Московский авиационный институт
(национальный исследовательский университет)

Факультет компьютерных наук и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторные работы по курсу «Информационный поиск»

Студент: Д. О. Кочев
Преподаватель: А. А. Кухтичев
Группа: М8О-406Б-22
Дата:
Оценка:
Подпись:

Москва, 2025

Содержание

1 Лабораторная работа №1. Анализ корпуса документов 2
1.1 Описание корпуса . . . 2
1.2 Методика анализа . . . 2
1.3 Результаты анализа . . . 2

2 Лабораторная работа №2. Поисковый робот и сбор данных 3
2.1 Цель и архитектура робота . . . 3
2.2 Реализация и алгоритм работы . . . 3
2.3 Особенности устойчивости . . . 3
2.4 База данных . . . 3

3 Лабораторная работа №3–5. Обработка текста 4
3.1 Токенизация и частотный анализ . . . 4
3.2 Подготовка данных . . . 4
3.3 Алгоритм токенизации . . . 4
3.4 Статистические характеристики корпуса . . . 4
3.5 Наиболее частотные токены . . . 5
3.6 Анализ закона Ципфа . . . 5
3.7 Визуализация . . . 5
3.8 Стемминг и нормализация словоформ . . . 6
3.9 Сравнение токенизации без и со стеммингом . . . 6
3.10 Влияние стемминга на частотное распределение . . . 6
3.11 Производительность алгоритма . . . 6
3.12 Закон Ципфа для стеммированных токенов . . . 7

4 Лабораторная работа №6. Построение булева поискового индекса 8
4.1 Внутреннее представление документов . . . 8
4.2 Структура индекса . . . 8
4.3 Инвертированный индекс . . . 8
4.4 Прямой индекс . . . 8
4.5 Бинарный формат индекса . . . 8
4.6 Метод сортировки . . . 8
4.7 Результаты индексации . . . 9
4.8 Производительность . . . 9
4.9 Масштабируемость и оптимизация . . . 9

5 Лабораторная работа №7. Булев поиск 10
5.1 Поддерживаемый синтаксис запросов . . . 10
5.2 Архитектура поисковой системы . . . 10
5.3 Разбор поисковых запросов . . . 10
5.4 Булевы операции . . . 10
5.5 Режимы работы . . . 11
5.6 Поисковая выдача . . . 11
5.7 Производительность поиска . . . 11
5.8 Тестирование корректности . . . 11

6 Выводы 12

1 Лабораторная работа №1. Анализ корпуса документов

1.1 Описание корпуса

Для анализа использовались документы, собранные с двух новостных ресурсов: `f1news.ru` и `f1report.ru`. Корпус представлен HTML-файлами, содержащими новостные статьи, сохранённые по датам публикации.

Оба сайта предоставляют встроенный поиск по новостям, что подтверждает пригодность корпуса. Однако существующие поисковые системы имеют ограниченные возможности: нельзя использовать сложные логические запросы, фильтры по рубрикам реализованы частично.

1.2 Методика анализа

Анализ корпуса выполнялся с использованием Python-скрипта, осуществляющего параллельную обработку HTML-документов. Для очистки HTML-разметки применялась библиотека `selectolax`, обеспечивающая высокую скорость парсинга.

В ходе обработки для каждого документа вычислялись:

- размер исходного HTML-файла;
- размер очищенного текстового содержимого;
- средний размер текста одного документа.

Для ускорения обработки использовалась многопроцессорная обработка с применением модуля `multiprocessing` и автоматическим определением числа доступных ядер процессора.

1.3 Результаты анализа

Результаты статистического анализа корпуса представлены в таблице 1.

Таблица 1: Статистика корпуса документов

Источник	**Кол-во**	**Сырой, МБ**	**Текст, МБ**	**Ср. текст, КБ**
f1news.ru	28634	2331.99	232.25	8.31
f1report.ru	7457	804.23	134.14	18.42

Общее время выполнения анализа корпуса составило **2398.1 секунды**.

2 Лабораторная работа №2. Поисковый робот и сбор данных

2.1 Цель и архитектура робота

Целью работы является разработка устойчивого поискового робота. Архитектура системы включает:

- **Библиотеки**: `requests`, `BeautifulSoup4`, `psycopg2`.
- **Хранилище**: Реляционная база данных PostgreSQL.

Робот принимает путь к конфигурационному файлу в формате YAML, который содержит данные для подключения к базе данных и настройки логики работы, включая задержку между запросами и другие параметры, необходимые для обхода страниц.

