ПРИМЕНЕНИЕ LLM-АГЕНТОВ ДЛЯ АВТОМАТИЗИРОВАННОГО КОД-РЕВЬЮ

Рыбаченко И.А[1].
[1]*НИ ТПУ, ИШИТР, гр. А3-36, аспирант, iar12@tpu.ru*

Аннотация

В работе представлены результаты экспериментального исследования по применению LLM-агента для автоматизированного проведения код-ревью. Жизненный цикл предложенного LLM-агента основан на последовательном применении инструмента уточняющих вопросов и инструмента сохранения комментариев. Проведено сравнение сгенерированных LLM-агентом комментариев с эталонными комментариями, написанными человеком. Результаты демонстрируют потенциал LLM-агентов в автоматизации код-ревью, однако требуют дальнейшей доработки для повышения точности.

Ключевые слова: LLM, LLM-агент, software engineering, программная инженерия, автоматизация, код-ревью.

Введение

Большие языковые модели демонстрируют высокую эффективность в решении задач обработки естественного языка. По сравнению с более ранними моделями, например рекуррентными сетями, большие языковые модели демонстрируют более глубокое понимание контекста и неявных семантических связей. Специфические характеристики больших языковых моделей делают их хорошим вариантом для использования в качестве вычислительного ядра для агентных систем. В настоящее время активно исследуется возможность применения агентов, основанных на больших языковых моделях, для решения задач, связанных с разработкой программного обеспечения [1].

Разработка программного обеспечения включает множество задач, которые выглядят перспективными для автоматизации при помощи LLM-агентов. Агенты могут применяться для проектирования доменной модели приложения, для написания кода, для создания вопросно-ответной системы по кодовой базе и для множества других задач. В настоящей работе рассматривается вопрос применения LLM-агентов для решения задачи автоматизированного код-ревью.

Код-ревью – это процесс рецензирования кода, написанного разработчиками. Код-ревью применяется для повышения качества кода и для распространения знаний внутри команды. Сроки выполнения код-ревью напрямую влияют на метрику time-to-market, отражающую время между написанием кода и запуском кода для конечного пользователя. Своевременное получение комментариев разработчиком комментария к написанному коду позволит коду быстрее попасть в релиз. Автоматизация код-ревью позволяет получить обратную связь по написанному коду почти мгновенно в сравнении с ручным анализом. В настоящее время решения для автоматизации код-ревью активно развиваются [2].

Существует большое количество исследований, направленных на повышение качества работы LLM-агентов. Известно, что важным фактором, влияющим на качество ответа, сгенерированного большой языковой моделью, является контекст вопроса. Для формирования контекста может применяться технология RAG (Retrieval Augmented Generation). Для больших кодовых баз применяется деление кода на фрагменты и векторизация [3]. Одним из передовых направлений для решения задачи формирования контекста является агентный RAG [4].

Целью настоящей работы является разработка LLM-агента для автоматизации код-ревью. Для формирования контекста применяется агентный RAG. Для оценки качества разработанного агента выполняется экспертное сравнение комментариев, сгенерированных агентом и написанных человеком.

Архитектура LLM-агента

LLM-агент для код-ревью представляет собой программу, которая принимает на вход пулл-реквест (pull request) с изменениями кода, а на выходе выдает список комментариев к пулл-реквесту. Внутри LLM-агента происходит последовательный вызов предложенных инструментов до тех пор, пока ревью не будет завершено. Схема работы LLM-агента представлена на рисунке 1.

