Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторные работы по курсу «Информационный поиск»

Студент: В. А. Сухов
Преподаватель: А. А. Кухтичев
Группа: М8О-401Б
Дата:
Оценка:
Подпись:

Москва, 2025

Лабораторная работа №1 «Добыча корпуса документов»

Необходимо подготовить корпус документов, который будет использован при выполнении остальных лабораторных работ:

- Скачать его к себе на компьютер. В отчёте нужно указать источник данных.
- Ознакомиться с ним, изучить его характеристики. Из чего состоит текст? Есть ли дополнительная мета-информация? Если разметка текста, какая она?
- Разбить на документы.
- Выделить текст.
- Найти существующие поисковики, которые уже можно использовать для поиска по выбранному набору документов (встроенный поиск Википедии, поиск Google с использованием ограничений на URL или на сайт). Если такого поиска найти невозможно, то использовать корпус для выполнения лабораторных работ нельзя!
- Привести несколько примеров запросов к существующим поисковикам, указать недостатки в полученной поисковой выдаче.

В результатах работы должна быть указаны статистическая информация о корпусе:

- Размер «сырых» данных.
- Количество документов.
- Размер текста, выделенного из «сырых» данных.
- Средний размер документа, средний объём текста в документе.

Описание

Требуется выбрать корпус документов, по которому в следующих лабораторных работах буду их использовать, ознакомиться с корпусами и проанализировать их HTML код, привести примеры поисковых запросов к выбранному корпусу документов.

Источник данных

Для формирования корпуса были выбраны ресурсы, посвященные аниме-культуре и рецензиям:

- **Shikimori** `https://shikimori.one` — крупнейшая русскоязычная энциклопедия аниме и манги.
- **Kanobu** `https://kanobu.ru` — популярный портал о видеоиграх и поп-культуре, содержащий развернутые рецензии.

Обоснование выбора: Тексты рецензий на данных ресурсах написаны естественным языком, содержат специфическую терминологию, сленг и оценочные суждения. Это делает корпус репрезентативным для проверки алгоритмов стемминга и поиска (закон Ципфа на таких текстах проявляется особенно ярко).

Предварительный анализ структуры документов

Каждый документ представляет собой HTML-страницу. В ходе анализа выявлены следующие особенности:

- **Shikimori**: Текст рецензии находится в блоках `<div>` с классом `b-review-topic`. Заголовки часто отсутствуют (используется никнейм автора), поэтому в качестве заголовка берется название тайтла.
- **Kanobu**: Использует сложную верстку. Текст статьи расположен в блоках `c-entry__body` или размечается атрибутом `itemprop="articleBody"`.

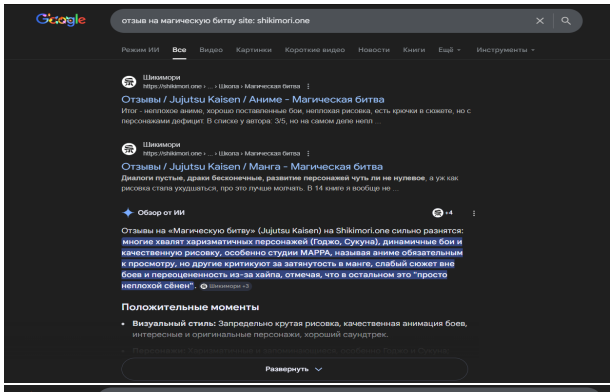
Примеры документов

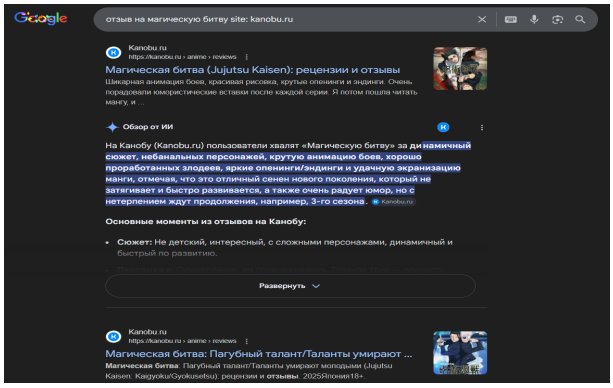
Пример документа с **shikimori.one**:

