

**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 $\approx 7\,227$ 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 \approx 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 `token → 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

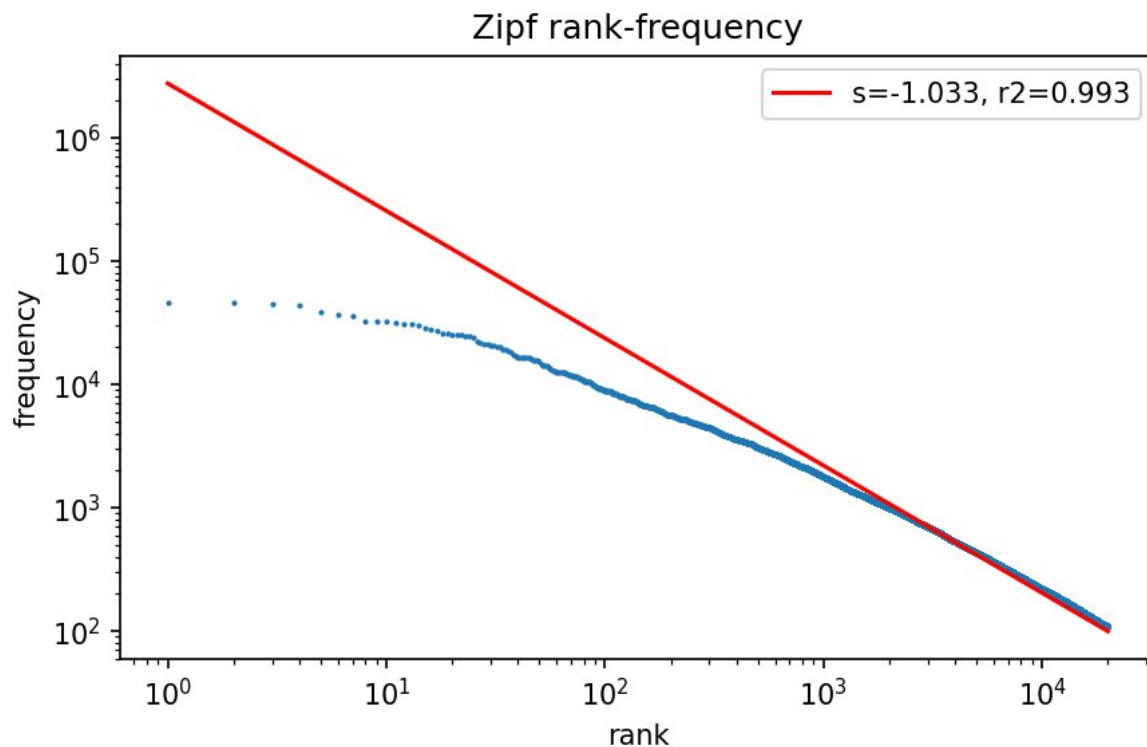
- `vocab.tsv` — словарь: строка на термин с полями `df`, `offset`, `length` (текстовый, UTF-8).
- `postings.bin` — бинарный контейнер постингов: `varint(df)` затем `gap-varint` для каждого `docnum`.
- `forward.tsv` — прямой индекс: `docnum \t docid \t title` (текстовый).
- `doclens.json` — длины документов (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 $\log(\text{freq})$ от $\log(\text{rank})$. 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.



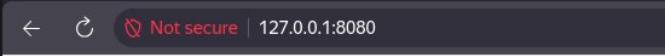
Практическое значение для проекта: сильная неравномерность распределения токенов объясняет, почему в индексах доминируют стоп-слова и почему важно фильтровать нерелевантные служебные токены. Анализ Zipf также помогает выбирать пороги для сжатия и предсказывать эффективность операций, зависящих от длин 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 \rightarrow последовательность docnum). Операции над результатами реализованы стандартными множественными

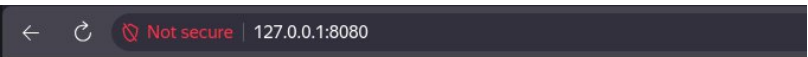
операциями: пересечение для AND, объединение для OR и дополнение относительно множества всех документов для NOT. После вычисления итогового множества результаты сортируются по внутреннему номеру документа и выводятся пользователю (в CLI показывается до 50 первых результатов, веб-интерфейс поддерживает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



Boolean search

Введите запрос, например: (актёр || режиссёр) && !сериял

Operators: && (AND), || (OR), ! (NOT), parentheses.

