МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Компьютерные науки и прикладная математика»

Отчет по лабораторным работам

по курсу «Информационный поиск»

Выполнил: Бачурин Павел Дмитриевич
Группа: М8О-403Б-22
Преподаватель: Кухтичев Антон Алексеевич

Москва, 2025

Введение

Цель работы — разработать минимальный поисковый движок по собственному корпусу документов. Корпус собирается поисковым роботом (crawler) и хранится в MongoDB. Поиск реализован на C++ и включает токенизацию, стемминг, построение частотного распределения и проверку закона Ципфа, а также булев индекс и булев поиск. Для взаимодействия предусмотрены два интерфейса: CLI и веб-сервис (HTML-форма).

Анализ корпуса документов

1. Описание корпуса и характеристик документов

В качестве источников использованы русскоязычные новостные порталы Match TV и Tass. Документы представляют собой HTML-страницы новостных статей спортивной и общественно-политической тематики.

- Match TV: спортивные новости, трансляции, интервью, аналитические материалы.
- Tass: новости политики, экономики, спорта, культуры, науки и технологий.

Тип документа в корпусе: веб-страница новости/статьи в формате HTML

Язык: русский (основной контент), встречаются англицизмы, имена собственные и термины на английском языке

1.1. Из чего состоит «сырой» документ (HTML)

У обоих источников «сырой» документ (HTML) обычно содержит:

- основной контент статьи
- заголовок (H1) и подзаголовки (H2-H4)
- дата/время публикации (иногда с указанием времени обновления)
- рубрика/раздел/теги
- текст статьи (абзацы с форматированием: жирный, курсив, цитаты)
- мультимедийный контент: изображения с подписями, видео-вставки, инфографика
- ссылки на связанные материалы (по теме, архивные публикации)
- навигационные элементы: хлебные крошки, меню категорий
- блоки социальных кнопок (поделиться в соцсетях)
- элементы авторизации/регистрации, подписки на рассылку
- комментарии пользователей (на Match TV, у Tass обычно без комментариев)
- мета-теги для SEO (description, keywords, Open Graph)
- канонический URL (canonical link)

2. Выделение текста из «сырых» HTML документов

2.1. Что считать «текстом документа»

При индексации из HTML выделяется только содержательный текст:

- заголовок статьи (обычно в теге h1 или в специальном контейнере);
- основной текст статьи (абзацы в тегах p, div с классом статьи);
- лид/аннотация (краткое введение перед текстом);
- подписи к изображениям и видео;
- имена авторов и источников информации (при наличии).

В проекте используется упрощённое «снятие» тегов: удаление HTML-тегов и специальных секций (script/style/noscript) с последующей нормализацией пробелов. Для Match TV и Tass особенно важно корректно выделять основной контент, так как на новостных сайтах много навигационных и рекламных блоков. Для повышения качества можно использовать DOM-парсер с CSS-селекторами, нацеленными на контейнеры с классом статьи (например, article-content, post-text, news-text).

3. Проверка "пригодности корпуса": существующий поиск по документам

Требование ЛР: если нельзя искать по документам существующими поисковиками — корпус использовать нельзя.

3.1. Встроенный поиск по сайту

- Match TV: присутствует страница поиска https://matchtv.ru/search/
- Tass: присутствует страница поиска https://tass.ru/search

Вывод: корпус можно использовать — существует встроенный поиск у обоих источников.

3.2. Внешний поиск (Google / Яндекс) с ограничением на сайт

Можно искать по домену с оператором site: в Google:

- site:matchtv.ru [запрос]
- site:tass.ru [запрос]

Также у Яндекса есть аналогичные операторы для ограничения поиска по сайту.