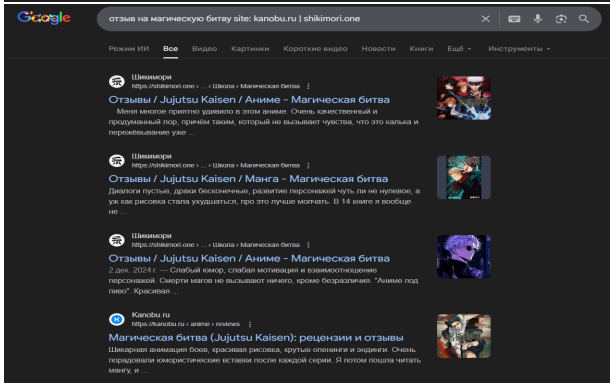
- *Размер 'сырых' данных (HTML)*: 1630.89 KB
- *Размер выделенного текста*: 548.01 KB
- *Средний размер документа (HTML)*: 12945.96 bytes
- *Средний объем текста текста*: 4350.07 bytes

Поисковые запросы и анализ выдачи

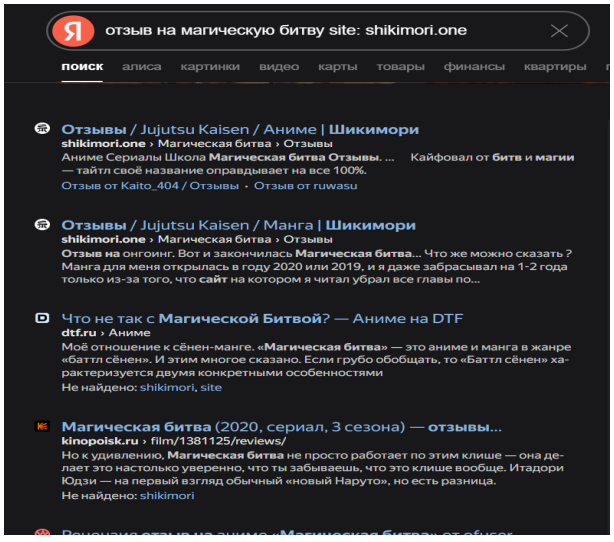
Для анализа использовался Google и Яндекс с оператором *site:*.
Запросы в Google в 3ех видах - запрос к сайду shikimori.one, запрос к сайту kanobu.ru и к ним обоим:

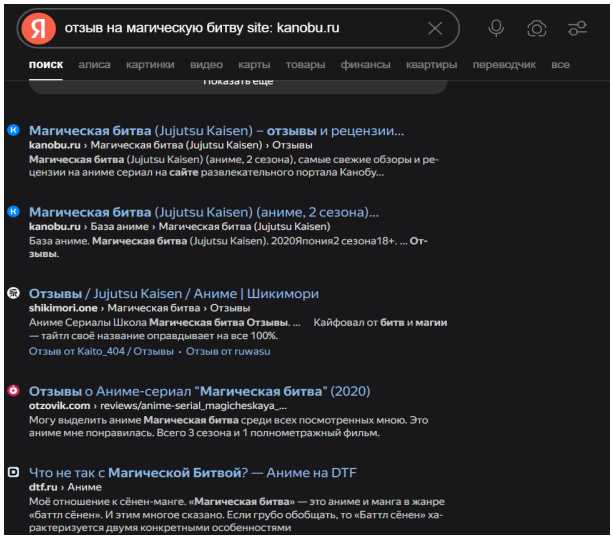


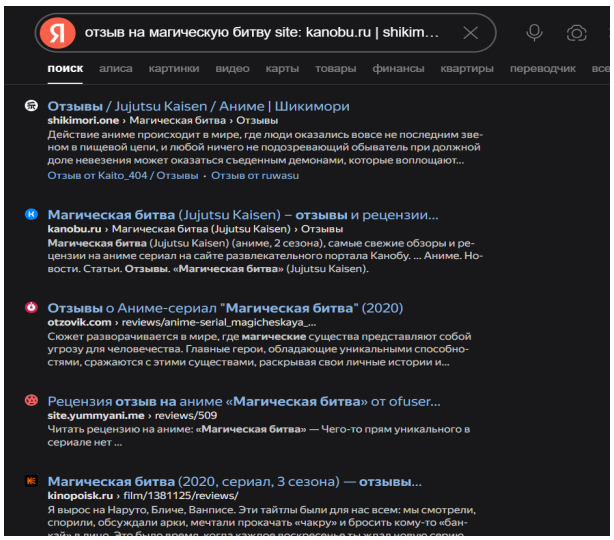




Запросы в Yandex в 3ех видах - запрос к сайду shikimori.one, запрос к сайту kanobu.ru и к ним обоим:







Вывод

В ходе работы был сформирован корпус из 547 документов. Соотношение объема чистого текста к сырому HTML составляет примерно 30%, что говорит об эффективной очистке данных от тегов и служебной информации. Собранный материал готов для использования в задачах индексации.

