Санкт-Петербургский государственный университет
Кафедра информатики

Реализация автоматизированной системы парсинга данных на основе больших языковых моделей

Мурадян Денис Степанович, группа 23.Б16-мм

Научный руководитель: старший преподаватель кафедры информатики, Бушмелев Федор Витальевич
Консультант: аналитик данных ПАО "Сбербанк России", Попов Артем Петрович

Санкт-Петербург
2025

Введение

- Веб-парсинг: извлечение структурированных данных из веб-страниц
- Проблема: сайты бывают статическими и динамическими, для каждого нужен свой парсер и его постоянная поддержка
- Идея: автоматизация генерации parser-скриптов с помощью больших языковых моделей (LLM)

Существующие решения и их недостатки

- **Статический парсинг (requests + BeautifulSoup)**
  - + Лёгкость и скорость запуска
  - – Не работает с динамическим JS-контентом, требует ручных селекторов
- **Динамический парсинг (Selenium)**
  - + Обходит любой JS
  - – Тоже требует писать и поддерживать скрипты-парсеры под каждую структуру
- **Общий недостаток**: отсутствует единый самонастраивающийся механизм — при изменении верстки ручной код ломается

LLM-подход и семантический кэш

- **Structuring**: извлечение ответа на запрос пользователя напрямую из очищенного текста страницы
- **Codegen (двойное применение LLM)**:
  - *HintGen*: генерирует подсказки, селекторы и few-shot примеры
  - *CodeGen*: на основе подсказок формирует Python-скрипт-парсер и выполняет его
- **Проблемы**: дороговизна запросов в LLM и время инференса ответа
- **Решение**: семантический кэш (SQLite + ChromaDB), для одинаковых ссылок и схожих embedding запросов - запросы в LLM не отправляются, а используются ранее кешированные парсеры

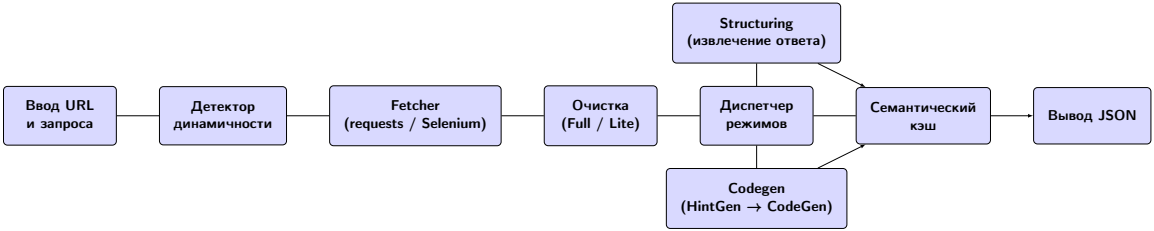
Постановка задачи

Цель проекта: оптимизировать сбор данных из открытых веб-источников за счёт автоматизации написания парсеров веб-страниц с помощью больших языковых моделей (LLM)

Задачи:

- Определять тип страницы (статическая или динамическая)
- Реализовать две стратегии очистки HTML (Full / Lite)
- Внедрить режимы LLM-парсинга: Structuring и Codegen
- Построить семантический кэш (SQLite + ChromaDB)
- Разработать интерфейсы: REST API, веб-фронтенд, Gradio
- Провести экспериментальное исследование

Архитектура решения



Результаты экспериментального исследования

Сайт / Запрос	Codegen (cold)	Codegen (warm)	Structuring
Gismeteo			
«Текущая температура в СПБ»	23.28	5.61	8.62
СПбГУ			
«Выведи последние новости СПбГУ»	40.91	6.09	7.34
Яндекс.Финанс			
«Курс доллара к рублю»	17.59	4.61	12.23

Примеры JSON-ответов

Gismeteo: «Текущая температура в Санкт-Петербурге»

```
{
  "query_data": Текущая" температура в СанктПетербурге—: +11°C\
   Температурап по ощущению: +10°C\n"
}
```

Яндекс.Финанс: «Курс доллара США к рублю»

```
{
  "query_data": Курс" доллара США к рублю: 78,62 руб\Изменениеп
   курса: −2,88 руб −(3,53%)\Источникп: ЦБ РФ\n"
}
```

Итоги работы

- Разработана модульная система для парсинга статических и динамических веб-страниц
- Реализованы два режима LLM-генерации парсеров:
  - прямое извлечение ответа из текста (Structuring)
  - генерация Python-скрипта с помощью подсказок (Codegen)
- Внедрён семантический кэш для повторного использования сгенерированных скриптов
- Предоставлены разные интерфейсы: библиотека, REST API, веб-приложение и Gradio
- Экспериментально подтверждена эффективность подхода на реальных сайтах

Воспользоваться проектом можно, отсканировав QR-код для перехода на Hugging Face Space