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

Отчет по лабораторной работе №2
«Булев поиск»
по курсу
«Информационный поиск»

Группа: М80-108М-17
Выполнил: Забарин Н.И.
Преподаватель: А.Л. Калинин

Москва, 2018

Задание

Реализовать ввод поисковых запросов и их выполнение над индексом, получение поисковой
выдачи.

Синтаксис поисковых запросов:

- Пробел или два амперсанда, «&&», соответствуют логической операции «И».
- Две вертикальных «палочки», «||» – логическая операция «ИЛИ»
- Восклицательный знак, «!» – логическая операция «НЕТ»
- Могут использоваться скобки.

Для демонстрации работы поисковой системы должен быть реализован веб-сервис, реализующий базовую функциональность поиска из двух страниц:

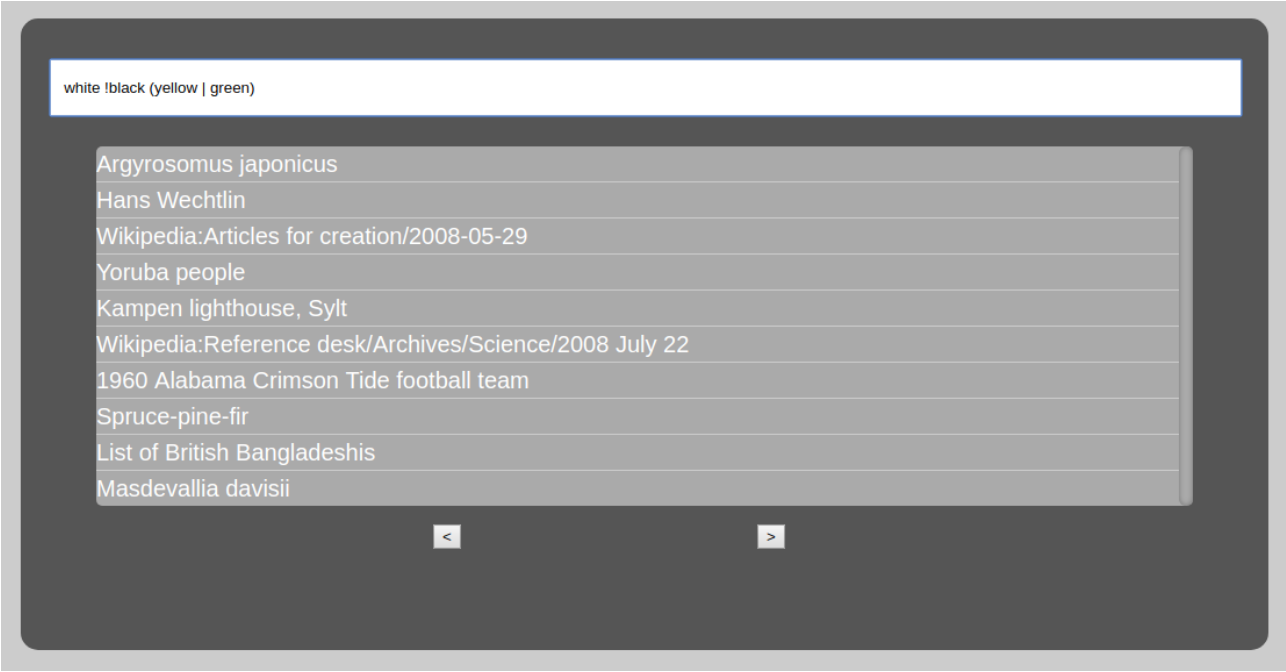
- Начальная страница с формой ввода поискового запроса.
- Страница поисковой выдачи, содержащая в себе форму ввода поискового запроса, 50 результатов поиска в виде текстов заголовков документов и ссылок на эти документы, а так же ссылку на получение следующих 50 результатов.

Метод решения

Веб-интерфейс написан на HTML+CSS+JQuery, backend составляющая реализована на Flask.

