УДК 004.89

ИСПОЛЬЗОВАНИЕ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА В РАЗРАБОТКЕ ВЕБ-ПРИЛОЖЕНИЙ

В. С. Романчик, А. Х. Перез Чернов

Белорусский государственный университет,
пр. Независимости, 4, 220030, г. Минск, Беларусь, romanchik@bsu.by, pereztchernov@bsu.by.

Рассмотрены современные возможности использования искусственного интеллекта для разработки фронтенд и бэкенд частей веб-приложений. Приведены популярные инструменты автоматизации, перечислены возможные механизмы улучшения больших языковых моделей для целей веб-программирования.

Ключевые слова: искусственный интеллект; большие языковые модели; автоматизация; обучение моделей; веб-разработка.

Возможности ИИ для фронтенд-разработки

Веб-разработка по созданию выбранного интернет-сервиса или сайта начинается с анализа требований. Аналитик обычно составляет или уже знаком со словарями доменной области, субъектами (акторами, заинтересованным лицами), их целями, типовыми сценариями взаимодействия с выбранными бизнес-функциями. Рассмотрим, как большие языковые модели типа GPT, далее – ИИ, могут ассистировать или автоматизировать эту работу. При использовании ИИ для целей анализа предметной области необходимо осуществлять преобразование неструктурированных диалогов с клиентом и описаний предметной области в согласованные требования. Для выполнения этих действий на наш взгляд для типовых сценариев достаточно использовать существующие коммерческие чатботы типа Antropic Claudi 3 и OpenAI GPT-4. Важно использовать соответствующие системные инструкции и подсказки в числе которых задание роли для ИИ, предоставление примеров для few-shot обучения, предоставления контекстных материалов до инструкций, задание возможностей ИИ модели брать паузы в размышлениях и действовать пошагово [1]. Дополнительным удобством может быть обработка с помощью Lumentis [2] неструктурированных транскриптов переговоров. Возможно и создание специальных fine-tuned моделей для роли аналитика предметной области, обученных на наборе примеров пользовательских противоречивых или неточных описаний, демонстрации вариативности выделения и приоритизации требований.

Следующим этапом в цепочке работ является дизайн пользовательского опыта. По предложенным ранее требованиям необходимо составить последовательности состояний интерфейса, композицию элементов интерфейса. Для составления списка страниц и более удобной визуализации удобно использовать различные flowchart диаграммы, в частности, в форматах Mermaid. Этот тип диаграммы поддерживается основными ИИ-моделями и способен проиллюстрировать ключевые переходы между состояниями. Часто на уровне дизайна пользовательского опыта может стоять задача создания логической модели данных и самого контента. В большинстве открытых и коммерческих системах задача генерации контента обычно успешно решается. В ряде случаев, если контент связан с последующей индексацией, то процедура становится много шаговой с дополнительными этапами задания ключевых слов [3, 4] и исследованием источников [5]. Возможно и создание специальных мультимодальных fine-tuned моделей для роли дизайнера пользовательского интерфейса, обученных на наборе популярных UX-решений и последовательностей страниц, типовых ошибок использования интерфейсов.

Дальнейшим этапом разработки обычно идет этап визуального дизайна, в частности создание и выбор шрифтов, схемы цветов, ритма размеров, сетки размещения, традиционных компонент, и общей дизайн-системы. Далее идет использование верстки визуального дизайна в набор HTML, CSS и простейшего React (JS) кода для управления состояниями страниц. Здесь же создаются наборы отдельных графических и медиа элементов – аудио, видео, фото, графических элементов для последующего использования на веб-страницах.

Создание визуального контента возможно как с помощью средств общих генеративных инструментов Midjourney, так и нишевых DesignAI, IconifyAI, Magic Studio. Прикладные возможности использования AI активно интегрируются в популярные средства графического моделирования Figma, Adobe или специализированные инструменты Krea AI [6].

Проект автоматизации создания HTML-верстки по дизайн-скриншотам [7, 8] иллюстрирует ключевые аспекты обучения прикладных мультимодальных моделей для целей веб-разработки. А именно, исследователи предоставили около 10 тыс примеров скриншотов веб-страниц и их HTML реализации. Определили набор метрик близости получаемых результатов по цветовой схеме, визуальной композиции, текстам. Предоставили возможность модели просматривать только что созданные страницы и итеративно улучшать результаты. В итоге, генерируемая HTML-верстка признана экспертами допустимой уже более чем в половине случаев. Создание одной HTML-страницы по изображению занимает около 5 минут на видеокарте объемом от 24Gb VRAM.

