АВТОМАТИЗАЦИЯ НАУЧНОЙ АНАЛИТИКИ: СРАВНЕНИЕ КОНВЕЙЕРОВ ДАННЫХ, СОЗДАННЫХ С ПОМОЩЬЮ ПЕРЕДОВОГО ИИ, И ПРОЕКТИРУЕМЫХ ВРУЧНУЮ

Негматов Улугбек[1], Жобиров Муродуллахон[2], Глеб Маёршин[2], Мадинабону Орифжонова[2]

[1] PhD, Наманганский государственный технический университет.
[2] Студент, Наманганский государственный технический университет.
E-mail: aflotun.team+murot.jobirov+glebmayorshin@gmail.com
DOI: 10.61587/mmit.tiue.uz.v1i1.312

Аннотация. В статье проводится эмпирическое сравнение конвейеров данных научной аналитики, созданных разработчиками вручную, с решениями, сгенерированными передовыми агентными ИИ-фреймворками (AutoGen, AutoGPT, OpenAgents, OpenHands, Claude и Gemini). Сначала описываются два независимых ручных конвейера, обеспечивающих стабильный сбор, семантическую фильтрацию и кластеризацию тысяч документов. Затем оценивается способность ИИ-агентов воспроизвести аналогичный функционал. Эмпирические результаты показывают, что сгенерированный ИИ код страдает от некорректного использования API, предметных ошибок и отсутствия механизмов надежности (кэширования, дедупликации). Время отладки ИИ-кода (35–150+ часов) значительно превысило время разработки ручной системы с нуля (~60 часов). На основе анализа авторы предлагают гибридную архитектуру, совмещающую детерминированное программное ядро и точечное использование LLM.

Ключевые слова: *научная аналитика, автоматизация исследований, агентный ИИ, генерация кода, конвейеры данных, многоагентные системы, гибридные архитектуры.*

AUTOMATING RESEARCH INTELLIGENCE: ADVANCED-AI-GENERATED VS MANUALLY DESIGNED PIPELINES

Negmatov Ulug'bek[1], Jobirov Murodullaxon[2], Gleb Mayorshin[2], Madinabonu Orifjonova[2]

[1] PhD, Namangan Davlat texnika Universiteti.
[2] Student, Namangan State Technical University.
E-mail: aflotun.team+murot.jobirov+glebmayorshin@gmail.com

Abstract. This paper presents an empirical comparison between manually designed research intelligence pipelines and those generated by advanced agentic AI frameworks (AutoGen, AutoGPT, OpenAgents, OpenHands, Claude, and Gemini). We first outline two independent manual pipelines developed for narrow technical fields, achieving stable multi-source ingestion, semantic filtering, and document clustering. We then evaluate the capability of AI agents to replicate this workflow. The results show that AI-generated code suffers from API misuse, domain-specific errors, and a lack of robustness features like caching and deduplication. The expert debugging time for AI-generated pipelines (35 to 150+ hours) far exceeded the time required to build the manual baseline from scratch (~60 hours). We conclude that current AI agents cannot build complex pipelines autonomously, and we propose a hybrid architecture combining a deterministic backbone with targeted LLM integration.

Keywords: *research intelligence, automated research, agentic AI, code generation, data pipelines, multi-agent systems, hybrid architectures.*

ВВЕДЕНИЕ

Экспоненциальный рост научных публикаций усложняет их отслеживание. В динамичных областях (машинное обучение, биоинженерия) ключевые открытия появляются в препринтах (arXiv, bioRxiv) задолго до индексации в Scopus или Web of

Science [1, 2]. Это требует создания автоматизированных систем научной аналитики (automated research intelligence) для сбора, фильтрации и кластеризации данных.

Развитие агентного ИИ (agentic AI) и фреймворков (AutoGen, AutoGPT, OpenHands) обещает автоматический синтез таких систем по описанию на естественном языке. Однако детальные сравнения ИИ-генерации с ручной разработкой отсутствуют. Данная работа сопоставляет эффективность ручных конвейеров и решений от ИИ-агентов в двух тестовых доменах: оценка неопределенности в ML и инженерия антител.

ОБЗОР ЛИТЕРАТУРЫ И МЕТОДОЛОГИЯ

Проблема информационного перегруза исследуется давно [1, 2]. Инструменты вроде Feedly или Inoreader требуют сложной настройки, тогда как агентные фреймворки ИИ позиционируются как автономные разработчики [11, 15]. Наше исследование сопоставляет их декларативные возможности с реальной практикой.

