Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Отчет по лабораторным работам

по курсу «Информационный поиск»

Студент: Н. С. Сарайкин
Группа: М8О-406Б-22
Преподаватель: А. А. Кухтичев
Дата: 28.12.2025
Оценка:
Подпись:

Москва, 2025

Содержание

1 Добыча корпуса документов

1.1 Цель и общая идея

Цель работы — сформировать текстовый корпус, пригодный для задач информационного поиска: токенизации, стемминга, анализа распределения частот (закон Ципфа) и построения булевого индекса.

Корпус формируется автоматически многопоточным поисковым роботом. Сбор организован так, чтобы избегать повторной загрузки уже сохранённых документов, сохранять метаданные (URL, источник, время получения) и обеспечивать продолжение работы после остановки.

Объём корпуса задаётся лимитом **31 000 документов**.

1.2 Выбор источников (научная тематика)

Для корпуса использовались два русскоязычных открытых источника, ориентированные на научные материалы и энциклопедические статьи:

- **Википедия (разделы и категории по науке)** — страницы научной тематики (физика, математика, информатика, биология, химия и др.), доступные через MediaWiki API;
- **Викиучебник/Викикниги (научные и учебные материалы)** — тексты учебного характера, доступные через MediaWiki API и стандартные страницы.

1.3 Хранилище и структура MongoDB

Данные хранятся в MongoDB в базе `IR_MAI`, коллекция документов — `labs_doc`. Коллекция содержит очищенные тексты и метаданные.

Каждая запись в `labs_doc` включает:

- `url` — исходный адрес страницы;
- `url_norm` — нормализованный URL (без фрагмента, единый вид);
- `source` — источник (`wikipedia_science`, `wikibooks_science` и т.п.);
- `fetched_at` — время загрузки (UNIX-время);
- `clean_text` — очищенный текст, используемый далее как корпус;
- `content_hash` — хэш содержимого для контроля дублей и изменений.

MongoDB развёрнута на нестандартном порту (отличном от `27017`); подключение задаётся строкой вида `mongodb://localhost:<PORT>` и хранится в конфигурации робота.

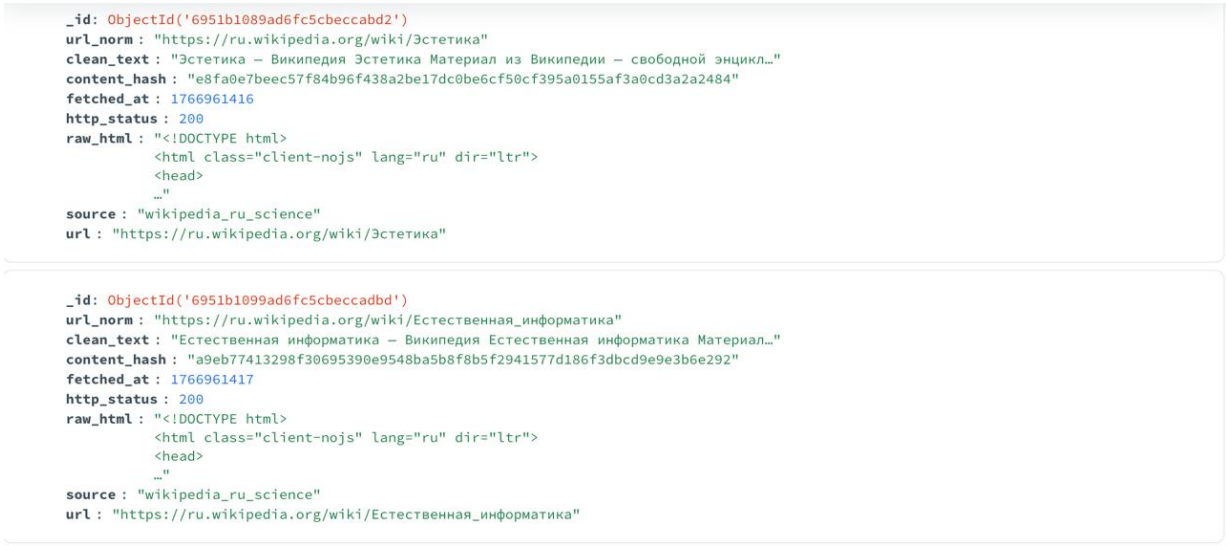

Рис. 1: Пример содержимого коллекции `IR_MAI.labs_doc`

2 Поисковый робот

2.1 Назначение и принцип работы

Поисковый робот предназначен для автоматического сбора документов и формирования корпуса. Робот реализован на Python и использует MongoDB как для хранения корпуса, так и для хранения состояния обхода.