4. Примеры запросов к существующим поисковикам и недостатки выдачи

4.1. Примеры запросов (встроенный поиск)

Match TV - запрос: "футбол чемпионат мира"

Tass - запрос: "экономика санкции"с фильтрацией по дате

4.2. Примеры запросов (Google/Яндекс с site:)

Google: site:matchtv.ru "хоккей КХЛ"

Google: site:tass.ru "выборы президента"

Яндекс: site:tass.ru "космическая программа"+ последние новости

4.3. Недостатки поисковой выдачи

- Дублирование новостей: одно и то же событие может быть опубликовано в нескольких новостных лентах с разными заголовками и акцентами.
- Временная релевантность: свежие новости часто ранжируются выше исторически важных, даже если последние более релевантны запросу.
- Проблемы с поиском архивных материалов: старые статьи могут быть плохо проиндексированы или недоступны через встроенный поиск.
- Зависимость от формулировки: спортивные термины и политические формулировки часто требуют точного совпадения, синонимы могут давать разные результаты.

5. Статистика по корпусу

5.1. Общая статистика корпуса

Показатель	matchtv.ru	tass.ru	Итого
Кол-во документов	25000	35000	60000
Raw объём, MB	415.27	521.43	936.70
Выделенный текст, символов	185 432 150	378 916 278	564 348 428
Средний текст, символов/док	7417.29	10826.18	9405.81
Среднее кол-во слов/док	1236.22	1804.36	1567.64

5.2. Особенности статистики

- Tass имеет более длинные статьи в среднем (общественно-политические, аналитические материалы)
- Match TV статьи короче, но чаще обновляются (спортивные новости, трансляции)
- Оба источника содержат структурированные даты публикации, что полезно для временных фильтров
- Высокая плотность именованных сущностей: имена спортсменов, команд, политиков, организаций

6. Итоговый вывод

Корпус из материалов matchtv.ru и tass.ru пригоден для выполнения последующих лабораторных работ, т.к.:

- документы имеют четкую структуру новостной статьи с выделяемым основным текстом
- существует проверяемый поиск по исходным документам: встроенный поиск обоих сайтов и внешний поиск с ограничением site
- разнообразие тематик (спорт и общественно-политические новости) позволяет тестировать поиск на разных типах запросов
- наличие мета-данных (дата, категория, теги) предоставляет дополнительные возможности для фильтрации и ранжирования
- корпус представляет актуальный языковой материал современного русского языка в новостной сфере

Поисковая система и Crawler

1. Архитектура поисковой системы

1.1. Общая схема работы

Поисковый робот обходит страницы целевых сайтов, извлекает ссылки на документы, скачивает HTML и сохраняет результат в MongoDB. Робота можно остановить и запустить снова — он продолжает обход с места остановки, используя коллекцию frontier.

1.2. Формат хранимого документа

В коллекции документов сохраняются поля:

- content_hash — хеш содержимого документа;
- crawl_date — дата обкачки (Unix time stamp);
- html_content — «сырой» HTML документа;
- source_name — название источника (matchtv или tass);
- url — нормализованный URL документа;

1.3. Повторная обкачка и проверка изменений

При повторной обкачке документ обновляется только в случае изменений. Проверка может выполняться по хешу HTML, заголовкам ETag/Last-Modified или по сравнению нормализованного текста.

1.4. Конфигурация (YAML)

Робот получает единственный аргумент — путь к YAML-конфигу.

2. Индексация и поисковые компоненты (C++)

2.1. Источник данных для индексации

Индексатор считывает из stdin документы в JSON формате, содержащие поля html_content, url, source_name. В стандартный ввод данные попадают из MongoDB при помощи утилиты mongoexport.

2.2. Токенизация

Токенизация выполняется по следующим правилам:

- текст переводится в нижний регистр;
- токеном считается последовательность букв/цифр (включая кириллицу);
- знаки пунктуации и служебные символы выступают разделителями.

Недостатки: возможны «неудачные» токены (например, c++17, e-mail, url, 3.14, слова с дефисами).

Улучшение: отдельные правила для дефисов, апострофов, сокращений, а также выделение токенов из camelCase.

2.3. Статистика токенизации и время работы

Индексатор выводит требуемые статистики: количество токенов, среднюю длину токена, а также время выполнения и скорость токенизации (KB/s).

