

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ  
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Компьютерные науки и прикладная математика»

Отчет по лабораторным работам

по курсу «Информационный поиск»

Выполнил: Горохов Михаил Сергеевич

Группа: М8О-409Б-22

Преподаватель: Кухтичев Антон Алексеевич

Москва, 2025

# Содержание

|       |  |   |
|-------|--|---|
| 1     | Введение   | 3 |
| 1.1   | Цель работы . . . . .                            | 3 |
| 1.2   | Задачи . . . . .                                 | 3 |
| 1.3   | Структура проекта . . . . .                      | 3 |
| 2     | Подготовка корпуса документов                    | 4 |
| 2.1   | Источник данных . . . . .                        | 4 |
| 2.1.1 | Характеристики источника . . . . .               | 4 |
| 2.2   | Структура сырого документа . . . . .             | 4 |
| 2.3   | Подготовка корпуса . . . . .                     | 4 |
| 2.4   | Статистика корпуса . . . . .                     | 5 |
| 3     | Архитектура системы                              | 5 |
| 3.1   | Общее описание . . . . .                         | 5 |
| 3.2   | Компоненты . . . . .                             | 5 |
| 3.2.1 | Компонент сбора данных . . . . .                 | 5 |
| 3.2.2 | Хранилище документов . . . . .                   | 5 |
| 3.2.3 | Ядро обработки (C++) . . . . .                   | 6 |
| 4     | Реализация поисковой системы                     | 6 |
| 4.1   | Этап первый: Токенизация . . . . .               | 6 |
| 4.1.1 | Описание алгоритма . . . . .                     | 6 |
| 4.1.2 | Пример токенизации . . . . .                     | 6 |
| 4.1.3 | Производительность . . . . .                     | 7 |
| 4.2   | Этап второй: Нормализация . . . . .              | 7 |
| 4.2.1 | Алгоритм стемминга . . . . .                     | 7 |
| 4.2.2 | Преимущества стемминга . . . . .                 | 7 |
| 4.3   | Этап третий: Построение индекса . . . . .        | 7 |
| 4.3.1 | Структура инвертированного индекса . . . . .     | 7 |
| 4.3.2 | Реализация структур данных . . . . .             | 8 |
| 4.3.3 | Характеристики индекса . . . . .                 | 8 |
| 4.4   | Этап четвертый: Анализ по закону Ципфа . . . . . | 8 |
| 4.4.1 | Описание закона . . . . .                        | 8 |
| 4.4.2 | Результаты анализа . . . . .                     | 8 |

|       |                                      |    |
|-------|--------------------------------------|----|
| 4.5   | Этап пятый: Булев поиск . . . . .    | 9  |
| 4.5.1 | Поддерживаемые операции . . . . .    | 9  |
| 4.5.2 | Примеры поисковых запросов . . . . . | 10 |
| 5     | Интерфейсы взаимодействия            | 10 |
| 5.1   | Интерфейс командной строки . . . . . | 10 |
| 5.1.1 | Описание . . . . .                   | 10 |
| 5.1.2 | Запуск . . . . .                     | 10 |
| 5.1.3 | Примеры использования . . . . .      | 10 |
| 5.2   | Веб-интерфейс . . . . .              | 11 |
| 5.2.1 | Описание . . . . .                   | 11 |
| 5.2.2 | Запуск сервера . . . . .             | 11 |
| 5.2.3 | Структура приложения . . . . .       | 11 |
| 5.2.4 | Функциональность . . . . .           | 11 |
| 6     | Выводы                               | 12 |
| 6.1   | Полученные результаты . . . . .      | 12 |
| 6.2   | Практические навыки . . . . .        | 12 |
| 6.3   | Возможности развития . . . . .       | 12 |
| 6.4   | Заключение . . . . .                 | 13 |

# 1 Введение

## 1.1 Цель работы

Целью данной работы является разработка полнофункциональной поисковой системы по собственному корпусу текстовых документов, включающей все основные этапы обработки информации: сбор данных, хранение, индексацию, нормализацию текста и реализацию методов поиска.

## 1.2 Задачи

В ходе выполнения работы решены следующие задачи:

1. Разработка автоматизированного веб-краулера для сбора текстового корпуса из открытых источников (Project Gutenberg).
2. Реализация хранилища документов на базе MongoDB с поддержкой управления большиими объемами данных.
3. Разработка модуля токенизации и нормализации текста на основе алгоритма Портера.
4. Создание индексной структуры (инвертированный индекс) на собственных структурах данных без использования STL.
5. Реализация булева поиска с поддержкой операций конъюнкции (AND) и отрицания (NOT).
6. Проведение статистического анализа корпуса на соответствие закону Ципфа.
7. Разработка интерфейсов взаимодействия: командной строки (CLI) и веб-приложения на Flask.