Сбор выполняется параллельно в **7 потоков**. Для устойчивости к остановке применяется проверка наличия документа в базе и использование уникальности по `url_norm`; при повторном запуске робот продолжает работу и не загружает повторно уже сохранённые документы.

2.2 User-Agent и ограничения нагрузки

Для корректной идентификации робота используется User-Agent:

`Nikita Saraykin`

Для уважительного обхода источников задаются таймауты, задержки между запросами и ограничение числа повторов при ошибках.

2.3 Нормализация URL и уникальность

Для предотвращения дублей применяется нормализация URL:

- удаление фрагмента (`#...`);
- приведение домена/схемы к нижнему регистру;
- сохранение пути и query-части.

Нормализованный URL сохраняется в поле `url_norm` и используется как ключ уникальности.

2.4 Текстовая схема работы робота

Работа робота организована в виде конвейера:

1. формирование стартового набора URL (seed) по научным разделам/категориям источников;
2. параллельная обработка URL в пуле потоков: загрузка страницы по HTTP с учётом ограничений нагрузки;
3. очистка HTML и выделение чистого текста;
4. сохранение результата в MongoDB (`IR_MAI.labs_doc`) с метаданными и контролем уникальности по `url_norm`;
5. повторение шагов до достижения целевого количества документов.

2.5 Очистка и формирование поля `clean_text`

Очистка выполняется удалением служебных HTML-блоков (`script`, `style` и др.), извлечением текстового содержимого и нормализацией пробелов. Итоговый текст записывается в `clean_text` и используется далее для статистики и индекса.

3 Токенизация

3.1 Назначение токенизации

Токенизация разбивает текст документа на термы, пригодные для дальнейшей нормализации и индексирования. В работе токенизация применяется для подсчёта частот термов (анализ Ципфа) и для построения булевого индекса.

3.2 Реализация

Токенизация выполняется по очищенному тексту документа из поля `clean_text`. Алгоритм ориентирован на русскоязычный корпус и предназначен для выделения термов, пригодных для последующего стемминга и индексирования.

В качестве токенов выделяются последовательности буквенно-цифровых символов. Разделителями считаются пробельные символы и знаки пунктуации, которые не входят в состав терма. При проходе по строке текст просматривается слева направо; при обнаружении начала слова формируется токен до тех пор, пока не встретится разделитель.

Для повышения устойчивости к вариативности написания применяется нормализация регистра: латинские символы приводятся к нижнему регистру, что позволяет считать одинаковыми термы вида `Algorithm` и `algorithm`. Для кириллицы, учитывая особенности представления UTF-8, обработка выполняется как для многобайтовых символов, при этом сохраняется корректность выделения символов и целостность токена.

Дополнительно вводится фильтрация шумовых элементов. Короткие токены (например, длиной менее 2 символов) отбрасываются, поскольку они часто соответствуют предлогам, одиночным буквам и артефактам очистки. Также отбрасываются токены, состоящие только из цифр либо содержащие явно неязыковые последовательности, если такие встречаются в корпусе после очистки.

Результатом токенизации является последовательность термов для каждого документа. Далее эти термы передаются на этап стемминга и используются при построении частотного словаря и булевого индекса.

4 Стемминг

4.1 Назначение стемминга

Стемминг применяется для приведения словоформ к общей основе (стему), что уменьшает размер словаря и повышает полноту поиска за счёт уменьшения морфологической вариативности.

4.2 Реализация

Используется эвристический стеммер (rule-based suffix stripping), удаляющий распространённые суффиксы и окончания и возвращающий основу терма. Подход не является полноценным морфологическим анализом, однако эффективен для быстрого построения индекса и статистик.

5 Закон Ципфа и распределение частот

5.1 Суть закона Ципфа

Для естественных языков характерно неравномерное распределение частот слов: небольшое число термов встречается очень часто, а большинство — редко. Это описывается законом Ципфа:

$$f(r) \approx \frac{C}{r^s},$$

где r — ранг терма, $f(r)$ — частота, C — константа, s — параметр, обычно близкий к 1.

5.2 Формирование частот и CSV

Для построения распределения из MongoDB считывается поле `clean_text`. После токенизации и стемминга для каждого терма увеличивается счётчик частоты. Результат сохраняется в CSV-файл `frequencies.csv` формата `term,frequency`. Далее частоты сортируются по убыванию, вводится ранг r, и строится график в log–log координатах.

5.3 Аппроксимация законом Ципфа и причины расхождения

В log–log координатах распределение частот имеет выраженный степенной характер. Теоретическая кривая закона Ципфа хорошо согласуется с экспериментальными данными на среднем диапазоне рангов.

