Московский авиационный институт

(национальный исследовательский университет)

Институт информационные технологии и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №2 по курсу

«Информационный поиск»

Студент: Е.М. Стифеев
Преподаватель: А.А. Кухтичев
Группа: М8О-109М-21
Дата: 29.11.21
Оценка:
Подпись:

Москва, 2021

Лабораторная работа №2 «Оценка качества поиска»

Необходимо оценить качество своего поиска и сравнить их с двумя альтернативами (для Википедии можно собственный поиск по Википедии, поиск Google или Яндекса с ограничением по сайту Википедии). Как минимум, нужно измерить P, DCG, NDCG и ERR уровней @1, @3 и @5, приветствуется использование дополнительных метрик качества.

Для оценки качества необходимо придумать 30 запросов, отражающих интересы пользователей или, если есть доступ к настоящим запросам пользователей, то выбрать репрезентативную подборку.

В качестве примера посмотрите на 10 запросов к поиску по всей Википедии, подумайте о том, почему именно они были выбраны и какую сложность для поисковой системы они представляют:

1. [из каких книг состоит библия]
2. [что где когда]
3. [игра]
4. [российские авиазаводы]
5. [без меня народ не полный]
6. [как называют жителей набережных челнов]
7. [где короновали николая 2]
8. [товарищ прокурора]
9. [цари газы]
10. [административный кодекс]

Проведите анализ результатов оценки качества. Какие у какой поисковой системы сильные и слабые стороны? Как можно бороться с недостатками, что можно сделать, чтобы улучшить качество?

Описание

Корпус

Напомню, что корпус документов имеет следующую структуру, полученную по результатам ЛР1 (доступен по ссылке https://cloud.mail.ru/public/ZfkX/gccM7hnDR):

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

$$\Sigma_{Gb} = 2{,}899\ \text{Gb}, \Sigma_{\text{docs}} = 186109$$

Также, напомню, что получение одного документа могло включать проход по нескольким html-страницам и обработку динамически подгружаемых страниц, поэтому общее количество обработанных страниц было >800’000.

Корпус документов собран с трёх сайтов:

- 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:

- **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. Координаты упорядочены по возрастанию документа.

- **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. См. подробности в ЛР по ранжированию.

Готовый индекс доступен по ссылке https://cloud.mail.ru/public/wynT/adagiBjh9

Запросы

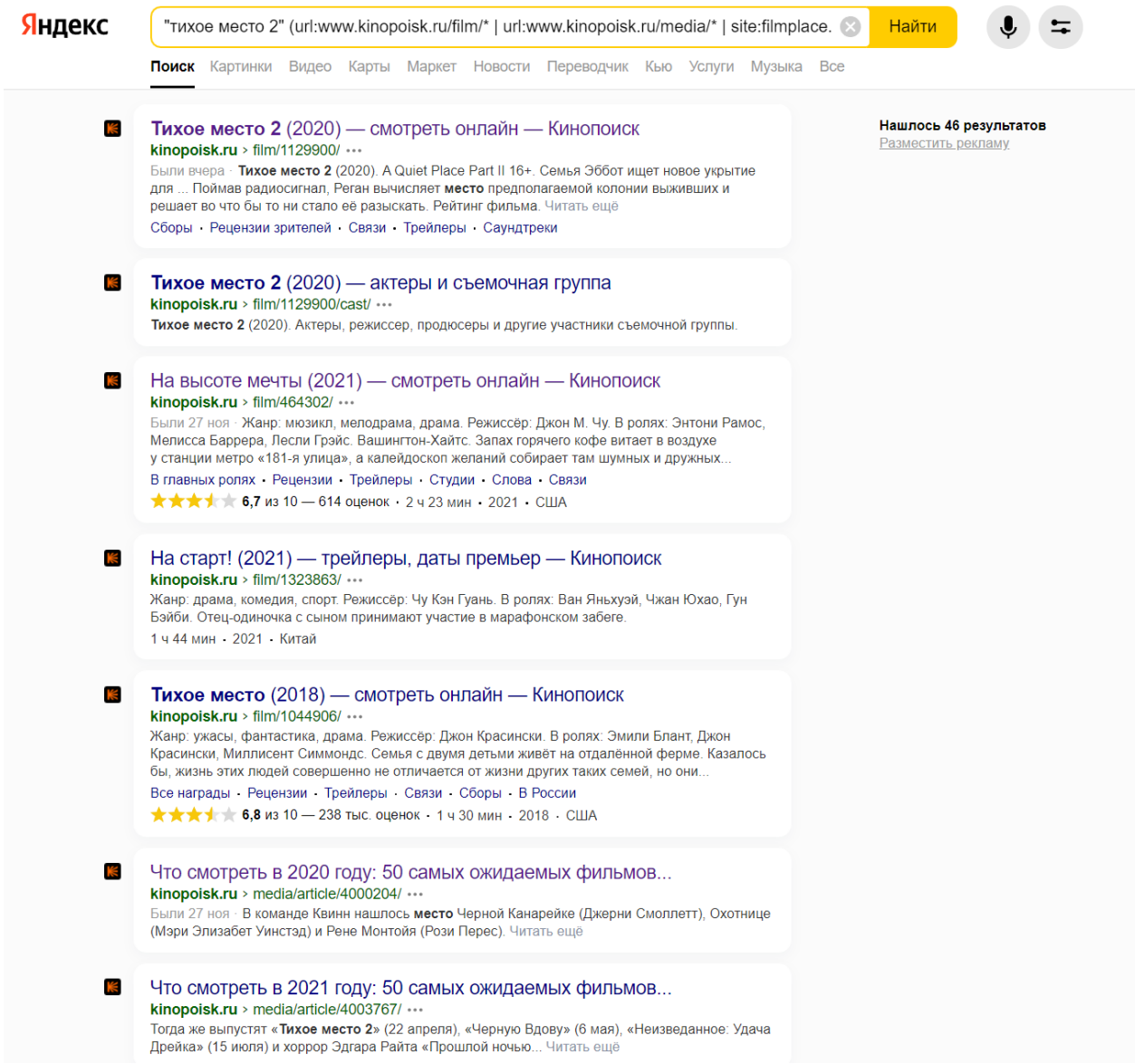
Учитывая тематику корпуса были придуманы следующие запросы (в форме для обработки поисковиком):

