

Научная статья
DOI 10.66424/2071-8217-2026-3-7
УДК 004.89:004.738.5

ИССЛЕДОВАНИЕ МЕТОДОВ АВТОМАТИЗИРОВАННОГО ИЗВЛЕЧЕНИЯ ДАННЫХ ИЗ ВЕБ-РЕСУРСОВ С ДИНАМИЧЕСКИ ИЗМЕНЯЕМЫМ КОНТЕНТОМ С ПОМОЩЬЮ ИИ-АГЕНТОВ

Р. А. Абитов*, А. Д. Дахнович, Д. А. Москвин

Санкт-Петербургский политехнический университет Петра Великого, Санкт-Петербург, Россия

✉ *abitov.ra@edu.spbstu.ru

ДЛЯ ЦИТИРОВАНИЯ

Абитов Р. А., Дахнович А. Д.,
Москвин Д. А. Исследование
методов автоматизированного
извлечения данных из веб-ресурсов
с динамически изменяемым
контентом с помощью ИИ-агентов // Проблемы информационной безопасности. Компьютерные системы. 2026. № 3. С. 96–109.
DOI: 10.66424/2071-8217-2026-3-7

ПОСТУПИЛА 03.07.2026

ПРИНЯТА 28.07.2026

ОПУБЛИКОВАНА 13.10.2026

© Абитов Р. А., Дахнович А. Д.,
Москвин Д. А.

Издатель: Санкт-Петербургский
политехнический университет
Петра Великого

АННОТАЦИЯ

Рассматривается задача автоматизированного извлечения данных из веб-ресурсов с динамически изменяемым контентом с применением ИИ-агентов. Показано, что практика сбора данных остается фрагментированной: неоднородность DOM-структур, активные действия пользователя, средства антибот-защиты и высокая доля неструктурированных веб-данных повышают стоимость сопровождения классических парсеров. Предложен двухэтапный метод, разделяющий дорогостоящий этап анализа ресурса языковой моделью и экономичный этап массового детерминированного сбора. Метод реализован в программном макете на базе мультиагентной оркестрации OpenClaw, изолированного браузерного контура Chromium, протокола Model Context Protocol и контура авторизации. Выполнены сравнительный анализ средств сбора и больших языковых моделей, классификация типов лент и активных действий, а также экспериментальная оценка качества извлечения. На эталонном наборе данных основной показатель качества составил $F1 = 0,89$. Ограничения связаны с зависимостью от устойчивости селекторов, изменчивостью антибот-механизмов и вычислительной стоимостью этапа конфигурирования.

КЛЮЧЕВЫЕ СЛОВА

ИИ-агенты, извлечение данных, веб-скрапинг, динамический контент, большие языковые модели, асинхронная обработка, отказоустойчивость, качество данных, Model Context Protocol, Chromium, OpenClaw

Original article
DOI 10.66424/2071-8217-2026-3-7

RESEARCH ON METHODS FOR AUTOMATED DATA EXTRACTION FROM WEB RESOURCES WITH DYNAMICALLY CHANGING CONTENT USING AI AGENTS

R. A. Abitov*, A. D. Dakhnovich, D. A. Moskvina

Peter the Great St. Petersburg Polytechnic University, St. Petersburg, Russia

✉ *abitov.ra@edu.spbstu.ru

FOR CITATION

Abitov R. A., Dakhnovich A. D.,
Moskvin D. A. Research on methods for
automated data extraction from web
resources with dynamically changing
content using AI agents. *Problems of
information security. Computer systems*.
2026. No. 3, pp. 96–109.
DOI: 10.66424/2071-8217-2026-3-7
(In Russian)

RECEIVED 03.07.2026

ACCEPTED 28.07.2026

PUBLICATION 13.10.2026

ABSTRACT

This paper addresses automated extraction of data from web resources with dynamically changing content using AI agents. Data collection practice remains fragmented because of heterogeneous DOM structures, interactive user actions, anti-bot controls, and the predominance of unstructured web data, which increases maintenance costs of classical parsers. We propose a two-stage method that separates an expensive large language model analysis stage from an economical deterministic mass-collection stage. The method is implemented in a software prototype based on OpenClaw multi-agent orchestration, an isolated Chromium browser contour, the Model Context Protocol, and an authentication contour. We performed comparative analysis of scraping tools and large language models, classified feed types and active actions, and experimentally assessed extraction quality. On a reference dataset the primary quality metric reached $F1 = 0.89$. Limitations include selector fragility, evolving anti-bot mechanisms, and the computational cost of the configuration stage.

KEYWORDS

AI agents, data extraction, web scraping, dynamic content, large language models, asynchronous processing, fault tolerance, data quality, Model Context Protocol, Chromium, OpenClaw

1. ВВЕДЕНИЕ

Рост объемов цифровых данных и расширение спектра прикладных задач, связанных с аналитикой, обучением интеллектуальных систем, финансовым и экономическим прогнозированием, а также мониторингом рисков информационной безопасности, усиливают потребность в надежных и масштабируемых процедурах автоматизированного извлечения данных из открытых веб-источников [1–3]. По оценкам отраслевых исследований, значительная доля глобальных данных формируется веб-средой, социальными сетями и медиа, при этом преобладают неструктурированные представления (веб-страницы, мультимедиа, пользовательский контент) [1, 2]. В таких условиях классические парсеры, жестко связанные с конкретной разметкой HTML/CSS, быстро теряют работоспособность при изменении верстки, логики подгрузки контента или сценариев взаимодействия.

На рис. 1 и 2 приведены количественные индикаторы, иллюстрирующие актуальность задачи: рост мирового рынка технологий Big Data и структуру глобальных данных по источникам и степени струк-

турированности. Эти показатели не определяют алгоритм извлечения напрямую, но задают внешний контекст, в котором стоимость ручной адаптации средств сбора и потери полноты данных становятся критичными для прикладных контуров.