Отклонения от модели объясняются следующими причинами:

- в области малых рангов наиболее частотные термы встречаются чаще, чем предсказывает простая модель, что связано с общеязыковыми словами и повторяемостью ключевых научных терминов;
- в области больших рангов “хвост” распределения чувствителен к размеру корпуса, качеству очистки и особенностям токенизации/стемминга, а также содержит множество редких и уникальных термов;
- тематическая специфика научного корпуса усиливает частотность ограниченного набора терминов, смещая распределение относительно универсальной языковой модели.

Дополнительно может использоваться модель Ципфа–Мандельброта:

$$f(r) \approx \frac{C}{(r+\beta)^s},$$

которая позволяет более точно описывать начальный участок распределения.

5.4 График распределения частот

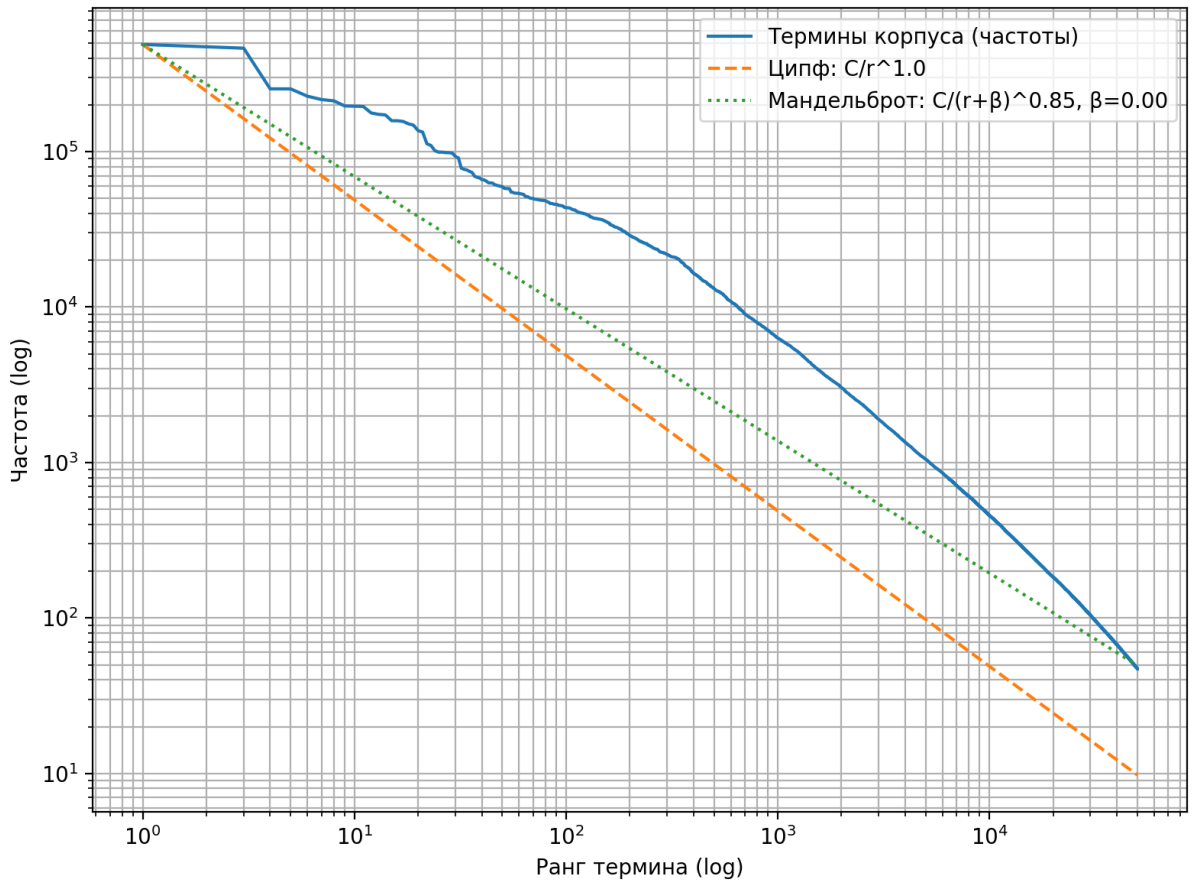

Рис. 2: Распределение частот термов корпуса в log–log координатах и аппроксимация законом Ципфа (и законом Ципфа–Мандельброта)

6 Булев индекс и булев поиск

6.1 Модель булевого поиска

Булев поиск относится к базовой (нулевой) модели информационного поиска, в которой документ либо удовлетворяет запросу, либо нет. Запрос формируется из термов и логических операторов `AND`, `OR`, `NOT`, а также скобок.