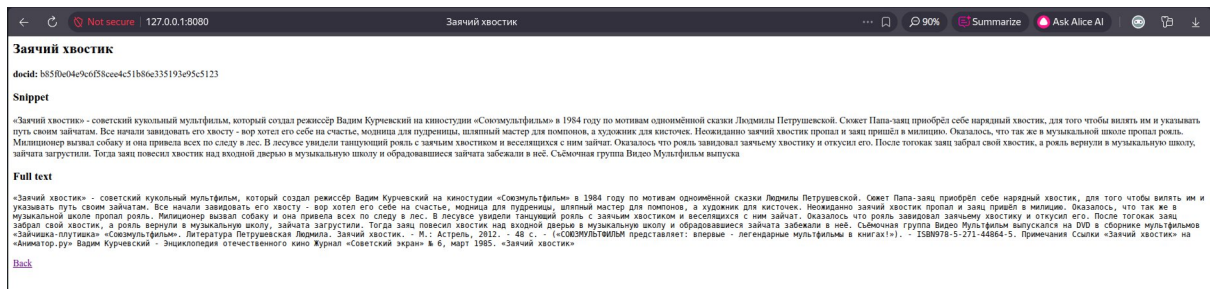


[New search](#) — results for: **советский && режиссёр**

Found 981 documents. Showing 1–50

1. [Кинематограф](#) (da44545061fa219aacc6e2db1bf470966727f8b)
2. [Бриллиантовая рука](#) (ae3d3929354f54308a365a06d40a0453eb3a004a)
3. [Вечер поэзии в Политехническом музее](#) (0ea7b5a3168c9ae9f501c3470e82b62cf3b9b75c)
4. [Владимир Высоцкий в кино](#) (7ed0f0ed6b797fe8b54bee570a6bd7c102184e06)
5. [Гусарская баллада](#) (35c8fe09a2e73c20854a92fed69d4cdc2d8da244)
6. [Двадцать восемь панфиловцев \(фильм\)](#) (2edc04d8b82a2929587f7330667b571b82a0f37c)
7. [Ёжик в тумане](#) (2e7502aadcb5f865e70b41fe91ed2ab6f215866)
8. [Завотной апельсин \(фильм\)](#) (a9dace0560c54d677b995954d4c0a915f559b2f9)
9. [Золушка \(фильм, 1947\)](#) (80939ff497a06beb0c4958c45b2608b75ad8c787)
10. [Иди и смотри](#) (93134ed2dceea74cd937dbbf2562f9b81f4fe50)
11. [Кавказская пленница, или Новые приключения Шурика](#) (8f224b1442f783be63ba32bfe72789a7995cf89f)
12. [Конформист \(фильм\)](#) (521b056d4a9bfa10b9c3d294fb3a72aa5e95c238)
13. [Место встречи изменить нельзя](#) (22e22805d09109c3052960db4ff620986e2d8431)
14. [Мимино](#) (178c7827b914efe24527349e3783cc57a840522)
15. [Первые на Луне](#) (0eb19cfb4c4b83dac489d922183ace61858dab09)
16. [Подкидыш \(фильм\)](#) (45a5ea31552f78ce152dbb498bc51282329ab745)
17. [Покровские ворота \(фильм\)](#) (4adde6dcdb1f98bb878275c1cab2e7ab01950bfe)
18. [Полёты во сне и наяву](#) (733be2d050645e39c18d5874851a6c9e9edcb577)
19. [Фантомас \(фильм, 1964\)](#) (0f7f7e4738c36e15a8750d2192243eae1623e0b5)
20. [Шахматная горячка](#) (a677b7deaa5c080d786b12ad6760d0e6a9854855)
21. [Я - Куба](#) (ca04ba2fcbecb4caa89725545e07bcc90856ead9)
22. [Я шагаю по Москве](#) (d12028f9eb47bd10e4e3f638e5b21fa09cf226eb)
23. [Даль, Олег Иванович](#) (205c1a62edf53be320c33281a64c40d0f6f0d9d)
24. [Ефремов, Олег Николаевич](#) (1875efcd1f3bf9c6cc1fe01896c65d6da371f081)
25. [Попов, Алексей Дмитриевич](#) (4585d3b7d2dbca6e06cb2301c91bbe6cf29c2d5)
26. [Смокуновский, Иннокентий Михайлович](#) (a3bd1dd6cbc813e3f5a61fad554daf0b442d305b)
