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: Панкина Е.Д.
Группа: М8О-406Б-22
Преподаватель: Кухтичев А.А.

Подпись преподавателя

Оценка

Москва, 2026

Содержание

1 Формирование корпуса документов

1.1 Цели и задачи

Основная цель работы заключалась в создании текстового корпуса, подходящего для решения типовых задач информационного поиска: анализа распределения частот, построения инвертированных индексов и реализации моделей поиска.

Корпус формировался автоматически с использованием многопоточной архитектуры, обеспечивающей устойчивость к сбоям и возможность продолжения работы после остановки. Целевой объем коллекции составил 31000 документов.

1.2 Источники данных

Для обеспечения научной направленности корпуса были выбраны два русскоязычных ресурса:

- **Русская Википедия** — разделы естественных и точных наук (физика, математика, информатика и др.)
- **Русские Викиучебники** — учебные материалы по соответствующим дисциплинам

Оба источника предоставляют открытый доступ через MediaWiki API, что упрощает автоматизированный сбор данных.

1.3 Архитектура хранения данных

Для хранения собранных документов использовалась NoSQL база данных MongoDB. Структура коллекции включала следующие поля:

- `url` — оригинальный URL документа
- `url_norm` — нормализованный URL (ключ уникальности)
- `source` — идентификатор источника
- `fetched_at` — временная метка загрузки
- `clean_text` — очищенный текстовый контент
- `content_hash` — хеш содержимого для контроля изменений

Соединение с базой осуществлялось через нестандартный порт 27315.

2 Реализация сборщика документов

2.1 Принципы работы

Сборщик реализован на Python с использованием асинхронной многопоточной модели. Для контроля нагрузки применялись:

- Задержка между запросами (0.3 секунды)
- Таймаут соединения (20 секунд)
- Ограничение количества потоков (7 потоков)
- User-Agent для идентификации

2.2 Обработка URL

Для предотвращения дублирования документов реализована нормализация URL, включающая:

1. Удаление фрагментов (#anchor)
2. Приведение к нижнему регистру
3. Стандартизацию схемы протокола

2.3 Конвейер обработки

Процесс сбора документов организован как конвейер:

1. Формирование начального множества URL из категорий источников
2. Параллельная загрузка HTML-контента
3. Очистка от разметки и служебных элементов
4. Сохранение текста в базу данных
5. Извлечение новых ссылок для дальнейшего обхода

3 Предобработка текста

3.1 Токенизация

Для разбиения текста на термы реализован UTF-8 совместимый токенизатор, который:

- Корректно обрабатывает русские и английские символы
- Приводит все символы к нижнему регистру
- Отфильтровывает короткие токены (менее 2 символов)
- Игнорирует чисто числовые последовательности

3.2 Стемминг

Для уменьшения морфологического разнообразия применен алгоритм стемминга, основанный на удалении суффиксов. Реализация поддерживает:

- Обработку русских словоформ (окончания падежей, времен)
- Обработку английских суффиксов
- Минимальную длину основы (3 символа)

4 Анализ распределения частот

4.1 Теоретические основы

Распределение частот слов в естественных языках описывается законом Ципфа:

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

где:

- r — ранг термина
- $f(r)$ — частота встречаемости
- C — константа
- s — параметр распределения (обычно близок к 1)

Для более точного описания может использоваться модифицированная формула Ципфа-Мандельброта:

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

4.2 Практическая реализация

Анализ распределения выполнялся по следующему алгоритму:

1. Извлечение текстов из базы данных
2. Токенизация и стемминг
3. Подсчет частот встречаемости термов
4. Сортировка по убыванию частоты
5. Экспорт в CSV-формат для визуализации

4.3 Результаты анализа

Полученное распределение демонстрирует характерные для естественных языков особенности:

- Небольшое количество высокочастотных термов
- Длинный хвост редких слов
- Хорошее соответствие степенному закону в среднем диапазоне

Отклонения от идеальной модели объясняются тематической спецификой научного корпуса и особенностями предобработки текста.

