Генерация входных данных для тестирования значений границ с использованием проектирования подсказок с большими языковыми моделями: обнаружение ошибок и анализ покрытия

Дата: 2025-01-24 00:00:00

Ссылка на исследование: https://arxiv.org/pdf/2501.14465

Рейтинг: 71
Адаптивность: 75

Ключевые выводы:

Исследование оценивает эффективность использования больших языковых моделей (LLM) для генерации тестовых входных данных с граничными значениями в контексте тестирования программного обеспечения методом белого ящика. Основные результаты показывают, что LLM, при правильном использовании промптов, могут генерировать тестовые входные данные, сравнимые или превосходящие по эффективности традиционные методы в обнаружении ошибок и покрытии кода в определенных случаях.

Объяснение метода:

Исследование предлагает практичную методологию использования LLM для генерации тестовых данных через простые промпты, которые любой пользователь может адаптировать. Демонстрирует эффективность LLM в обнаружении сложных ошибок и важность качества тестов над количеством. Однако полная ценность требует понимания концепций тестирования и доступа к исходному коду, что ограничивает применимость для некоторых пользователей.

Ключевые аспекты исследования: 1. **Методология использования LLM для генерации тестовых входных данных**: Исследование предлагает фреймворк для оценки эффективности LLM в создании граничных тестовых значений для программного обеспечения, используя инженерию промптов для направления моделей на создание специфических тестовых входных данных.

Сравнение с традиционными методами: Авторы сравнивают тестовые данные, сгенерированные LLM, с данными, полученными традиционными методами (случайное тестирование, конколическое тестирование, машинное обучение для анализа граничных значений), оценивая способность обнаружения ошибок и охват

кода.

Оценка эффективности обнаружения ошибок: Исследование анализирует способность LLM-генерированных тестовых наборов выявлять различные типы ошибок в коде, включая ошибки "off-by-one", которые часто встречаются на границах условий.

Влияние количества тестовых данных: Авторы изучают взаимосвязь между количеством сгенерированных тестовых входных данных и эффективностью тестирования, выявляя, что больший объем тестов не всегда гарантирует лучшие результаты.

Корреляция между охватом кода и обнаружением ошибок: Исследование выявляет положительную корреляцию между охватом ветвей кода и обнаружением ошибок, особенно для тестовых данных, ориентированных на граничные значения.

Дополнение: Исследование не требует дообучения или специального API для применения основных методов. Авторы использовали GPT-4o с простыми промптами для генерации тестовых входных данных. Хотя для анализа результатов применялись специальные инструменты (gcov), сам процесс генерации тестов доступен в стандартном чате.

Концепции и подходы, которые можно применить в стандартном чате:

Генерация граничных тестовых случаев: Используя промпт "Generate boundary value test inputs for c code delimited by triple backticks", можно получить тестовые данные, ориентированные на граничные условия. Этот подход применим к любому коду, который пользователь хочет протестировать.

Сбалансированный подход к количеству тестов: Исследование показывает, что качество тестовых данных важнее их количества. Пользователи могут запрашивать небольшие, но хорошо продуманные наборы тестов.

Адаптация промптов для разных языков программирования: Хотя исследование фокусируется на C/C++, тот же подход можно применять для Python, JavaScript и других языков.

Фокус на конкретных типах ошибок: Можно модифицировать промпты для поиска конкретных типов ошибок, например: "Generate test cases that would identify off-by-one errors in this function".

Итеративное улучшение тестов: Пользователи могут анализировать результаты выполнения сгенерированных тестов и запрашивать уточненные тесты на основе обнаруженных проблем.

Применяя эти концепции, пользователи могут значительно улучшить качество своего тестирования без необходимости в специализированных инструментах. Результаты включают более надежный код, раннее обнаружение ошибок и лучшее

понимание потенциальных проблемных мест в программах.