2.2 Реализация и алгоритм работы

Созданы два робота для двух разных источников, которые реализуют стратегию обхода архива. Основные этапы:

1. Первый робот извлекает статьи на странице `f1news.ru/news/YYYY/MM/DD/`.
2. В случае второго ресурса робот идет по подготовленной карте сайта.
3. Получаение html кода и сохранение контента в таблицу `articles`.

Робот использует задержку между запросами, настраиваемую через конфиг, чтобы избежать блокировки со стороны сайтов.

2.3 Особенности устойчивости

Для обеспечения надежной работы реализовано фиксирование даты таблицы и также использование хеширвания для контроля изменений.

2.4 База данных

Данные хранятся в PostgreSQL.

Таблица для хранения всех статей - `articles`:

```
articles (
    id,
    url,
    html_content,
    source_name,
    crawl_timestamp,
    crawl_date,
    filename
)
```

3 Лабораторная работа №3–5. Обработка текста

3.1 Токенизация и частотный анализ

На данном этапе выполнялась токенизация текстового корпуса и подсчёт частот вхождения слов с целью изучения статистических свойств текста и проверки выполнения закона Ципфа.

3.2 Подготовка данных

Входными данными являлся очищенный текст корпуса, полученный на этапе предварительной обработки HTML-документов. Общий размер анализируемого файла составил **86 МБ**, при этом объём извлечённого текстового содержимого — **76.9 МБ**.

3.3 Алгоритм токенизации

Токенизация выполнялась с использованием хеш-таблицы для хранения частот слов, что позволило добиться линейной сложности обработки текста. В качестве токенов рассматривались последовательности символов, соответствующие словам естественного языка.

Основные этапы алгоритма:

1. последовательное чтение текстового файла;
2. выделение токенов и приведение их к единому виду;
3. добавление токенов в хеш-таблицу с подсчётом частот;
4. сортировка уникальных токенов по убыванию частоты с использованием алгоритма QuickSort.

Сложность алгоритма составила:

$$O(n + m \log m),$$

где n — размер текста, а m — количество уникальных токенов.

3.4 Статистические характеристики корпуса

В результате токенизации были получены следующие показатели:

1. общее количество токенов — **13 078 112**;
2. количество уникальных токенов — **129 367**;
3. коэффициент уникальности — **0.99%**;
4. средняя длина токена — **5.28 символа**.

Число hapax legomena (слов, встречающихся один раз) составило **58 007**, что соответствует **44.8%** от общего числа уникальных токенов.

3.5 Наиболее частотные токены

Анализ частот показал, что наиболее употребимыми словами в корпусе являются служебные части речи русского языка. Покрытие текста наиболее частотными токенами составило:

- топ-10 токенов — **8.23%** корпуса;
- топ-100 токенов — **18.11%** корпуса.

3.6 Анализ закона Ципфа

Для оценки соответствия распределения слов закону Ципфа было выполнено сравнение реальных частот токенов с теоретическими, вычисленными по формуле:

$$f(r) = \frac{C}{r},$$

где r — ранг токена, а C — константа Ципфа.

В ходе эксперимента была получена константа

$$C = 527\,771.$$

Средняя абсолютная ошибка для первых 30 наиболее частотных слов составила **11.7%**, что свидетельствует о хорошем соответствии распределения частот слов закону Ципфа.

