**Московский авиационный институт**

**(национальный исследовательский университет)**

**Институт информационные технологии и прикладной математики**

**Кафедра вычислительной математики и программирования**

**Лабораторная работа №4 по курсу**

**«Обработка текстов на естественном языке»**

|  |  |
| --- | --- |
| Студент: | Е.М. Стифеев |
| Преподаватель: | А.А. Кухтичев |
| Группа: | М8О-109М-21 |
| Дата: | 04.12.21 |
| Оценка: |  |
| Подпись: |  |

**Москва, 2021**

# Лабораторная работа №4 «Построение сниппетов»

Необходимо добавить в поисковую систему построение цитат (сниппетов), реферирование документов, найденных по запросу.

Сниппеты должны содержать слова запроса и давать пользователю представление о том, насколько документ отвечает поисковому запросу. Длинна сниппета должна быть ограничена двумя-тремя строчками.

В отчёте нужно привести описание алгоритма построения сниппетов, примеры.

# Описание

## Корпус

Поисковая система обрабатывает запросы для корпуса документов, хранящегося на диске.

По итогам лабораторных работ по курсу с помощью веб-скрапинга по нескольким сайтам был получен корпус документов (доступен по ссылке <https://cloud.mail.ru/public/ZfkX/gccM7hnDR>), который имеет следующую структуру:

* films1.txt (94 Мб, 15000 документов, UTF-8)
* films2.txt (96 Мб, 15000 документов, UTF-8)
* films3.txt (184 Мб, 15000 документов, UTF-8)
* films4.txt (219 Мб, 15000 документов, UTF-8)
* films5.txt (322 Мб, 15000 документов, UTF-8)
* films6.txt (711 Мб, 15000 документов, UTF-8)
* films7.txt (823 Мб, 15000 документов, UTF-8)
* films8.txt (226 Мб, 15000 документов, UTF-8)
* films9.txt (67 Мб, 15000 документов, UTF-8)
* films10.txt (75 Мб, 15000 документов, UTF-8)
* films11.txt (99 Мб, 15000 документов, UTF-8)
* films12.txt (78 Мб, 15000 документов, UTF-8)
* films13.txt (41 Мб, 6109 документов, UTF-8)

Получение одного документа зачастую включало проход по нескольким html-страницам и обработку динамически подгружаемых страниц, поэтому общее количество обработанных страниц было >800’000.

Всего была обкачено три сайта:

* <https://www.kinopoisk.ru/>
* <https://shikimori.one/>
* <http://filmplace.ru/>

В каждом файле \*.txt документы хранятся следующим образом:

* 1 строка 1 документ {….}
* 2 строка 2 документ {….}
* *n* строка *n* документ {….}

Каждый документ снабжён прямой ссылкой на источник, откуда был скачен, и хранит только выделенный из html-кода текст в кодировке UTF-8. Например, 234 строка файла films1.txt выглядит так:

{"page\_url": "https://www.kinopoisk.ru/media/article/1773537/", "title": "Артур Смольянинов: «Я сомневался, что смогу сыграть ангела»", "body": "2 января в российский прокат вышла романтическая комедия Веры Сторожевой „Мой парень — ангел“, главные роли в которой исполнили Артур Смольянинов и Анна Старшенбаум. Мы подготовили небольшой видеосюжет с участием создателей картины...Студентка Саша с большим трудом верит в чудеса. Ангелу Серафиму приходится приложить немало усилий, чтобы доказать ей, что ангелы существуют. Но он не учел одного: если девушка тебе поверит, она, скорее всего, тебя полюбит.\n\n\n\n\n\n\n\nАвтор: Дарико Цулая", "comments": ""}.

## Индекс

Готовый индекс хранится в четырёх файлах (доступен по ссылке <https://cloud.mail.ru/public/wynT/adagiBjh9>):

* **docs\_id.data** (42 Мб)

Файл служит для отображения индекса документа (doc\_id) в его текстовое представление в файлах \*.txt. Поддерживается переменная длина пути до файлов с документами.

* **terms.data** (54 Мб)

Файл служит для хранения словаря с терминами и ссылок (смещений) на файл с словопозициями и координатами. Поддерживается переменная длина термина. Термины упорядочены в лексикографическом порядке.

* **postings\_list.data** (2.68 Гб)

Файл служит для хранения словопозиций и координат терминов в документе. Слопозиции упорядочены по возрастанию идентификаторов документов.

* **tf.data** (939 Мб)

Файл служит для быстрого получения компонент документа, как вектора в пространстве терминов. Он нужен для быстрого ранжирования на основе косинуса между вектором запроса и вектором документа.