Мы разработали два независимых ручных конвейера. Конвейер 1 собирает статьи из arXiv, bioRxiv, Semantic Scholar, Crossref, OpenAlex, LinkedIn, Reddit, фильтрует их с помощью Sentence Transformers (all-MiniLM-L6-v2) по 10 шаблонам, снижает размерность через UMAP и кластеризует методами HDBSCAN и KMeans. Конвейер 2 обрабатывает PubMed, bioRxiv/medRxiv с векторизацией TF-IDF, оптимизирует кластеризацию по силуэтному коэффициенту и суммирует тексты с помощью TextRank и BART. Обе системы контейнеризованы в Docker.

Аналогичные задачи были поставлены перед фреймворками AutoGen, AutoGPT, OpenAgents, OpenHands, а также моделями Claude (Opus 4.5) и Gemini (3 Pro) в средах Aider и MetaGPT. Для количественной оценки эффективности разработанных конвейеров мы предлагаем модель оценки эффективности (Ep):

$$E\_p = f(F\_c, S\_v, R\_o) = \alpha \cdot F\_c + \beta \cdot S\_v + \gamma \cdot R\_o \ (1)$$

Здесь: Fc — функциональная корректность кода; Sv — предметная валидность алгоритмов; Ro — операционная надежность; α, β, γ — весовые коэффициенты (α + β + γ = 1). Оценка параметров приведена в Таблице 1.

Таблица 1. Сравнительный анализ параметров ручной разработки и ИИ-генерации конвейеров научной аналитики

Параметр сравнения	Ручная реализация (Человеческий дизайн)	ИИ-генерация (Агентные фреймворки)
Функциональная корректность (Fc)	Полная функциональность, стабильная интеграция компонентов и предсказуемая логика.	Быстрая генерация кода, но частые синтаксические ошибки, неверные вызовы API и бесконечные циклы.
Научная валидность (Sv)	Глубокое соответствие предметной области, корректная фильтрация и валидация научных текстов.	Смешение предметных данных (путаница нуклеотидов/аминокислот), использование ASCII-значений для UMAP.
Операционная надежность (Ro)	Наличие кэширования, пакетной обработки, надежная обработка ошибок и Docker-контейнеризация.	Бриттл-код, отсутствие обработки ошибок, утечки памяти (до 8 ГБ на UMAP), отсутствие кэширования (высокие затраты).
Затраты и отладка	Предсказуемые затраты на API ($5.38), около 60 часов экспертного времени на разработку с нуля.	Скрытые расходы на API (до $8000), 35 часов начальной отладки и 150+ часов для достижения базовой стабильности.

РЕЗУЛЬТАТЫ И ОБСУЖДЕНИЕ

Ручные конвейеры продемонстрировали высокую стабильность. Конвейер 1 при мониторинге публикаций по неопределенности в ML успешно обработал 5631 документ, достигнув F1-меры 86.8% (точность 87.7%, полнота 86.0%). Встроенное кэширование снизило время повторного запуска с 12.3 минут до 5 секунд. Конвейер 2 успешно распределил 5028 статей по инженерии антител на 18 кластеров за 33 минуты общего времени (чистый анализ — 343 секунды), сгенерировав качественные обзоры с помощью BART.

ИИ-агенты показали существенные ограничения. AutoGen (gpt-5-mini) не смог настроить API без готовых функций-инструментов, зависая в бесконечных диалогах.

AutoGPT нашел 8 документов через Perplexity, но не справился с крупным сбором из-за лимита в 5-10 элементов. OpenAgents оказался неработоспособен из-за устаревшего бэкенда. OpenHands передавал в LLM сырой HTML с рекламой, а при чтении двухколоночных научных PDF терял структуру текста (смешивал разделы и подписи), а также не выполнял дедупликацию.

При прямой генерации кода Claude создал 3800+ строк в 31 файле за 8 минут, но UMAP падал при попытке обработать текстовые строки как ASCII с евклидовым расстоянием, скрепер путал нуклеотиды и аминокислоты, а память превышала 8 ГБ. Gemini 3 Pro (Aider, MetaGPT) вызвал конфликты версий библиотек (protobuf, grpcio, aiohttp) и накладывал поверхностные заглушки (try-except) вместо исправления ошибок, уходя в циклы самокоррекции.