Интересным примером является демонстрация в рамках проекта OpenUI [9] создания HTML-верстки и одновременного итеративного исправления дизайна с помощью обратной связи от человека-оператора. Похожие сценарии работы сейчас интегрируются в различных визуальных инструментах создания сайтов, в частности, в WIX ADI.

Примером использования ИИ для разработки фронтенда может служить проект React-Agent [10], способный составить React-компоненты с учетом библиотек и проектов TailwindCSS, Typescript, Radix UI, Shadcn UI.

Возможности ИИ для бэкенд-разработки

Разработка бэкенд составляющей интернет-сервисов и сайтов часто сопряжена с большой вариативностью и глубиной прикладной логики. Базовые алгоритмы программирования и стандартизированные композиции элементов в целом решаются с помощью доступных ИИ моделей. В качестве популярных свободных моделей, используемых для генерации кода стоит отметить WizardCoder, CodeLLama, DeepSeek Coder, Mistral Code. Существуют специальные доски анализа существующих моделей для целей программирования [11, 12].

Для упрощения работы целесообразно использовать специальные интеграции ИИ с IDE, в числе которых Amazon CodeWhisperer, OpenAI ChatGPT and IntelliJ CodeGPT, Microsoft Co-Pilot, Replit, Cursor, Sourcegraph Cody, Cognition AI Devin, Pythagora GPT-Pilot.

Моделирование логической моделей данных хранения уже возможно языковыми моделями общего типа или специализированными моделями как SQLCoder [13].

Крупные корпорации уже используют ИИ для улучшения юнит-тестирования программного обеспечения [14]. Использование ИИ для создание полностью автоматического создания юнит-тестов, впрочем, еще требует улучшений [15].

При этом, программная разработка в настоящий момент имеет ряд пока нерешенных проблем, над которым активно работают. В числе них: увеличение размера контекста, которым ИИ модель может оперировать. Предоставление возможности модели выполнять несколько шагов самостоятельно. Взаимодействовать с человеком асинхронно вместо пошагового диалога. Самостоятельно запускать компиляцию и сборку, отслеживать поведение работающего программного модуля. Планировать и выполнять большие цепочки действий.

Часть разрабатываемых улучшений находится и в организации механизма размышлений. В частности, обучение на цепочках принятия решений вида вариативность исходов – действие – результат [16], размышления с поиском по деревьям решений [17], итеративной работы [18],

использовании вероятностного вывода [19], взаимодействия несколько агентов над общей задачей, прямой работы с индексируемыми пользовательскими артефактами кода, управлением внимания.

Опишем потенциальные концептуализации и возможные примеры для обучения ИИ-моделей для выбранных ролей.

Роль технического архитектора. Данные: библиотеки, схемы взаимодействия систем, деревья эволюции требований, деревья эволюции технологий и бизнес-контекстов. Примеры: конверсия описаний систем и контекста их использования в соответствующие согласованные архитектурные схематизации и описания.

Роль программиста-разработчика. Данные: спецификации библиотек, спецификации лучших практик, описания типовых способов и границ тестирования. Примеры: конвертация описания требуемого изменения в набор согласованных изменений уже разработанного или нового программного обеспечения.

Роль технического лидера. Данные: последовательности работ, описания технических решений, оценки и риски. Примеры: конвертация описаний продуктовых изменений в цепочку декомпозированных работ.

Роль тестировщика. Данные: типовые ошибки, типовые архитектуры, операции отладки и сборки. Примеры: конверсия задач с обнаруженными или скрытыми ошибками в последовательности операций по анализу данных логов, локализации проблем, дебага, замеров производительности.

Заключение

При решении вышеупомянутых задач автоматизации профессия веб-разработчика серьезно изменится. Те специфические навыки, которые были свойственны преимущественно техническим лидерам и проектировщикам программного обеспечения, могут стать обязательными для большинства веб-программистов. Постепенно будет сглаживаться знание конкретных библиотек в сторону опыта координации ИИ над популярными и стандартизированными библиотеками. Возможно, новые инструменты и библиотеки будут создаваться с первоочередной (AI first) интеграцией с ИИ для обучения и настройки. Серьезно возрастут объемы и скорость разработки, и при должном развитии систем дизайна и тестирования – качество программного кода над стандартизированными библиотеками.