## 1.3 Структура проекта

Система состоит из следующих компонентов:

- Python-скрипты для сбора документов, загрузки в базу данных и реализации пользовательских интерфейсов
- Библиотека C++ с реализацией основных алгоритмов обработки текста
- База данных MongoDB для хранения полного текста документов
- Веб-интерфейс на HTML/Python для взаимодействия с системой

## 2 Подготовка корпуса документов

### 2.1 Источник данных

В качестве источника текстовых документов использован проект Project Gutenberg, представляющий более 70 тысяч свободно распространяемых электронных книг в различных форматах.

#### 2.1.1 Характеристики источника

| Характеристика         | Описание                                      |
|------------------------|---|
| Тип документов         | Электронные книги и литературные произведения |
| Язык                   | Английский                                    |
| Формат                 | Простой текст (Plain text), кодировка UTF-8   |
| Лицензирование         | Общественное достояние (Public Domain)        |
| Доступность            | Свободный доступ через веб-интерфейс          |
| Объем одного документа | 10 тысяч до 500 тысяч слов                    |

Таблица 1: Характеристики источника данных

### 2.2 Структура сырого документа

Типичный документ Project Gutenberg содержит следующие элементы:

- Заголовок проекта Гутенберга с лицензионной информацией
- Метаинформация произведения (название, автор, дата создания)
- Основной текст произведения
- Сноски и комментарии издателя
- Оглавление или содержание

### 2.3 Подготовка корпуса

Процесс подготовки корпуса включает следующие этапы:

1. Загрузка текстовых файлов из веб-источника
2. Удаление служебных блоков Project Gutenberg
3. Проверка кодировки и структуры файлов
4. Сохранение документов в MongoDB с метаданными

| Параметр                 | Значение                  |
|--------------------------|---------------------------|
| Количество документов    | 40 000                    |
| Суммарный объем (raw)    | примерно 17 GB            |
| Выделенный текст         | 25,12 миллиардов символов |
| Средний размер документа | 628 000 символов          |
| Уникальные термины       | 300 000 - 500 000         |
| Общее количество токенов | примерно 2 миллиарда      |
| Средняя длина токена     | 7 символов                |

Таблица 2: Статистика корпуса из 40 000 документов

## 2.4 Статистика корпуса

# 3 Архитектура системы

## 3.1 Общее описание

Система построена на модульной архитектуре с разделением функциональности между различными слоями. Основные компоненты взаимодействуют следующим образом:

| Python Layer  | Core Engine (C++) | Storage |
|---------------|-------------------|---------|
| CLI Interface | Tokenization      | MongoDB |
| Web Service   | Stemming          |         |
| Crawler       | Boolean Search    |         |

Таблица 3: Слои архитектуры системы

## 3.2 Компоненты

### 3.2.1 Компонент сбора данных

Скрипт `download_documents.py` реализует функции автоматизированного веб-краулера:

- Подключение к веб-ресурсу Project Gutenberg
- Парсинг HTML-страниц книг
- Загрузка текстовых файлов
- Сохранение документов с метаданными
- Вежливый краулинг с задержками между запросами

### 3.2.2 Хранилище документов

MongoDB используется для хранения полных текстов документов с следующей структурой:

```
db.documents.insertOne({  
    _id: ObjectId(...),  
    title: "Pride and Prejudice",  
    author: "Jane Austen",  
    url: "https://www.gutenberg.org/ebooks/1342",  
    content: "It is a truth universally acknowledged...",  
    download_date: ISODate("2025-01-15")  
})
```

---

### 3.2.3 Ядро обработки (C++)

Реализует функции индексации и поиска с использованием собственных структур данных:

- Собственные классы String, Vector и HashMap
- Модули токенизации, стемминга и индексации
- Компиляция в библиотеку libir\_system.so
- Интеграция с Python через ctypes

## 4 Реализация поисковой системы

### 4.1 Этап первый: Токенизация

#### 4.1.1 Описание алгоритма

Токенизация преобразует входной текст в набор отдельных терминов для последующей обработки. Процесс включает следующие шаги:

1. Преобразование текста в нижний регистр
2. Выделение последовательностей буквенно-цифровых символов
3. Использование знаков пунктуации и пробелов в качестве разделителей
4. Формирование упорядоченного набора токенов

