# Как ученые используют большие языковые модели для программирования

Дата: 2025-02-24 00:00:00

Ссылка на исследование: https://arxiv.org/pdf/2502.17348

Рейтинг: 75
Адаптивность: 85

## Ключевые выводы:

Исследование направлено на изучение того, как ученые используют большие языковые модели (LLM) для программирования в научных исследованиях. Основные результаты показывают, что ученые часто используют LLM как инструмент для поиска информации о незнакомых языках программирования и библиотеках, причем большинство предпочитает интерфейсы чата (например, ChatGPT) встроенным инструментам IDE (например, GitHub Copilot).

## Объяснение метода:

Исследование высоко полезно, раскрывая как ученые используют LLM для программирования. Предоставляет практические стратегии навигации в незнакомых языках, методы верификации кода и выявляет типичные заблуждения о работе моделей. Результаты применимы для широкой аудитории, хотя требуют некоторой адаптации из научного контекста.

## Ключевые аспекты исследования: 1. **Исследование использования LLM учеными для программирования**: Статья изучает, как научные исследователи применяют генеративные модели для написания кода в своей работе.

**Интерфейсы взаимодействия с LLM**: Исследуется предпочтение ученых между двумя интерфейсами - браузерными чат-интерфейсами (Chat GPT) и интегрированными в IDE средствами автодополнения кода (GitHub Copilot).

**Мотивация использования**: Ученые часто используют LLM как инструмент для получения информации о незнакомых языках программирования и библиотеках, заменяя традиционные источники документации и Stack Overflow.

**Стратегии проверки кода**: Выявлены основные подходы к проверке сгенерированного кода, включая его запуск с проверкой вывода, построчное чтение и анализ структуры кода.

**Ментальные модели и заблуждения**: Исследование показывает, что у некоторых

пользователей есть неверные представления о работе LLM (например, что они работают как поисковые системы или могут выполнять вычисления).

## Дополнение:

### Применимость методов в стандартном чате

Исследование не требует дообучения или API для применения основных выявленных подходов. Большинство описанных методов можно непосредственно использовать в стандартном чат-интерфейсе без дополнительных инструментов.

### Ключевые адаптируемые концепции:

**Использование LLM как инструмента для изучения незнакомых языков программирования** Можно просить модель объяснить синтаксис, функции и методы в любом языке Запрашивать примеры использования конкретных библиотек Использовать для разбора ошибок и отладки

**Стратегии декомпозиции сложных задач**

Разбивать сложные программные задачи на более мелкие части Запрашивать генерацию кода для каждой подзадачи отдельно Постепенно объединять результаты

**Методы верификации сгенерированного кода**

Проверка построчно с анализом логики Поиск знакомых паттернов и структур Запуск кода с разными входными данными для проверки Использование модели для объяснения собственного кода с последующей проверкой объяснения

**Взаимодействие с ментальными моделями**

Понимание ограничений модели (не является поисковиком или калькулятором) Формирование запросов с учетом этих ограничений Критическая оценка ответов, особенно для длинных блоков кода ### Ожидаемые результаты от применения:

• Повышение эффективности изучения новых языков программирования

• Снижение риска принятия некорректного кода

• Более структурированный подход к решению сложных задач программирования

• Лучшее понимание возможностей и ограничений LLM в контексте программирования

Исследователи использовали расширенные техники в основном для удобства анализа, но ключевые принципы и методы полностью применимы в стандартном чате.

## Анализ практической применимости: 1. **Использование LLM как инструмент для навигации в незнакомых языках и библиотеках** - Прямая применимость: Высокая. Пользователи могут сразу использовать LLM для поиска функций, методов и примеров использования в незнакомых языках и библиотеках. - Концептуальная ценность: Средняя. Помогает понять, что LLM могут быть эффективным инструментом для обучения новым языкам программирования. - Потенциал для адаптации: Высокий. Любой пользователь может адаптировать этот подход для изучения новых технологий, независимо от уровня подготовки.

