Намерение — это всё, что нужно: уточнение вашего кода на основе вашего намерения

Дата: 2025-02-12 00:00:00

Ссылка на исследование: https://arxiv.org/pdf/2502.08172

Рейтинг: 73
Адаптивность: 85

Ключевые выводы:

Исследование предлагает новый подход к улучшению кода на основе извлечения намерений из комментариев рецензентов. Основная цель - повысить эффективность процесса доработки кода путем разделения задачи на два этапа: извлечение намерения и генерация исправлений на основе этого намерения. Результаты показывают, что этот подход достигает 79% точности в извлечении намерений и до 66% точности в генерации исправленного кода, что значительно превосходит существующие методы.

Объяснение метода:

Исследование предлагает эффективный двухэтапный подход к улучшению кода через LLM: сначала извлечение намерения, затем генерация улучшений. Типология намерений и стратегии промптов непосредственно применимы пользователями. Хотя полная реализация требует технических навыков, ключевые концепции могут быть адаптированы для повседневного использования. Подход показывает значительные улучшения точности (до 66%) и работает с различными моделями.

Ключевые аспекты исследования: 1. **Декомпозиция процесса улучшения кода**: Исследование предлагает разбить процесс улучшения кода на два последовательных этапа: извлечение намерения (Intention Extraction) и генерация улучшений, управляемая намерением (Intention-Guided Revision Generation). Это позволяет лучше понимать цель рецензента и создавать более точные улучшения кода.

Типология намерений рецензентов: Авторы выделяют три основные категории намерений: явные предложения кода (Explicit Code Suggestions), предложения отката изменений (Reversion Suggestions) и общие предложения (General Suggestions) с шестью подкатегориями. Эта классификация структурирует понимание целей рецензентов.

Гибридный подход к извлечению намерений: Для извлечения намерений используется комбинация правил и LLM-классификаторов, что позволяет более

точно определить намерение рецензента из комментариев к коду.

Стратегии промптов для генерации улучшений: Исследование тестирует различные стратегии промптов (простые промпты, RAG-промпты, самогенерируемые промпты) для создания улучшений кода на основе извлеченных намерений.

Очистка данных на основе намерений: Авторы демонстрируют, что использование намерений может улучшить качество данных для задач улучшения кода, повышая согласованность между комментариями рецензентов и фактическими изменениями кода.

Дополнение:

Применимость методов исследования в стандартном чате

Исследование фокусируется на улучшении процесса рецензирования кода с использованием языковых моделей, и хотя авторы использовали API для экспериментов, большинство концепций можно адаптировать для использования в стандартном чате с LLM без дообучения или специального API.

Концепции и подходы, применимые в стандартном чате:

Двухэтапный процесс запросов: Пользователь может сначала попросить LLM проанализировать комментарий рецензента и выделить конкретное намерение Затем использовать это намерение для формулирования более точного запроса на изменение кода

Структурированные шаблоны намерений:

Пользователи могут использовать предложенные категории намерений (explicit, reversion, general) для структурирования своих запросов Например: "Проанализируй этот комментарий к коду и определи, предлагает ли рецензент конкретный код, откат изменений или общие изменения"

Стратегии промптов:

RAG-подход можно имитировать, предоставляя релевантные примеры в запросе Self-generated промпты можно реализовать, попросив модель сначала создать примеры, а затем использовать их для решения исходной задачи

Пост-обработка на основе правил:

Пользователь может попросить модель следовать определенным правилам при генерации кода: Сохранять неизменными строки, не затронутые в намерении Поддерживать согласованность комментариев Включать только необходимые изменения **Ожидаемые результаты от адаптации:**

Повышение точности генерации кода (на 5-15% согласно исследованию) **Улучшение понимания комментариев рецензентов Более структурированные и целенаправленные изменения кода Снижение вероятности избыточных или нежелательных изменений** Хотя полная автоматизация процесса потребовала бы системы с несколькими агентами и API, основные концепции исследования могут быть успешно применены в обычном чате с LLM, что делает их доступными для широкого круга пользователей без необходимости в специальных технических знаниях или инструментах.