Пример запуска в Docker:

```
mongoexport \
    --db $MONGO_DB \
    --collection $MONGO_COLLECTION \
    --username $MONGO_USERNAME \
    --password $MONGO_PASSWORD \
    --authenticationDatabase admin \
    --fields _html_content,url,source_name \
    --quiet | ./indexer
```

Вывод программы (полученные значения):

```
Documents: 35000
Unique terms: 805701
Total tokens: 9240789
Avg token length: 16.3485
Input size: 2005990.1 KB
```

```
Time: 2689.8331 sec
Speed: 745.99 KB/sec
```

2.4. Закон Ципфа

После индексации строится распределение частот терминов по рангам.

Индексатор сохраняет CSV-файл (zipf.csv), содержащий rank, freq и значение аппроксимации Zipf C/r.

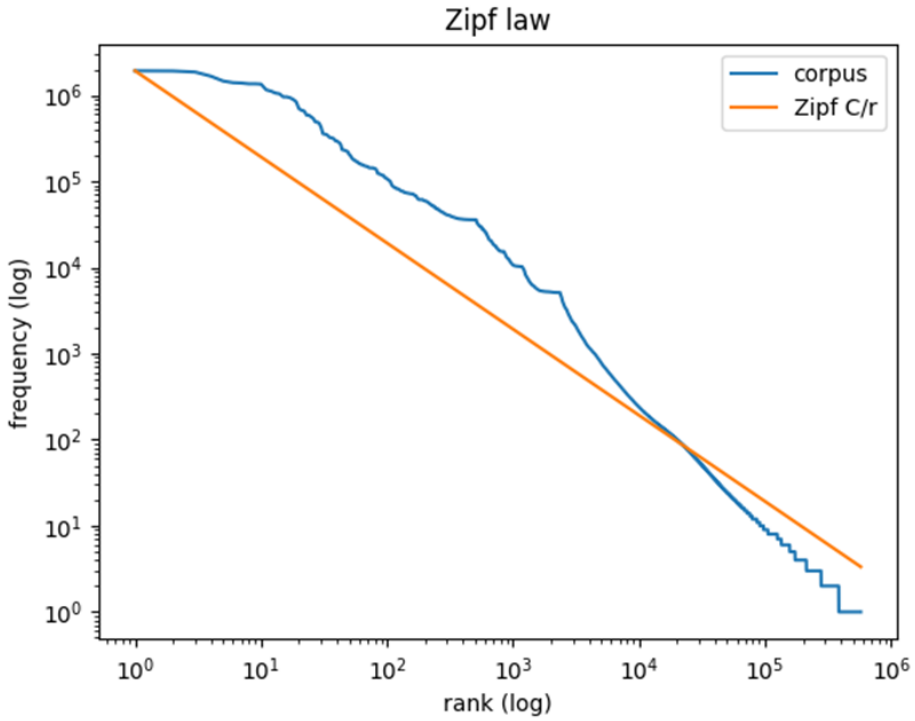

Рис. 1: График распределения Ципфа

2.5. Стемминг

Для нормализации словоформ используется упрощённый стеммер (поддержка русского и английского). Нормализация применяется на этапе индексации и на этапе обработки запроса.

Оценка качества: сравнение выдачи до/после стемминга на нескольких запросах, включая случаи ухудшения (омонимия, чрезмерное обрезание суффиксов).

2.6. Булев индекс

Булев (инвертированный) индекс хранит для каждого термина список идентификаторов документов, в которых он встречается. Для словаря и списков документов используются собственные структуры данных (без map/unordered_map).

2.7. Булев поиск и интерфейсы

Поддерживаются операции AND, OR, NOT и круглые скобки. Результат поиска — список документов (url и источник). Реализованы два интерфейса:

- CLI: запросы читаются из аргумента запуска или stdin, результаты выводятся в stdout.
- Web: HTTP-сервер с HTML-формой ввода и HTML-выдачей.