В задачах информационной безопасности автоматизированный сбор открытых данных используется также для мониторинга признаков мошеннических профилей, оценки цифрового следа сотрудников в контексте рисков таргетированного фишинга и социальной инженерии, контроля утечек секретов в публичных репозиториях, а также отслеживания упоминаний организации на специализированных площадках [3, 4]. Общим для этих сценариев является требование воспроизводимого извлечения структурированных полей из динамических интерфейсов при ограничениях на стоимость инференса, задержку и устойчивость к сбоям.

Основная проблема исследования состоит в выборе метода и конфигурации агентской системы извлечения по совокупности критериев: стоимость, быстродействие, возможность асинхронной обработки, отказоустойчивость и качество извлекаемых данных.

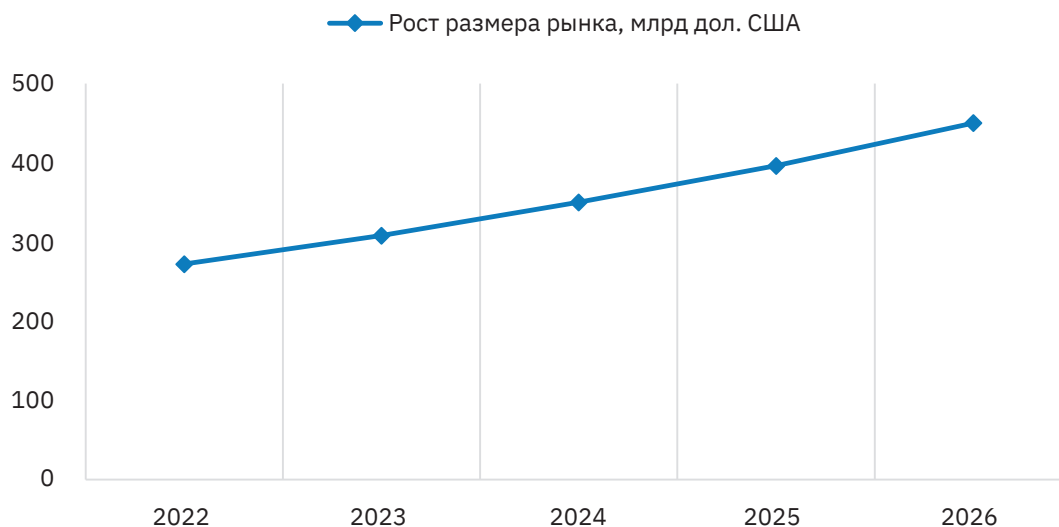


Рис. 1 | Рост размера мирового рынка технологии Big Data, млрд дол. США

Fig. 1 | Growth in the size of the global Big Data technology market, USD billion

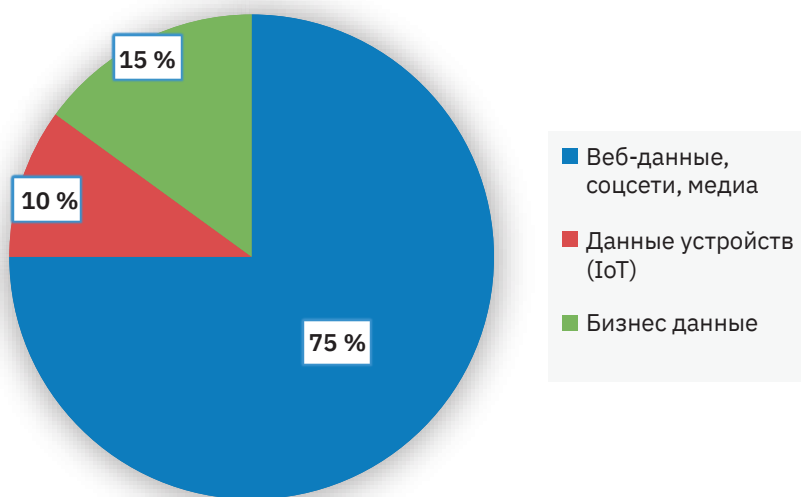


Рис. 2 | Распределение глобальных данных по источникам и уровень структурированности Big Data

Fig. 2 | Distribution of global data by source and the level of Big Data structuring

Цель работы – разработать подход для извлечения данных из веб-ресурсов с динамически изменяемым контентом на базе ИИ-агентов и экспериментально оценить его качество. Объект исследования – веб-ресурсы с динамически изменяемым контентом. Предмет исследования – методы автоматизированного извлечения данных из таких ресурсов с использованием ИИ-агентов.

Для достижения цели поставлены следующие задачи: проанализировать средства сбора данных из веб-ресурсов и большие языковые модели (БЯМ) применительно к извлечению данных и генерации программ сбора; разработать подход к сбору данных на основе ИИ-агентов; реализовать программный макет предложенного подхода; провести экспериментальные исследования макета и оценить качество сбора.

В связанных работах автономные ИИ-агенты рассматриваются как системы, способные планировать многошаговые действия в браузере и извлекать информацию в реалистичных средах [5, 6]. Отдельное направление образуют подходы к генерации правил и скриптов сбора на основе языковых моделей и структуры HTML [7], а также агентные схемы агрегации веб-информации [8].

Протокол Model Context Protocol (MCP) задает стандартизированный способ подключения моделей к инструментам выполнения, включая управление браузером и доступ к HTML [9]. Вместе с тем в литературе недостаточно проработан вопрос разделения «однократного интеллектуального конфигурирования» и «многократного детерминированного исполнения», хотя именно такое разделение критично для снижения стоимости инференса при массовом сборе.

Научная новизна заключается в формализации двухэтапного метода автоматизированного извлечения данных для динамических веб-ресурсов, в котором ИИ-агент используется для анализа ресурса, унификации DOM-представления, синтеза схемы полей и генерации исполнимых средств сбора, а регулярный сбор

выполняется без обращения к БЯМ на каждую единицу данных. Практическая значимость состоит в снижении трудозатрат на сопровождение парсеров при сохранении приемлемого качества извлечения и управляемой стоимости эксплуатации.

2. МЕТОДЫ

Методология исследования включала: сравнительный анализ средств сбора и БЯМ; эмпирическую классификацию веб-ресурсов по типам лент и активным действиям; проектирование двухэтапного метода и программной архитектуры; экспериментальную оценку качества извлечения на эталонном наборе «HTML/JSON»; оценку контура авторизации на наборе ресурсов с различными схемами входа.