Номер	Запрос	Вид
1	"тихое место 2"	цитатный
2	аватар скачать	фразовый (транслируется в нечёткий)
3	"re zero" / 4	цитатный
4	режиссёр назад в будущее	фразовый
5	фильм для интеллектуалов	фразовый
6	фильм сериал с самым большим рейтингом	фразовый
7	фильмы Макото Синкая	фразовый
8	лучшие фильмы квентина тарантино	фразовый
9	как звали главного героя коносубы	фразовый
10	самый лучший фильм	фразовый
11	сериалы с рейтингом 18+	фразовый
12	высшая школа демонов	фразовый
13	джокер	фразовый
...	...	...

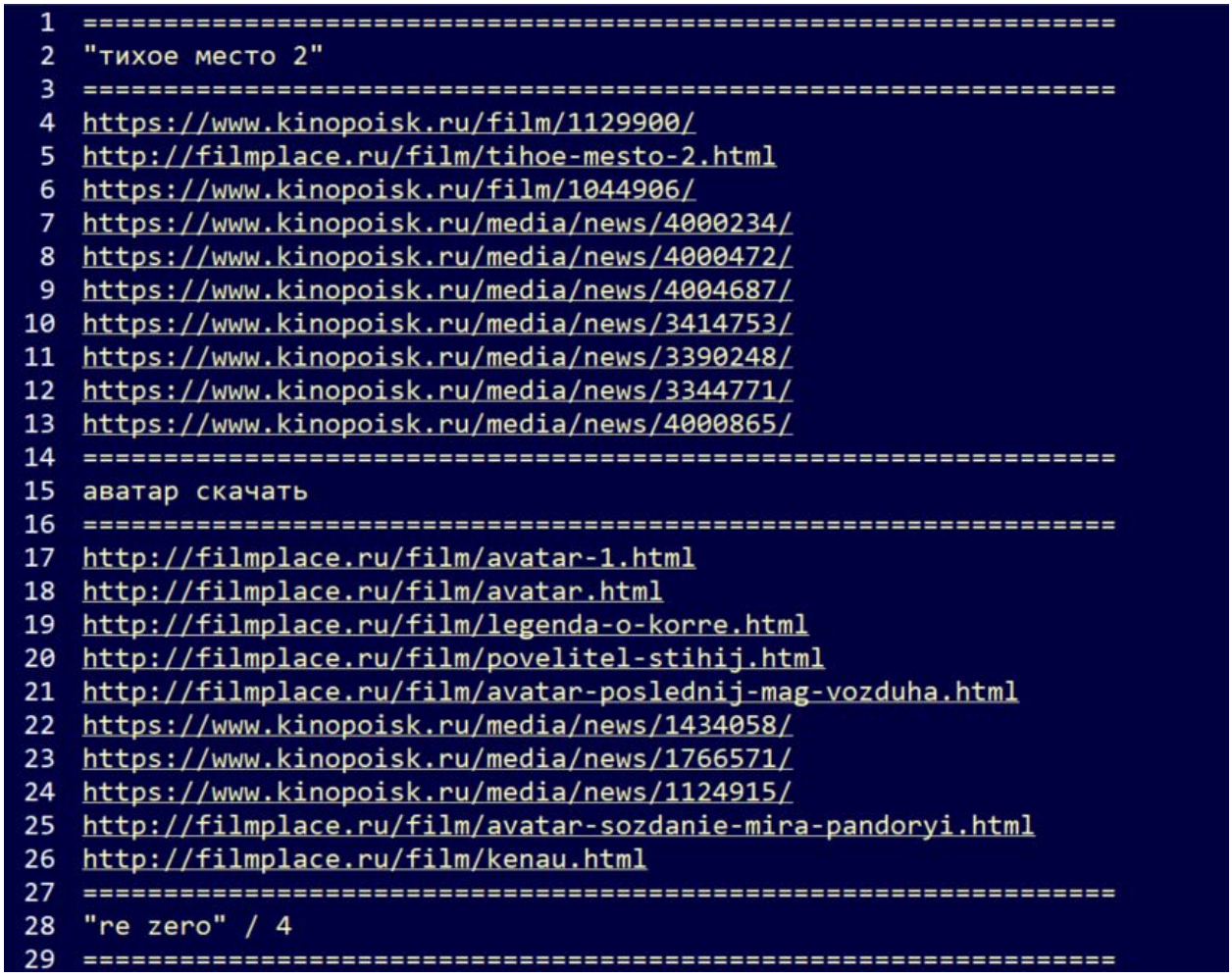
Далее, запросы были адресованы поисковой системе Yandex с ограничением по сайтам, дате и с приведением к соответствующему виду.