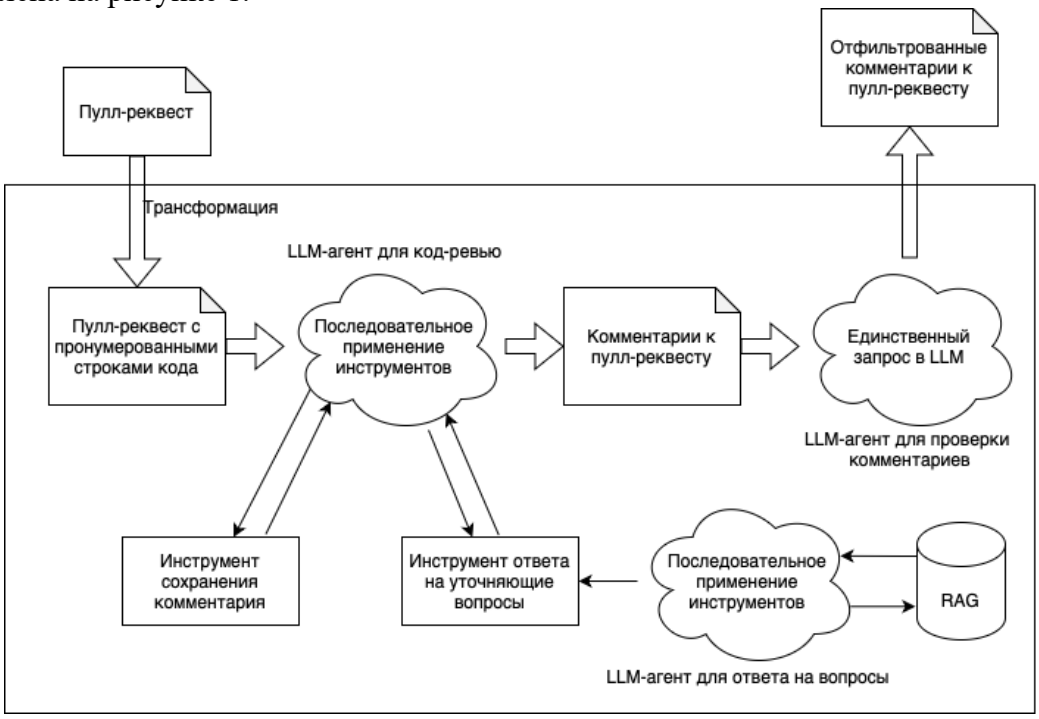

Рис. 1. Схема работы LLM-агента для код-ревью

Код-ревью выполняется двумя агентами – генерирующим и фильтрующим. Такая схема позволяет исключить малоинформативные и неоднозначные комментарии. Каждый комментарий, сгенерированный агентом, требует времени разработчика на ознакомление, поэтому точность (precision) является более приоритетной метрикой, чем полнота (recall).

Исходный пулл-реквест имеет формат следующего вида.

diff --git a/example.txt b/example.txt
index 83db48f..f735d3d 100644
--- a/example.txt
+++ b/example.txt
@@ -1,4 +1,4 @@
-Hello, world!
+Hello, Git world!
This is a sample file.

Пулл-реквест преобразуется к другому формату, в котором указываются номер строки в старом файле, номер строки в новом файле и тип операции. Ниже приведен пример преобразованного пулл-реквеста.

a/example.txt b/example.txt
O1 N- [DELETED] Hello, world!
O- N1 [ADDED] Hello,Git world!
O2 N2 [SAME] This is a sample file.

Преобразование пулл-реквеста из первого формата во второй необходимо для того, чтобы LLM-агент лучше понимал, к какой строке кода относятся изменения и имел возможность привязать комментарий в наиболее релевантной строке кода. Разработчикам проще воспринимать комментарии, привязанные к конкретной строке кода, чем к файлу целиком.

Ключевыми составляющими частями LLM-агента является системный промпт и набор инструментов. Системный промпт описывает набор правил, которыми должен руководствоваться агент при выполнении поставленной задачи. Набор инструментов должен быть достаточным для выполнения задачи, должен быть спроектирован в удобной для агента форме и должен быть хорошо задокументирован.

В качестве вычислительного движка агента использовался deepseek-r1 [5]. Для системного промпта использовался английский язык исходя из предположения, что deepseek-r1 лучше адаптирован к размышлениям на английском или китайском языке, чем на русском. Системный промпт агента для код-ревью имеет следующий формат.