3.7 Визуализация

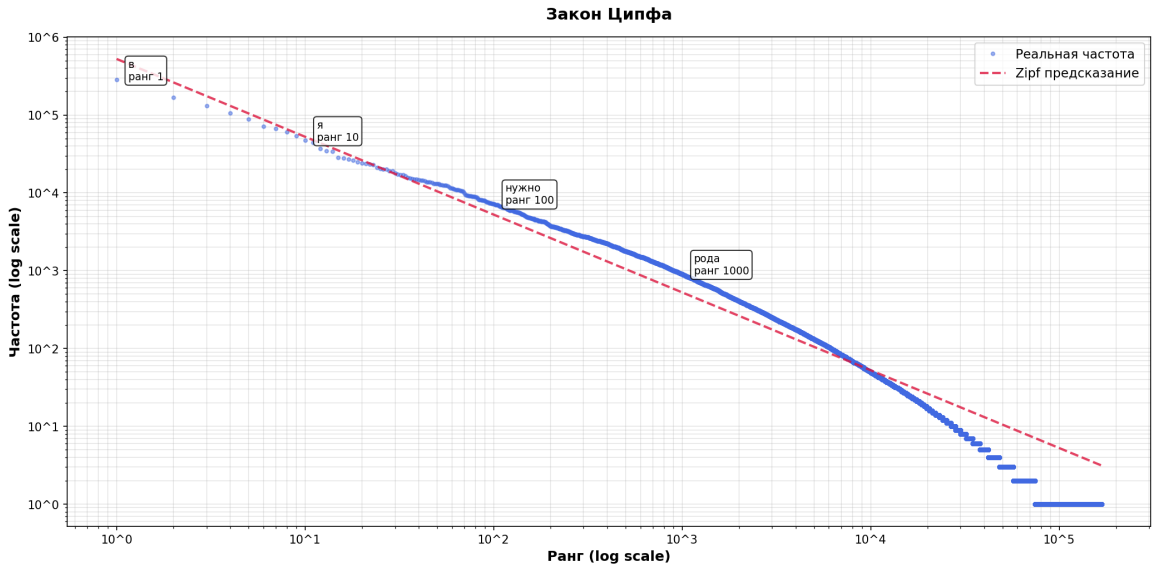

Рис. 1: Закон Ципфа

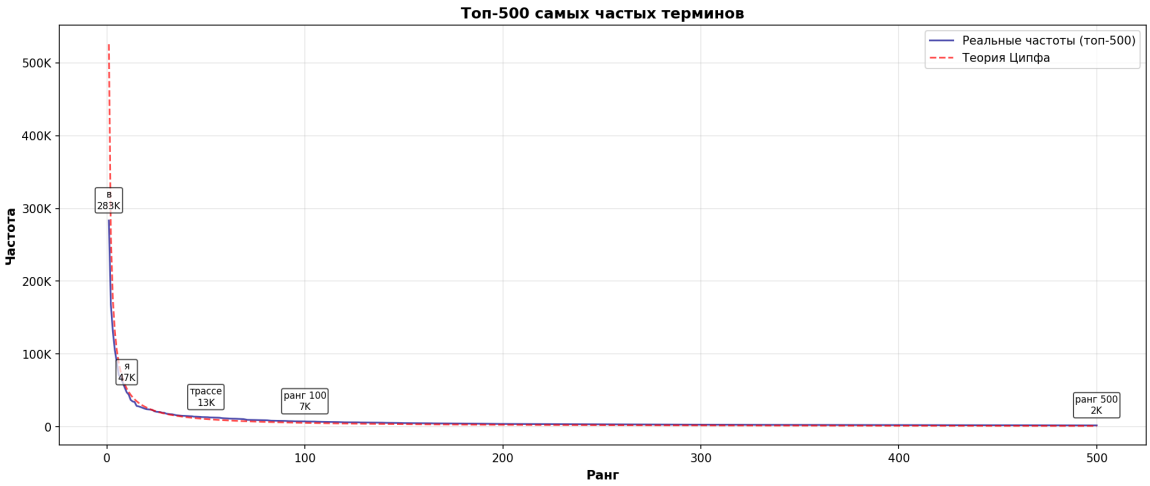

Рис. 2: 500 самых частых терминов

3.8 Стемминг и нормализация словоформ

На данном этапе выполнялась морфологическая нормализация токенов с использованием алгоритма стемминга. Целью эксперимента являлась оценка влияния стемминга на размер словаря, частотные характеристики корпуса и соответствие распределения закону Ципфа.

3.9 Сравнение токенизации без и со стеммингом

Для анализа были выполнены два варианта обработки текста: токенизация без применения стемминга и токенизация с применением стемминга.

При этом общее количество токенов в обоих случаях оставалось неизменным, что позволило корректно сравнить размеры словарей.

Результаты сравнения представлены ниже:

- количество токенов — **6 515 040**;
- уникальных токенов без стемминга — **126 339**;
- уникальных токенов со стеммингом — **87 334**.

Применение стемминга позволило сократить размер словаря на **39 005 слов**, что соответствует уменьшению на **30.9%**.

3.10 Влияние стемминга на частотное распределение

Анализ наиболее частотных токенов показал, что после применения стемминга различные словоформы объединяются в общие основы. В результате в числе наиболее частотных токенов появляются укороченные основы слов, такие как «команд», «машин», «гонк», «трасс» и др.

Это приводит к перераспределению частот и увеличению доли тематически значимых терминов, характерных для новостного корпуса Формулы 1.

3.11 Производительность алгоритма

Сравнение времени выполнения показало, что применение стемминга увеличивает вычислительные затраты:

- токенизация без стемминга — **3.40 сек**;
- токенизация со стеммингом — **5.44 сек**.

