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

Отчет по лабораторной работе №4
«Булев Поиск»
по курсу
«Информационный поиск»

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

Москва, 2019

Задание

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

Выполнение лабораторной работы

Для выполнения работы была немного изменена структура запросов, так как использовались логические операции над set(). По этому для поисковых запросов синтаксис следующий:

1. Пробел или один амперсанд, «&», соответствуют логической операции «И»

2. Одна вертикальных «палочки», «|» – логическая операция «ИЛИ»

3. «!» – логическая операция «НЕ»

Условие устойчивости к количеству пробелов осталось неизменным.

Примеры поисковых запросов из задания выглядят следующим образом:

1. [московский авиационный институт]

2. [(красный | желтый) автомобиль]

3. [руки !ноги]

Работа поиска происходит в консольной утилите.

Поиск работает следующим образом:

0. При запуске утилиты загружается инвертированный индекс.

1. Пользователь вводит поисковый запрос.

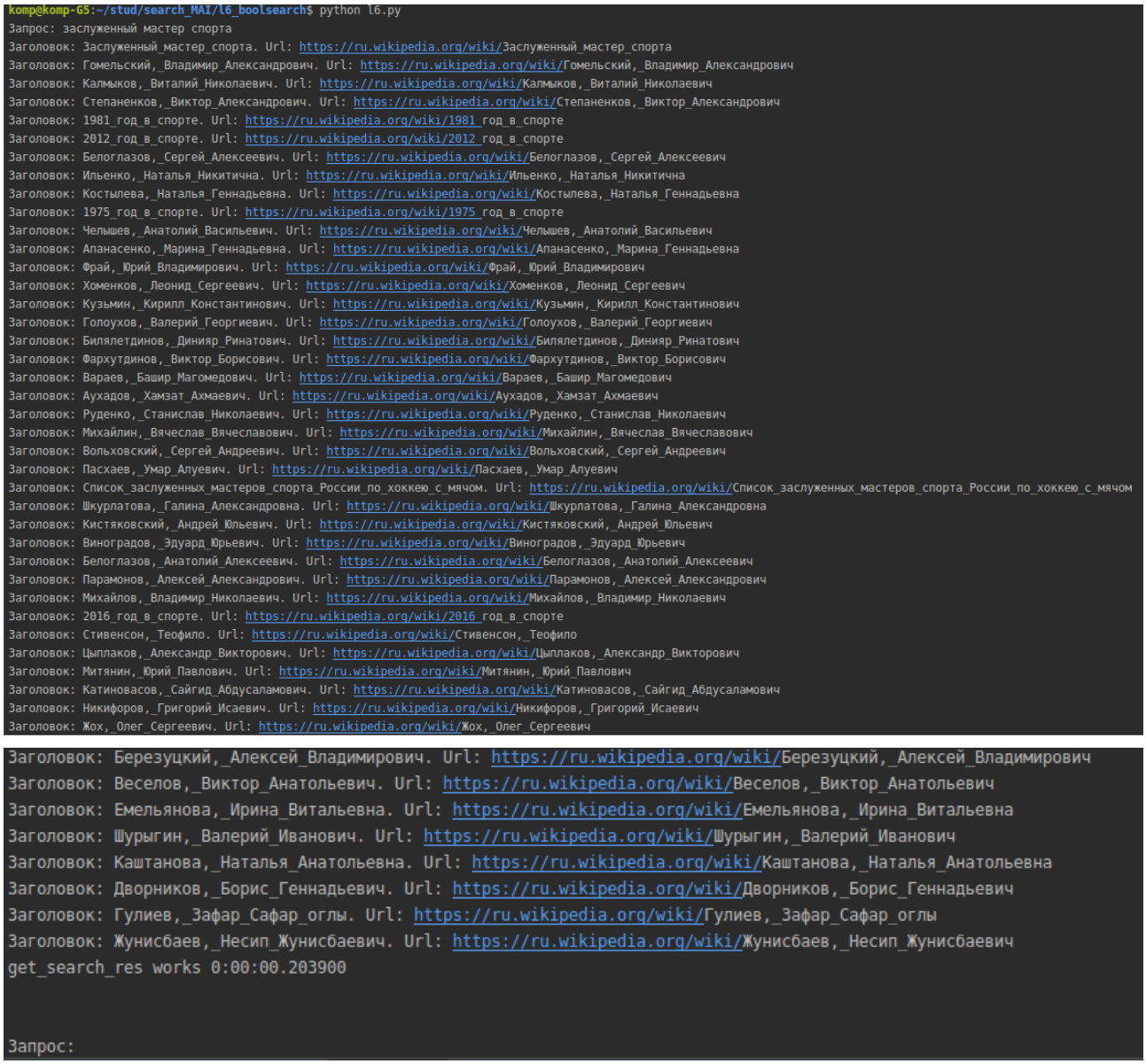
2. Запрос проходит предпроцессинг запроса:

- удаляются лишние пробелы
- происходит замена символа пробела на « & »
- происходит замена символа «!» на « - »
- запрос разбивается на отдельные слова

3. Для каждого слова получается hash, происходит поиск в инвертированном индексе и затем из бинарных файлов получаем статьи и формируем выдачу. (Ограничения по выдаче было решено не делать, для проверки результатов. В случае необходимости добавляется в одну строчку кода).

Описание работы получения статей. Для этого был использована встроенная функция python eval(), которая берет строку, переданную ей в качестве аргумента, и превращает ее в исполняемый код. В качестве такой строки передается запрос, в котором на место слова ставится набор doc_id в виде set(). После этого производятся логические операции над set-ами и формируется результат. Учтены случаи, когда одни слова являются частью другими слов, например «известный» и «в». Так же не производится повторный поиск по уже обработанным словам.

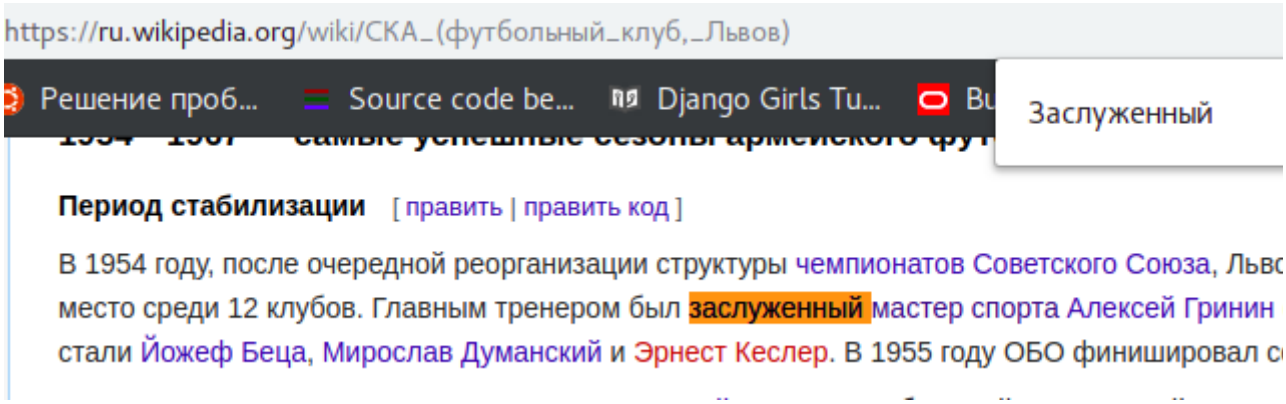
Проверка работы осуществлялась при помощи сравнения руками. Использовался встроенный поиск в PyCharm.



Результат для поискового запроса «заслуженный мастер спорта» - время работы 2 мск — это время поиска + вывода в консоль данных. Время работы поиска - 0.002 с. Запрос является не сложным.

```
(base) ivan@ivan-G5:~/study/search/search_MAI/l6_boolsearch$ python l6.py
Запрос: заслуженный мастер спорта
Результат получен за 0:00:00.002327
```

Для наглядности не будем использовать поиск PyCharm, а воспользуемся поиском на странице wiki. Первый запрос слишком очевиден, возьмем статью про https://ru.wikipedia.org/wiki/СКА_(футбольный_клуб,_Львов).