В рамках задачи классификации динамических веб-ресурсов на выборке тематических ресурсов с карточными анкетами выделены наиболее распространенные типы лент и активных действий, необходимых для полного извлечения полей (табл. 1). Классификация использовалась как методический каркас для построения шаблонов сценариев сбора и для интерпретации отказов автоматизации.

Таблица 1 | Наиболее популярные типы лент веб-ресурсов с динамическим контентом

Table 1 | The most popular types of feeds for web resources with dynamic content

Тип ленты	Число ресурсов	Характеристика типа	Особенности сбора
Классическая постраничная лента	7	Анкеты разделены на страницы каталога. Переход между частями списка выполняется через номера страниц или кнопки перехода	Сбор выполняется через последовательный обход страниц пагинации
Лента с ручной подгрузкой через кнопку	4	Новые анкеты появляются после нажатия кнопки загрузки дополнительных карточек	Сбор требует повторного поиска и нажатия кнопки подгрузки до окончания списка
Смешанная лента	4	Ресурс использует несколько способов навигации по анкетам: постраничную пагинацию и ручную подгрузку через кнопку	Нужно учитывать оба сценария: переход по страницам и дозагрузку анкет через кнопку
Одностраничная лента без пагинации	2	Все доступные анкеты размещены на одной странице, без перехода по страницам и без кнопки подгрузки	Достаточно обработать одну страницу и извлечь все карточки или ссылки на анкеты
Нестандартная структура ленты	2	Лента устроена не как обычный список карточек или имеет специфическую механику раскрытия, переходов или отображения анкет	Для каждого ресурса требуется отдельная логика сбора

В результате анализа типов лент чаще всего встречаются ленты классического формата, а именно ленты с query параметрами с выбором страниц.

Наиболее популярные активные действия на страницах, необходимые для сбора: открытие модальных окон – 9; раскрытие скрытых блоков – 5; переходы на вложенные страницы – 2; переключение табов – 1; прокрутка страницы – 1; капча – 1.

Выполнен сравнительный анализ средств сбора, где в качестве основных решений рассматривались ReaderLM-v2, nvidia/gliner-PII, browser-use, ScrapeGraphAI, Ui. Vision и Skyvern. Критерии сравнения включали возможность локального запуска БЯМ, поддержку активных действий, способность генерировать переносимый код и сценарий, устойчивость к CAPTCHA и 2FA, скорость и предсказуемость ошибок.

Обобщенные результаты представлены в табл. 2.

По итогам анализа в качестве базовой идеи выбран гибридный подход, близкий к парадигме ScrapeGraphAI: БЯМ исследует страницу и формирует сценарий/код, а детерминированное исполнение обеспечивает массовый сбор. При этом для практической эксплуатации потребовалась дополнительная оркестрация агентов, изолированный браузерный контур и отдельный контур авторизации.

Выполнен анализ больших языковых моделей, где БЯМ оценивались в двух постановках: извлечение данных из HTML (табл. 3 и 4) и генерация программы сбора (табл. 5).

Для постановки извлечения данных из HTML рационально использовать модель лишь для разметки классов CSS, xpath

Таблица 2 | Анализ средств автоматизированного сбора данных на веб-ресурсах

Table 2 | Analysis of automated data collection tools on web resources

Средство	Описание	Преимущества	Недостатки
ReaderLM-v2	Преобразование HTML в markdown/JSON; локальная модель для извлечения данных из страницы	Локальный запуск; быстрый GPU-инференс; снижает объем ошибок в HTML при генерации	На CPU не хватает ресурсов; качество падает на больших и сложных страницах
gliner-PII	NER-модель для извлечения персональных и заданных сущностей из HTML или markdown	Может работать локально; полезна для PII и типовых полей; не требует полного ИИ	Не покрывает всю бизнес-логику страницы; нужна разметка/датасет и предобработка
browser-use	БЯМ управляет браузером через инструменты: клики, ввод, скролл, чтение DOM	Подходит для активных действий; компактных моделей достаточно для команд управления	Медленнее статического кода; чувствительно к CAPTCHA, 2FA и нестабильному DOM
ScrapeGraphAI	ИИ-агент для извлечения данных или генерации Python-скрипта для извлечения данных	Умеет генерировать код, есть режимы прямого извлечения и решение для создания Python-скриптов	Ошибки валидации на малых моделях; генерация скрипта занимает сотни секунд; нет авторизации
Ui.Vision	Расширения для заполнения форм и выполнения действий в браузере, которое позволяет выполнить действия и получить уникальные HTML	Мгновенное выполнение; удобно подготавливать разметку; стабильно для авторизационных степперов	При смене разметки страницы или изменения ее контента форму нужно настраивать заново
Skyvern	Автоматизация сбора с помощью браузера через БЯМ и компьютерное зрение, поддержка MCP	Проходит сложную авторизацию; гибкие выходные статусы ошибок; высокая кастомизация	Низкая скорость сбора одной единицы данных, режим компьютерного зрения часто галлюцинирует и неправильно классифицирует объекты на странице

и локализации фрагментов, тогда как штатное извлечение выполняет программа. В результате чего сформирован вывод о том, что запуск моделей на CPU малого размера приемлем по качеству, но медленный, а запуск на GPU предпочтительен при жестких требованиях к задержке, а также позволяет использовать тяжелые модели.