Построение индекса для корпуса с учётом лемматизации терминов с помощью отечественной NLP-системы Natasha [1] занимает 4 часа при распараллеливании на 4 OMP-потока (больше не позволяет размер оперативной памяти) процессора Intel Core i7 9700K (3.6 GHz). Блочный индекс (до слияния) доступен по ссылке <https://cloud.mail.ru/public/F3Fe/eTEiUPHt6>.

## Построение сниппетов

Был реализован простейший алгоритм построения сниппетов. Пусть дан запрос в любой форме, состоящий из терминов . Для документа, для которого требуется построить сниппет, с помощью координатного индекса возвратим все координатные вхождения в него терминов из запроса и отберём из них по *одному* для каждого термина запроса. В простейшем случае можно выбрать только первое или случайное вхождение. В более сложном нужно учитывать зону вхождения: заголовок, тело, метаданные, вступление, заключение; тепловую карту документа (с помощью eye-трекинга), машинное обучение и т.д. В лабораторной работе рассматривается только первое по порядку вхождение относительно начала документа. Далее, для каждого термина в определённом радиусе (в реализации – 20 символов) выводится текст находящийся слева и справа (с учётом границ документа, других терминов, которые могу попасть в радиус). Сам термин выделяется цветов или капитализацией.

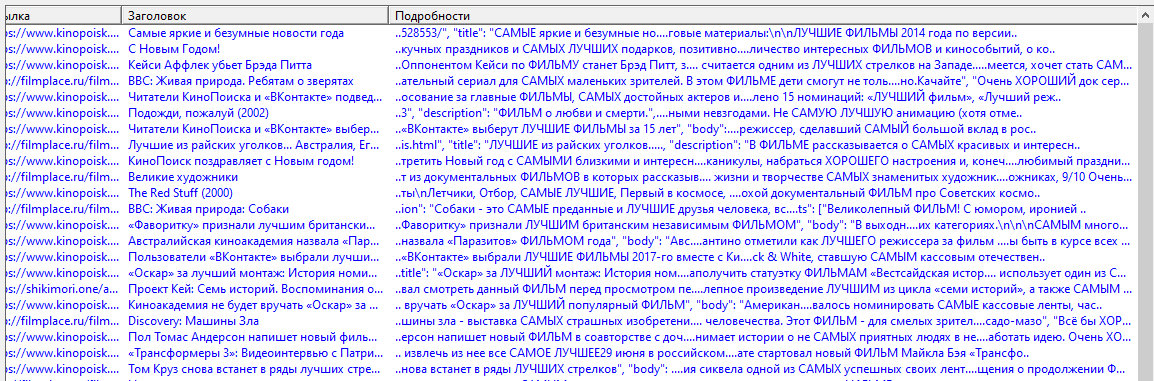


Рис. – Построение сниппетов для выборки документов по запросу «самый лучший фильм» в графическом интерфейсе

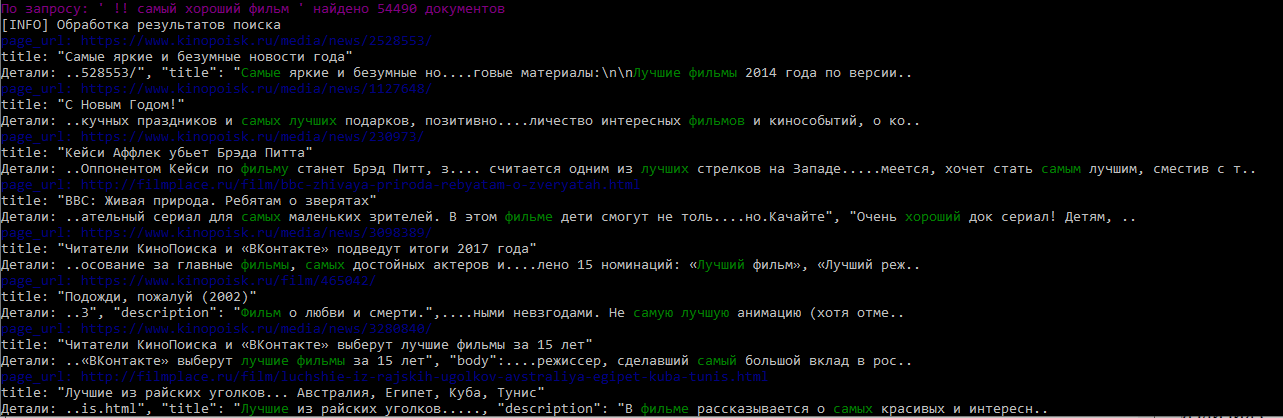


Рис. – Построение сниппетов для выборки документов по запросу «самый лучший фильм» в консольном интерфейсе

## Поисковая система. Интерфейс

Реализовано декстоп-приложение для ОС семейства Windows.