На странице все 3 этих слова встречаются.

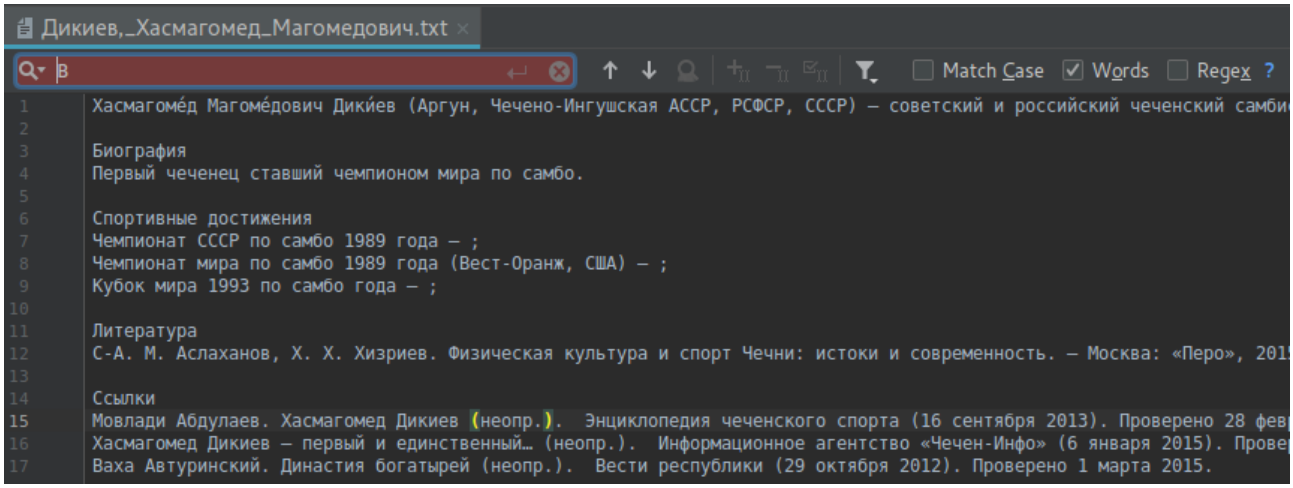
Попробуем запрос «(заслуженный мастер) !спорта». Количество таких статей должно быть очень малым.

```
komp@komp-G5:~/stud/search_MAI/l6_boolsearch$
Запрос: (заслуженный мастер) !спорта
get_search_res works 0:00:00.007720
```

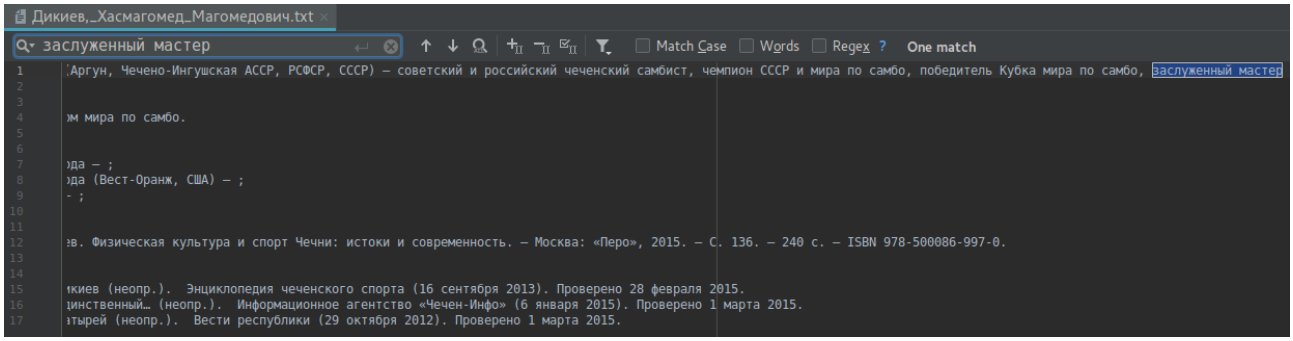
Как видно - ожидания оправдались. Поисковая выдача равна 0. PyCharm — так же не нашел файлов с таким сочетанием (использовался поиск по регулярным выражениям).

