МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский Авиационный Институт»

(Национальный Исследовательский Университет)

Институт: №8 «Информационные технологии и прикладная математика»
Кафедра: 806 «Вычислительная математика и программирование»

Отчет по лабораторной работе №2

по предмету «Информационный поиск»

«Поисковый робот»

Группа: М8О-412Б-22

Студент(ка): Кайдалова А. А.

Оценка:

Дата сдачи:

Москва, 2025

Задание

Необходимо написать парсер на любом языке программирования.

- Написать поисковый робот — компоненты обкачки документов, используя любой язык программирования;
- Единственным аргументом поисковому роботу подаётся путь до yaml-конфига, содержащий:
  - Данные для базы данные в секции db;
  - Данные для робота в секции logic: задержка между обкачкой страницы;
  - Любые другие данные, необходимые для реализовации логики поискового робота.
- Сохранять в базе данных (например, MongoDB) документы со следующими полями:
  - url, нормализованный;
  - «сырой» html-текст документа;
  - название источника;
  - Дата обкачки документа в формате Unix time stamp.
- Поисковый робот можно остановить в любой момент и при повторном запуске робот должен начать 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.

Описание метода решения задачи

Я реализовала поискового робота на языке Python.

Хранение данных: PostgreSQL. Таблица documents содержит все требуемые поля, также метаданные: заголовок, автор, дата публикации, и служебные поля last_fetch_attempt, etag_header для управления обновлениями. DDL:

Начальный обход начинается с заданных стартовых URL сайта 7ya.ru, также используются sitemapы сайтов mama.ru и letidor.ru.

Обход документов реализован на основе очереди, в которую изначально добавляются стартовые URL и ссылки полученные из sitemapов. При обработке каждой ссылки из очереди программа:

1. Загружает HTML-документ с заданной задержкой (1 сек);
2. Извлекает из него внутренние ссылки;
3. Добавляет новые подходящие ссылки в конец очереди;

4. Фильтрует URL.

Сохранение состояния: при каждом цикле обхода состояние (очередь, набор уже добавленных URL, счётчик сохранённых документов) сериализуется в файл crawler_state.pkl.

Фильтрация URL: реализована функция is_valid_article_link(), которая допускает только статьи например, /article/ на 7ya.ru, /articles/ на mama.ru, исключая категории, форумы, профили.

Извлечение текста: с помощью BeautifulSoup удаляются навигационные блоки, извлекается только полезный текст статьи. Для некоторых сайтов используется структурированный JSON-LD.

Обновление документов: после первичного сбора документов запускается фаза обновления. Документы, не обновлявшиеся последние 7 дней, проверяются на изменения с помощью условных HTTP-запросов и хэширования по полям title, author, date, text.

Вежливость: соблюдение robots.txt с кэшированием на 24 часа, задержка 0.5 сек между запросами, корректный User-Agent.

Журнал выполнения задания

Реализация фильтрации URL. На mama.ru в sitemapах много ссылок на категории и форумы. Добавила проверку: только пути /articles/... без /category/. На letidor.ru разрешила только .html/.htm, на 7ya.ru — только /article/....

Нормализация URL. Изначально один и тот же документ мог сохраниться несколько раз из-за различий в http/https, www и конечных /. Реализовала функцию normalize_document_url(), приводящую все URL к единому виду.

Извлечение текста и метаданных. Написала отдельные блоки для каждого сайта, отличительные черты:

- 7ya.ru — текст в .articlebody;
- mama.ru — отдельно лид и основной блок;
- letidor.ru — дата в формате 12 декабря 2024 преобразована через словарь месяцев.

Обработка ошибок 404. Некоторые страницы возвращали статус 200, но содержали текст «страница не найдена». Реализовала функцию is_bad_content(), проверяющую <title>, <h1> и общий текст на наличие «страница не найдена» или «404».

Обеспечение отказоустойчивости. После каждого обработанного URL (для того чтобы можно было возобновить работу именно с ссылки. На которой закончили) состояние сохраняется в crawler_state.pkl. При Ctrl+C программа корректно завершает работу, при повторном запуске возобновляет обход.

Реализация фазы обновления. После сбора 54 543 документов запускается обновление:

1. выбор документов, не обновлявшихся 7+ дней;
2. условный запрос с If-Modified-Sinc, If-None-Match;
3. при изменении - пересчёт хэша по title, author, date, text и обновление записи.

Результаты

За время выполнения работы собрано 54 543 уникальных статей из трёх источников:

	source	article_count
1	7ya.ru	8 417
2	letidor.ru	40 685
3	mama.ru	5 441

Все документы прошли фильтрацию по содержимому и тематике. Программа успешно возобновляла работу после пяти принудительных остановок. Поограмма корректно искала обновившееся документы (для теста брала разницу в 10 минут).

Выводы

Работа выполнена в полном соответствии с требованиями. Реализован надёжный, вежливый и отказоустойчивый поисковый робот, способный собирать и обновлять корпус документов.

Недостатки и пути улучшения:

- Производительность: задержка 1 секунда, хоть и не большая, но сильно замедляет сбор. Улучшение: можно динамически регулировать задержку в зависимости от ответа сервера или использовать асинхронные запросы.
- Зависимость от структуры HTML: при изменении верстки сайта парсер может сломаться. Улучшение: добавить fallback-механизмы и логирование ошибок извлечения текста.

Несмотря на указанные недостатки, программа стабильно работает, корректно сохраняет состояние и обеспечивает хорошее качество собранного корпуса.

Приложения

Исходный код: crawler.py, config.yaml, .env (см. ниже).
Требования к запуску:

- PostgreSQL с созданной таблицей documents:

```
CREATE TABLE public.documents (
	id bigserial NOT NULL,
	url text NOT NULL,
	normalized_url text NOT NULL,
	"source" text NOT NULL,
	created_at timestamptz DEFAULT now() NULL,
	html text NULL,
	clean_text text NULL,
	title text NULL,
	author text NULL,
	publish_date text NULL,
	last_modified_header text NULL,
	etag_header text NULL,
	fetch_timestamp int8 NULL,
	last_fetch_attempt int8 NULL,
	CONSTRAINT documents_normalized_url_key UNIQUE (normalized_url),
	CONSTRAINT documents_pkey PRIMARY KEY (id)
);
CREATE INDEX idx_last_fetch ON public.documents USING btree (last_fetch_attempt);
CREATE UNIQUE INDEX idx_normalized_url ON public.documents USING btree
(normalized_url);
```

- Переменная окружения DB_PASSWORD