Для генерации программы сбора экономически оправдано применение модели высокого качества: однократная

Таблица 3 | Анализ работы моделей запущенных на GPU
в задаче извлечения данных из HTML (запуск производился с помощью GPU)

Table 3 | Analysis of the performance of models run on a GPU
in the task of extracting data from HTML (the run was performed using a GPU)

Модель	Время на один HTML, с	Точность (Precision)	Полнота (Recall)	F1
qwen3.5-27b	78	0,89	0,89	0,89
qwen3.5-9b	81	0,82	0,82	0,82
jinaai/ReaderLM-v2	20	0,65	0,63	0,64

Таблица 4 | Анализ работы моделей запущенных на GPU
в задаче извлечения данных из HTML (запуск производился на CPU)

Table 4 | Analysis of the performance of models run on the GPU
in the task of extracting data from HTML (the models were run on the CPU)

Модель	Время на один HTML	Точность (Precision)	Полнота (Recall)	F1
qwen3.5-27b	> 24 ч (остановлена)	—	—	—
qwen3.5-9b	2122 с	0,8	0,76	0,78
jinaai/ReaderLM-v2	> 24 ч (остановлена)	—	—	—
gemma4: e2b	900 с	0,85	0,79	0,82
qwen3.5:4b	1411 с	0,85	0,82	0,83

Таблица 5 | Анализ больших языковых моделей
в решении задачи генерации программы для извлечения данных из HTML

Table 5 | Analysis of large language models
in solving the problem of generating a program to extract data from HTML

Модель	Время на один HTML с генерацией программы	Точность (Precision)	Полнота (Recall)	F1
qwen3.5-27b	335 с	0,91	0,67	0,76
qwen3.5-9b	288 с	0,88	0,68	0,76
deepseekV4-pro-1.6t	65 с	0,98	0,99	0,98
qwen3.6-397b	102 с	0,96	0,97	0,96

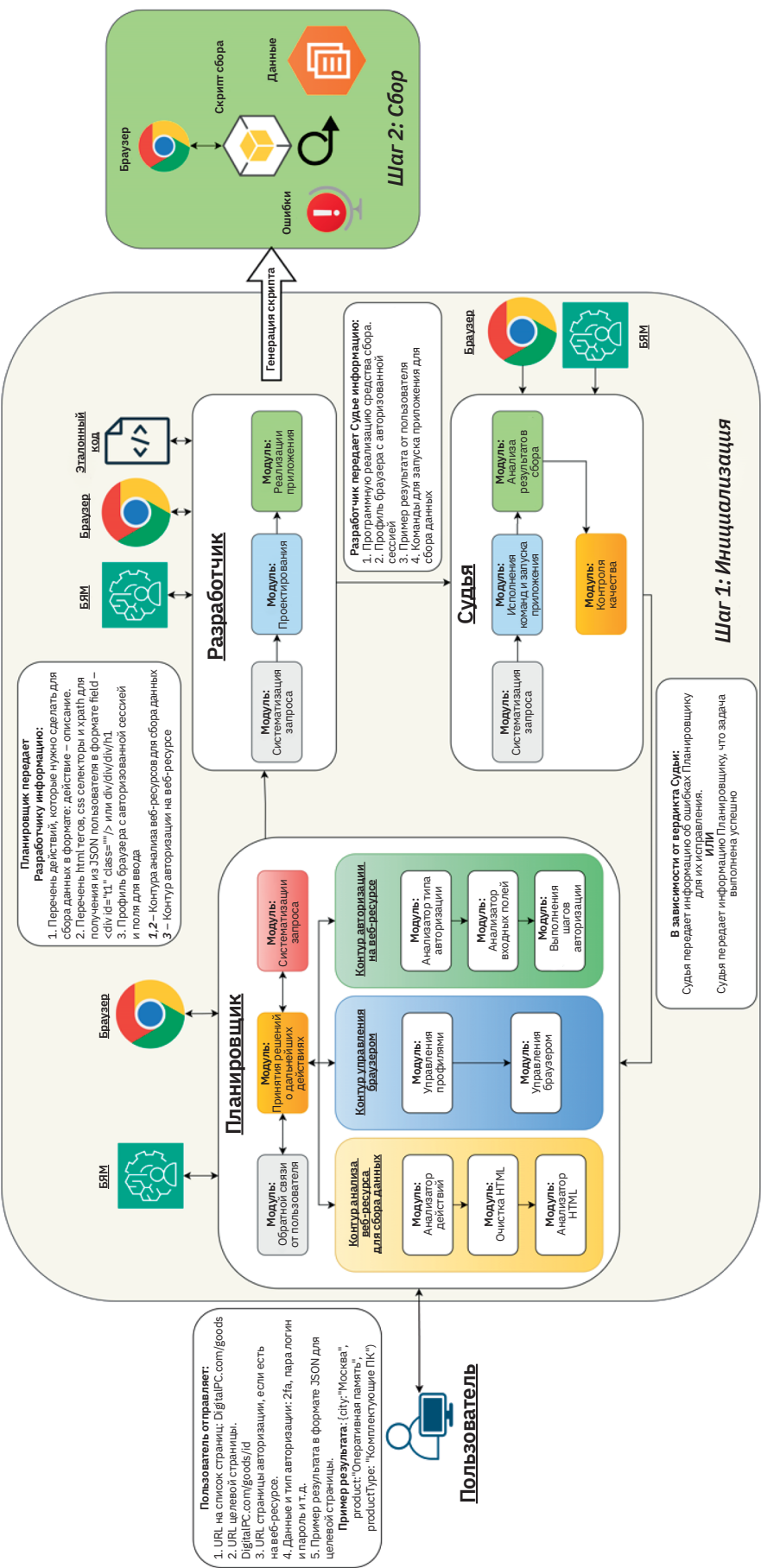


Рис. 3 | Двухэтапный подход к сбору данных: инициализация с помощью ИИ и массовый сбор

Fig. 3 | Two-stage approach to data collection: AI-based initialization and large-scale collection

успешная генерация предпочтительнее серии повторных запусков с исправлением ошибок [7, 10].

Рассмотрим предлагаемый метод, который разделяет процесс на два этапа (рис. 3). На этапе конфигурирования ИИ-агент анализирует целевой ресурс: выполняет навигацию, стабилизацию и унификацию DOM, обнаруживает необходимые поля, формализует схему результата и синтезирует набор статических средств сбора (селекторы, сценарии действий, программный шаблон).

На этапе эксплуатации регулярный сбор выполняется без обращения к БЯМ на каждую единицу извлекаемой информации, что снижает совокупную стоимость инференса и повышает пропускную способность. При изменении верстки иницируется повторное конфигурирование, а не непрерывный «агентный» проход по всему объему данных.

