Московский авиационный институт (национальный исследовательский университет)

Институт информационных технологий и прикладной математики
«Кафедра вычислительной математики и программирования»

Отчёт по лабораторным работам по курсу «Информационный поиск» IV курс, VII семестр

Студентка: Соколова В.Д.

Преподаватель: Кухтичев А.А.

Группа: М8О-401Б-22

Дата: 28.12.2025

Оценка: _____

Подпись: _____

Москва 2025 г.

Содержание

1 Введение

Целью работы является построение учебной поисковой системы по собственному корпусу документов: от добычи корпуса и хранения «сырых» документов до токенизации, стемминга, анализа частот терминов и реализации булевого индекса и булевого поиска (CLI и Web). Реализованный пайплайн обеспечивает воспроизводимую обработку документов: сбор, извлечение текста, нормализация, построение индекса и выполнение булевых запросов.

2 Лабораторная работа №1. Добыча корпуса документов

2.1 Источник данных

Корпус сформирован из двух независимых источников на движке MediaWiki:

- Русская Википедия: `https://ru.wikipedia.org/w/api.php`.
- Русская Викитека: `https://ru.wikisource.org/w/api.php`.

2.2 Характеристики документов и мета-информация

Для каждого документа сохраняются:

- «сырой» документ в формате HTML, полученный через `action=parse, prop=text`;
- выделенный текст в виде плоского текста (без вики-разметки), полученный через `prop=extracts, explaintext=1`;
- мета-информация: `doc_id`, `page_id`, `title`, `source`, а также имена файлов raw/text.

Дополнительно для выделенного текста применяется обрезка хвостовых разделов («Ссылки», «Примечания», «Литература», «Внешние ссылки»), что снижает долю шумовых фрагментов в дальнейшей индексации.

2.3 Выделение текста

Выделенный текст получен средствами MediaWiki API (`extracts`) и приведён к единому виду: UTF-8, без HTML и без вики-разметки, с минимальной эвристической очисткой хвостовых разделов. Такой текст используется как вход для токенизации и последующих лабораторных работ.

2.4 Готовые поисковики и примеры запросов

Для обоих источников существует готовый поиск:

- встроенный поиск на сайтах Википедии и Викитеки;
- поиск Google с ограничением по домену: `site:ru.wikipedia.org` и `site:ru.wikisource.org`.

Примеры запросов и характерные недостатки выдачи:

- `site:ru.wikipedia.org фильм триллер 1999` — в выдаче нередко доминируют страницы-списки, категории и навигационные страницы, а не конкретные статьи.

- `site:ru.wikisource.org стихотворение осень` — встречаются оглавления, страницы категорий и служебные страницы, что снижает точность полнотекстового поиска.
- `site:ru.wikipedia.org режиссер снял фильм` — результаты содержат шум из шаблонов и повторяющихся блоков, а также страдают от разной морфологии слов без учёта словоформ.

2.5 Статистика корпуса

Корпус содержит 37 420 документов. Для оценки длины документов использована выборка из 2000 документов; длина измерялась как число символов в выделенном тексте. Получены следующие значения:

- минимальная длина: 263;
- медиана: 2066;
- средняя длина: 3648;
- максимальная длина: 77221.

Статистика по «сырым» документам (HTML) оценивается отдельно по размеру сохранённых raw-файлов; выделенный текст имеет меньший объём за счёт отсутствия разметки и служебных блоков.

3 Лабораторная работа №2. Поисковый робот

3.1 Требования и формат конфигурации

Поисковый робот реализован как консольная программа, которой передаётся единственный аргумент: путь до `YAML`-конфига. Конфиг содержит:

- секцию `db` с параметрами подключения к базе данных;
- секцию `logic` с параметрами робота, включая задержку между запросами;
- секцию источников (`sources`), содержащую данные MediaWiki (endpoint API, категории/стартовые точки).

3.2 Схема хранения в базе данных

Робот сохраняет документы в базе данных со следующими полями:

- `url` — нормализованный URL документа;
- `html` — «сырой» HTML-текст документа;
- `source` — название источника;
- `fetch_ts` — дата обкачки в формате Unix timestamp;
- `hash` — контрольная сумма содержимого (для проверки изменений).

