МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский Авиационный Институт»
(Национальный Исследовательский Университет)

Институт: №8 «Информационные технологии и прикладная математика»
Кафедра: 806 «Вычислительная математика и программирование»

Лабораторные работы по курсу «Информационный поиск»

Группа: М8О-407Б-22

Студент: Диёров Д.У.

Преподаватель: Кухтичев А.

Оценка:

Дата: 11.12.2025

Москва, 2025

Цель работы:

- Найти и проанализировать корпус документов, который будет использован при выполнении остальных лабораторных работ
- написать парсер на любом языке программирования
- реализовать процесс разбиения текстов документов на токены, который потом будет использоваться при индексации. Для этого потребуется выработать правила, по которым текст делится на токены. Реализовать стемминг
- Реализовать булев индекс и булев поиск
- Написать юнит-тесты для проверки корректности работы

Задание:

Парсер

В качестве корпуса документов был взят форум https://anime-characters-fight.fandom.com , который хранит 1 200 000 страниц из 23 000 статей с персонажами и их способностями. Для скачивания и парсинга страниц был использован скрипт на языке программирования Go, который брал несколько ссылок и рекурсивно обходил каждую из них и проверял что статьи находятся в том же домене что и основные ссылки, чтобы скрипт не парсил страницы, которые не принадлежат домену сайта. Для скачивания страниц была использована стандартная библиотека http, для парсинга использовалась open-source библиотека github.com/PuerkitoBio/goquery. Проверка работы парсера проверена юнит-тестом. При скачивании структура Crawler проверяет, что код не нарушает принципы файла robots.txt, если какая-то ссылка недоступна, то парсер пропустит эту страницу и не будет ее запрашивать.
Метод Crawl скачивает страницу отдает парсеру и парсер возвращает удобочитаемое json представление страницы. Далее Crawler сохраняет полученные данные в Postgres. Main функция скрипта через паттерн WorkerPool вызывает метод Crawl, далее горутина засыпает на 100ms чтобы не давать большую нагрузку на форум.
Производительность токенизации:

Для оценки скорости токенизации был сгенерирован текст объёмом **1024 КБ**.

- Количество полученных токенов: **149 798**
- Средняя длина токена: **7,0 символов**
- Время токенизации: **0,04 секунды**

- Скорость обработки: **≈ 25 МБ/с**

Для анализа данных в БД был реализован python скрипт.

При использовании стемминга были получены следующие усреднённые показатели качества:

- **Precision@5:** 0.32
- **Recall@5:** 0.80
- **F1@5:** 0.45

Качество поиска без стемминга

При отключении стемминга качество поиска заметно улучшилось:

- **Precision@5:** 0.60
- **Recall@5:** 0.80
- **F1@5:** 0.67

Полнота осталась на том же уровне, однако точность значительно возросла. Это означает, что без стемминга поисковый индекс формирует более точные результаты, особенно для терминологических и имен собственных запросов.

Для анализа распределения частот токенов были аппроксимированы два закона.

Закон Ципфа показал параметры:

- $C=2.53 \cdot 10^4C = 2.53 \cdot 10^4C=2.53 \cdot 104$
- $s=0.79s = 0.79s=0.79$

Значение параметра $s<1s < 1s<1$ говорит о медленном убывании частот: редкие слова составляют значительную часть словаря.

Закон Мандельброта дал параметры:

- $C=2.75 \cdot 10^4C = 2.75 \cdot 10^4C=2.75 \cdot 104$
- $b=0.137b = 0.137b=0.137$

- s=0.81s = 0.81s=0.81

Добавление параметра смещения b улучшает аппроксимацию распределения, особенно для наиболее частотных слов.

Токенизация

Функция tokenize выполняет токенизацию входного текста, то есть разбивает строку на отдельные слова (токены), корректно обрабатывая символы в кодировке UTF-8.

На первом этапе входная строка в формате std::string (UTF-8) преобразуется в широкую строку std::wstring. Это необходимо для корректной работы с Unicode-символами и использования стандартных функций классификации символов (буквы, цифры и т.д.).

Далее устанавливается UTF-8 локаль, что позволяет правильно определять типы символов (буквенно-цифровые или разделители) для многоязычного текста.

Затем функция последовательно проходит по каждому символу широкого текста:

- если символ является буквенно-цифровым (iswalnum), он приводится к нижнему регистру и добавляется к текущему формируемому токену;

- если встречается любой другой символ (пробел, знак пунктуации, спецсимвол), текущий токен считается завершённым и добавляется в результирующий список, после чего начинается формирование нового токена.

После обработки всего текста функция дополнительно проверяет, остался ли незавершённый токен, и при необходимости добавляет его в результат.

Каждый сформированный токен перед добавлением в итоговый массив преобразуется обратно из std::wstring в UTF-8 строку (std::string). Результат работы функции сохраняется в БД. Чтение и запись в БД происходит батчами по 5 документов.