Формально качество извлечения оценивалось на уровне пар «атрибут – значение». Пусть E – множество эталонных пар,

R – множество пар, возвращенных системой. Тогда:

$$TP = |E \cap R|, FP = |R \setminus E|, FN = |E \setminus R|,$$
$$Precision = \frac{TP}{(TP + FP)}, Recall = \frac{TP}{(TP + FN)},$$
$$F1 = \frac{2Precision \cdot Recall}{Precision + Recall}.$$

Пример интерпретации классов ошибок для аппаратных характеристик для сбора товаров приведен в табл. 6: совпадение пары трактуется как TP , пропуск эталонной пары – FN , появление лишней/ошибочной пары – FP , а отсутствие нерелевантных полей – корректный TN -кейс на уровне схемы.

В табл. 7 представлены результаты экспериментального исследования метода, реализованного в программном макете.

Рассмотрим архитектуру программного макета, реализованного в виде мультиагентной системы под управлением OpenClaw (рис. 4). Пользователь задает

Таблица 6 | Пример сопоставления эталона и результата извлечения

Table 6 | Example of comparing the reference and the extraction result

Класс	Пара	Пояснение
TP	(CPU, 16gb)	Результат совпал с эталоном
FN	(GPU, RTX 4090)	Результат не совпал с эталоном
FP	(GPU, RTX 4080)	Лишняя/ошибочная пара
TN	Отсутствие полей RAM, SSD ...	Лишние поля не добавлены

Таблица 7 | Результаты экспериментального исследования разработанного метода и программного макета

Table 7 | Results of the experimental study of the developed method and software prototype

Решение	Время на генерацию скрипта и сбора одной анкеты	Точность (Precision)	Полнота (Recall)	F1
Программный макет	200 с	0,90	0,88	0,89
ScrapeGraphAI с помощью генерации приложения	335 с	0,91	0,67	0,76

URL списка, URL целевой страницы, параметры авторизации (при наличии) и пример результата в JSON. Оркестратор маршрутизирует задачу между агентами: планировщиком, контуром анализа веб-ресурса, контуром авторизации, разработчиком и судьей.

Браузерный контур построен на Chromium и движке Botosaurs (CDP/Playwright) с возможностью повторного воспроизведения цепочки действий без БЯМ. Инструментальный доступ агентов к HTML и действиям реализован через разработанный MCP-сервер. Состояния сессий,

журналы ошибок и цепочки авторизации хранятся в PostgreSQL, а извлеченные данные – в Elasticsearch. Приложения сбора упаковываются в Docker контейнеры и предоставляют REST API для управления.

Для контура анализа, разработки и судейства использовалась модель семейства DeepSeek (deepseekv4-pro) как модель высокого качества на этапе генерации спецификации и кода. Для контура авторизации выбрана отдельная конфигурация на базе qwen3.5-27b, показавшая наилучшую успешность прохождения схем входа в сравнительных испытаниях.

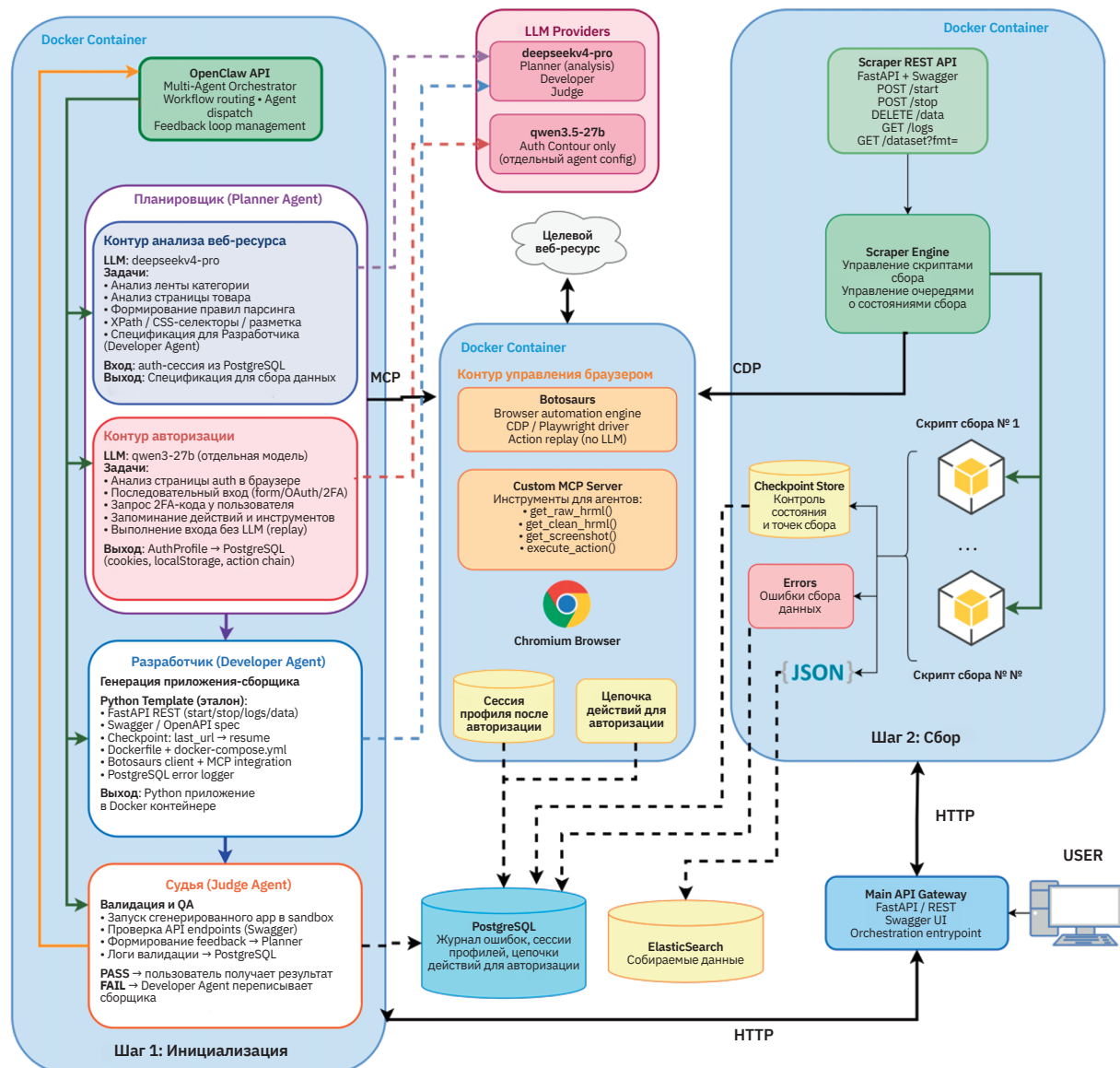


Рис. 4 | Архитектура программного макета сбора данных на базе ИИ-агентов

Fig. 4 | Architecture of the software prototype for data collection based on AI agents