Литература

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

[2] Shikimori.one: Шикимори - энциклопедия аниме и манги
`https://shikimori.one`

[3] Kanobu.ru: Канобу — медиа о поп-культуре, кино, сериалах и аниме
`https://kanobu.ru`.

Лабораторная работа №2 «Поисковый робот»

Необходимо написать парсер на любом языке программирования:

- Написать поисковый робот — компоненты обкачки документов, используя любой язык программирования.
- Единственным аргументом поисковому роботу подаётся файл конфигурации (формата YAML или JSON), в котором содержатся параметры работы программы.
- База данных должна быть запущена в docker-контейнере, можно использовать docker-compose.
- В качестве хранилища результатов использовать MongoDB или PostgreSQL.
- Робот должен применять нормализацию URL-адресов.
- Робот должен сохранять в БД сырой HTML документа.
- Необходимо сохранять метаинформацию о каждом документе (дата скачивания, источник URL и т.д.).
- При остановке работы робот должен сохранять контрольную точку так, чтобы при повторном запуске он мог продолжить работу с того документа, с которого он остановился.
- Периодически он должен уметь переобкачивать документы, которые уже есть в базе, но только в том случае, если они изменились.

Задание

Разработать поисковый робот (crawler), который:

- Обходит целевые веб-ресурсы (Shikimori, Kanobu).
- Сохраняет данные в базу данных MongoDB.
- Поддерживает конфигурацию через YAML-файл.
- Умеет останавливаться и возобновлять работу (checkpoint).
- Избегает повторного скачивания актуальных документов.

Выбор языка программирования и технологий

Для реализации поискового робота был выбран язык программирования **Python**

Архитектура и технологии

Язык программирования: Python 3.10
База данных: MongoDB (использована библиотека `pymongo`).
Библиотеки: `requests` (HTTP-запросы), `BeautifulSoup4` (парсинг HTML), `PyYAML` (конфигурация).

Реализация

1. Конфигурация (config.yaml)

Параметры робота вынесены в отдельный файл. Это позволяет менять задержку (delay) и источники без правки кода.

```
db:
host: "localhost"
port: 27017
name: "ir_lab"
collection: "documents"
logic:
delay: 2.0
reindex_days: 7
```

2. Логика обкачки и парсинга

Робот использует адаптивные селекторы. Для сайта Kanobu реализован механизм поиска текста по нескольким вероятным классам (`c-entry__body`, `itemprop="articleBody"`), что делает парсер устойчивым к изменениям верстки.

3. Сохранение состояния (Checkpoint)

Для реализации остановки и возобновления робот сохраняет номер последней обработанной страницы в файл `crawler_state.json`.

```
def save_state(state):
with open('crawler_state.json', 'w') as f:
json.dump(state, f)
```

4. Работа с базой данных

Документы сохраняются в MongoDB. Перед записью проверяется поле `timestamp`. Если документ уже есть в базе и с момента последнего скачивания прошло менее 7 дней (настройка `reindex_days`), повторное скачивание не производится.

Результаты работы

Робот успешно обошел целевые сайты. В базу данных было сохранено 547 документов за примерно 25-30 минут работы программы.

- **Стабильность:** Робот корректно обрабатывает ошибки 404 и временные разрывы соединения.
- **Производительность:** Среднее время обработки одной страницы (с учетом задержки вежливости) составило ≈ 2.5 сек.