Анализ затрат опровергает экономичность ИИ. Отладка кода Claude потребовала 35 часов работы эксперта (4 ч — тесты, 12 ч — алгоритмы, 6 ч — схемы БД, 5 ч — вызовы API, 8 ч — конвейер) для достижения базовой работоспособности. Полная интеграция ИИ-решения потребует более 150 часов, тогда как создание ручной системы заняло всего 60 часов. Код OpenHands без кэширования привел бы к расходам на API более $8000 при тестировании, в то время как ручная система требует менее $200 в месяц при обработке 10 000 статей.

На основе результатов мы предлагаем гибридную архитектуру (Рисунок 1), где сбор, очистка и кластеризация данных выполняются детерминированным ручным ядром, а ИИ-компоненты используются точечно для суммаризации и разметки.

Рисунок 1. Четырехуровневая архитектура гибридной системы научной аналитики (Research Intelligence)

Уровень 1. Детерминированный сбор данных и скрепинг (Специализированные классы скреперов, обработка rate-limits с экспоненциальным бэк-оффом, прокси-ротация).

Уровень 2. Предварительная обработка и контроль качества (Очистка HTML от рекламы и скриптов, хэш-дедупликация, двухдвижковое извлечение текста из двухколоночных PDF-файлов).

Уровень 3. Алгоритмический анализ и кластеризация (Векторизация текстов, размерное сжатие через UMAP, плотностное сегментирование HDBSCAN/KMeans с силуэтной валидацией).

Уровень 4. Интеграция LLM для суммаризации (Генерация кратких обзоров на базе BART/BART-like моделей, разметка кластеров, интерактивный интерфейс и экспертный контроль).

ЗАКЛЮЧЕНИЕ

Современные ИИ-агенты не способны автономно создавать сложные специализированные конвейеры данных. Хотя ИИ быстро генерирует код, он отличается хрупкостью, отсутствием механизмов надежности и предметными ошибками. Время на аудит и отладку ИИ-кода превышает время ручной разработки с нуля.

Наиболее эффективным подходом является построение гибридных систем научной аналитики, в которых сбор и алгоритмическая обработка данных доверяются детерминированному ручному ядру, а ИИ-компоненты используются как ассистенты для выполнения ограниченных и легко проверяемых задач: суммаризации текстов и генерации поисковых шаблонов.

СПИСОК ЛИТЕРАТУРЫ

1. Bornmann, L., Haunschild, R., and Mutz, R. (2021). Growth rates of modern science: a latent piecewise growth curve approach to model publication numbers from established and new literature databases. Humanities and Social Sciences Communications, 8(1), 224. https://doi.org/10.1057/s41599-021-00903-w
2. Landhuis, E. (2016). Scientific literature: Information overload. Nature, 535(7612), 457–458. https://doi.org/10.1038/535457a
3. Wu, Q., Bansal, G., Zhang, J., et al. (2024). AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation. Proceedings of the Conference on Language Modeling (COLM 2024).
4. Richards, T. B. (2023). AutoGPT: An open-source autonomous AI agent. GitHub Repository. https://github.com/Significant-Gravitas/AutoGPT
5. Xie, T., Zhou, F., Cheng, Z., et al. (2023). OpenAgents: An Open Platform for Language Agents in the Wild. arXiv preprint arXiv:2310.10634.
6. Wang, X., Li, B., Song, Y., et al. (2024). OpenHands: An Open Platform for AI Software Developers as Generalist Agents. arXiv preprint arXiv:2407.16741.
7. Hong, S., Zhuge, M., Chen, J., et al. (2023). MetaGPT: Meta Programming for A Multi-Agent Collaborative Framework. arXiv preprint arXiv:2308.00352.
8. Teece, D. J., Pisano, G., and Shuen, A. (1997). Dynamic Capabilities and Strategic Management. Strategic Management Journal, 18(7), 509–533.
9. Brynjolfsson, E., Li, D., and Raymond, L. R. (2023). Generative AI at Work. Quarterly Journal of Economics, 138(4), 1875–1925. https://doi.org/10.1093/qje/qjad027
10. Chen, Z., Chen, S., Ning, Y., et al. (2024). ScienceAgentBench: Toward Rigorous Assessment of Language Agents for Data-Driven Scientific Discovery. arXiv preprint arXiv:2410.05080.