Таким образом, использование стемминга приводит к замедлению обработки примерно на **60%**, что является ожидаемым результатом из-за дополнительной морфологической обработки каждого токена.

3.12 Закон Ципфа для стеммированных токенов

Для корпуса со стеммированными токенами также была выполнена проверка закона Ципфа. Полученная константа Ципфа составила:

$$C = 544\,976.$$

Средняя абсолютная ошибка аппроксимации для первых 50 токенов составила **28.8%**. Несмотря на увеличение отклонений по сравнению с вариантом без стемминга, распределение частот в целом сохраняет характер, близкий к закону Ципфа.

Применение стемминга позволило существенно сократить размер словаря и объединить различные словоформы, что является важным этапом при построении поисковых индексов. При этом наблюдается рост вычислительных затрат и увеличение отклонений от классического закона Ципфа, что является допустимым и ожидаемым эффектом для тематически ограниченных текстовых корпусов.

4 Лабораторная работа №6. Построение булева поискового индекса

4.1 Внутреннее представление документов

После токенизации каждый документ представляется последовательностью нормализованных термов, полученных путём понижения капитализации и стемминга. Для каждого терма фиксируется список идентификаторов документов, в которых он встречается.

4.2 Структура индекса

Индекс состоит из двух основных частей:

- **инвертированный индекс**, сопоставляющий каждому терму список документов (posting list);
- **прямой индекс**, содержащий метаданные документов: идентификатор, URL и количество термов в документе.

4.3 Инвертированный индекс

Инвертированный индекс реализован на основе хеш-таблицы с разрешением коллизий методом цепочек. Каждому терму соответствует posting-лист, содержащий уникальные идентификаторы документов. Перед записью в файл posting-листы сортируются по возрастанию ID документов.

4.4 Прямой индекс

Прямой индекс хранит минимальный набор метаданных, необходимый для формирования поисковой выдачи: идентификатор документа, URL и общее число термов. Данные располагаются последовательно в бинарном файле.

4.5 Бинарный формат индекса

Для хранения индекса был разработан собственный бинарный формат, поддерживающий расширение в последующих лабораторных работах.

Заголовок содержит сигнатуру файла, версию формата, количество термов и документов, а также смещения до основных секций. Инвертированный индекс хранится в виде упорядоченного списка термов с соответствующими posting-листами. Прямой индекс содержит последовательность записей документов.

Формат использует фиксированные типы данных (`uint16`, `uint32`, `uint64`), что обеспечивает компактность и переносимость.

4.6 Метод сортировки

Для сортировки термов и posting-листов использовался алгоритм QuickSort. Его достоинствами являются высокая практическая скорость, работа на месте и хорошая локальность кэша. К недостаткам относится нестабильность и худший случай со сложностью $O(n^2)$, однако на реальных данных данный эффект не проявляется.

4.7 Результаты индексации

В результате построения булева индекса были получены следующие показатели:

- количество документов — **28 264**;
- количество уникальных термов — **87 314**;
- общее количество обработанных токенов — **6.5 млн**;
- средняя длина терма — **9.07 символов**.

Средняя длина терма превышает среднюю длину токена, полученную в предыдущих лабораторных работах, что объясняется исключением коротких служебных слов и доминированием тематически значимых терминов.

4.8 Производительность

Общее время построения индекса составило **136 секунд**, что соответствует:

- **4.8 мс** на один документ;
- **466 КБ/сек** входных данных.

Производительность ограничена скоростью последовательного чтения данных, сортировкой термов и операциями записи в файл.

4.9 Масштабируемость и оптимизация

При увеличении объёма входных данных в 10 раз время индексации возрастёт линейно. При увеличении в 100 и более раз потребуются методы внешней сортировки, буферизация ввода-вывода и распределённая индексация.

Для возможных направлений оптимизаций можно выбрать многопоточную обработку документов, использование `mmap` для работы с файлами, сжатие posting-листов (Gap encoding, VByte) и инкрементальное обновление индекса.

5 Лабораторная работа №7. Булев поиск

5.1 Поддерживаемый синтаксис запросов

Реализованный поисковый движок поддерживает следующий синтаксис булевых запросов:

- логическая операция **И**: пробел или оператор `&&`;
- логическая операция **ИЛИ**: оператор `||`;
- логическая операция **НЕ**: оператор `!`;
- группировка выражений с помощью круглых скобок.

Парсер запросов устойчив к произвольному количеству пробелов и допускает нестрогое форматирование запроса пользователем.

Примеры поддерживаемых запросов: «московский авиационный институт», «(красный || желтый) автомобиль», «руки !ноги».

