МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Компьютерные науки и прикладная математика»

Лабораторная работа
по курсу «Информационный поиск»

Поисковой движок.

Выполнил: Кудинов Д.В.
Группа: М8О-412Б-22
Оценка:
Дата сдачи:
Преподаватели: Кухтичев А. А.

Москва, 2025

Подготовка корпуса данных

Я собрал корпус из русскоязычной Википедии по категории «Кинематограф» и вложенным подкатегориям. Скачивание выполнялось скриптом, который обходит категорию и сохраняет каждую страницу в файл wiki_cinema/docs.jsonl (по одному JSON-объекту на строку). Для надёжности работы скрипт ведёт файл processed.txt, что позволяет продолжить прерванный сбор без повторной загрузки уже сохранённых страниц.

Далее я запустил предобработку: из JSONL получаю набор TSV-файлов по 1000 документов в каждом (part_001.tsv, …). В процессе текст очищается — табы и переводы строк заменяются пробелами, множественные пробелы сводятся к одному, табы в заголовках заменяются пробелами. В каждом TSV-файле три поля: id (SHA-1 от заголовка), title и text (вся статья в одной строке). Параллельно составляется info.txt со статистикой корпуса.

Итоговые числа: всего 49 662 документа; размер «сырых» данных (JSONL) — 370 787 856 байт; объём текста после очистки — 358 908 634 байт; средний размер текста в документе ≈ 7 227 байт; корпус разбит на 50 файлов (по 1000 документов, последний файл заполнен частично). Эти данные и примеры строк присутствуют в info.txt и в файлах part_*.tsv.

Ограничения, которые важно отметить в отчёте: при предобработке я убирал переносы строк и табуляции, поэтому в TSV—тексте потеряна исходная разбивка на абзацы — это не мешает базовой индексации и булевому поиску, но усложняет задачи, где нужны точные позиции токенов (координатный индекс, цитатный поиск). Идентификатор документа получен как SHA-1 от заголовка — это удобно, но при смене заголовка в оригинальной Википедии id изменится; при потребности можно сохранить и pageid/URL.

```
СТАТИСТИКА КОРПУСА
========================================


Всего документов (входных строк): 49662
Всего уникальных документов: 47500
Размер сырых данных: 370,787,856 байт
Размер текста (после очистки): 357,430,554 байт
Средний размер документа: 7525 байт
Файлов в корпусе: 49
Документов в файле (максимум): 1000
Источник: Википедия (категория Кинематограф)
Формат: TSV (id\ttitle\ttext)
Кодировка: UTF-8
```

Предобработка и токенизация

Корпус был очищен от полных дубликатов и пустых записей, выполнена нормализация Unicode, устранены переносы слов и удалены простые HTML-вставки и лишние пробелы. Токенизация проводилась с учётом Unicode: извлекались слова, числа и буквенно-цифровые сочетания; все токены приводились к нижнему регистру. Элементы, не содержащие букв или цифр (отдельная пунктуация и т.п.), отфильтровывались как шум. В результате обработано 47 500 документов, получено 26

429 331 токенов; число уникальных терминов — 857 797, средняя длина документа ≈ 556 токенов. Самые частые термины: «в» (1 116 827), «и» (772 401), «на» (449 400). Лемматизация в базовом пайплайне не применялась; при необходимости строился отдельный стеммированный индекс. Полные статистики сохранены в tokens_stats.json.

Построение индекса

Индекс формировался по частям: скрипт обходил файлы корпуса (part_*.tsv) в лексикографическом порядке, последовательно обрабатывал каждый документ и присваивал ему внутренний порядковый номер (docnum). Для каждого документа выполнялась токенизация текста, затем для каждого уникального токена в документе записывалось соответствие токен → docnum (по одному вхождению на документ). При желании производился опциональный этап нормализации — стемминг: если включён флаг стемминга, токены преобразовывались вспомогательным стеммером перед учётом в индексах. После прохода по корпусу собранные списки постингов (терм → список docnum) сортировались детерминированно по термам, а затем записывались подряд в бинарный файл. В дополнение к обратному индексу формировался прямой индекс (таблица docnum → docid, title) и таблица длин документов, которые затем сохранялись как отдельные файлы. В процессе сборки вычислялись и сохранялись метаданные (число документов, число уникальных терминов, хеши исходных скриптов, метка о стемминге), что упрощает воспроизводимость и верификацию индекса.