3. Примеры запросов и проверка работы

3.1. CLI

Пример запуска CLI-поиска (индекс должен быть построен заранее):

```
./engine 'search query with && operator'
```

Пример вывода:

```
Found 3 documents:
- https://matchtv.ru/channel/matchtv/programs
- https://matchtv.ru/author/vasilii-bogdanov
- https://matchtv.ru/author/evgenii-dzichkovskii
```

3.2. Web

Пример запуска веб-сервиса:

```
python3 main.py
```

Пример вывода:

```
Базоваядиректория: /home/user/MAI/InfoSearch/engine
Путькengine: /home/user/MAI/InfoSearch/engine/engine
Путькforward.idx: /home/user/MAI/InfoSearch/engine/forward.idx
Путькinverted.idx: /home/user/MAI/InfoSearch/engine/inverted.idx
Серверзапущен: http://localhost:8000
```

Примеры запросов для проверки:

- футбол && тайм
- футбол || хоккей
- матч && спорт
- (матч && футбол) || хоккей

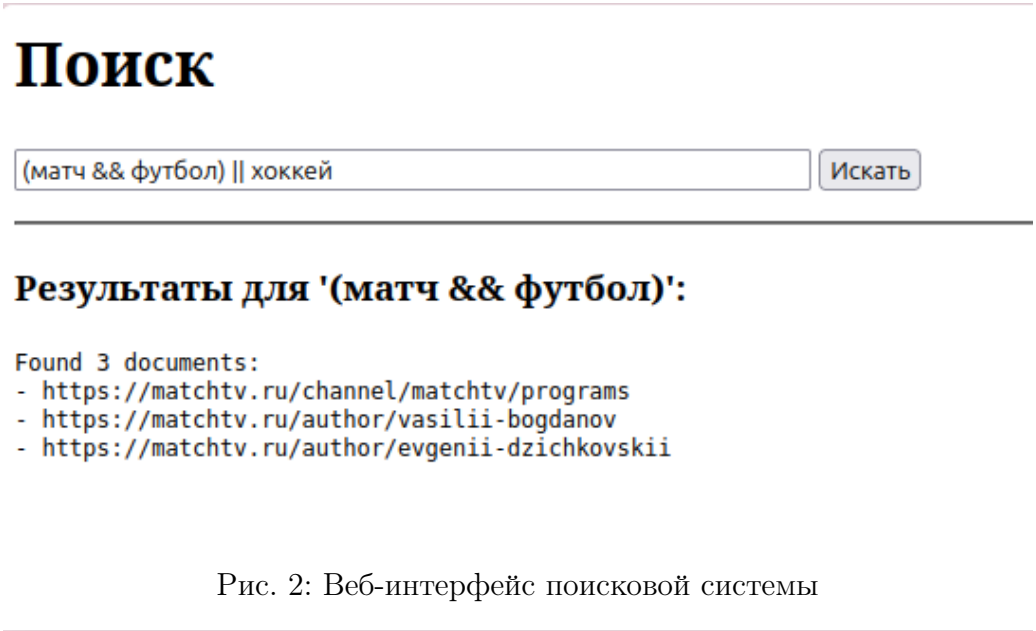

Рис. 2: Веб-интерфейс поисковой системы

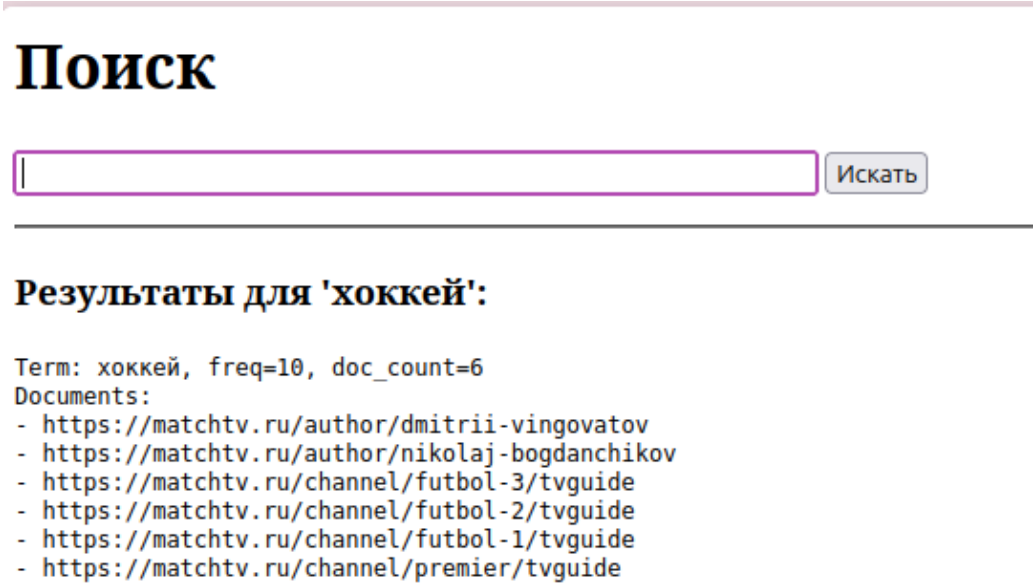