6.2 Построение инвертированного индекса

Индекс строится по документам корпуса (поле `clean_text`). Для каждого терма хранится постинг-лист — отсортированный список идентификаторов документов, в которых встречается терм.

Используются собственные структуры данных на базе `vector` и `string`. Отсортированность постинг-листов достигается за счёт последовательной обработки документов с монотонно растущими `doc_id`, что позволяет эффективно выполнять операции пересечения и объединения списков.

6.3 Детали построения индекса

Индекс строится по документам корпуса, где каждому документу соответствует внутренний идентификатор `doc_id` (нумерация от 0 до $N - 1$). Для каждого терма хранится постинг-лист — упорядоченный список `doc_id`, в которых встречается данный терм.

Построение индекса выполняется следующим образом:

1. документ токенизируется;
2. каждый токен нормализуется стеммингом;
3. терм добавляется в словарь, а `doc_id` добавляется в соответствующий постинг-лист.

Постинг-листы формируются в отсортированном виде благодаря тому, что документы обрабатываются последовательно, а добавление выполняется только при отличии от последнего добавленного идентификатора. Это обеспечивает отсутствие повторов и позволяет выполнять операции булевой алгебры линейными алгоритмами.

6.4 Разбор и вычисление булевых запросов

Запрос пользователя поддерживает логические операторы `AND`, `OR`, `NOT` и скобки. Для корректной обработки приоритетов используется стандартный подход:

- сначала запрос разбирается на токены (термы и операторы);
- затем выражение преобразуется в обратную польскую нотацию (RPN) с учётом приоритетов: `NOT` имеет наивысший приоритет, затем `AND`, затем `OR`;
- после этого RPN вычисляется стековым алгоритмом.

На этапе вычисления операции выполняются над постинг-листами:

- **AND** реализован как пересечение двух отсортированных списков двумя указателями;
- **OR** реализован как объединение двух отсортированных списков с устранением дублей;

- **NOT** реализован как дополнение относительно универсального множества документов (все `doc_id` корпуса), также алгоритмом двух указателей.

Данный подход обеспечивает предсказуемую асимптотику операций, так как пересечение и объединение выполняются за линейное время от суммарной длины обрабатываемых списков.

6.5 Локальный интерфейс поиска

Для тестирования реализован консольный интерфейс. Пользователь вводит булев запрос, система вычисляет множество релевантных документов и выводит список идентификаторов найденных документов. Дополнительно реализована команда открытия документа по номеру результата, что позволяет просмотреть текст и убедиться в корректности поиска.

Режимы работы приложения разделены:

- `build` — построение индекса из дампа корпуса и сохранение на диск;
- `ui` — загрузка индекса и интерактивный поиск.

Разделение на режимы позволяет один раз построить индекс и многократно использовать его для экспериментов с запросами без повторной индексации.

6.6 Вычисление булевых запросов

Запрос разбирается на токены, после чего выражение переводится в обратную польскую нотацию (RPN) с учётом приоритетов операций: `NOT` выше `AND`, `AND` выше `OR`. Далее выражение вычисляется стековым алгоритмом. Операции над множествами реализуются линейными алгоритмами по отсортированным массивам:

- **AND** — пересечение постинг-листов;
- **OR** — объединение постинг-листов;
- **NOT** — дополнение относительно универсального множества документов.

6.7 Локальный интерфейс поиска

Приложение реализовано как консольная программа с двумя режимами: `build` (построение индекса) и `ui` (интерактивный поиск). В интерактивном режиме поддерживаются ввод булевых запросов, просмотр списка найденных документов и команда `:open N` для открытия текста документа по номеру результата.

```
Local Boolean Search UI
Operators: AND OR NOT, parentheses: ( )
Commands:
  :open N    open document by result number (from last search)
  :exit

search> Наука AND Математика
hits: 3661
1) 6951b1039ad6fc5cbecca0d5
2) 6951b1039ad6fc5cbecca3f7
3) 6951b1049ad6fc5cbecca8d1
4) 6951b1059ad6fc5cbecca95c
5) 6951b1079ad6fc5cbeccaa28
6) 6951b1079ad6fc5cbeccaadb
7) 6951b1079ad6fc5cbeccab84
8) 6951b1089ad6fc5cbeccabd2
9) 6951b1099ad6fc5cbeccadbd
10) 6951b1099ad6fc5cbeccae4e
11) 6951b10a9ad6fc5cbeccaf1c
12) 6951b1109ad6fc5cbeccb4d5
13) 6951b1129ad6fc5cbeccb7cd
14) 6951b1139ad6fc5cbeccbdd1
15) 6951b1149ad6fc5cbeccc0a9
16) 6951b1149ad6fc5cbeccc106
17) 6951b1149ad6fc5cbeccc16a
18) 6951b1149ad6fc5cbeccc18d
19) 6951b1159ad6fc5cbeccc1b1
20) 6951b1159ad6fc5cbeccc368
```