Кратко о файлах

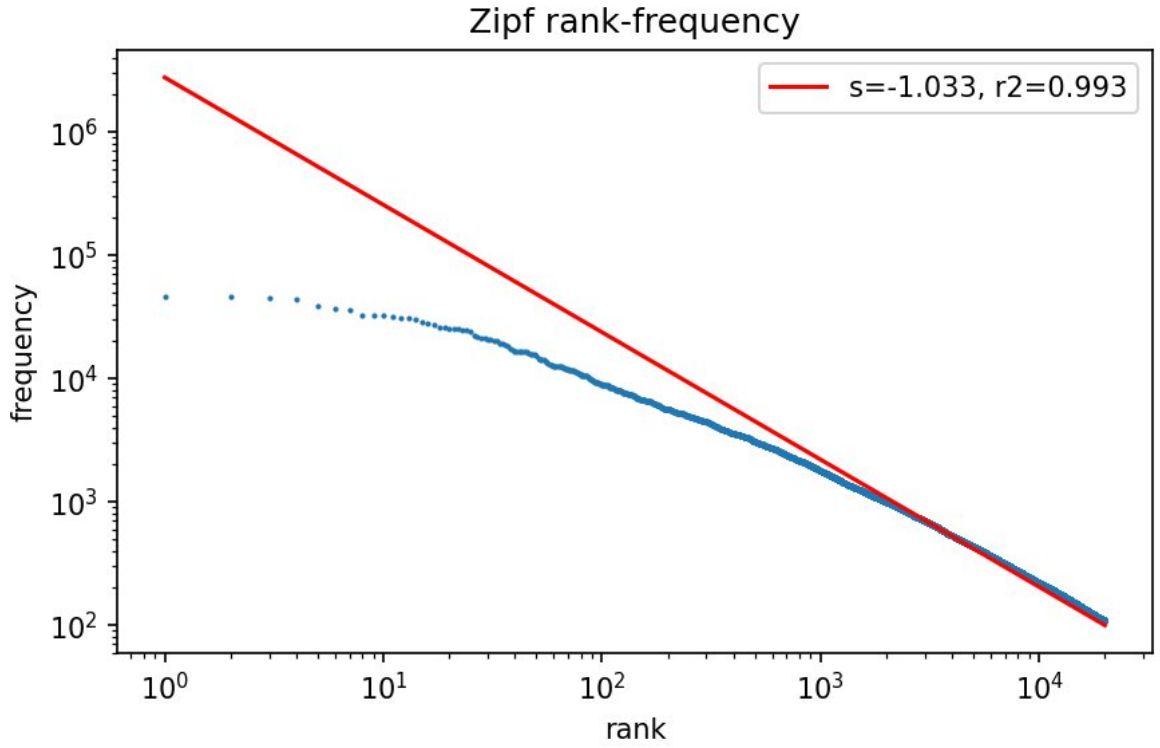
- vocab.tsv — словарь: строка на термин с полями df, offset, length (текстовый, UTF-8).
- postings.bin — бинарный контейнер постингов: varint(df) затем gap-varint для каждого docnum.
- forward.tsv — прямой индекс: docnum \t docid \t title (текстовый).
- doclens.json — длины документов (число токенов) по docnum (JSON).
- meta.json — метаданные сборки: время, количество документов, уникальных терминов, флаг стемминга, контрольные хеши.

Закон Ципфа

Закон Ципфа описывает эмпирическое распределение частот слов в естественных языках: частота слова обратно пропорциональна его рангу в упорядоченном по убыванию списке. На практике это означает, что несколько самых частых слов встречаются крайне часто, а большая часть слов — очень редко.