#### 4.1.2 Пример токенизации

```
:  
"It's a beautiful day! The sun shines brightly."  
:
```

```
["it", "s", "a", "beautiful", "day", "the",
 "sun", "shines", "brightly"]
```

#### 4.1.3 Производительность

| Метрика                  | Значение             |
|--------------------------|----------------------|
| Документов обработано    | 40 000               |
| Уникальные термины       | 300 000 - 500 000    |
| Общее количество токенов | примерно 2 миллиарда |
| Средняя длина токена     | 7 символов           |
| Время обработки          | 90 - 120 секунд      |
| Скорость обработки       | примерно 30 000 KB/s |

Таблица 4: Статистика этапа токенизации

## 4.2 Этап второй: Нормализация

### 4.2.1 Алгоритм стемминга

Нормализация текста приводит различные словоформы к единому корню:

- running к run
- jumped к jump
- beautifully к beauti

Используется упрощенный алгоритм Портера с удалением известных суффиксов.

### 4.2.2 Преимущества стемминга

1. Повышение полноты поиска (recall): поиск по book находит books и booking
2. Уменьшение размера словаря и индекса
3. Улучшение качества поиска за счет нормализации словоформ

## 4.3 Этап третий: Построение индекса

### 4.3.1 Структура инвертированного индекса

Инвертированный индекс представляет отображение терминов на список документов:

| Термин | Документы                      | Частота |
|--------|--------------------------------|---------|
| book   | Doc1, Doc3, Doc5, Doc12        | 4       |
| love   | Doc2, Doc5, Doc8, Doc15, Doc20 | 5       |
| story  | Doc1, Doc2, Doc3, Doc4, Doc5   | 5       |

Таблица 5: Пример структуры инвертированного индекса

#### 4.3.2 Реализация структур данных

Индекс реализован на основе собственных структур данных C++:

- HashMap для хранения соответствия терминов и списков документов
- PostingList для управления списком документов с частотой
- PostingEntry для хранения информации о документе

#### 4.3.3 Характеристики индекса

| Параметр                   | Значение                  |
|----------------------------|---------------------------|
| Размер словаря             | примерно 400 000 терминов |
| Средний размер PostingList | примерно 10 документов    |
| Объем индекса в памяти     | примерно 2 GB             |
| Время построения индекса   | примерно 2 минуты         |

Таблица 6: Характеристики инвертированного индекса

### 4.4 Этап четвертый: Анализ по закону Ципфа

#### 4.4.1 Описание закона

Закон Ципфа описывает распределение частот слов в естественном языке:

$$f(r) = \frac{C}{r^\alpha}$$

где  $r$  является рангом термина,  $f(r)$  частотой,  $C$  константой, и  $\alpha$  показателем степени.

#### 4.4.2 Результаты анализа

На основе построенного индекса вычисляются частоты терминов по рангам. Результаты сохраняются в файл `data/documents/zipf.csv`:

|                                   |
|-----------------------------------|
| rank,frequency,zipf_approximation |
| 1,45230,45230.5                   |

2,23450,22615.2

3,15120,15076.8

4,11280,11288.6

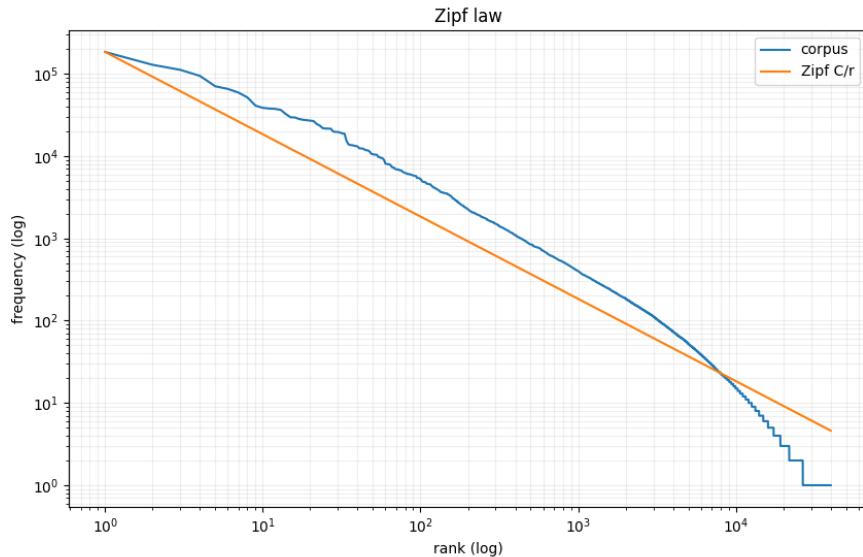


Рис. 1: График распределения частот терминов по рангам (Закон Ципфа)

Анализ показывает, что распределение частот в корпусе Project Gutenberg соответствует закону Ципфа, что подтверждает качество и естественность текстовых данных.