Рис. 3: Пример выполнения запроса и вывод результатов

```
search> :open 2

DOC_ID: 6951b1039ad6fc5cbecca3f7
──────────────────────────────────
Наука в средневековом исламском мире — Википедия Наука в средневековом исламском мире Материал из Википедии — свободной энциклопедии Перейти к навигации Перейти к поиску Современные исследователи сходятся во мнении,
то на протяжении значительной части средневекового периода исламские страны были мировыми лидерами в сфере науки и технологий [ 1 ] . Исламская наука развивалась интенсивно в эпоху « Золотого века ислама (762–1258)»
Переводческое движение , сосредоточенное в багдадском Доме мудрости , перевело на арабский язык индийские, ассирийские, иранские и греческие научные труды, что положило начало развитию наук и философии в исламском м
е. Среди исламских учёных было много персов [ 2 ] [ 3 ] , арабов [ 4 ] , берберов, таджиков, мавров, тюрков и египтян. В конфессиональном плане, большинство учёных были мусульманами [ 5 ] [ 6 ] [ 7 ] , но встречались
также христиане [ 8 ] , иудеи [ 8 ] [ 9 ] , сабии и др. Возрождение XII века в Европе началось после латинских переводов трудов мусульманских учёных. Содержание 1 История 1.1 Зарождение исламской науки 1.2 Монгольско
 нашествие и последующий упадок науки 2 Науки 2.1 Астрономия 2.2 Математика 2.3 Другие науки 3 Оценки 4 См. также 5 Комментарии 6 Примечания 7 Литература 8 Ссылки История [ править | править код ] Зарождение исламско
 науки [ править | править код ] Основные статьи: Исламская наука , Золотой век ислама и Переводческое движение В начале IX века научным центром халифата становится « Дом мудрости » в Багдаде , в который халифами пр
лашались виднейшие учёные со всего исламского мира. Большинство багдадских учёных до XI века были выходцами из Средней Азии ( Аль-Хорезми , Хаббаш аль-Хасиб , Аль-Ферга
...[cut]

search>
```

Рис. 4: Открытие документа командой `:open`

7 Как запустить и протестировать

7.1 Формирование дампа корпуса

Очищенные тексты (`clean_text`) выгружаются в файл `corpus_dump.txt` в формате:

```
==DOC_START==
<doc_id>
==CONTENT_START==
<multiline text>
==DOC_END==
```

7.2 Сборка и запуск

Сборка C++ приложения:

```
g++ -O2 -std=c++17 IR/tokenizer.cpp IR/stemmer.cpp IR/term_dict.cpp \
  IR/boolean_query.cpp IR/corpus_io.cpp IR/main.cpp -o search_app
```

Построение индекса:

```
./search_app build corpus_dump.txt boolean_index.bin
```

Запуск интерфейса поиска:

```
./search_app ui boolean_index.bin corpus_dump.txt
```

7.3 Примеры тестовых запросов

- `математика AND анализ`
- `алгоритм OR вычисление`
- `информатика AND NOT история`
- `(математика AND теория) OR (физика AND модель)`

8 Выводы

В ходе выполнения лабораторных работ был реализован полный базовый конвейер информационного поиска: от сбора и очистки корпуса до построения статистик и булевого индекса.

Разработан многопоточный поисковый робот, сохраняющий состояние в MongoDB и обеспечивающий продолжение работы после остановки. На основе очищенных текстов выполнены токенизация и стемминг, что позволило снизить размер словаря и повысить полноту поиска.

Построен график распределения частот термов в log–log координатах и выполнено сравнение с законом Ципфа (и, при необходимости, с законом Ципфа–Мандельброта). Наблюдаемая форма распределения подтверждает естественную статистическую структуру корпуса и его пригодность для задач информационного поиска.

Реализован булев индекс и консольный интерфейс поиска, поддерживающий логические операции `AND/OR/NOT` и просмотр текстов документов.

9 Список литературы

1. Солтон Дж., Макгилл М. Введение в современный информационный поиск. — М.: Мир, 1983. — 416 с.
2. Кузнецов С. Д. Основы информационного поиска. — М.: Физматлит, 2009. — 320 с.
3. Мэннинг К., Рагхаван П., Шютце Х. Введение в информационный поиск. — М.: Диалектика, 2011. — 528 с.