3.3 Нормализация URL

Для MediaWiki-страниц URL приводится к единому виду `https://<host>/?curid=<page_id>` без лишних параметров. Это обеспечивает устойчивое совпадение ключа документа в БД и корректность дедупликации.

3.4 Продолжение после остановки

Состояние робота хранится в отдельной записи (`state`), включающей курсор обхода (позицию/cont token). При повторном запуске робот читает состояние и продолжает обкачку с последнего сохранённого документа без повторной загрузки уже обработанных элементов.

3.5 Переобкачка и обновление только при изменениях

Для переобкачки используется сравнение контрольной суммы `hash`. Документ переобкачивается по расписанию, но запись обновляется только если новая контрольная сумма отличается от сохранённой. Это снижает нагрузку на источник и объём лишних операций записи в БД.

4 Лабораторная работа №3. Токенизация

4.1 Правила токенизации

Токенизация реализована для входного текста в UTF-8. Токеном считается максимальная последовательность символов из множества:

- кириллица (включая Ё/ё);
- латиница;
- цифры.

Все буквенные символы приводятся к нижнему регистру. Любые символы вне указанного множества являются разделителями токенов. Результат сохраняется в файлы `*.tok` (один токен на строку).

4.2 Достоинства и недостатки

Достоинства:

- линейная сложность по объёму входных данных;
- одинаковые правила для разных источников;
- устойчивость к «грязному» тексту и случайным символам.

Недостатки:

- токены с внутренними символами (`C++`, `e-mail`) дробятся;
- дефисные конструкции («С.-Петербург») распадаются на части;
- отдельные числовые и буквенно-числовые шаблоны требуют отдельных правил, если нужна повышенная точность.

4.3 Статистические данные и производительность

Для токенизации собираются:

- количество токенов;
- средняя длина токена.

Время выполнения измеряется на уровне всего корпуса. Скорость токенизации рассчитывается как число килобайт входного текста, обработанных за секунду:

$$V = \frac{\text{bytes\_in}/1024}{T}.$$

Для ускорения токенизации применимы: потоковая обработка без чтения файла целиком в память, переиспользование буферов, параллельная обработка файлов и уменьшение количества аллокаций.

5 Лабораторная работа №4. Лемматизация / стемминг

5.1 Метод нормализации словоформ

В систему добавлен стемминг русского языка, выполняемый над токенами в UTF-8. Применены правила:

- токены с цифрами не стеммятся;
- токены без кириллицы не стеммятся;
- задаётся минимальная длина основы, предотвращающая чрезмерную обрезку;
- снимаются типовые русские окончания и возвратность («ся/сь») по списку суффиксов.

5.2 Место применения стемминга

Стемминг применяется на этапе индексации и на этапе выполнения запроса:

- индекс строится по стеммированным токенам, что увеличивает полноту;
- термы запроса приводятся к нижнему регистру и стеммируются теми же правилами, что обеспечивает совпадение с индексом.

5.3 Оценка влияния на качество поиска

Стемминг улучшает качество поиска на запросах с разными словоформами (рост полноты), например: «фильм/фильмы/фильма», «режиссёр/режиссёра/режиссёры». Ухудшения возможны из-за слияния разных слов в одну основу при агрессивной обрезке окончаний, а также на коротких словах и именах собственных. Снижение таких ошибок достигается повышением минимальной длины основы, ограничением набора снимаемых окончаний и комбинированием точного совпадения со стеммингом.

6 Лабораторная работа №5. Закон Ципфа

6.1 Построение распределения частот

По токенизированному корпусу построено распределение частот терминов. Термины отсортированы по убыванию частоты, каждому термину назначен ранг r, и построен график в логарифмических шкалах $\log f(r)$ от $\log r$.

6.2 Наложение закона Ципфа

Использована модель Ципфа:

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

Коэффициент C выбран по частоте термина ранга 1, параметр s принят близким к 1. Наложение закона выполнено на график распределения частот терминов.