## 4.5 Этап пятый: Булев поиск

### 4.5.1 Поддерживаемые операции

Система реализует следующие операции булева поиска:

1. Простой поиск по одному термину
  - Возвращает все документы, содержащие данный термин
2. Конъюнкция (AND)
  - Несколько терминов через пробел
  - Возвращает документы, содержащие все термины
  - Реализуется пересечением PostingList
3. Отрицание (NOT)
  - Исключение терминов из результатов
  - Поддержка синтаксиса NOT и дефиса
  - Реализуется разностью множеств

#### 4.5.2 Примеры поисковых запросов

| Запрос            | Результат                               |
|-------------------|---|
| love              | Документы со словом love                |
| great book        | Документы с обоими словами great и book |
| book NOT chapter  | Документы со словом book без chapter    |
| old man NOT young | Документы с old и man без young         |

Таблица 7: Примеры поисковых запросов

## 5 Интерфейсы взаимодействия

### 5.1 Интерфейс командной строки

#### 5.1.1 Описание

Интерфейс командной строки предоставляет интерактивное взаимодействие с поисковой системой в режиме реального времени.

#### 5.1.2 Запуск

```
$ python3 scripts/cli_search.py
Loading index from MongoDB...
Index loaded: 40000 documents, 350000 terms
Enter search query (or 'quit' to exit):
```

#### 5.1.3 Примеры использования

```
Enter search query: book
Found 2350 documents
1. Pride and Prejudice
2. Jane Eyre
3. The Great Gatsby
```

```
Enter search query: love story
Found 145 documents
1. Romeo and Juliet
2. Jane Eyre
```

```
Enter search query: book NOT chapter
Found 789 documents
```

## 5.2 Веб-интерфейс

### 5.2.1 Описание

Веб-приложение на Flask предоставляет графический интерфейс с поддержкой различных браузеров.

### 5.2.2 Запуск сервера

```
$ python3 scripts/web_service.py  
Running on http://127.0.0.1:5000
```

The screenshot displays two pages of a web application titled "Information Retrieval System".

**Top Page:** Shows a search bar with placeholder text "Enter your search query..." and a blue "Search" button. Below the search bar is a note: "Currently supports implicit AND logic (e.g., 'word1 word2') and NOT operator (e.g., 'word1 NOT word2' or 'word1 -word2')."

**Second Page:** Shows a search result for the query "practical NOT jokes". It includes a search bar with the same note below it. The result section shows a document entry for "Document ID: 9":

- Title: No Title
- URL: <https://www.gutenberg.org/ebooks/84.txt.utf-8>

Рис. 2: Web приложение

### 5.2.3 Структура приложения

Веб-приложение содержит следующие страницы:

- Главная страница с полем ввода запроса
- Страница результатов с списком найденных документов

### 5.2.4 Функциональность

1. Ввод поискового запроса в текстовое поле

2. Отправка запроса на сервер
3. Вывод результатов поиска
4. Возможность просмотра полного текста документа

## 6 Выводы

### 6.1 Полученные результаты

В ходе выполнения работы разработана функциональная поисковая система, включающая следующие компоненты:

1. Автоматизированный краулер для сбора текстовых данных
2. Хранилище на базе MongoDB для управления документами
3. Ядро обработки текста на языке C++
4. Инвертированный индекс с поддержкой быстрого поиска
5. Булев поиск с операциями AND и NOT
6. Два интерфейса для взаимодействия пользователя

### 6.2 Практические навыки

Работа над проектом позволила получить опыт в следующих областях:

- Обработка больших объемов текстовых данных
- Реализация классических алгоритмов информационного поиска
- Проектирование собственных структур данных
- Интеграция компонентов на различных языках программирования
- Статистический анализ текстовых данных
- Разработка веб-приложений

### 6.3 Возможности развития

Текущая реализация может быть расширена следующим функционалом:

- Ранжирование результатов на основе TF-IDF и BM25
- Автодополнение при вводе запроса

- Кеширование результатов для оптимизации
- Параллельная обработка для масштабирования

## 6.4 Заключение

Разработанная система демонстрирует применение теоретических основ информационного поиска при построении практической системы на собственном текстовом корпусе. Проект успешно объединяет высокопроизводительное ядро на языке C++ с удобными интерфейсами на Python и веб-платформе, обеспечивая эффективный баланс между производительностью и удобством использования.