Анализ практической применимости: **Декомпозиция процесса улучшения кода**: - Прямая применимость: Высокая. Пользователи могут применять двухэтапный подход при взаимодействии с LLM для улучшения кода, сначала четко формулируя намерение, а затем запрашивая конкретные изменения. - Концептуальная ценность: Значительная. Понимание важности четкого выражения намерения помогает пользователям формулировать более эффективные запросы к LLM. - Потенциал для адаптации: Высокий. Принцип декомпозиции сложных задач на более простые применим к широкому спектру взаимодействий с LLM.

Типология намерений рецензентов: - Прямая применимость: Средняя. Пользователи могут использовать классификацию для структурирования своих запросов к LLM при работе с кодом. - Концептуальная ценность: Высокая. Понимание различных типов намерений помогает пользователям осознать, какие виды запросов LLM обрабатывает лучше всего. - Потенциал для адаптации: Высокий. Классификация может быть расширена или адаптирована для различных контекстов разработки.

Гибридный подход к извлечению намерений: - Прямая применимость: Низкая для обычных пользователей, так как требует технической реализации. - Концептуальная ценность: Средняя. Понимание того, что комбинирование правил и LLM может улучшить результаты. - Потенциал для адаптации: Средний. Принцип комбинирования различных подходов может быть применен в упрощенной форме.

Стратегии промптов для генерации улучшений: - Прямая применимость: Очень высокая. Пользователи могут непосредственно использовать описанные стратегии промптов при работе с LLM. - Концептуальная ценность: Высокая. Пользователи получают понимание того, как различные типы промптов влияют на качество генерации кода. - Потенциал для адаптации: Высокий. Стратегии могут быть адаптированы для различных задач, связанных с кодом.

Очистка данных на основе намерений: - Прямая применимость: Низкая для обычных пользователей, больше подходит для исследователей и разработчиков систем. - Концептуальная ценность: Средняя. Показывает важность согласованности между намерением и результатом. - Потенциал для адаптации: Средний. Принцип может быть применен для валидации результатов генерации LLM.

Prompt:

Использование знаний из исследования "Намерение — это всё, что нужно" для создания эффективных промптов Исследование демонстрирует, что двухэтапный подход с выделением намерений значительно повышает качество улучшения кода. Вот как можно применить эти знания в промптах для GPT.

Ключевой принцип Вместо прямой передачи комментария или задачи, сначала выделите конкретное намерение, а затем используйте его для генерации решения.

Пример промпта на основе исследования

[======] # Запрос на улучшение кода

Исходный код [======]python def calculate_total(items): total = 0 for item in items: total += item.price return total [======]

Комментарий рецензента "Этот код не учитывает случай, когда items может быть пустым списком или None. Также стоит учесть налог."

Инструкции: 1. Сначала выдели конкретные намерения из комментария рецензента (что именно нужно изменить) 2. Для каждого намерения предложи конкретное исправление кода 3. Представь окончательную версию улучшенного кода с учетом всех намерений 4. Кратко объясни, как твои изменения соответствуют выделенным намерениям [======]

Почему такой промпт работает лучше

Структурированное извлечение намерений: Промпт явно требует выделить конкретные намерения, что соответствует первому этапу метода из исследования (79% точности в извлечении намерений).

Поэтапная генерация решений: Вместо попытки сразу решить всю проблему, промпт разбивает задачу на логические шаги, как в исследовании.

Верификация соответствия: Требование объяснить, как изменения соответствуют намерениям, обеспечивает дополнительную проверку, что повышает точность (как показано в исследовании, где точность повышается до 66% при таком подходе).

Структурированный вывод: Четкая структура промпта обеспечивает более организованный ответ, что облегчает понимание и применение предложенных изменений.

Другие применения метода

• Для более сложных задач можно использовать гибридный подход, комбинируя правила для простых случаев и LLM для сложных

• При работе с большими проектами можно включать контекст из базы кода (RAG-подход)

• Для разных типов задач можно создавать специализированные шаблоны промптов, ориентированные на конкретные типы намерений

Этот метод особенно эффективен для задач улучшения кода, но может быть адаптирован и для других областей, где важно точно понять намерение запроса перед генерацией ответа.