

Московский авиационный институт  
(национальный исследовательский университет)

Факультет информационных технологий и прикладной  
математики

Кафедра вычислительной математики и программирования

Лабораторная работа №4 по курсу «Информационный поиск»

Студент: Н. С. Федоров  
Преподаватель: А. А. Кухтичев  
Группа: М8О-410Б  
Дата:  
Оценка:  
Подпись:

Москва, 2025

# Лабораторная работа №4 «Разработка булевого индекса и булевого поиска по индексу»

В рамках данной работы была реализована подсистема индексирования и поиска по текстовой коллекции документов. В отличие от предыдущих заданий, формальная формулировка задачи не была задана заранее, поэтому цель работы определялась исходя из общей архитектуры информационно-поисковых систем и уже реализованных этапов обкатки, очистки и токенизации документов.

Целью данной работы является разработка и реализация булевого инвертированного индекса, обеспечивающего:

- добавление документов в индекс на основе их токенизированного представления;
- хранение для каждого термина списка документов, в которых он встречается;
- вычисление частоты термина по всей коллекции;
- обработку булевых поисковых запросов с операторами AND, OR, NOT;
- сохранение и загрузку индекса;

Реализуемый индекс предназначен для работы с корпусом новостных и аналитических статей, предварительно собранных поисковым роботом и очищенных от HTML-разметки.

## 1 Реализация и анализ решения

### 1 Общая архитектура индекса

В работе реализован булев инвертированный индекс, в основе которого лежит отображение термина в список идентификаторов документов, содержащих данный термин. Индекс реализован на языке C++ для повышения производительности и интегрирован с Python с помощью библиотеки `pybind11`.

Основной структурой данных является хеш-таблица фиксированного размера, в которой каждый бакет содержит список терминов, попавших в данный хеш.

Каждому термину сопоставляется структура, включающая:

- строковое представление термина;
- коллекционную частоту термина (collection term frequency);
- список идентификаторов документов (posting list).

## 2 Хеширование и хранение данных

Для распределения терминов по таблице используется полиномиальная хеш-функция с основанием 31. Размер таблицы задаётся при создании индекса и по умолчанию составляет 100 000 элементов, что позволяет снизить количество коллизий при работе с большой коллекцией документов.

При добавлении документа каждый токен:

- хешируется;
- либо добавляется как новый термин;
- либо обновляет существующую запись, увеличивая частоту и дополняя список документов.

Добавление идентификаторов документов в posting list происходит с контролем уникальности, что предотвращает повторное добавление одного и того же документа.

## 3 Обработка булевых запросов

Индекс поддерживает выполнение простых булевых запросов следующего вида:

- одиночный термин;
- `term1 AND term2`;
- `term1 OR term2`;
- `NOT term`.

Для обработки запросов используются стандартные алгоритмы слияния отсортированных списков:

- пересечение списков для операции AND;
- объединение списков для операции OR;
- дополнение относительно полного множества документов для операции NOT.

Все операции выполняются за линейное время относительно размеров posting list, что является классическим подходом в булевых информационно-поисковых системах.

## 4 Сериализация индекса

Для обеспечения повторного использования индекса реализованы методы сохранения и загрузки в бинарном формате. В файл записываются:

- размер хеш-таблицы;
- общее количество документов;
- содержимое всех бакетов таблицы, включая термины и posting list.

Данный подход позволяет полностью восстановить состояние индекса без повторного анализа корпуса документов.

## 5 Интеграция с Python

С использованием библиотеки `pybind11` класс индекса экспортируется в Python-модуль, что позволяет:

- добавлять документы из Python-кода;
- выполнять поисковые запросы;
- сохранять и загружать индекс;
- использовать индекс совместно с ранее реализованными модулями токенизации и стемминга.

Это делает индекс удобным компонентом гибридной системы, в которой ресурсоёмкие операции выполняются на C++, а управляющая логика остаётся на Python.

## 2 Выводы

В ходе выполнения данной работы были изучены и реализованы основные принципы построения поисковых систем, в частности механизм инвертированного индекса и булевого поиска. В процессе разработки была получена практическая реализация структуры данных, лежащей в основе большинства классических информационно-поисковых систем.

В ходе работы был получен опыт проектирования и реализации инвертированного индекса, включая хранение posting lists, подсчёт частот термов и выполнение логических операций AND, OR и NOT над результатами поиска. Также были изучены алгоритмы эффективного слияния отсортированных списков документов.

Отдельное внимание было уделено сериализации и десериализации сложных структур данных, что позволило реализовать сохранение индекса на диск и его последующую загрузку без повторной обработки документов. В рамках работы был получен практический опыт интеграции C++-кода с Python с использованием библиотеки `pybind11`, что продемонстрировало возможности комбинирования языков для создания производительных и удобных в использовании решений.

В результате выполнения работы было сформировано целостное понимание процесса индексирования текстовых данных и выполнения поисковых запросов, а также заложена основа для дальнейшего изучения более сложных моделей поиска и ранжирования документов.

## Список литературы

- [1] Маннинг, Рагхаван, Шютце *Введение в информационный поиск* — Издательский дом «Вильямс», 2011. Перевод с английского: доктор физ.-мат. наук Д. А. Ключина — 528 с. (ISBN 978-5-8459-1623-4 (рус.))