Заменим слово «спорта» на какое-то высокочастотные слово — например «в».

```
Запрос: (заслуженный мастер) !в
Заголовок: Дикиев,_Хасмагомед_Магомедович. Url: https://ru.wikipedia.org/wiki/Дикиев,_Хасмагомед_Магомедович
get_search_res works 0:00:00.011679
```



Была найдена одна статья. На этот раз я приведу пример поиска используя PyCharm, так как на web-странице не в тексте статьи есть предлог «в».



Теперь приведем время поиска статей с высокочастотными словами. Используем запрос «оно | в | на | он | я». В результат вошла почти вся тестовая выборка.

```
Заголовок: Летние_Олимпийские_игры_1940. Url: https://ru.wikipedia.org/wiki/Летние_Олимпийские_игры_1940
Заголовок: Панамериканские_игры_1971. Url: https://ru.wikipedia.org/wiki/Панамериканские_игры_1971
Заголовок: Чемпионат_мира_по_крикету_2011. Url: https://ru.wikipedia.org/wiki/Чемпионат_мира_по_крикету_2011
Заголовок: Виера,_Юсниер. Url: https://ru.wikipedia.org/wiki/Виера,_Юсниер
Заголовок: Григорьев,_Владимир_Яковлевич. Url: https://ru.wikipedia.org/wiki/Григорьев,_Владимир_Яковлевич
Заголовок: Зимние_Паралимпийские_игры_2010. Url: https://ru.wikipedia.org/wiki/Зимние_Паралимпийские_игры_2010
Заголовок: Летние_Всемирные_военные_игры_2015. Url: https://ru.wikipedia.org/wiki/Летние_Всемирные_военные_игры_2015
Заголовок: Список_главных_тренеров,_выигравших_Кубок_УЕФА_и_Лигу_Европы_УЕФА. Url: https://ru.wikipedia.org/wiki/Список_главных_тренеров,_выигравших_Кубок_УЕФА_и_Лигу_Европы_УЕФА
Заголовок: Чемпионат_СССР_по_тяжёлой_атлетике_1986. Url: https://ru.wikipedia.org/wiki/Чемпионат_СССР_по_тяжёлой_атлетике_1986
Заголовок: Бадминтон_на_Европейских_играх_2019. Url: https://ru.wikipedia.org/wiki/Бадминтон_на_Европейских_играх_2019
Заголовок: Хансен,_Алан. Url: https://ru.wikipedia.org/wiki/Хансен,_Алан
Заголовок: Гран-при_Марсельезы. Url: https://ru.wikipedia.org/wiki/Гран-при_Марсельезы
Articles count: 2705
get_search_res works 0:00:02.290107
```

Как видно — время работы почти 2 секунды. Однако стоит учитывать, что это полное время работы консольной утилиты, то есть львиная часть этого времени — это вывод запросов Так сделано для более красивого вывода

данных. Если оставить только получение поисковой выдачи без ее печати, то время выполнение — 0.01 секунды.

```
(base) ivan@ivan-G5:~/study/search/search_MAI/l6_boolsearch$ python l6.py
Запрос: оно | в | на | он | я
Результат получен за 0:00:00.013194
```

Оценка точности

Точность считается как

$$P = \frac{a}{a + b}.$$

a — количество релевантных документов

b — число нерелевантных документов.

Возьмем 5 верхних статей из выдачи для 5ти запросов из ЛР2.

"спорт экспресс" — 0.4

"виды спорта" — 0.4

"активный отдых" — 0.4

"профессиональный бокс" — 0.6

"боевые искусства" — 0.8

Средняя точность — 0.52

Вывод

В ходе лабораторной работы был реализован булев поиск. Изучены инструменты Python для работы с парсингом выражений.