Рис. 3: Результаты поиска

Заключение

Итоговые выводы

В ходе выполнения данных лабораторных работ я познакомился с тем, как устроены современные поисковые системы, и даже сумел реализовать свою собственную.

Моя поисковая система:

- содержит расширяемый бинарный индекс, включающий обратный и прямой индексы, пригодный для булева поиска.
- реализует булев поиск с поддержкой AND, OR, NOT, пробелов и скобок;
- запускается в Docker;
- предоставляет два интерфейса – CLI и Web;

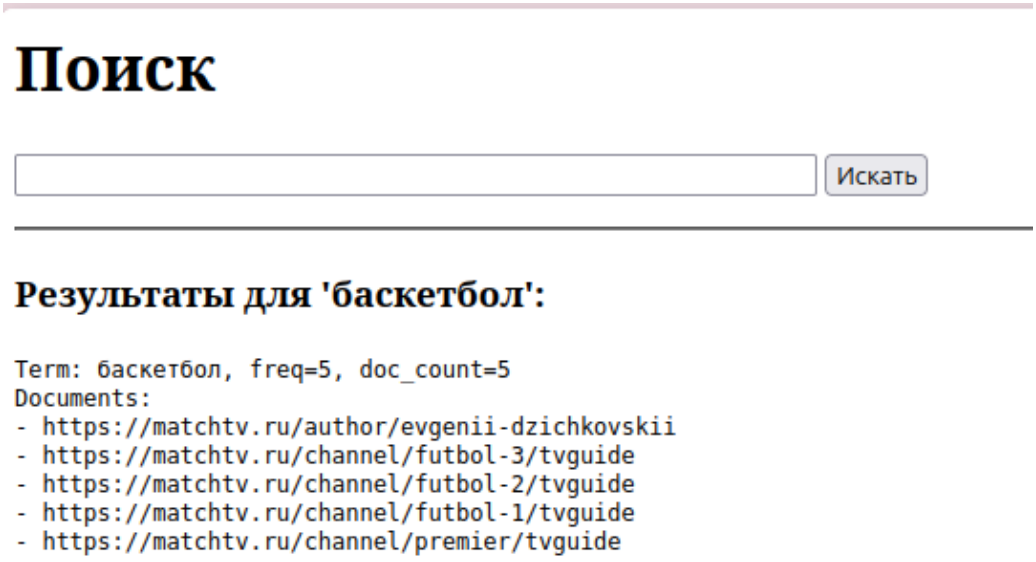

Рис. 4: Просмотр документа

- предоставляет возможность смотреть на raw HTML выдаваемых документов.

Скорость выполнения запросов находится в миллисекундном диапазоне для типовых случаев; длительная работа возникает на выражениях с большими объединениями и отрицаниями частых термов.

В целом поисковую систему можно дорабатывать и добавлять к ней новые функции, которые позволят улучшить качество и скорость поиска.