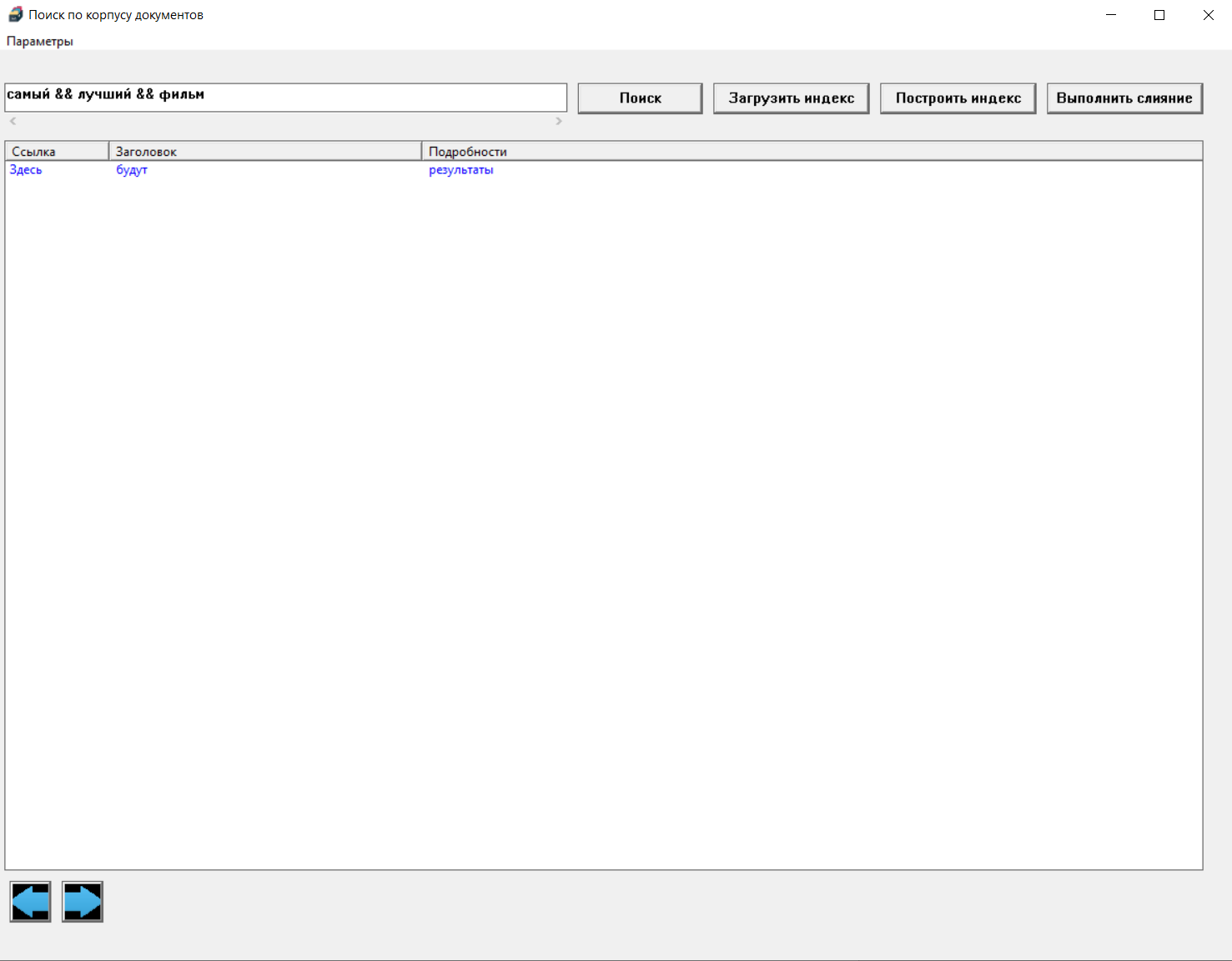


Рис. – Стартовое графическое окно приложения

При запуске приложения пользователь видит два окна: графическое (с элементами управления) и консольное – для логирования.