Анализ практической применимости: **Методология использования LLM для генерации тестовых входных данных**: - Прямая применимость: Высокая. Пользователи могут непосредственно использовать метод инженерии промптов для направления LLM на создание тестовых входных данных. Простые промпты, представленные в исследовании, легко адаптируются для различных программ. - Концептуальная ценность: Значительная. Демонстрирует способность LLM анализировать код и генерировать осмысленные тестовые входные данные, что расширяет понимание возможностей LLM в программировании. - Потенциал для адаптации: Высокий. Подход можно применять к программам на разных языках программирования и модифицировать промпты для различных типов тестирования.

Сравнение с традиционными методами: - Прямая применимость: Средняя. Результаты сравнения информативны, но требуют понимания традиционных методов тестирования для практического применения. - Концептуальная ценность: Высокая. Помогает понять сильные и слабые стороны LLM в контексте тестирования, что позволяет выбрать подходящий метод для конкретных задач. - Потенциал для адаптации: Средний. Знание эффективности разных методов позволяет комбинировать подходы, но требует дополнительной экспертизы.

Оценка эффективности обнаружения ошибок: - Прямая применимость: Высокая. Пользователи могут использовать LLM для генерации тестов, которые фокусируются на обнаружении конкретных типов ошибок, особенно граничных случаев. - Концептуальная ценность: Высокая. Демонстрирует, что LLM способны обнаруживать сложные ошибки, такие как ошибки "off-by-one", что расширяет понимание их аналитических возможностей. - Потенциал для адаптации: Высокий. Подход можно адаптировать для поиска различных типов ошибок путем модификации промптов.

Влияние количества тестовых данных: - Прямая применимость: Высокая. Понимание того, что качество тестовых данных важнее их количества, может непосредственно применяться при разработке тестовых стратегий. - Концептуальная ценность: Значительная. Помогает осознать важность баланса между количеством и качеством тестов. - Потенциал для адаптации: Средний. Принцип универсален, но определение оптимального количества тестов для конкретной задачи требует экспериментов.

Корреляция между охватом кода и обнаружением ошибок: - Прямая применимость: Средняя. Знание о корреляции полезно, но его практическое применение требует инструментов для измерения охвата кода. - Концептуальная ценность: Высокая. Подтверждает важность тестирования граничных условий для обнаружения ошибок. - Потенциал для адаптации: Средний. Принцип может быть применен к различным типам программ, но требует адаптации под конкретные случаи.

Prompt:

Использование знаний из исследования о граничных значениях в промптах для GPT
Ключевые выводы исследования для промптов

Исследование показывает, что большие языковые модели (LLM) могут эффективно генерировать тестовые данные с граничными значениями, иногда превосходя традиционные методы тестирования. Особенно важно качество промптов, а не количество сгенерированных тестов.

Пример промпта для тестирования граничных значений

[======] # Запрос на генерацию тестовых данных с граничными значениями

Контекст программы Я разрабатываю функцию, которая проверяет валидность возраста пользователя для регистрации на сайте. Возраст должен быть от 18 до 120 лет.

Код функции [======]python def validate_age(age): if isinstance(age, (int, float)) and 18 <= age <= 120: return True return False [======]

Запрос Сгенерируй набор тестовых входных данных с граничными значениями для этой функции. Для каждого значения укажи: 1. Само значение 2. Ожидаемый результат (True/False) 3. Граничное условие, которое проверяется

Особенно сфокусируйся на: - Точных граничных значениях (17, 18, 119, 120) - Значениях рядом с границами - Экстремальных значениях - Нетипичных входных данных (строки, None, отрицательные числа) [======]

Почему это работает

Данный промпт эффективен, потому что:

Содержит конкретную информацию о программе — указаны тип, назначение и ограничения функции **Включает исходный код** — позволяет модели точно определить граничные условия **Структурирует запрос** — четко указывает, какие именно данные нужно сгенерировать **Направляет внимание на граничные значения** — явно запрашивает проверку граничных случаев **Запрашивает обоснование** — просит указать, какое граничное условие проверяется Согласно исследованию, такой подход позволяет получить более качественные тестовые данные, которые с большей вероятностью выявят ошибки, особенно на границах допустимых значений, где часто возникают проблемы.

Применение в других контекстах

Этот подход можно адаптировать для различных задач тестирования, включая проверку функций обработки текста, валидации данных, математических вычислений и других областей, где важно тестирование граничных случаев.