27. [Ульянов, Михаил Александрович](#) (29ba7eb17f292355866685cd130f3b3149efa53b)
28. [Винни-Пух](#) (ada67ef778f823702356cb03ca795dd2c82f5171)
29. [Война и мир \(фильм, 1967\)](#) (8ad4e71fe17f9c7d51b96f16ec849ba4271ee929)
30. [Добро пожаловать, или Посторонним вход воспрещён](#) (47565c781a007beab542057889eece03ec5bb3c2c)
31. [Идиот \(фильм, 1958\)](#) (5bed72cbde43a4cc2bcc2f1760dace3a2af7e0f3)
32. [Кашей Бессмертный \(фильм\)](#) (81e6d07b550356f920c5b5dd27a6932dae429eb)
33. [Молчание моря \(фильм, 1947\)](#) (bd64d531581b2f10cb30d10ca92927be2cf56efd)
34. [Опасный поворот \(фильм\)](#) (d38684d1462b33fcb4fbcca91ca10565894266e)
35. [Пираты XX века](#) (39c4f4b6b0298226a54bb09b2bd0c4892358490e)
36. [Сталкер \(фильм\)](#) (b335d3804dbd9e7f832033789745d984bec60cee)
37. [Стачка \(фильм\)](#) (f6ee5f45205d575eb29e2f2380a40b4a6c6ba464)
38. [Андреева, Мария Фёдоровна](#) (4c2d63d1da1accbca36b1db6515f02ef232bd1bc)
39. [Барabanов, Николай Фёдорович](#) (5f1ae1ed33a7d2a95ae562aaa683d405c6bdf54)
40. [Смирнов, Алексей Макарович](#) (0e61c882a68c321c9509c6afca48a28672c4f4386)
41. [Янковский, Олег Иванович](#) (bdf0d13e0880931a3389fde899a2903c6ebd0d77)
42. [1904 год в кино](#) (b7ecec0f9dcb4966947fba095ac85aed772cd5f0)
43. [1921 год в кино](#) (8ed65e39291dd44eaa7143b9d648d2043aa0e76b)
44. [Автомобилисты \(мультфильм\)](#) (3f91fc95fd589111dac1fc67b3f3fd0f38fd8b50)
45. [Василенко, Нина Константиновна](#) (f8bc9c70bfe7432926c03559d8fc6c89604e3b4e)
46. [Викен, Александр Владимирович](#) (146c3d5406191eb2a2eed936044c129283dc6b32)
47. [Воробышко](#) (59c8e589dc667b995f06920950ddc4bdae5b0cac)
48. [Дым коромыслом \(мультфильм\)](#) (b956f5764cd1774994241dab611b6cb525b47331)
49. [Заячий хвостик](#) (b85f0e04e9c6f58cee4c51b86e335193e95c5123)
50. [Киракосян, Акон Гургенович](#) (92d2863862217cf9d0fc98f92ed39f76f4cb950d)

[Next](#)



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 выполнен анализ распределения частот и собраны статистики по корпусу и индексам. Поддержка ранжирования и позиционных постингов оставлена как потенциальное направление для дальнейшего развития.