Например, первый запрос из списка был представлен в форме:

"тихое место 2" (url:www.kinopoisk.ru/film/* | url:www.kinopoisk.ru/media/* | site:filmplace.ru | site:shikimori.one) date:<20210930

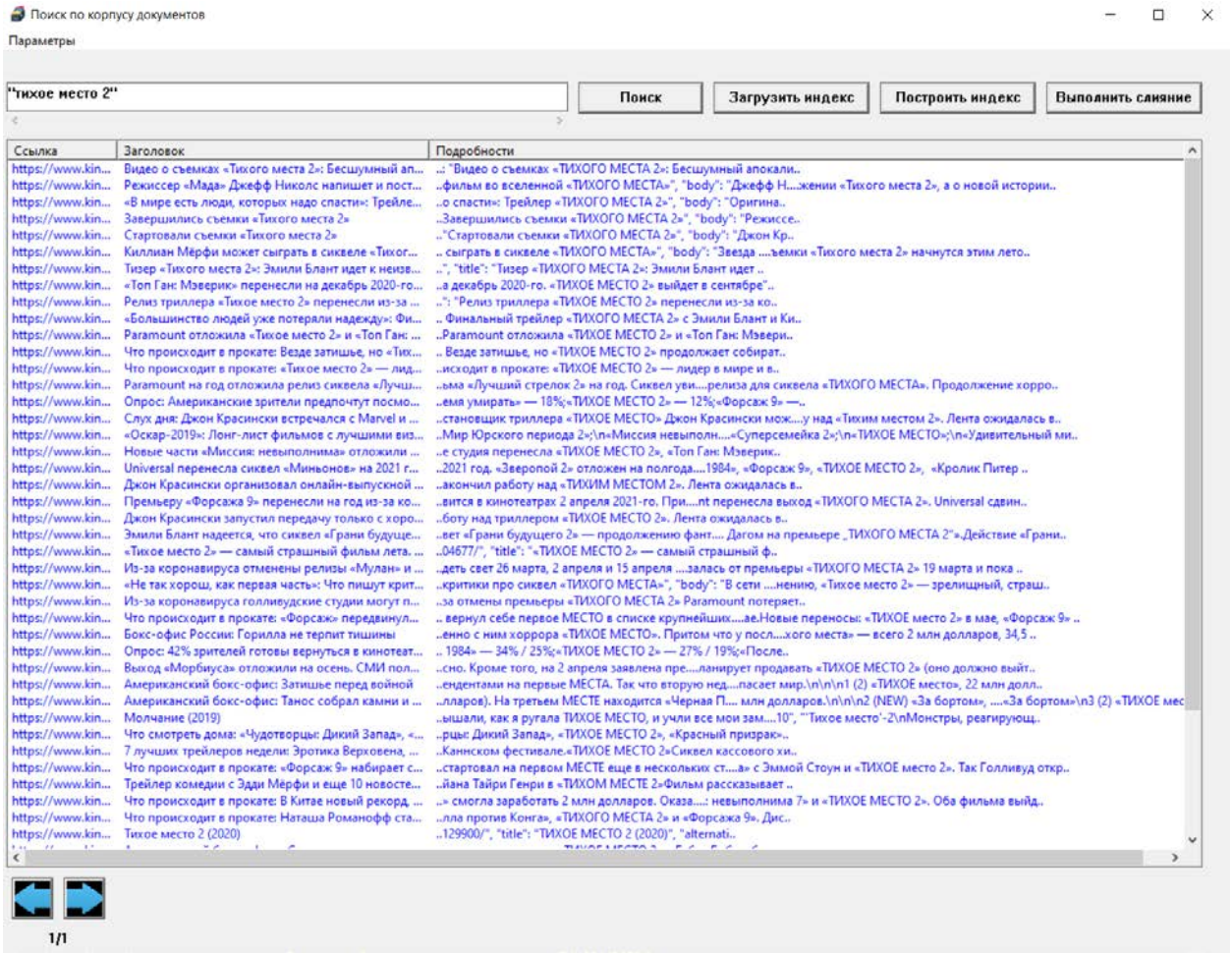


Далее, для каждого запроса из списка вручную был отобран список релевантных ссылок, ввиду того, что поисковая система Yandex’а даёт сбои.



С полным списком запросов и релевантных ссылок читатель может ознакомить по ссылке: https://cloud.mail.ru/public/Hp9R/KMn7Vo1HJ (эта директория также подаётся на вход программе-оценщику).

Далее, запросы из списка были адресованы поисковой системе, созданной по результатам ЛР по курсу.



Извлекая ссылки из ответов и сравнивая их с ссылками, полученными Yandex’ом можно высчитать требуемые метрики.

Метрики

Для просчёта метрик были взяты формулы из статьи [2].

Пусть

$$r: E \to [0,1],$$

функция переводящая документ $d \in E$ в число – релевантность запроса. В моём случае множество значений было решено ограничить значениями $\{0,1\}$ (нерелевантен/релевантен);

$$\pi: E \to [1, K],$$

функция сопоставляющая документу $e \in E$ позицию в отсортированном по убыванию косинус-веса списке найденных документов, ограниченных уровнем K.

Тогда

$$p@K = \frac{1}{K}\sum_{k=1}^{K} r\left(\pi^{-1}(k)\right);$$

$$DCG@K = \sum_{k=1}^{K} \frac{r\left(\pi^{-1}(k)\right)}{\log_2 k + 1};$$

$$IDCG@K = \sum_{k=1}^{K} \frac{1}{\log_2(k+1)};$$

$$NDCG@K = \frac{DCG@K}{IDCG@K};$$

$$ERR@K = \sum_{k=1}^{K} \frac{1}{k} r\left(\pi^{-1}(k)\right) \prod_{i=1}^{k-1} \left(1 - r\left(\pi^{-1}(k)\right)\right).$$

