Генерация тестов на основе LLM cGuidedMutation в Meta

Дата: 2025-01-22 00:00:00

Ссылка на исследование: https://arxiv.org/pdf/2501.12862

Рейтинг: 70
Адаптивность: 75

Ключевые выводы:

Исследование описывает систему ACH (Automated Compliance Hardener) от Meta для генерации тестов на основе мутаций с использованием больших языковых моделей (LLM). Основная цель - создание тестов, которые могут обнаруживать определенные типы ошибок (в данном случае связанных с приватностью), которые не обнаруживаются существующими тестами. ACH генерирует относительно небольшое количество мутантов (симулированных ошибок) по сравнению с традиционным мутационным тестированием, но фокусируется на создании тестов для обнаружения конкретных проблем.

Объяснение метода:

Исследование демонстрирует эффективный подход к мутационному тестированию с использованием LLM для выявления специфических проблем (приватность). Высокая принимаемость тестов инженерами (73%) и их релевантность (36%) свидетельствуют о практической ценности. Концепции генерации направленных мутантов, определения их эквивалентности и создания тестов применимы широкой аудиторией, хотя полная реализация требует адаптации.

Ключевые аспекты исследования: 1. **Мутационное тестирование для специфических проблем**: ACH генерирует мутанты (симулированные ошибки), нацеленные на конкретные проблемы (в данном случае - приватность), вместо случайных ошибок.

LLM для генерации тестов: Система использует языковые модели для создания мутантов и тестов, способных обнаружить эти мутанты, что "закаляет" код против регрессий.

Автоматическое обнаружение эквивалентных мутантов: ACH включает агента на базе LLM, который определяет, действительно ли мутант изменяет поведение программы.

Интеграция в рабочий процесс: Система встраивается в стандартный процесс разработки через механизм "диффов", которые рассматриваются обычным

способом.

Оценка инженерами: Тесты, созданные ACH, достигли 73% принятия инженерами, причем 36% тестов были признаны релевантными для проблем приватности.

Дополнение: Исследование ACH от Meta демонстрирует подход, который во многом может быть адаптирован для использования в стандартных чатах с LLM, хотя оригинальная реализация интегрирована в CI/CD системы компании.

Рассмотрим ключевые концепции, которые можно применить в обычном чате:

Генерация мутантов для конкретных проблем: Пользователь может предоставить код и описание проблемы (например, "проблемы с приватностью данных") и попросить LLM создать версию кода с потенциальной ошибкой этого типа. Это не требует дообучения или API, а использует способность LLM понимать контекст и генерировать код.

Проверка эквивалентности: Пользователь может показать LLM две версии кода и спросить, эквивалентны ли они функционально. Исследование показывает, что LLM могут эффективно выполнять эту задачу (с точностью 79-95%).

Генерация тестов для обнаружения ошибок: После получения мутанта пользователь может запросить LLM создать тест, который обнаружит введенную ошибку. Это является ключевым компонентом метода и полностью выполнимо в обычном чате.

Итеративное улучшение: Пользователь может повторять этот процесс для разных частей кода или разных типов проблем.

Результаты, которые можно получить: - Тесты, ориентированные на конкретные типы проблем - Лучшее понимание потенциальных уязвимостей в коде - Повышение качества тестирования без необходимости в сложной инфраструктуре

Важно отметить, что хотя в исследовании используется Llama 3 170B, основные концепции работают и с другими современными LLM. Исследователи сами отмечают, что не использовали продвинутые методы промптинга или дообучение, что делает подход еще более доступным для адаптации в обычном чате.

Анализ практической применимости: 1. **Мутационное тестирование для специфических проблем** - Прямая применимость: Высокая. Пользователи могут адаптировать подход для генерации тестов, нацеленных на конкретные проблемы, описывая их текстом. - Концептуальная ценность: Очень высокая. Демонстрирует, как можно преобразовать неформальные описания проблем в конкретные тесты. - Потенциал для адаптации: Высокий. Подход может быть адаптирован для любых типов проблем, не только приватности.

LLM для генерации тестов Прямая применимость: Средняя. Требует интеграции с CI/CD системами, но сам принцип использования LLM для генерации тестов может

быть применен в обычных чатах. Концептуальная ценность: Высокая. Показывает, как LLM могут генерировать тесты, обнаруживающие специфические ошибки. Потенциал для адаптации: Высокий. Можно применять в различных контекстах и для различных языков программирования.

Автоматическое обнаружение эквивалентных мутантов

Прямая применимость: Средняя. Принцип использования LLM как "судьи" для оценки эквивалентности кода может быть применен пользователями. Концептуальная ценность: Высокая. Решает классическую проблему мутационного тестирования. Потенциал для адаптации: Средний. Требует понимания концепции эквивалентных мутантов.

Интеграция в рабочий процесс

Прямая применимость: Низкая. Специфична для процессов Meta, требует адаптации. Концептуальная ценность: Средняя. Показывает, как автоматизированные инструменты могут интегрироваться в рабочий процесс. Потенциал для адаптации: Средний. Идеи интеграции можно адаптировать к другим процессам.

Оценка инженерами

Прямая применимость: Низкая. Специфична для контекста Meta. Концептуальная ценность: Высокая. Демонстрирует реальное принятие подхода разработчиками. Потенциал для адаптации: Средний. Методология оценки может быть адаптирована.

Prompt:

Применение знаний из исследования ACH в промптах для GPT На основе исследования Meta о системе ACH (Automated Compliance Hardener) можно создавать эффективные промпты для GPT, которые помогут генерировать тесты для выявления конкретных типов ошибок в коде.

Пример промпта для генерации тестов на проверку приватности данных

[======] Действуй как система мутационного тестирования, основанная на исследовании ACH (Automated Compliance Hardener) от Meta.

Я предоставлю тебе код класса на Kotlin для Android приложения. Твоя задача:

Проанализировать код и определить потенциальные проблемы с приватностью данных пользователя Создать 3-5 мутантов кода, которые симулируют типичные ошибки приватности (например, отсутствие проверки разрешений, неправильное хранение чувствительных данных, утечка данных через логи) Для каждого мутанта разработать тест, который обнаружит эту проблему Объяснить, как каждый тест защищает от конкретной проблемы приватности Вот код класса: [======] [код

класса] [======] [======]

Почему это работает

Данный промпт использует ключевые аспекты исследования ACH:

Целевая генерация мутантов — промпт фокусируется на конкретной области (приватность данных), что соответствует подходу ACH генерировать небольшое количество целевых мутантов вместо случайных.

Многоэтапный процесс — промпт разбивает задачу на этапы (анализ, создание мутантов, генерация тестов), что соответствует компонентной архитектуре ACH.

Объяснение релевантности — требование объяснить, как тесты защищают от проблем, помогает оценить их эффективность, что согласуется с оценкой релевантности тестов в исследовании (где 36% были признаны релевантными для приватности).

Фокус на конкретных проблемах — вместо общего покрытия кода, промпт направлен на обнаружение специфических проблем, что отражает вывод исследования о том, что 49% тестов не добавляют покрытие кода, но все равно обнаруживают важные ошибки.

Этот подход можно адаптировать для других областей безопасности, таких как защита от инъекций, проверка авторизации или соответствие нормативным требованиям, просто изменив фокус в пункте 1 промпта.