You are a professional software developer. You are asked to make code review for given pull-request. You should follow this rules during the review:
- You should be polite. Add some compliment, if possible.
- Your comments should be concise.
- If you can't figure out the author's logic, you MUST try to clarify it.
- If you can't figure out the logic of code, you MUST try to clarify it.
- If you need some additional information elsewhere in the codebase, you MUST request it.
- Try to pin your comments to the most relevant code line.
- Finish the chat when you finished reviewing.
BEGIN DIFF
{place_holder}
END DIFF
You may use following tools to answer: {place_holder}
Your answer should be formatted as json and contain exactly one tool call. For example, {place_holder}

Агенту были предоставлены инструменты со следующими интерфейсами.

put_comment(is_critical: bool, file_name: str, line_number: str, comment: str)
ask_question(question: str)

Для каждого инструмента была написана документация с указанием примеров использования. Документация так же как и системный промпт писалась на английском языке.

Реализация агентов для RAG и для фильтрации сгенерированных комментариев остается за рамками данной работы. В данной работе вместо этих агентов использовались агенты-заглушки. Агент ответа на вопрос всегда отвечал сообщением «*I received you question and will answer later. Continue reviewing.*». Агент фильтрации комментариев всегда оставлял исходный список комментариев.

Результаты

Работа LLM-агента для код-ревью была проверена на 2 реальных пулл-реквестах, которые ранее были проверены человеком. Для каждого пулл-реквеста агент запускался 3 раза, дубликаты комментариев были устранены. В таблице 1 представлено сравнение комментариев, написанных человеком и LLM-агентом. Полезным комментарием назван комментарий, который был принят автором пулл-реквеста и привел к изменению кода.

Таблица 1. Сравнение комментариев, составленных человеком и LLM-агентом

	Человек		**LLM-агент**		**Сравнение человека и LLM-агента**		
Описание пулл-реквеста	Общее кол-во	Кол-во полезных	Общее кол-во	Кол-во полезных	Кол-во одинаковых	Кол-во новых полезных	Кол-во упущенных полезных
Рефакторинг	4	2	2	0	0	0	2
Сложная логика	5	2	10	1	0	1	2

Для проверки одного пулл-реквеста LLM-агент затрачивает в среднем 3 минуты. Сгенерированные комментарии являются лаконичными и обоснованными. Агент демонстрирует тенденцию к формулированию вопросов, направленных на уточнение логики кода. Это хорошо иллюстрируют следующие примеры:

Пример 1. JpaQueriesCount(48) seems high for this success scenario. Are all 48 queries necessary, or could there be opportunities for batch optimizations or caching to reduce database load?

Пример 2. The logic for retrieving the last PREPARING segment assumes there's exactly one COURIER or MOVEMENT segment. Could there be cases with multiple preparing segments of these types? If yes, how is the correct one selected?

Заключение

В настоящей работе была предложена структура мультиагентной программы для автоматизации код-ревью и продемонстрирована реализация LLM-агента, генерирующего комментарии для код-ревью. Анализ результатов работы агента показал, что сгенерированные комментарии являются обоснованными и лаконичными. Разработанный агент имеет тенденцию задавать уточняющие вопросы в комментариях. Перспективным направлением развития агента является интеграция RAG-системы и оптимизация системного промпта.

Список использованных источников

1. He J., Treude C., Lo D. LLM-Based Multi-Agent Systems for Software Engineering: Vision and the Road Ahead // arXiv e-prints. – 2024. – DOI: 10.48550/arXiv.2404.04834

2. Sun T., Xu J., Li Y., Yan Z., Zhang G., Xie L., Geng L., Wang Z., Chen Y., Lin Q., Duan W., Sui K. BitsAI-CR: Automated Code Review via LLM in Practice // arXiv e-prints. – 2025. – DOI: 10.48550/arXiv.2501.15134

3. Sheffer T. RAG for a codebase with 10k repos. – Текст: электронный. – 2024. – URL: qodo.ai/blog/rag-for-large-scale-code-repos/ (дата обращения 31.03.2025)

4. Singh A., Ehtesham A., Kumar S., Khoei T.T. Agentic Retrieval-Augmented Generation: A Survey on Agentic RAG // arXiv e-prints. – 2025. – DOI: 10.48550/arXiv.2501.09136

5. DeepSeek-AI. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. – 2025. – [Электронный ресурс]. – URL: arxiv.org/pdf/2501.12948 (дата обращения 31.03.2025)