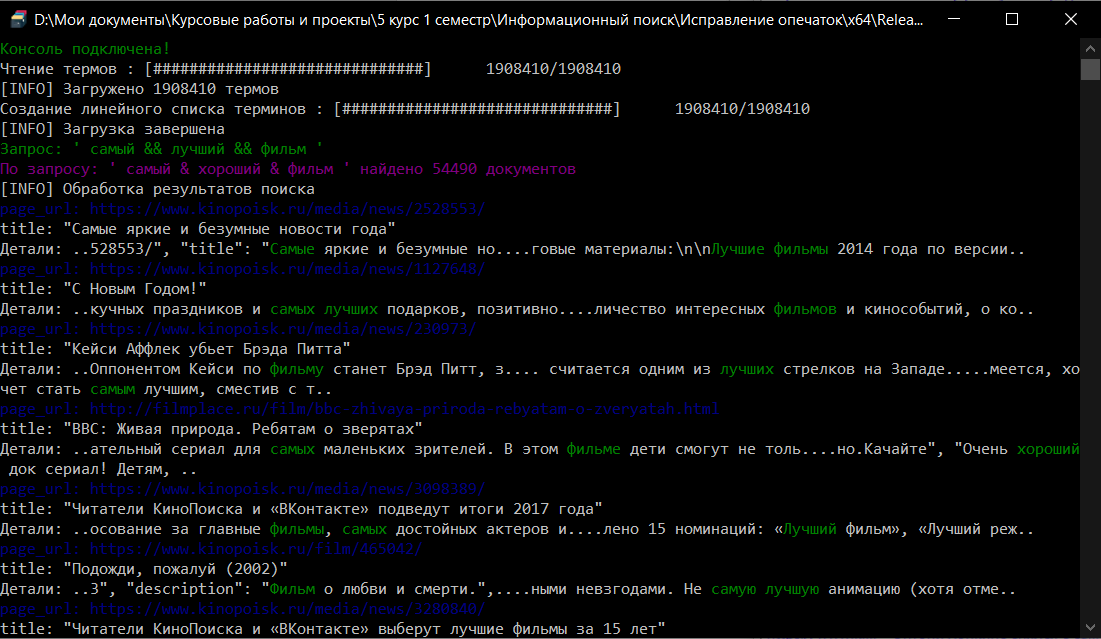


Рис. – Консольное окно приложения

При взаимодействии пользователя с системой последняя через консольное окно ведёт оповещение, чем она «занята» в данный момент. Прежде делать запрос к корпусу необходимо загрузить индекс с помощью кнопки «Загрузить индекс» или, если индекс не создан, то необходимо создать блочный индекс и затем выполнить слияние с помощью соответствующих кнопок. Корпус должен храниться в директории на диске в формате, описанном в информации о корпусе (нумерация файлов необязательна – названия - произвольные). При этом один файл с документами считается системой одним блоком, которые обрабатываются параллельно, так что к выбору размера файла нужно подходить разумно.

В меню «параметры» настраивается количество потоков и пути до нужных директорий. При закрытии и запуске приложения система запоминает все пути, количество потоков, а также последний сделанный пользователем запрос.

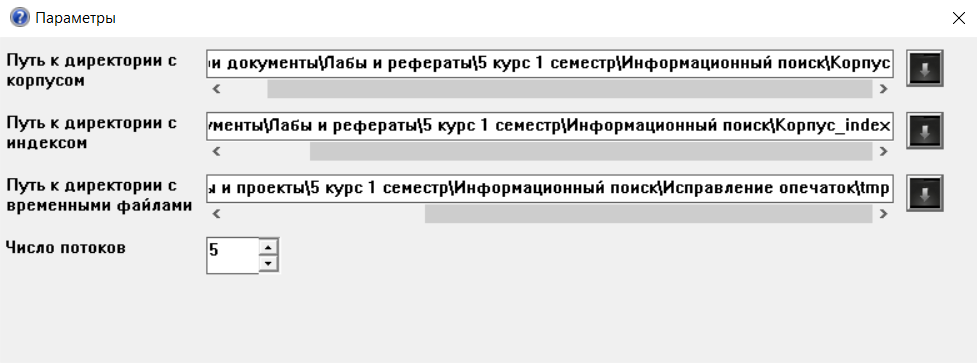


Рис. – Выбор параметров

После загрузки индекса, можно делать запросы в формах, описанных в предыдущем разделе. Результаты сортируются по косинусному правилу TF-IDF.

