Московский Авиационный Институт
(Научный Исследовательский Институт)

Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Отчёт по лабораторной работе №3 по курсу «Информационный поиск»

Выполнил: Ефименко Н. А.

Группа: 8О-106М

Преподаватель: Калинин А.Л.

Москва, 2020

ЛР3: Булев индекс

Постановка задачи:
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?

Этапы выполнения:

- Выбрать структуру файлов
- Преобразовать файлы полученные на предыдущем этапе в выбранную структуру ﬧ Выбор метода сортировки
- Реализация
- Тестирование
- Расчет статистической информации
- Анализ решения

Структура файлов:

Выходные данные это три бинарных файла. Которые представляют прямой и обратный индекс. Обратный индекс – словарь и файл смещения токенов. Прямой индекс содержит информацию о статьях.

Обратный индекс. Файл представляет собой набор из следующих записей <term_hash><offset><count>, где term_hash – это 32 байта, которые мы получаем применением хэш-функции md5 к термину (термины могу быть разного размера, так мы ограничиваем хранимого «слепка» термина — тем самым уменьшая размер файла, однако теряем по времени т.к появляется слой md5) offset количество байт, для отступа от начала - словаря, для того, чтобы начать считывать информацию о документах в которых встречается данный токен, count – это количество записей, которые содержат информацию о термине. Для того, чтобы найти статьи, содержащие определенный термин, необходимо найти его хэш в индексе, считать смещение и count, затем сместиться в файле-словаре на offset байт и считать count записей из словаря. Offset и count занимает 4-8 байта (на моей системе 8байт). Таким образом, на один термин в обратном индексе приходится 48 байт.

Словарь - вспомогательная структура для обратного индекса. Состоит из записей вида - <doc_id><position>. Где doc_id – это 4 байта, который определяют идентификатор документа. Так как количетсво документов влезает в представление unsigned int. Для преобразования Python пердставление int в его байтовое представление, использовалась функцию int.to_bytes. tf, df – это частота слова, и обратная частота документа соответственно, необходим для оценки релевантности результата для запроса, tf и df занимает по 4 байта каждый. Position – это так же 4 байта, которые определяют позицию вхождения термина в документ. Таким образом на одно вхождение термина в статью приходится 16 байт.

Прямой индекс содержит информацию о документах в корпусе. Его структура следующая: <doc_id><offset><title_len><url…<title><url>. doc_id – 4 байта, хранящие идентификатор документа. Offset – сдвиг относительно начала словаря, на который нужно переместиться, чтобы считать урл и заголовок, занимает 4 байта. title_len – два байта, содержащие длину заголовка статьи. Url_len – 4 байта, содержащие длину ссылки на статью. title – заголовок статьи с длинной title_len. Url – ссылка на статью, длиной url_len.

Тестирование

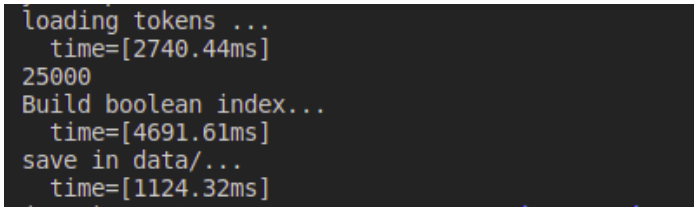
В ходе тестирования был построен индекс по корпусу документов из ЛР1

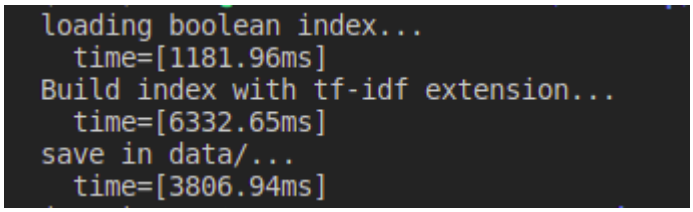
Статистическая информация

Токенов:	36937361 / 14646464
AVG	6.1050 / 7.2154

Изменения в количестве и средних длин связаны с тем, что все токены, повторяющиеся в пределах одного документа, были удалены во втором случае, поскольку такие, наиболее

часто встречающиеся слова, которые как правило имеют меньшую длину, были удалены, то средняя длина была увеличина. Кроме того привод всех термов к одному регистру дал возможность получить больше дублей, которые были исключены, а средняя длина увеличина.





Среднее время индексации на объеме данных 163МБ (25000 статей) составило 6.332сек.

(расширение по готовому булеву индексу)

Среднее время индексации на объеме данных 163МБ (25000 статей) составило 4,691сек.

(булев индекс)

Среднее время индексации на объеме данных 72МБ (10000 статей) составило 2.1сек.

Среднее время индексации на объеме данных 6МБ (1000 статей) составило 0.31мсек.

Копирование такого файла(163МБ) занимает около 5сек. время является приемлемым, однако улучшаемым.

Анализ решения

Полученное решение обладает существенным недостатком, не обрабатывается ситуация, когда данных из tokens становиться настолько много, что они не помещаются в RAM. Решение – переходить к внешней сортировке.

Вывод:

В ходе выполнения лабораторной работы была разработана и реализована модель индексации документов. Полученное решение было проанализировано.