Булев поиск и булев индекс

Сначала строковый запрос разбирается функцией parse_bool_query. Она ищет в строке логические операторы AND или OR, разделяет запрос на левый и правый токены и сохраняет тип операции. Если оператор не найден, запрос помечается как некорректный.

Для работы с результатами поиска используется структура IntArray, которая хранит динамический массив идентификаторов документов и его размер.

Функция get_token_id по тексту токена выполняет SQL-запрос к базе данных и возвращает его числовой идентификатор. Если токен отсутствует в таблице, возвращается -1.

Функция get_documents_by_token получает из базы данных список идентификаторов документов, связанных с заданным токеном, и формирует массив IntArray.
Логические операции над результатами поиска реализованы отдельными функциями:

- intersect вычисляет пересечение двух массивов документов и возвращает только совпадающие идентификаторы;

- unite формирует объединение двух массивов, добавляя элементы без дублирования.

Функция boolean_search связывает все этапы поиска. Она получает идентификаторы токенов, извлекает соответствующие списки документов и в зависимости от операции (AND или OR) вызывает пересечение или объединение массивов.
Функция boolean_search_http предназначена для использования в HTTP-обработчике. Она вызывает булев поиск, а затем по каждому найденному идентификатору документа запрашивает из базы данных URL страницы и формирует JSON-ответ со списком найденных ссылок.

График распределения терминов:

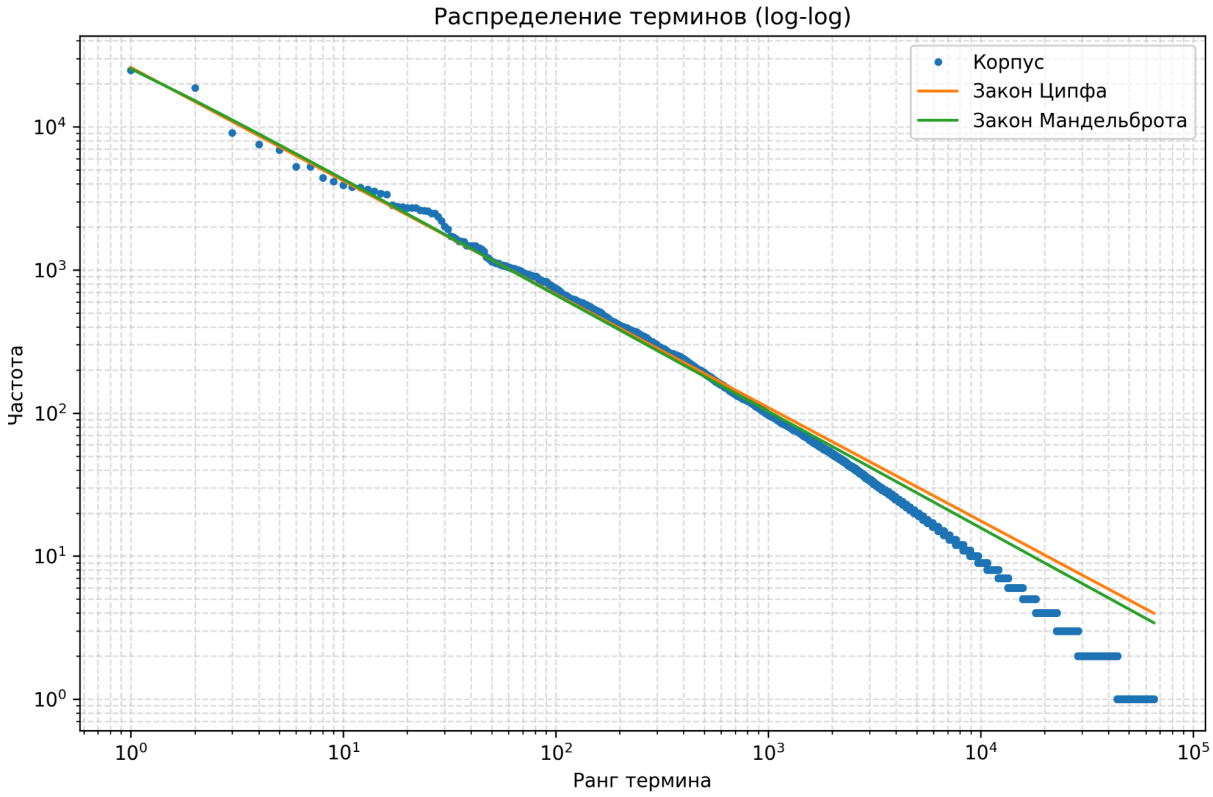
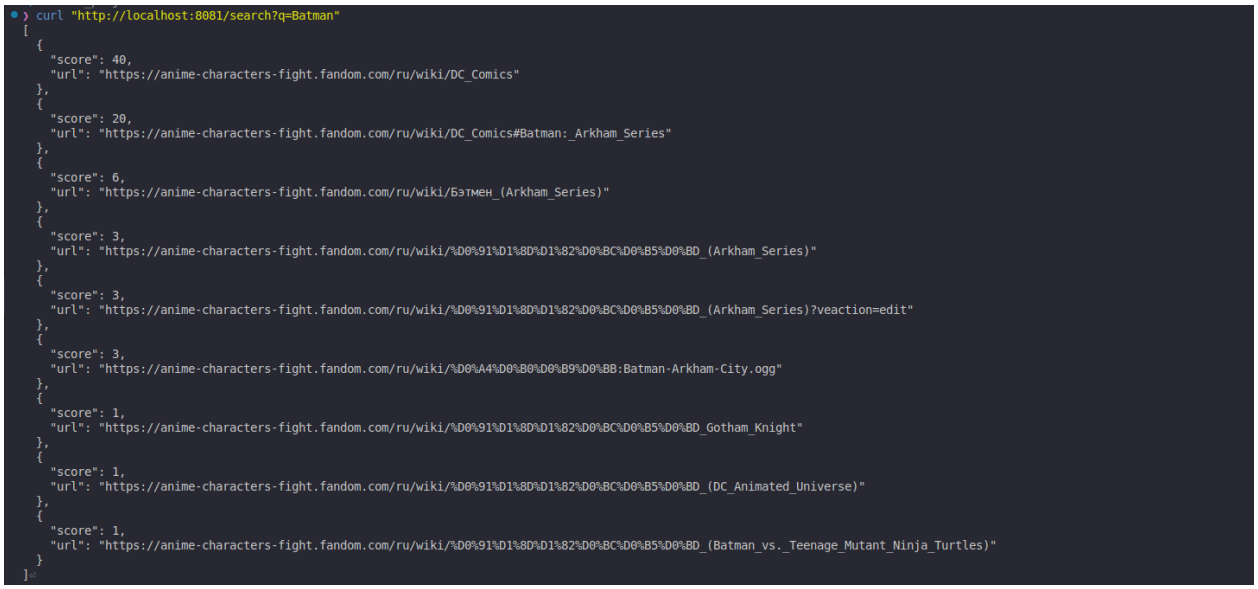