Такое разделение моделей отражает различие задач: генерация переносимого сборщика требует более сильной модели «один раз», тогда как авторизация требует устойчивого пошагового управления формой, 2FA и запоминания цепочки действий для последующей авторизации.

Алгоритм работы макета включает: систематизацию пользовательского запроса; авторизацию и сохранение сессии; анализ ленты и целевой страницы с формированием селекторов и перечня действий; генерацию приложения-сборщика по эталонному шаблону; проверку сгенерированного кода агентом-судьей. Тем самым агентная система используется как «фабрика» исполнимых процедур, а не как постоянный исполнитель на всем потоке данных [6–9].

Контур анализа веб-ресурса получает авторизованную сессию (если требуется) и выполняет обход ленты и целевой страницы анкеты. На выходе формируется спецификация: перечень действий «действие – описание», соответствие полей JSON селекторам (CSS/XPath), рекомендации по стабилизации DOM. Контур авторизации анализирует страницу входа, классифицирует тип схемы (форма, OAuth, 2FA), запрашивает у пользователя одноразовые коды при необходимости и сохраняет профиль сессии вместе с цепочкой действий для последующего воспроизведения без БЯМ. Модуль разработчика генерирует Python-приложение по эталонному шаблону. Модуль судьи запускает приложение в песочнице, проверяет эндпоинты и сравнивает результат с эталонным JSON, формируя вердикт.

Такая организация соответствует идее ReAct-контур «рассуждение – действие» на этапе конфигурирования [10] и принципу LLM-as-a-judge при контроле качества [11]. Существенное отличие от непрерывного агентного исполнения заключается в том, что после PASS агенты выводятся из пути сбора: дальнейшая работа опирается на детерминированный код, журналы и точки восстановления. Это снижает влияние известных ограничений длинного контекста языковых моделей [12] и умень-

шает накопление ошибок планирования на тысячах однотипных страниц.

Для сокрытия признаков автоматизации браузерный контур использует изолированные профили, управляемые параметры окружения и воспроизводимые цепочки действий. Устойчивость к антибот-системам не является гарантированной характеристикой метода: при усилении защит возможны отказы, требующие ручного вмешательства или изменения стратегии доступа. В экспериментах фиксировались случаи, когда ошибка относилась к инструменту автоматизации, а не к качеству ответа БЯМ.

Порядок эксперимента включал подготовку эталонных пар для выборки страниц, генерацию сборщика, запуск на отложенных страницах того же ресурса и расчет Precision/Recall/F1. Для контура авторизации фиксировались успешность входа, среднее число шагов и среднее время. Для сравнения агентских систем дополнительно измерялись покрытие ресурсов и задержки классификации страницы. Все метрики интерпретировались с учетом стратификации по типам лент и сложности активных действий (см. табл. 1).

3. РЕЗУЛЬТАТЫ И ОБСУЖДЕНИЕ

Результаты анализа средств сбора показали, что ни одно из рассмотренных решений в исходном виде не обеспечивает одновременно высокую скорость массового сбора, устойчивость к динамическим действиям, управляемую авторизацию и приемлемую стоимость инференса. ReaderLM-v2 и NER-модели полезны как компоненты предобработки, но не закрывают сценарии с модальными окнами, пагинацией и 2FA. Browser-use и Skyvern демонстрируют гибкость активных действий, однако проигрывают по скорости и предсказуемости при большом объеме однотипных записей. Ui.Vision ускоряет запись действий, но сохраняет хрупкость к изменению разметки. Далее сравнивались агентские системы, представленные в табл. 8.

Таблица 8 | Сравнение агентских систем в задаче авторизации**Table 8** | Comparison of agent systems in the authorization problem

Критерий	Skyvern	OpenClaw с Ui. Vision	OpenClaw с ToolCalling
Вход при инициализации	13/15 (~87 %)	15/15 (100 %)	15/15 (100 %)
Покрытие решением	8/15 (~53 %)	9/15 (60 %)	11/15 (~73 %)
Среднее время шага авторизации	1–4 мин	13–30 с	5–20 с
Максимальное время шага авторизации	До 8 мин	1 мин 20 с	30 с
Среднее время классификации страницы	Около 4 мин	5–15 с	2–10 с
Конфигурируемость	Низкая	Средняя	Высокая

Сравнение Skyvern и OpenClaw подтвердило, что vision-ориентированная автоматизация обеспечивает вход на большинство ресурсов, но существенно уступает по времени классификации страницы и шага авторизации.

Конфигурация OpenClaw с ToolCalling показала лучшее покрытие и меньшие задержки, однако сохранила зависимость от корректности инструментального описания динамических элементов. OpenClaw с Ui. Vision оказалась медленнее и менее покрывающей, оставаясь чувствительной к сбоям записи макросов. Следовательно, для целевого макета рационален путь ToolCalling с явной спецификацией действий и последующим запуском без БЯМ.

Экспериментальная оценка программного макета на эталонном наборе данных «страница → JSON» показала, что основной показатель качества извлечения составил $F1 = 0,89$. Высокое значение $F1$ достигнуто за счет комбинации: явной схемы целевых полей, задаваемой примером JSON; генерации детерминированного сборщика вместо непрерывного LLM-извлечения; контура судьи, отклоняющего заведомо некорректные реализации до передачи в эксплуатацию.