**Предпочтения интерфейсов (чат vs. IDE-интеграция)** Прямая применимость: Средняя. Помогает выбрать подходящий интерфейс в зависимости от типа задачи. Концептуальная ценность: Высокая. Исследование объясняет различия между интерфейсами и их влияние на процесс программирования и проверки кода. Потенциал для адаптации: Средний. Знание о различиях интерфейсов помогает пользователям сделать осознанный выбор, но применимость зависит от доступных инструментов.

**Стратегии проверки сгенерированного кода**

Прямая применимость: Высокая. Описанные стратегии проверки кода могут быть немедленно применены всеми пользователями. Концептуальная ценность: Очень высокая. Понимание ограничений и рисков при использовании сгенерированного кода критически важно. Потенциал для адаптации: Высокий. Предложенные методы верификации можно адаптировать для различных задач и уровней сложности.

**Заблуждения о работе LLM**

Прямая применимость: Средняя. Понимание распространенных заблуждений помогает избегать ошибок при использовании. Концептуальная ценность: Высокая. Правильное представление о принципах работы LLM позволяет эффективнее их использовать. Потенциал для адаптации: Средний. Знание ограничений моделей полезно, но требует определенного уровня технической грамотности.

**Случаи пропуска ошибок в сгенерированном коде**

Прямая применимость: Высокая. Примеры реальных ситуаций, когда ошибки не были замечены, служат предупреждением. Концептуальная ценность: Очень высокая. Демонстрирует реальные риски при использовании сгенерированного кода. Потенциал для адаптации: Высокий. Примеры могут быть использованы как руководство по тому, на что обращать внимание при проверке кода.

# Prompt:

Использование знаний из исследования о применении LLM учеными в промтах для GPT ## Ключевые инсайты из исследования

Исследование показывает, что: - 71% ученых предпочитают интерфейсы чата вместо встроенных IDE-решений - Разбиение сложных задач на подзадачи повышает эффективность - Существуют специфические стратегии верификации кода (запуск, визуальная проверка, построчное чтение) - Код длиннее 40 строк чаще содержит ошибки - Необходимо тщательно проверять модификации существующего кода

## Пример промта с использованием знаний из исследования

[======] # Запрос на оптимизацию кода для научных вычислений

## Контекст Я ученый, работающий с анализом данных в области [указать область науки]. Мне нужно оптимизировать следующий код для обработки экспериментальных данных.

## Мой код [======]python [вставить код, не более 40 строк] [======]

## Что мне нужно 1. Проанализируй мой код и предложи оптимизации, сохраняя все основные параметры анализа. 2. Разбей свой ответ на следующие секции: - Краткое объяснение того, что делает текущий код - Предлагаемые изменения с объяснением каждого изменения - Оптимизированный код - 2-3 простых модульных теста для проверки корректности работы

Особенно обрати внимание на параметры, которые могут повлиять на научные результаты (например, [указать критичные параметры]).

Предложи документированный код с комментариями для каждого ключевого шага. [======]

## Почему этот промт эффективен на основе исследования

**Разбиение задачи**: Промт четко структурирован и запрашивает разбивку ответа на логические секции, что соответствует рекомендации о разбиении сложных задач.

**Ограничение размера кода**: Указано ограничение в 40 строк, что согласно исследованию снижает вероятность ошибок.

**Стратегия верификации**: Запрос включает создание модульных тестов, что исследование выделяет как эффективную стратегию проверки.

**Внимание к критическим параметрам**: Промт явно акцентирует внимание на параметрах, которые могут повлиять на научные результаты, что адресует выявленный в исследовании риск.

**Запрос документации**: Требование комментировать код помогает пользователю лучше понять изменения и снижает риск пропустить важные детали.

Используя структуру промта, основанную на результатах исследования, вы значительно повышаете шансы получить качественный, проверяемый и надежный код для научных вычислений.