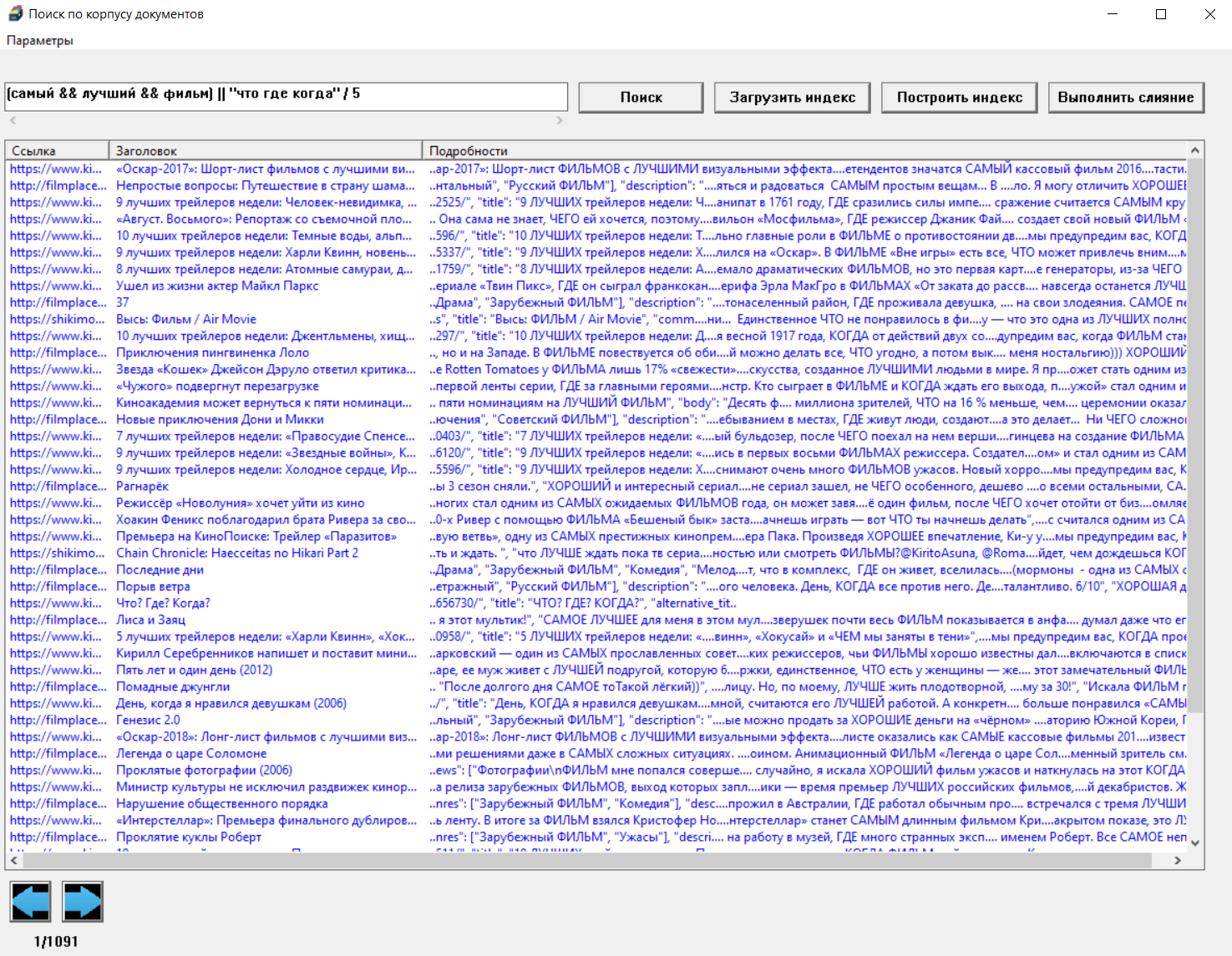


Рис. – Поисковая выдача по комбинированному запросу (графическое окно)