Анализ ошибок позволяет выделить несколько устойчивых источников деградации качества.

Динамическая подгрузка контента приводит к извлечению «пустых» узлов,

если агент не дождался стабилизации DOM. Модальные окна и скрытые блоки создают ложные негативы, когда поле существует, но недоступно без активного действия. Антибот-механизмы и CAPTCHA вносят отказы, не связанные с качеством языковой модели как таковой: часть неудач относится к инструментам автоматизации браузера. При генерации кода малые модели чаще допускают ошибки валидации и требуют повторных итераций «судья → разработчик», увеличивая стоимость этапа конфигурирования.

С точки зрения стоимости эксплуатации предложенный метод смещает расходы на этап однократного конфигурирования. После генерации сборщика пропускная способность определяется скоростью браузерной автоматизации и I/O хранилищ, а не токеной стоимостью БЯМ. Это особенно важно для длинных лент и регулярного обновления витрин. Если структура ресурса меняется чаще, чем допустимый бюджет повторного конфигурирования, преимущество метода сокращается.

Сравнение с автономными веб-агентами в постановках WebArena-подобных и multimodal web-agent сред [5, 13, 14] показывает сходство в необходимости измерения успеха на многошаговых сценариях и учета ограничений среды. Аналогично относительно AutoScraper и агентных агрегаторов [7, 8] предлагаемый подход

акцентирует эксплуатационный контур, необходимый для прикладного сопровождения. Выбор сильных моделей общего назначения для этапа синтеза [15] согласуется с наблюдаемой чувствительностью качества генерации кода к способности модели следовать сложной спецификации инструментов и HTML-контексту.

Для ресурсов с редкими изменениями верстки целесообразно выполнять однократную генерацию сборщика моделью высокого качества. Для контура авторизации предпочтительна отдельная модель с максимальной успешностью пошагового прохождения форм. Массовый сбор следует выполнять детерминированным кодом и журналированием. Повторный запуск агентного конфигурирования оправдан при падении полноты ниже приемлемого порога или при структурных изменениях ленты/карточки.

Дополнительно проанализирована стратификация качества по типам лент. На одностраничных лентах без пагинации ошибки чаще связаны с полнотой извлечения скрытых блоков, чем с навигацией. На постраничных и смешанных лентах доминируют ошибки синхронизации состояния (пропуск страницы, повтор карточки, раннее завершение при «пустой» загрузке). Для нестандартных лент доля ручной корректировки спецификации остается наивысшей, что подтверждает отсутствие универсального решения для произвольного веб-источника. Этот вывод согласуется с исходной постановкой проблемы фрагментированности практики сбора.

4. ЗАКЛЮЧЕНИЕ

Исследованы методы автоматизированного извлечения данных из веб-ресурсов с динамически изменяемым контентом с помощью ИИ-агентов. Показано, что ключевым фактором эффективности является не непрерывное применение БЯМ к каждой странице, а разделение анализа ресурса и массового детерминированного сбора.

Проанализированы средства сбора данных – в качестве основы выбран гибридный подход, ориентированный на генерацию исполнимого кода. Проанализированы БЯМ: для генерации сборщика рациональна модель высокого качества (в экспериментах DeepseekV4-pro), для контура авторизации – отдельно подобранная модель (в экспериментах qwen3.5-27b). Разработан способ сбора и программный макет на базе OpenClaw, Chromium/Botosaurs, MCP, PostgreSQL и Elasticsearch. Экспериментальные исследования показали $F1=0,89$ на эталонном наборе для сбора данных, а также преимущество OpenClaw+ToolCalling по покрытию и задержкам относительно Skyvern и связки с Ui. Vision в задаче авторизации.

Дальнейшие исследования целесообразно направить на автоматическое обнаружение изменения разметки, снижение стоимости повторного конфигурирования, расширение типологии ресурсов, включая сложные SPA, а также на более строгую оценку устойчивости к антибот-механизмам в контролируемых стендах.

КОНФЛИКТ ИНТЕРЕСОВ / CONFLICT OF INTERESTS

Авторы заявляют об отсутствии конфликта интересов / The authors declare no conflict of interests.

СПИСОК ИСТОЧНИКОВ

1. Wang J., Li Y., Wu J. et al. Unify: A System For Unstructured Data Analytics // Proceedings of the VLDB Endowment. 2025. Vol. 18. № 12. P. 5287–5290. DOI: 10.14778/3750601.3750653.
2. Reinsel D., Gantz J., Rydning J. The Digitization of the World: From Edge to Core. IDC White Paper. Framingham: International Data Corporation, 2018. 28 p.

3. Дахнович А. Д., Богина В. М., Макеева А. А. Применение больших языковых моделей в задаче прогнозирования событий // Проблемы информационной безопасности. Компьютерные системы. 2025. № S2. С. 58–68.
4. Зегжда Д. П., Жуков И. Ю. Особенности обеспечения информационной безопасности вычислительных систем // Безопасность информационных технологий. 2021. Т. 28. № 1. С. 42–61.
5. Zhou S., Xu F. F., Zhu H. et al. WebArena: A Realistic Web Environment for Building Autonomous Agents // arXiv preprint arXiv:2307.13854. 2023.
6. Deng X., Gu Y., Zheng B. et al. Mind2Web: Towards a Generalist Agent for the Web // Advances in Neural Information Processing Systems. 2023. Vol. 36. 24 p.
7. Huang W., Zhou J., Gu X. et al. AutoScraper: A Progressive Understanding Web Agent for Web Scraper Generation // arXiv preprint arXiv:2404.12753. 2024.
8. Reddy R. G., Sagnik Mukherjee, Jeonghwan Kim et al. Infogent: An Agent-Based Framework for Web Information Aggregation // arXiv preprint arXiv:2410.19054. 2024.
9. Hou X., Zhao Y., Wang S., Wang H. Model Context Protocol (MCP): Landscape, Security Threats, and Future Research Directions // ACM Transactions on Software Engineering and Methodology. 2025. 37 p. DOI: 10.1145/3796519.
10. Yao S., Zhao J., Yu D. et al. ReAct: Synergizing Reasoning and Acting in Language Models // International Conference on Learning Representations (ICLR). URL: <https://react-lm.github.io/> (дата обращения: 11.05.2026).
11. Zheng L., Chiang W.-L., Sheng Y. et al. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena // Advances in Neural Information Processing Systems. 2023. Vol. 36. P. 46595–46623.
12. Liu N. F., Lin K., Hewitt J. et al. Lost in the Middle: How Language Models Use Long Contexts // Transactions of the Association for Computational Linguistics. 2024. Vol. 12. P. 157–173.
13. He H., Yao W., Ma K. et al. WebVoyager: Building an End-to-End Web Agent with Large Multimodal Models // arXiv preprint arXiv:2401.13919. 2024.
14. Koh J. Y., Lo R., Jang L. et al. VisualWebArena: Evaluating Multimodal Agents on Realistic Visual Web Tasks // arXiv preprint arXiv:2401.13649. 2024.
15. Jain N., Han K., Gu A. et al. LiveCodeBench: Holistic and Contamination Free Evaluation of Large Language Models for Code // International Conference on Learning Representations (ICLR). URL: <https://arxiv.org/pdf/2403.07974> (дата обращения: 11.05.2026).