Поскольку число запросов больше чем 1, в конце проводится усреднением по всем метрикам для множества запросов Q.

1. Исходный код

Структура проекта

- include
  - algebra.h (примитивные операции с математическими векторами)
  - create_index.hpp (создание, чтение индекса)
  - defs.hpp (подключение внешних библиотек, макросы)
  - docs_parse.hpp (извлечение полей из корпуса и индекса)
  - search.hpp (поиск)
  - token_parse.hpp (функции для преобразования токенов в термы)
- 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
  - main.cpp (точка входа в консольный интерфейс)

Проект был написан с помощью Microsoft Visual Studio 2019 эксклюзивно для ОС семейства Windows.

Запуск

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:

```
$ ./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
```

2. Выводы

Метрика \ Уровень	1	3	5	30
P	0.62	0.43	0.4	0.241
DCG	0.62	1.011	1.29	2.65
nDCG	0.62	0.47	0.44	0.29
ERR	0.62	0.71	0.71	0.73

В целом система работает удовлетворительно и со своими задачами справляется. Из улучшений остаётся: автоисправление опечаток (см. курсовую работу), автораспознавание вида запроса (булев, цитатный, нечёткий), ускорение ранжирования, ускорение индексации путём отказа от Natash'и (4 часа на полную индексацию корпуса в многопоточном режиме), сжатие индекса, прыжки по индексу, более умные сниппеты, зонный поиск и прочее.

В ходе выполнения лабораторной работы я научился считать метрики ранжирования для коллекции документов.

Литература

[1] Кристофер Д.Маннинг, Прабхакар Рагхаван, Хайнрих Шютце. Введение в информационный поиск. 2020, изд. Вильямс.

[2] https://habr.com/ru/company/econtenta/blog/303458/