Пример структуры сохраненного документа (JSON):

```
{
 "_id": ObjectId("..."),
 "url": "https://shikimori.one/animes/...",
 "source": "shikimori",
 "clean_text": "Потрясающий сюжет и...",
 "timestamp": 1703885000
}
```

Реализация механизмов отказоустойчивости

В процессе написания поискового робота особое внимание было уделено стабильности процесса обкачки при работе с внешними ресурсами. Реализованы следующие механизмы:

- **Обработка HTTP-ошибок:** Робот анализирует коды ответа сервера. При получении ошибок доступа (например, `403 Forbidden`) или сетевых таймаутов, программа не завершается аварийно, а логирует событие и переходит к следующему документу.
- **Адаптивный сбор данных:** При возникновении предупреждений об отсутствии контента (`WARN`), робот использует резервные эвристики (поиск по альтернативным селекторам или сбор всех параграфов текста), что позволяет минимизировать потери данных при изменении верстки сайта.
- **Безопасная остановка:** Перехвачен сигнал `KeyboardInterrupt` (`Ctrl+C`). При нажатии данной комбинации робот немедленно прекращает сетевую активность, финализирует текущую итерацию и сохраняет текущий индекс страницы каталога в файл состояния.

Управление состоянием и фильтрация дубликатов

Для обеспечения эффективности и возможности работы на больших объемах данных реализованы:

1. **Механизм контрольных точек (Checkpoint):** Текущий прогресс обкачки (номера страниц каталога для каждого источника) сохраняется в JSON-файл. При повторном запуске робот считывает состояние и продолжает работу именно с того места, где она была прервана.
2. **Исключение дубликатов:**
    - На этапе сбора ссылок используется структура `set()` для удаления повторяющихся URL на одной странице.
    - На этапе сохранения в MongoDB реализована логика `upsert` на основе уникального индекса по полю `url`. Если документ с таким адресом уже существует, робот проверяет дату его обновления. Если данные актуальны, обкачка «сырого» контента пропускается, что значительно экономит трафик и ресурсы БД.

Результаты тестирования механизмов управления

В ходе выполнения работы были проведены тесты на прерывание процесса:

- **Тест остановки:** Принудительная остановка через `Ctrl+C` на 5-й странице каталога. Лог подтвердил сохранение: *«Робот остановлен пользователем. Прогресс сохранен»*.
- **Тест восстановления:** Повторный запуск инициировал запрос сразу к 5-й странице, пропустив первые четыре.
- **Тест дубликатов:** Повторная обкачка уже существующих 547 документов заняла значительно меньше времени за счет срабатывания логики `[SKIP] Актуально`.

Вывод

Разработанный краулер удовлетворяет всем требованиям задания. Использование NoSQL базы данных MongoDB позволило эффективно хранить неструктурированные данные (HTML и текст) и метаинформацию.

Литература

[1] Маннинг, Рагхаван, Шютце *Введение в информационный поиск* — Издательский дом «Вильямс», 2011. — 528 с.

[2] MongoDB Go Driver Documentation
`https://www.mongodb.com/docs/drivers/go/current/`

[3] Shikimori.one: Шикимори - энциклопедия аниме и манги
`https://shikimori.one`

[4] Kanobu.ru: Канобу — медиа о поп-культуре, кино, сериалах и аниме
`https://kanobu.ru`.

[5] Web Crawler Architecture and Design
`https://en.wikipedia.org/wiki/Web_crawler`

[6] The Robots Exclusion Protocol
`https://www.robotstxt.org/`

Лабораторная работа №3 «Токенизация, стемминг, закон Ципфа, индексация и булев поиск»

Реализовать токенизацию собранного корпуса, построить инвертированный индекс, проверить закон Ципфа и реализовать булев поиск.

Часть 1. Токенизация

- Реализовать процесс разбиения текстов документов на токены.
- Выработать правила токенизации, описать их достоинства и недостатки.
- Привести примеры неудачно выделенных токенов и способы исправления.
- Указать статистику: количество токенов, среднюю длину, скорость обработки.

Часть 2. Закон Ципфа

- Построить график распределения терминов по частотности в логарифмической шкале.
- Наложить теоретический закон Ципфа на реальные данные.
- Объяснить причины расхождения.
- (Опционально) Подобрать константы для закона Мандельброта.

Часть 3. Лемматизация

- Добавить лемматизацию/стемминг в поисковую систему.
- Оценить качество поиска до и после внедрения.
- Проанализировать запросы, где качество ухудшилось, объяснить причины.

Часть 4. Булев поиск

- Реализовать инвертированный индекс.
- Реализовать булев поиск с операторами AND, OR, NOT.
- Провести тестирование на реальных запросах.