6.3 Причины расхождений

Расхождения с идеальной моделью объясняются:

- предметной спецификой корпуса и неоднородностью тем;
- наличием часто повторяющихся служебных фрагментов и шаблонов;
- влиянием нормализации: стемминг объединяет словоформы, увеличивая частоты «основ»;
- ограниченностью корпуса по категориям и неслучайностью выборки документов.

6.4 Графики

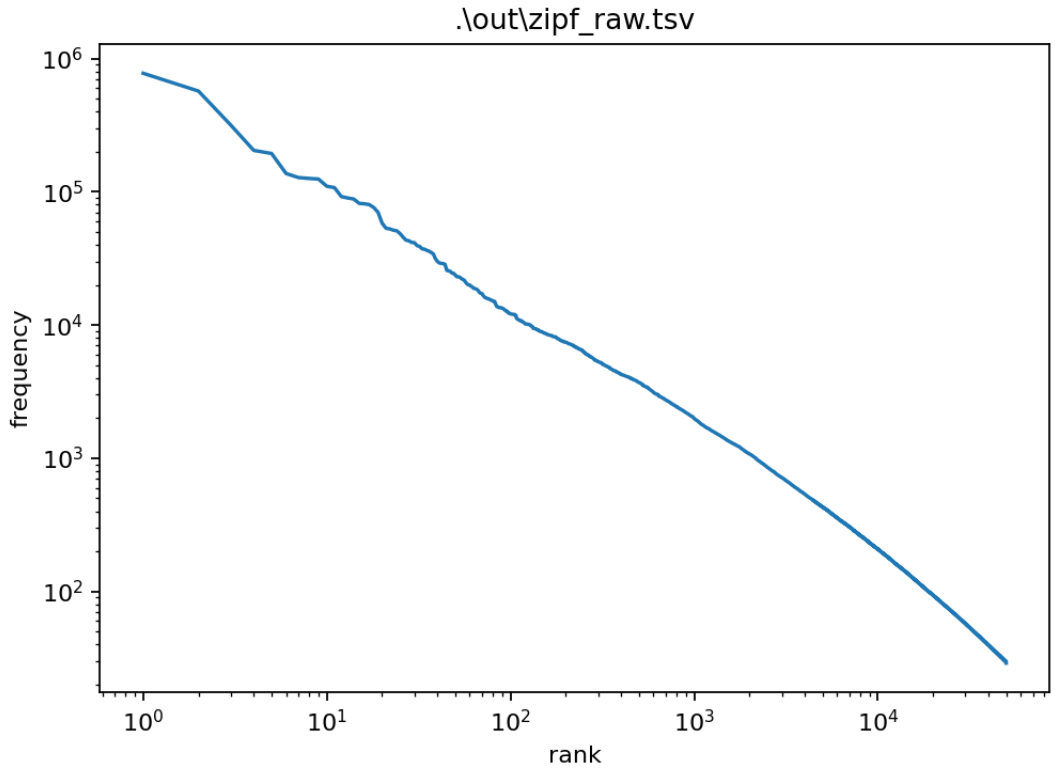

Рис. 1: Распределение частот терминов и закон Ципфа (без стемминга), логарифмические шкалы.