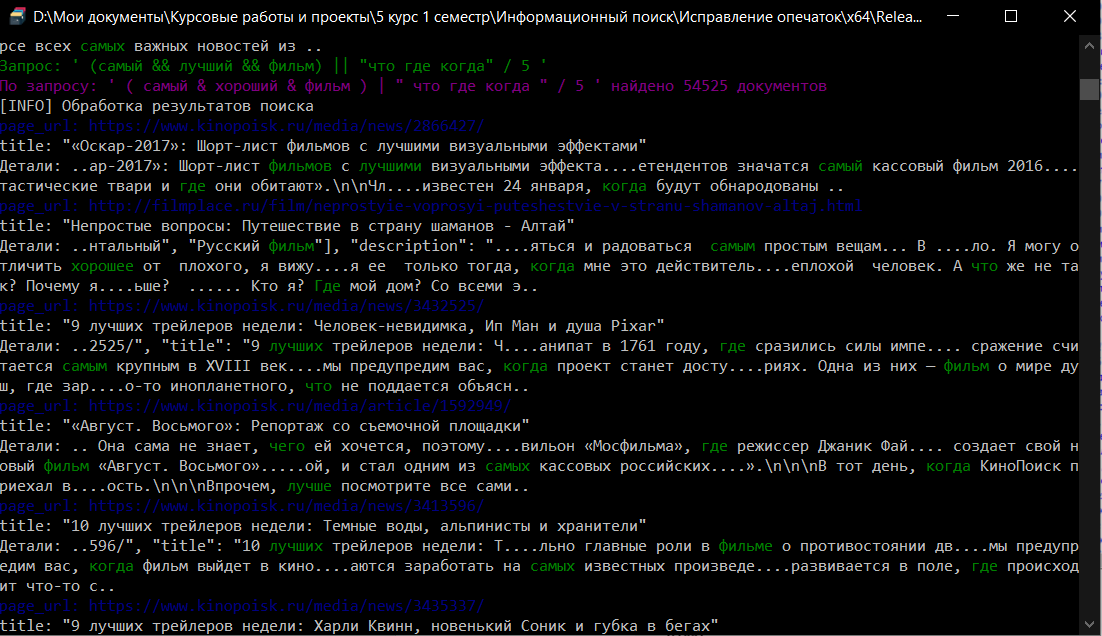


Рис. – Поисковая выдача по комбинированному запросу (консольное окно)

По результатам запроса пользователь открывает ссылку в первом столбце в любом, привычном ему, браузере.

Также существует полностью консольная версия приложения, существующая в т.ч. для тестирования и измерения метрик поисковой системы. Ей на вход в качестве аргументов программы подаются на вход все вышеуказанные параметры, а также путь к данным для просчёта метрик (см. подробности в соответствующей ЛР).

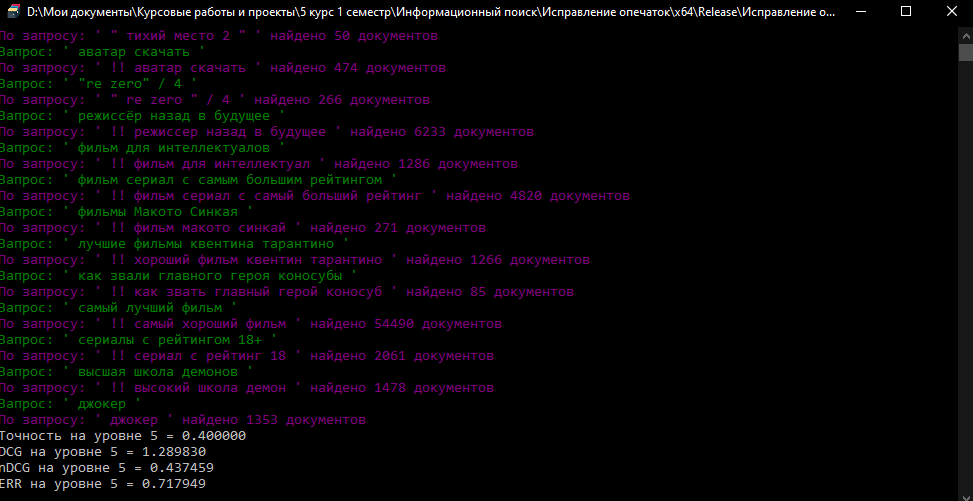


Рис. Утилита тестирования системы

# Исходный код

## Структура проекта

* include
  + algebra.hpp (простейшие операции с векторами)
  + create\_index.hpp (создание, чтение индекса)
  + defs.hpp (подключение внешних библиотек, макросы)
  + docs\_parse.hpp (извлечение полей из корпуса)
  + gui\_defs.hpp (подключение внешних библиотек, макросы, глобальные переменные)
  + gui\_params\_window.hpp (окно с выбором параметров)
  + resource.h (подключение изображений, иконок и прочего)
  + search.hpp (реализация всех видов поиска)
  + token\_parse.hpp (функции для преобразования токенов в термы)
  + typos\_correction.hpp (реализация исправления опечаток)
* python
  + lemmatizator.py (лемматизация документа)
  + lemmatizator\_setup.py (компиляция lemmatizator.py в exe-файл)
  + request\_parse.py (лемматизация запроса)
  + request\_parse\_setup.py (компиляция request\_parse.py в exe-файл)
* io
  + anwers.txt
  + requests.txt
* src
  + gui.cpp (точка входа в оконный интерфейс)
  + main.cpp (точка входа в консольный интерфейс тестирования программы)
* resourses (файлы ресурсов для оконного приложения)

Проект был написан с помощью Microsoft Visual Studio 2019 эксклюзивно для ОС семейства Windows. Исходный код доступен по <https://github.com/Stifeev/Information-retrieval/tree/main/Курсовой%20проект>.