Рисунок 1

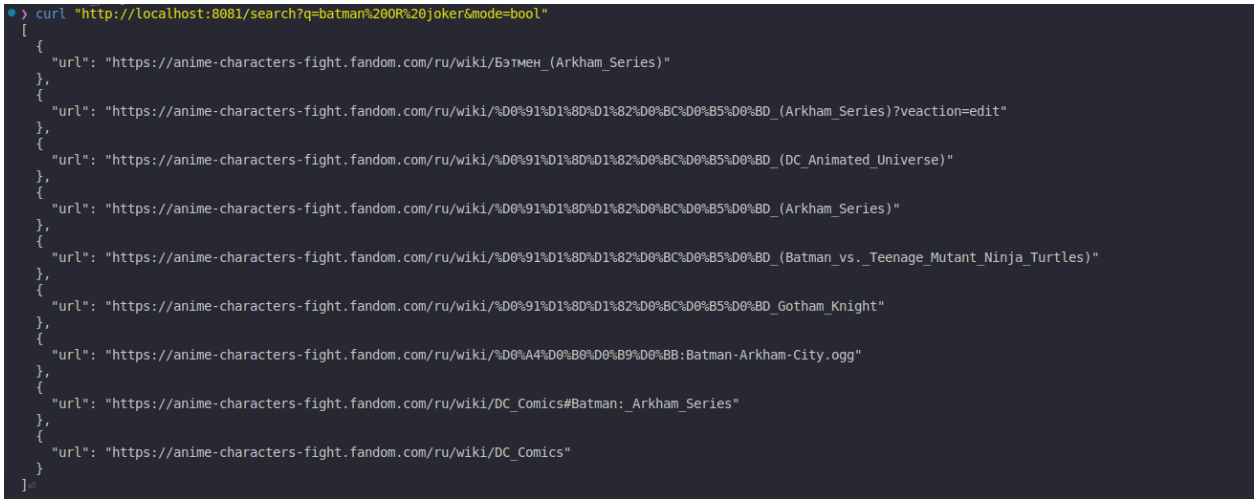
Для ввода и вывода в поисковик был реализован http сервер на c++ который принимает поисковые запросы и отдает ответ.

Примеры запросов и ответов:

Запрос 1



Запрос 2 с булевым поиском



Вывод:

В ходе лабораторной работы был реализован поисковый индекс с поддержкой токенизации и булевого поиска. Проведённый анализ корпуса подтвердил типичные статистические свойства текстовых данных, а реализованный токенизатор показал высокую производительность.