Frontend

Написана простая страничка, без каких либо особенностей. А так же скрипт на JavaScript который обрабатывает переходы на другие страницы и выполнение нового поискового запроса. CSS взят из личных запасов накопленных за прошлый семестр по курсу Сетевого Программирования.



Периодически среди статей встречается «мусор», как на 3 и 6 строках поисковой выдачи на изображении выше.

Backend

API состоит из двух запросов search и get_results. Первый принимает запрос обрабатывает и сохраняет результат на стороне сервера, возвращает id поиска *(случайный набор символов)*. Второй запрос принимает id поиска и номер страницы для отображения.

Во время составления корпуса документов мной не были сохранены в прямом индексе ссылки на статьи Википедии, поэтому запрос get_results обращается к API Википедии с целью получения ссылок на статью. Из-за этого второй запрос обрабатывается порядка 3-4 секунд, при времени обработки первого запроса в 25 мс.

Данные передаются в формате JSON.

search/	200	xhr	Other	313 B	25 ms
0	200	xhr	jquery.min.js:4	1.8 KB	3.49 s
1	200	xhr	jquery.min.js:4	1.5 KB	3.62 s
0	200	xhr	jquery.min.js:4	1.8 KB	3.51 s
1	200	xhr	jquery.min.js:4	1.5 KB	3.50 s
2	200	xhr	jquery.min.js:4	1.6 KB	3.59 s

На скриншоте выше сначала выполняется быстрый поисковый запрос*(он заканчивает выполняться когда поиск закончен и сохранен в кэш)*, далее идут несколько запросов результатов с разными страницами.

Обработка поискового запроса

Обработка запроса проходит в 3 этапа:

- «Очистка» запроса, то есть приведение в нижний регистр, удаление всех символов за исключением !, |, (,), букв латинского алфавита и цифр.
- Построение обратной польской записи с помощью конечного автомата.
- Обработка ОПЗ.

Таким образом поисковый запрос разбивается на отдельные термы, и поиск каждого терма происходит по отдельности. Для этого используется выгруженный в память на старте API заголовочный файл и бинарный поиск по файлу индекса.

Часть лога обработки поискового запроса «white !black (yellow | green)»

['white', 'black', '!', '&', 'yellow', 'green', '|', '&']
[2018.04.25 14:52:21] Search for "white" in ../data/index_tmp/tindex_20
[2018.04.25 14:52:21] Search finished, 203187 results, 0.05408525466918945 sec.
[2018.04.25 14:52:21] Search for "black" in ../data/index_tmp/tindex_20
[2018.04.25 14:52:22] Search finished, 184871 results, 0.04060697555541992 sec.
[2018.04.25 14:52:22] Search for "yellow" in ../data/index_tmp/tindex_20
[2018.04.25 14:52:22] Search finished, 59403 results, 0.010336160659790039 sec.
[2018.04.25 14:52:22] Search for "green" in ../data/index_tmp/tindex_20
[2018.04.25 14:52:22] Search finished, 121117 results, 0.03022170066833496 sec.

Как можно заметить поиск отдельного токена выполняется очень быстро, в худшем случае порядка 10-15мс*(напрямую зависит от количества вхождений, сам поиск нужного блока в файле выполняется почти мгновенно)*, при размере файла с индексом в 4,1 ГБ. В первой строке лога находится ОПЗ которую я получил после разбора запроса.

После обработки отдельных токенов идентификаторы по каждому поиску кладутся в set`ы и над ними выполняются логические операции, для учета логического отрицания написан класс хранящий сет и «знак» сета и перегружены все необходимые операции.

Выводы

Из-за потери ссылок на статьи смена страниц происходит очень долго*(3-4 сек)*, сам поиск при этом выполняется за сравнительно короткое время, если сделать случайный запрос в Google из нескольких слов, можно увидеть среднюю задержку порядка 300-400мс, то есть скорость поиска приемлема.

Без какого либо ранжирования результатов поиска очень сложно найти какую то конкретную статью.