## Запуск и сборка

Переключение между тремя точками входа осуществляется с помощью флага «исключить из сборки»:

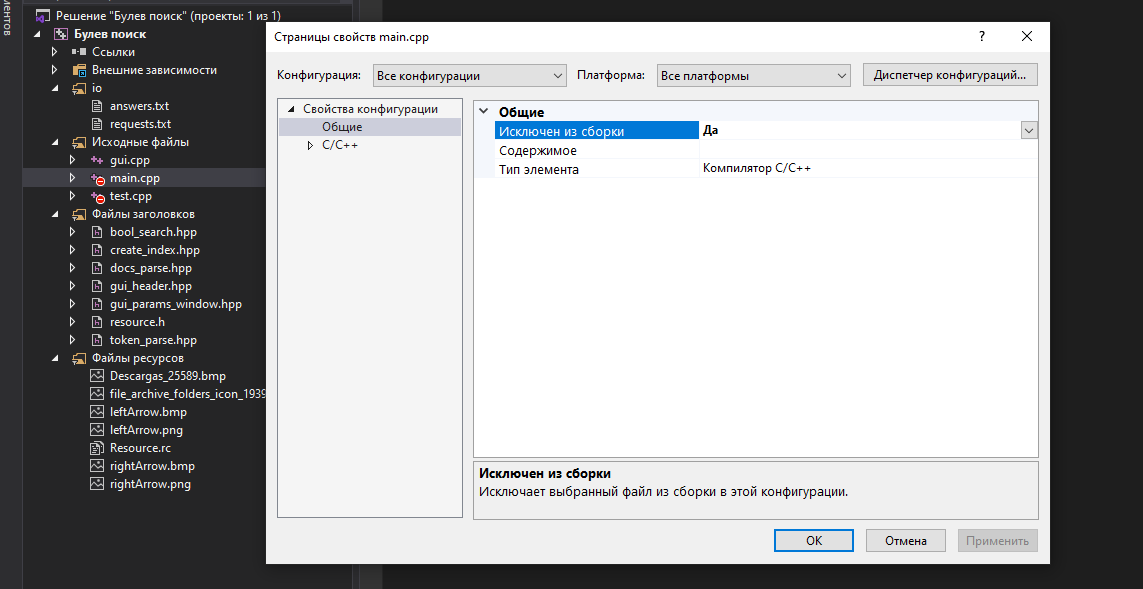


Рис. – Настойка переключения между точками входа

Не забудь при переключении между консольными и оконными приложениями менять подсистему в настройке проекта:

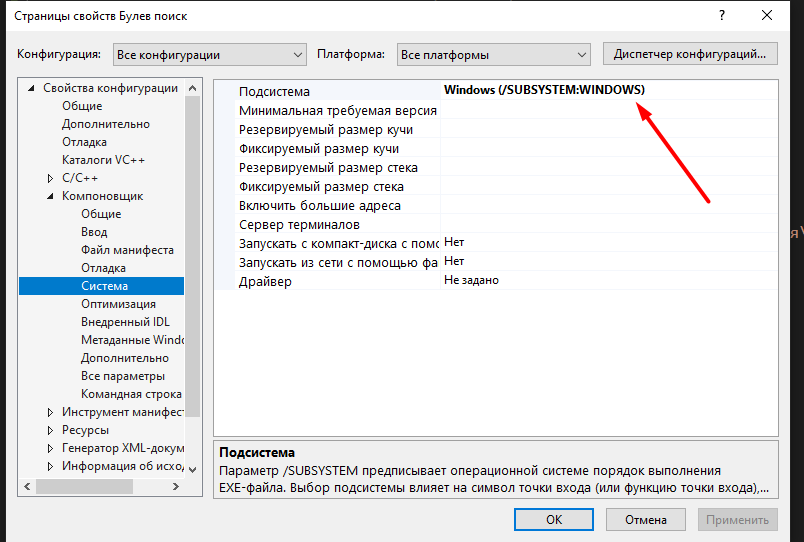


Рис. – Настройка подсистемы: консоль или окно

Консольное приложение поддерживает флаги запуска:

* -i 'путь к корпусу'
* -o 'путь к индексу'
* -t 'путь к директории с блочным индексом'
* -m 'путь к директории с эталонами для метрик'
* -p кол-во\_процессов\_для\_распараллеливания
* -create : создать блочный индекс
* -merge : выполнить слияние блочного индекса
* -clear : очистить папку с временными файлами после слияния
* -search : выполнить поиск
* -metric : высчитать метрики

**Пример создания блочного индекса из корпуса (время указано до внедрения лемматизации):**

$ ./prog.exe –p 4 –create -i "..\..\Корпус" -o -t "tmp"

**Вывод**

[INFO] Создание индекса для блоков

[INFO] Thread 0 processing block 1/13 : D:\Мои документы\Лабы и рефераты\5 курс 1 семестр\Информационный поиск\Корпус\films13.txt