Чтобы проверить соответствие корпуса закону Ципфа, я построил ранжированное распределение частот терминов и сделал логарифмическую аппроксимацию зависимости log(freq) от log(rank). Для устойчивости аппроксимации использовалась выборка верхних рангов (топ 100 000). Результаты сохранены в zipf.json

и иллюстрированы графиком. По данным анализа, получена оценка наклона регрессии примерно −1.033 с коэффициентом детерминации $R^2 \approx 0.993$. Это указывает на очень хорошее соответствие классической «степенной» модели, принятой в лингвистической статистике.



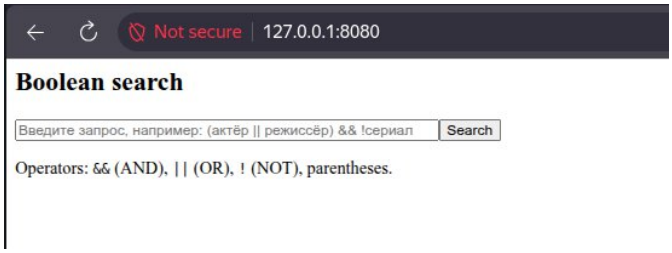
Практическое значение для проекта: сильная неравномерность распределения токенов объясняет, почему в индексах доминируют стоп-слова и почему важно фильтровать нерелевантные служебные токены. Анализ Zipf также помогает выбирать пороги для сжатия и предсказывать эффективность операций, зависящих от длин постинговых списков.

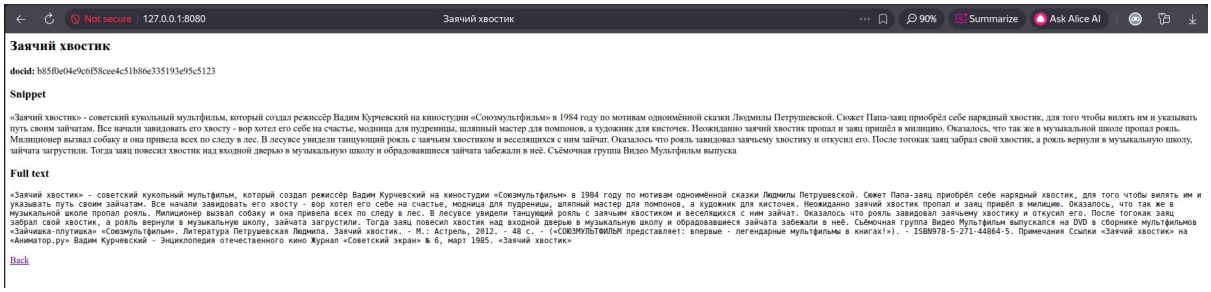
Булевый поиск и индекс

Поисковая часть реализована как булевский движок, поддерживающий три базовых оператора: логическое И (AND, в запросах обозначается &&), логическое ИЛИ (OR, обозначается ||) и отрицание (NOT, обозначается !), а также скобки для явной группировки выражений. Запросы проходят три этапа обработки: лексический разбор на токены и операторы, преобразование в постфиксную форму (для корректной обработки приоритетов и скобок) и оценка постфиксного выражения с использованием множественных операций над множествами docnum.

Для получения множества документов по терму система обращается к лексикону (vocab.tsv), читает смещение и длину блока в postings.bin, загружает соответствующий бинарный блок и декодирует его (varint + gap → последовательность docnum). Операции над результатами реализованы стандартными множественными

операциями: пересечение для AND, объединение для OR и дополнение относительно множества всех документов для NOT. После вычисления итогового множества результаты сортируются по внутреннему номеру документа и выводятся пользователю (в CLI показывается до 50 первых результатов, веб-интерфейс поддерживает постраничный вывод).





Выводы

В рамках проекта была спроектирована и реализована архитектура системы полнотекстового поиска. Работа охватила весь конвейер обработки данных: очистку и нормализацию корпуса, лингвистическую предобработку (токенизация, фильтрация нерелевантных токенов, опциональный стемминг), построение обратного и прямого индексов и инструменты верификации. На их основе реализован механизм булевого поиска и минимальным веб-интерфейсом; запросы обрабатываются через парсер и вычисление логических операций над множествами документов. В проекте также выполнён анализ распределения частот и собраны статистики по корпусу и индексам. Поддержка ранжирования и позиционных постингов оставлена как потенциальное направление для дальнейшего развития.