Библиографические ссылки

1. Kirkovska A. Claude 2.1 prompt engineering guide [Electronic resource] // Vellum.ai blog. 2023. URL: https://www.vellum.ai/blog/prompt-engineering-tips-for-claude (date of access: 31.03.2024).

2. Hrishioa. Lumentis: AI powered one-click comprehensive docs from transcripts and text [Electronic resource] // GitHub repository. 2023. URL: https://github.com/hrishioa/lumentis (date of access: 31.03.2024).

3. AJaySi. AI-Blog-Writer: AI Blog Creation and Management Toolkit [Electronic resource] // GitHub repository. 2023. URL: https://github.com/AJaySi/AI-Blog-Writer (date of access: 31.03.2024).

4. Gaurav18115. BLOGEN Blog Generation Application [Electronic resource] // GitHub repository. 2023. URL: https://github.com/gaurav18115/blogen (date of access: 31.03.2024).

5. Dzhng. Deep-seek: LLM powered retrieval engine [Electronic resource] // GitHub repository. 2023. URL: https://github.com/dzhng/deep-seek (date of access: 31.03.2024).

6. KREA.AI [Electronic resource]. 2023. URL: https://www.krea.ai/home (date of access: 31.03.2024).

7. NoviScl. Design2Code: How Far Are We From Automating Front-End Engineering? [Electronic resource] // GitHub repository. 2023. URL: https://github.com/NoviScl/Design2Code (date of access: 31.03.2024).

8. Design2Code: How Far Are We From Automating Front-End Engineering? [Electronic resource] / C. Si [et al.] // arXiv.org. 2024. URL: https://arxiv.org/abs/2403.03163 (date of access: 31.03.2024).

9. Openui: describe UI using your imagination, then see it rendered live [Electronic resource] / Wandb // GitHub repository. 2023. URL: https://github.com/wandb/openui (date of access: 31.03.2024).

10. Eylonmiz. React-agent: The open-source React.js Autonomous LLM Agent [Electronic resource] / Eylonmiz // GitHub repository. 2023. URL: https://github.com/eylonmiz/react-agent (date of access: 31.03.2024).

11. Ravkine M. A visual tool to explore the results of CanAiCode [Electronic resource] // Hugging Face. 2023. URL: https://huggingface.co/spaces/mike-ravkine/can-ai-code-results (date of access: 31.03.2024

12. What LLM to use? A perspective from the DevAI space [Electronic resource] / Continuedev // GitHub repository. 2023. URL: https://github.com/continuedev/what-llm-to-use (date of access: 31.03.2024).

13. Defog-ai. Sqlcoder: SoTA LLM for converting natural language questions to SQL queries [Electronic resource] // GitHub repository. 2023. URL: https://github.com/defog-ai/sqlcoder (date of access: 31.03.2024).

14. Alshahwan N. Automated Unit Test Improvement using Large Language Models at Meta [Electronic resource] // arXiv.org. 2024. URL: https://arxiv.org/pdf/2402.09171.pdf (date of access: 31.03.2024).

15. Huang Y. Generative Software Engineering [Electronic resource] // arXiv.org. 2024. URL: https://arxiv.org/abs/2403.02583 (date of access: 31.03.2024).

16. Yin D. Agent Lumos: Unified and Modular Training for Open-Source Language Agents [Electronic resource] // arXiv.org. 2024. URL: http://arxiv.org/abs/2311.05657 (date of access: 31.03.2024).

17. Feng X. Alphazero-like Tree-Search can Guide Large Language Model Decoding and Training [Electronic resource] // arXiv.org. 2024. URL: https://arxiv.org/abs/2309.17179 (date of access: 31.03.2024).

18. Zhu Y. KnowAgent: Knowledge-Augmented Planning for LLM-Based Agents [Electronic resource] // arXiv.org. 2024. URL: http://arxiv.org/abs/2403.03101 (date of access: 31.03.2024).

19. Liu Z. Reason for Future, Act for Now: A Principled Framework for Autonomous LLM Agents with Provable Sample Efficiency [Electronic resource] // arXiv.org. 2023. URL: https://arxiv.org/abs/2309.17382 (date of access: 31.03.2024).