1. Токенизация и Стемминг

Процесс преобразования текста в поисковые термины состоял из этапов:

1. **Очистка:** Удаление знаков препинания и спецсимволов (использовано регулярное выражение `[a-яёa-z]+`).
2. **Нормализация:** Приведение к нижнему регистру.
3. **Фильтрация:** Удаление стоп-слов (предлоги, союзы) с использованием библиотеки `NLTK`.
4. **Стемминг:** Приведение слов к основе (стемме) с помощью алгоритма Snowball (RussianStemmer

2. Статистика обработки

Анализ корпуса из 547 документов показал следующие результаты:

Таблица 1: Результаты токенизации

Метрика	**Значение**
Общее количество токенов	103,925
Средняя длина токена	6.99 символов
Время выполнения индексации	2.2542 сек
Скорость обработки	859.80 KB/sec

Скорость обработки ≈ 860 КБ/сек является приемлемой для скриптового языка Python, учитывая затраты на стемминг каждого слова.

3. Закон Ципфа

Теория

Закон Ципфа — эмпирическое наблюдение, согласно которому частота слова в тексте обратно пропорциональна его рангу в частотном словаре. Математически выражается как:

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

где:

- $f(r)$ — частота термина с рангом r
- r — ранг термина (1 для самого частого, 2 для второго и т.д.)
- C — константа, равная частоте самого частого слова

Закон Мандельброта — обобщение закона Ципфа с дополнительными параметрами:

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

где:

- B — сдвиг, учитывающий отклонение высокочастотных слов

- α — показатель степени (для классического Ципфа $\alpha = 1$)

Был построен график распределения частот терминов в логарифмической шкале (Log-Log).

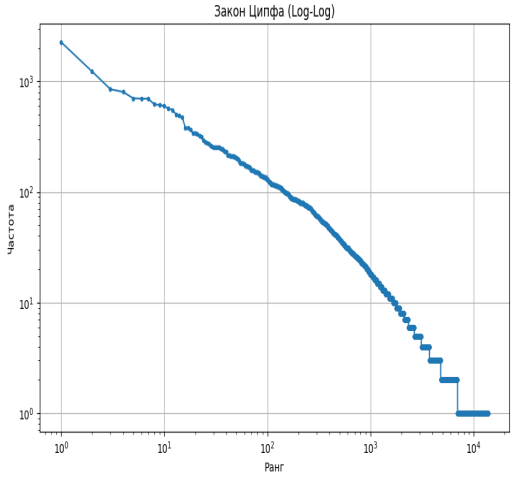


Рис. 1: График распределения частот терминов (Закон Ципфа)

Анализ графика: Полученная кривая близка к линейной, что подтверждает выполнение закона Ципфа для собранного корпуса. Наиболее частотные слова имеют ранг 1, а частота убывает обратно пропорционально рангу.

4. Реализация поиска

Реализован **инвертированный индекс** — структура данных, сопоставляющая каждому термину список ID документов. На его основе построен булев поиск с поддержкой операторов `AND`, `OR`, `NOT`.

Пример работы поиска

Запрос: `сюжет AND персонажи`
Логика: Поиск документов, где встречаются основы слов «сюжет» и «персонаж» одновременно.

Результаты:

- Найдено документов: 182
- Примеры выдачи:

1. `https://shikimori.one/animes/56752-shiguang-dailiren-yingdu-pian/reviews#580253`
2. `https://shikimori.one/animes/47194-summertime-render/reviews#611042`
3. `https://shikimori.one/animes/z37430-tensei-shitara-slime-datta-ken/reviews#614750`

Вывод

Реализованная поисковая система неплохо выполняет индексацию и поиск по корпусу из 547 документов. Использование стемминга позволяет находить документы, содержащие различные словоформы (например, «персонажи», «персонажей»), что повышает полноту поиска (Recall). Время отклика поиска по инвертированному индексу составляет доли секунды.

Литература

[1] Маннинг, Рагхаван, Шютце *Введение в информационный поиск* — Издательский дом «Вильямс», 2011. — 528 с.

[2] Zipf, G. K. (1949). *Human Behavior and the Principle of Least Effort*. Addison-Wesley.

[3] Mandelbrot, B. (1953). *An Informational Theory of the Statistical Structure of Language*. Communication Theory.

[4] Porter, M. F. (1980). *An algorithm for suffix stripping*. Program, 14(3), 130–137.

[5] Shikimori.one: Шикимори - энциклопедия аниме и манги `https://shikimori.one`

[6] Kanobu.ru: Канобу — медиа о поп-культуре, кино, сериалах и аниме `https://kanobu.ru`.