[INFO] Thread 1 processing block 2/13 : D:\Мои документы\Лабы и рефераты\5 курс 1 семестр\Информационный поиск\Корпус\films9.txt

[INFO] Thread 2 processing block 3/13 : D:\Мои документы\Лабы и рефераты\5 курс 1 семестр\Информационный поиск\Корпус\films10.txt

[INFO] Thread 3 processing block 4/13 : D:\Мои документы\Лабы и рефераты\5 курс 1 семестр\Информационный поиск\Корпус\films12.txt

[INFO] Block 1 has 232216 terms

<…>

[INFO] Block 10 has 597482 terms

[INFO] Block 11 has 669497 terms

[INFO] Block 12 has 921883 terms

[INFO] Block 13 has 1383148 terms

[INFO] Создание очередей термов: 13 блок из 13

[INFO] Слияние docs\_id: 13 блок из 13

[INFO] Слияние слопозиций термов

[INFO] Осталось термов: 0

[INFO] Очистка временных файлов

[INFO] Общее число термов в словаре = 2809203

Время выполнения = 145,5 sec, размер корпуса = 2,899 Gb, документов = 186109

Средняя скорость на документ = 0,782 ms

Средняя скорость на килобайт = 0,048 ms

**Пример слияние блочного индекса:**

$ ./prog.exe –p 4 –merge –clear –i "..\..\Корпус\_index" -t "tmp"

**Вывод**

[INFO] Слияние блочного индекса

[INFO] Создание очередей термов: 13 блок из 13

[INFO] Слияние docs\_id: 13 блок из 13

[INFO] Слияние слопозиций термов

[INFO] Осталось термов: 0

[INFO] Общее число термов в словаре = 1908410

Документов = 186109

[INFO] Время на слияние блочного индекса: 35 sec

[INFO] Вычисление статистики

Первый проход. Термов осталось: 0

Второй проход. Документов осталось: 0

[INFO] Вычисление статистики закончено

**Пример просчёта метрик:**

$ ./prog.exe –p 4 -metric -i "..\..\Корпус" -o "..\..\Корпус\_index"

-m "..\..\Корпус\_metric"

**Вывод**

Чтение термов : [##############################] 1908410/1908410

[INFO] Загружено 1908410 термов

Запрос: ' "тихое место 2" '

По запросу: ' " тихий место 2 " ' найдено 50 документов

Запрос: ' аватар скачать '

По запросу: ' !! аватар скачать ' найдено 474 документов

Запрос: ' "re zero" / 4 '

По запросу: ' " re zero " / 4 ' найдено 266 документов

Запрос: ' режиссёр назад в будущее '

По запросу: ' !! режиссер назад в будущее ' найдено 6233 документов

Запрос: ' фильм для интеллектуалов '

По запросу: ' !! фильм для интеллектуал ' найдено 1286 документов

Запрос: ' фильм сериал с самым большим рейтингом '

По запросу: ' !! фильм сериал с самый больший рейтинг ' найдено 4820 документов

Запрос: ' фильмы Макото Синкая '

По запросу: ' !! фильм макото синкай ' найдено 271 документов

Запрос: ' лучшие фильмы квентина тарантино '

По запросу: ' !! хороший фильм квентин тарантино ' найдено 1266 документов

Запрос: ' как звали главного героя коносубы '

По запросу: ' !! как звать главный герой коносуб ' найдено 85 документов

Запрос: ' самый лучший фильм '

По запросу: ' !! самый хороший фильм ' найдено 54490 документов

Запрос: ' сериалы с рейтингом 18+ '

По запросу: ' !! сериал с рейтинг 18 ' найдено 2061 документов

Запрос: ' высшая школа демонов '

По запросу: ' !! высокий школа демон ' найдено 1478 документов

Запрос: ' джокер '

По запросу: ' !! джокер ' найдено 1353 документов

Точность на уровне 30 = 0.241026

DCG на уровне 30 = 2.655877

nDCG на уровне 30 = 0.289893

ERR на уровне 30 = 0.732372

# Выводы

Построение сниппетов – важная часть работы поисковой системы. С помощью них пользователь решает, какие документы открывать, а какие – нет. Качественные сниппеты экономят время пользователя, а не качественные – наоборот.

В лабораторной работе реализована простейшая их реализация, однако даже она работает и даёт представление о чём документ, стоящий в поисковой выдаче.

В ходе лабораторной работы я научился выполнять строить сниппеты для коллекции документов.

# Литература

1. <https://natasha.github.io/>
2. Кристофер Д.Маннинг, Прабхакар Рагхаван, Хайнрих Шютце. Введение в информационный поиск. 2020, изд. Вильямс.