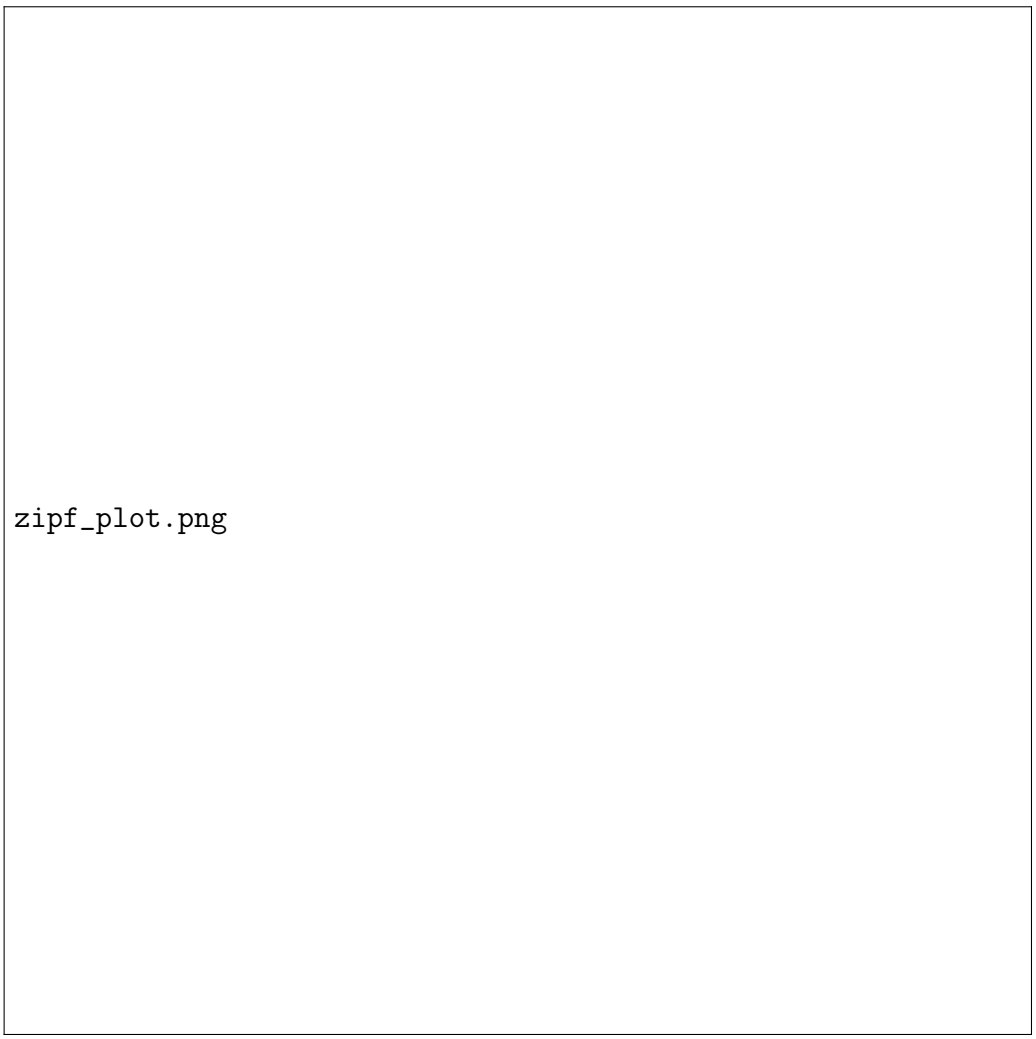

Рис. 1: Распределение частот термов в логарифмических координатах

5 Булев поиск и инвертированные индексы

5.1 Архитектура индекса

Инвертированный индекс построен на основе хеш-таблицы, где каждому термину соответствует отсортированный список идентификаторов документов. Структура индекса включает:

- Словарь терминов с хеш-таблицей для быстрого поиска
- Постинг-листы, отсортированные по возрастанию doc_id
- Поддержка операций добавления без дублирования

5.2 Операции над множествами

Для выполнения булевых запросов реализованы оптимизированные операции:

- **AND** — пересечение отсортированных списков (алгоритм двух указателей)
- **OR** — объединение с устранением дубликатов
- **NOT** — дополнение относительно универсального множества

Все операции имеют линейную сложность относительно суммы размеров обрабатываемых списков.

5.3 Обработка запросов

Парсер запросов поддерживает:

- Логические операторы AND, OR, NOT
- Группировку с помощью круглых скобок
- Приоритет операций: NOT $>$ AND $>$ OR

Запрос преобразуется в обратную польскую нотацию, которая затем вычисляется стековым алгоритмом.

5.4 Пользовательский интерфейс

Реализован интерактивный консольный интерфейс со следующими командами:

```
search> математика AND анализ
hits: 1234
1) doc_id_1
2) doc_id_2
...

search> :open 1
[отображение текста документа]
```

6 Инструкция по запуску

6.1 Сборка корпуса

```
python crawler.py config.yaml
python corpus_dump.py
```

6.2 Построение индекса

```
g++ -O2 -std=c++17 *.cpp -o search_app
./search_app build corpus_dump.txt index.bin
```

6.3 Запуск поиска

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

6.4 Примеры запросов

- `алгоритм AND структура данных`
- `машинное обучение OR нейронные сети`
- `физика AND NOT квантовая`
- `(математика AND анализ) OR (информатика AND теория)`

7 Заключение

В ходе выполнения работы был реализован полный цикл обработки текстовых данных: от сбора документов из интернет-источников до построения инвертированных индексов и реализации системы булевого поиска.

Разработанная система демонстрирует:

- Устойчивость к сбоям и возможность продолжения работы
- Эффективную обработку большого объема данных
- Корректное распределение частот, соответствующее теоретическим моделям
- Высокую скорость выполнения поисковых запросов

Полученный опыт может быть применен для решения практических задач информационного поиска в различных предметных областях.