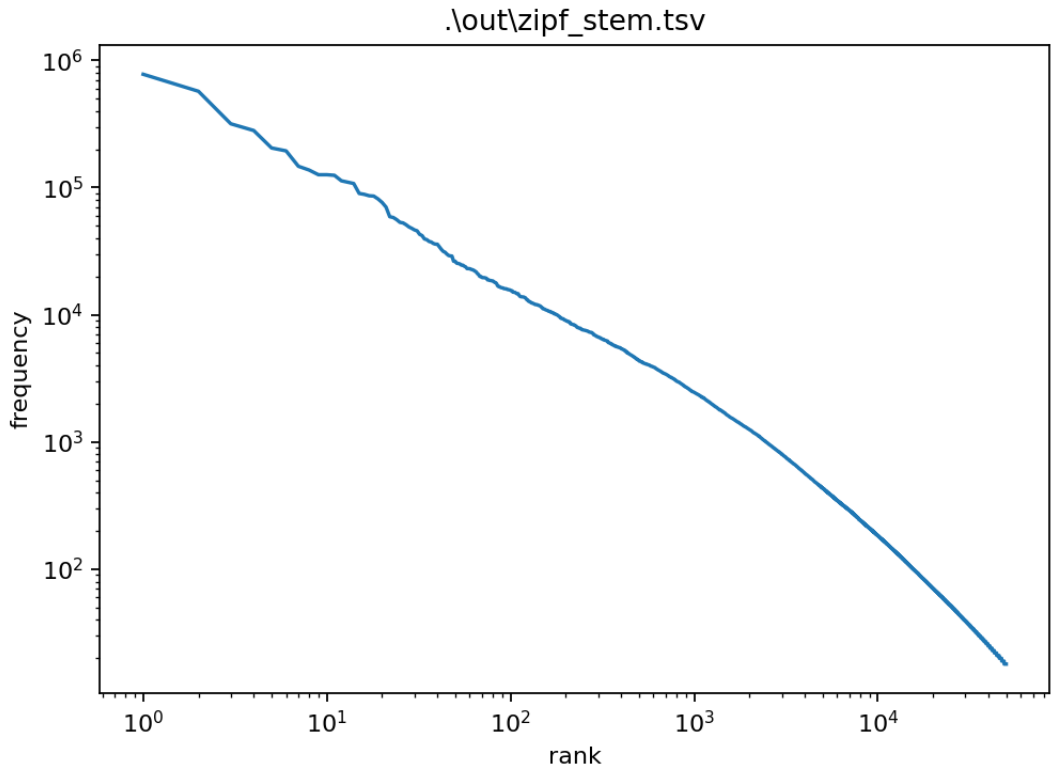

Рис. 2: Распределение частот терминов и закон Ципфа (со стеммингом), логарифмические шкалы.

7 Лабораторная работа №6. Булев индекс

7.1 Постановка

Реализован булев индекс (инвертированный индекс) для выполнения операций AND/OR/NOT по термам. Индекс строится по токенам (включая стеммированную версию) и хранит для каждого термина список идентификаторов документов, в которых он присутствует.

7.2 Формат индекса

Индекс сохранён в бинарном файле `index.bin`, содержащем:

- заголовок с версией, флагами и смещениями секций;
- словарь терминов (терм, смещение postings, длина postings);
- секцию postings (последовательности `doc_id`);
- секцию документов с метаданными: `doc_id`, `page_id`, `title`, `source`.

7.3 Учёт двух источников

В метаданных документа сохранён `source`. Это используется в поиске для формирования корректного URL:

- `ruwiki`: `https://ru.wikipedia.org/?curid=<page_id>`;
- `ru_wikisource`: `https://ru.wikisource.org/?curid=<page_id>`.

8 Лабораторная работа №7. Булев поиск

8.1 Синтаксис запросов

Поддержаны операции:

- AND: пробел или `&&`;
- OR: `||`;
- NOT: `!`;
- скобки: `( )`.

Термы запроса токенизируются по тем же правилам, что и документы: UTF-8, кириллица/латиница/цифры, приведение к нижнему регистру, стемминг.

8.2 Алгоритм обработки запроса

Запрос переводится в обратную польскую нотацию (алгоритм shunting-yard) с приоритетами NOT > AND > OR. Вычисление выполняется над postings-списками:

- AND — пересечение отсортированных списков;
- OR — слияние отсортированных списков;
- NOT — разность относительно множества всех документов.

8.3 Формирование выдачи

Для каждого найденного `doc_id` извлекаются метаданные (title, page_id, source), после чего формируется ссылка на источник и выводится результат. Реализована пагинация выдачи через параметры `offset` и `limit`. В веб-интерфейсе Streamlit выводятся кликабельные ссылки на документы.

9 Заключение

Выполнен полный цикл построения учебной поисковой системы. Сформирован корпус из двух источников MediaWiki, реализован робот обкачки с хранением raw HTML и метаданными в базе данных, выполнены токенизация и стемминг, исследовано распределение частот терминов и закон Ципфа, построен булев индекс и реализован булев поиск с поддержкой AND/OR/NOT и скобок, а также интерфейсом выдачи в CLI и Web.