняя длинна токенов — 6. Средняя длина терминов — 8. Это объясняется тем, что в токенах большое число высокочастотных слов имеют малую длину, из-за этого понижается среднее значение. Сортировка в ЛР не применялась, так как из-за особенностей используемых структур языка Python, данные не сортируются, но хранятся в практически отсортированном порядке. Происходит это из-за того, что set() является надстройкой над hash map, и значения в нем хранятся по хешу, а для положительных целых чисел хеш, который считает python под капотом, распределяет эти числа в отсортированном порядке. Средняя скорость получилась в районе 1000 статей/сек. Учитывая создание структур и тот факт, что python — интерпретируемый язык, то считаю это приемлемой скоростью.

Вывод

В ходе лабораторной работы была выполнена индексация корпуса документов для булева поиска. Изучены инструменты Python для работы с бинарными файлами. Весь процесс индексации 15ти тысяч статей занял около 15-ти секунд.