REFERENCES

1. Wang J., Li Y., Wu J. et al. Unify: A System For Unstructured Data Analytics. *Proceedings of the VLDB Endowment*. 2025. Vol. 18. No. 12, pp. 5287–5290. DOI: 10.14778/3750601.3750653.
2. Reinsel D., Gantz J., Rydning J. The Digitization of the World: From Edge to Core. IDC White Paper. Framingham: International Data Corporation, 2018, 28 p.
3. Dakhnovich A. D., Bogina V. M., Makeeva A. A. Application of large language models in event forecasting field. *Problems of information security. Computer systems*. 2025. No. S2, pp. 58–68. (In Russian)
4. Zegzhda D. P., Zhukov I. Y. Features of information security of computer systems. *IT Security*. 2021. Vol. 28. No. 1, pp. 42–61. (In Russian)
5. Zhou S., Xu F. F., Zhu H. et al. WebArena: A Realistic Web Environment for Building Autonomous Agents. *arXiv preprint arXiv:2307.13854*. 2023.
6. Deng X., Gu Y., Zheng B. et al. Mind2Web: Towards a Generalist Agent for the Web. *Advances in Neural Information Processing Systems*. 2023. Vol. 36, 24 p.
7. Huang W., Zhou J., Gu X. et al. AutoScraper: A Progressive Understanding Web Agent for Web Scraper Generation. *arXiv preprint arXiv:2404.12753*. 2024.
8. Reddy R. G., Sagnik Mukherjee, Jeonghwan Kim et al. Infogent: An Agent-Based Frame-

- work for Web Information Aggregation. *arXiv preprint arXiv:2410.19054*. 2024.
9. **Hou X., Zhao Y., Wang S., Wang H.** Model Context Protocol (MCP): Landscape, Security Threats, and Future Research Directions. *ACM Transactions on Software Engineering and Methodology*. 2025, 37 p. DOI: 10.1145/3796519.
 10. **Yao S., Zhao J., Yu D. et al.** ReAct: Synergizing Reasoning and Acting in Language Models. *International Conference on Learning Representations (ICLR)*. URL: <https://react-lm.github.io/> (accessed: 11.05.2026).
 11. **Zheng L., Chiang W.-L., Sheng Y. et al.** Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. *Advances in Neural Information Processing Systems*. 2023. Vol. 36. P. 46595–46623.
 12. **Liu N. F., Lin K., Hewitt J. et al.** Lost in the Middle: How Language Models Use Long Contexts. *Transactions of the Association for Computational Linguistics*. 2024. Vol. 12, pp. 157–173.
 13. **He H., Yao W., Ma K. et al.** WebVoyager: Building an End-to-End Web Agent with Large Multimodal Models. *arXiv preprint arXiv:2401.13919*. 2024.
 14. **Koh J. Y., Lo R., Jang L. et al.** VisualWebArena: Evaluating Multimodal Agents on Realistic Visual Web Tasks. *arXiv preprint arXiv:2401.13649*. 2024.
 15. **Jain N., Han K., Gu A. et al.** LiveCodeBench: Holistic and Contamination Free Evaluation of Large Language Models for Code. *International Conference on Learning Representations (ICLR)*. URL: <https://arxiv.org/pdf/2403.07974> (accessed: 11.05.2026).

СВЕДЕНИЯ ОБ АВТОРАХ / INFORMATION ABOUT AUTHORS

АБИТОВ Роман Альбертович – ассистент, Санкт-Петербургский политехнический университет Петра Великого, Россия, 195251, Санкт-Петербург, ул. Политехническая, д. 29
E-mail: abitov_ra@spbstu.ru
ORCID: 0009-0004-9733-3115

ДАХНОВИЧ Андрей Дмитриевич – старший преподаватель, научный сотрудник, Санкт-Петербургский политехнический университет Петра Великого, Россия, 195251, Санкт-Петербург, ул. Политехническая, д. 29
E-mail: add@ibks.spbstu.ru
ORCID: 0000-0001-9862-1507

МОСКВИН Дмитрий Андреевич – канд. техн. наук, доцент, Санкт-Петербургский политехнический университет Петра Великого, Россия, 195251, Санкт-Петербург, ул. Политехническая, д. 29
E-mail: moskvin@ibks.spbstu.ru
ORCID: 0000-0002-8715-8868

ABITOV Roman A. – Assistant, Peter the Great St. Petersburg Polytechnic University, Russia, 195251, St. Petersburg, Polytechnicheskaya str., 29

DAKHNOVICH Andrey D. – Senior Lecturer, Researcher, Peter the Great St. Petersburg Polytechnic University, Russia, 195251, St. Petersburg, Polytechnicheskaya str., 29

MOSKVIN Dmitry A. – Candidate of Engineering Sciences, Associate Professor, Peter the Great St. Petersburg Polytechnic University, Russia, 195251, St. Petersburg, Polytechnicheskaya str., 29