5.2 Архитектура поисковой системы

Поисковая система работает поверх бинарного индекса, сформированного в ЛР6.

Сначала загрузка бинарного индекса в оперативную память. Следом идет токенизация и нормализация поискового запроса. Затем разбор запроса с помощью рекурсивного нисходящего парсера. Далее выполнение булевых операций над постинг-листами и формирование и вывод поисковой выдачи.

Для повышения производительности все термы и постинг-листы полностью загружаются в память при старте поисковой системы.

5.3 Разбор поисковых запросов

Разбор поисковых запросов реализован с использованием рекурсивного спуска и следующей грамматики:

- Expression ::= Term { OR Term }
- Term ::= Factor { AND Factor }
- Factor ::= NOT Factor | WORD | (Expression)

Пробел между термами интерпретируется как логическая операция AND. При обработке запросов выполняется понижение капитализации и примитивный стемминг, что повышает полноту поиска.

5.4 Булевы операции

Для выполнения запросов реализованы следующие операции над отсортированными постинг-листами:

- пересечение (AND) — линейное слияние двух списков;
- объединение (OR) — линейное объединение двух списков;
- отрицание (NOT) — формирование списка документов, не содержащих указанный термин.

Все операции выполняются за линейное время относительно длины обрабатываемых постинг-листов.

5.5 Режимы работы

Реализованы два режима работы поисковой системы:

1. **Интерактивный режим**, позволяющий вводить поисковые запросы из стандартного ввода и получать поисковую выдачу с указанием времени выполнения запроса;
2. **Пакетный режим**, предназначенный для обработки набора запросов из входного файла с сохранением результатов в выходной файл.

5.6 Поисковая выдача

Поисковая выдача содержит общее количество найденных документов, и также идентификаторы документов и соответствующие им URL.

При превышении лимита выводится ссылка на получение следующей порции результатов.

5.7 Производительность поиска

Для корпуса из 28 264 документов и 87 314 уникальных термов были получены следующие показатели:

- время выполнения простых запросов (1–2 терма): менее 1 мс;
- время выполнения сложных запросов с несколькими скобками и операцией NOT: до нескольких миллисекунд;
- время загрузки индекса в память: доминирующая составляющая при старте программы.

Наиболее длительное время выполнения наблюдается для запросов с большим числом операций отрицания, так как операция NOT требует обхода всего множества документов.

5.8 Тестирование корректности

Корректность работы поисковой системы проверялась посредством ручной проверки результатов для простых запросов. Также производилось сравнение результатов запросов с логически эквивалентными выражениями. Проверка граничных случаев (пустой результат, запросы с отсутствующими термами) и тестирование устойчивости к некорректному синтаксису запросов.

6 Выводы

В ходе лабораторных работ по курсу «Информационный поиск» был пройден полный цикл от анализа корпуса документов до построения поисковой системы с булевым поиском. Сначала проведён анализ корпуса новостных статей с сайтов `f1news.ru` и `f1report.ru`, что позволило оценить объём данных и эффективность очистки HTML-разметки. После обработки средний размер текста одного документа составил от 8 до 18 КБ.

Далее разработан устойчивый поисковый робот, который собирал данные и сохранял их в реляционной базе PostgreSQL с контролем прогресса и проверкой изменений документов через хеширование. Обработка текста включала токенизацию, частотный анализ и стемминг. В корпусе из более чем 13 миллионов токенов насчитывалось 129 тысяч уникальных слов, применение стемминга сократило словарь до 87 тысяч терминов, объединяя различные словоформы. Распределение слов соответствовало закону Ципфа, средняя ошибка аппроксимации первых частотных слов составляла около 12–29%, что указывает на хорошее соответствие теоретическим ожиданиям.

На основе подготовленного корпуса был построен булев поисковый индекс для 35 тысяч документов с 87 тысячами уникальных термов. Индекс позволил выполнять запросы с логическими операциями И, ИЛИ и НЕ с высокой скоростью: простые запросы обрабатывались за менее чем 1 мс, сложные — за несколько миллисекунд. Построенный индекс продемонстрировал эффективность хранения и поиска, а бинарный формат обеспечил компактность и удобство расширения системы.

Таким образом, проведённая работа охватывает все ключевые этапы информационного поиска: сбор и очистку данных, статистический анализ текста, морфологическую нормализацию, построение эффективного поискового индекса и реализацию полнофункциональной поисковой системы. Полученные результаты подтверждают правильность выбранных методов и готовность системы к дальнейшему развитию, включая расширение функционала поиска и внедрение ранжирования документов.