HAFix: История-Увеличенные Большие Языковые Модели для Исправления Ошибок

Дата: 2025-01-15 00:00:00

Ссылка на исследование: https://arxiv.org/pdf/2501.09135

Рейтинг: 62
Адаптивность: 70

Ключевые выводы:

Исследование представляет подход HAFix (History augmented LLMs for Bug Fixing), который улучшает способность больших языковых моделей (LLM) исправлять программные ошибки путем использования исторического контекста из репозиториев программного обеспечения. Основные результаты показывают, что использование исторических эвристик значительно улучшает производительность исправления ошибок, с улучшением до 45% по сравнению с базовым подходом без исторического контекста.

Объяснение метода:

Исследование демонстрирует эффективность использования исторического контекста для улучшения работы LLM при исправлении ошибок в коде. Пользователи могут адаптировать ключевые концепции (использование имен измененных файлов, структурирование запросов в стиле Instruction), но полная реализация требует технической инфраструктуры. Основная ценность — в понимании важности исторического контекста и эффективных стилей запросов для работы с LLM.

Ключевые аспекты исследования: 1. **HAFix: новый подход к исправлению ошибок с помощью исторического контекста** - исследование предлагает метод, который использует историческую информацию из репозитория (данные из коммитов, изменения файлов) для улучшения работы LLM при исправлении ошибок в коде.

Семь исторических эвристик - авторы разработали различные способы извлечения исторической информации: имена измененных функций, имена всех измененных файлов, парные изменения кода функций и diff-патчи изменений, что позволяет предоставить LLM более богатый контекст.

HAFix-Agg: агрегированный подход - комбинирует результаты различных исторических эвристик, что улучшает производительность на 45% по сравнению с базовым подходом без исторического контекста.

Анализ стилей запросов (промптов) - исследование сравнивает три стиля запросов к LLM: Instruction, Instruction-label и Instruction-mask, выявляя наиболее эффективный для исправления ошибок.

Анализ компромисса между производительностью и затратами - оценка влияния различных подходов на время выполнения и стоимость инференса, предлагая стратегии для оптимизации затрат.

Дополнение:

Исследование HAFix действительно требует специализированной инфраструктуры и API для полной реализации, особенно для агрегированного подхода HAFix-Agg. Однако ключевые концепции и подходы можно адаптировать для использования в стандартном чате с LLM.

Вот что можно применить в обычном чате:

Использование исторического контекста: Пользователи могут включать информацию о предыдущих изменениях кода в свои запросы. Например, "Этот код раньше использовал другой метод для получения задач, но был изменен для оптимизации производительности."

Эвристика FLN-all (имена измененных файлов): Можно упоминать, какие файлы были затронуты в процессе разработки. Например, "При внедрении этой функции были изменены файлы scheduler.py, worker.py и tasks.py."

Стиль запроса Instruction: Использование четких, прямых инструкций оказалось наиболее эффективным. Вместо "Можешь посмотреть на этот код?" лучше писать "Исправь ошибку в этой функции, которая должна фильтровать задачи по статусу."

Предоставление функционального контекста: Включение не только проблемного кода, но и окружающего контекста функции.

Ожидаемые результаты от применения этих концепций: - Более точное понимание модели причин возникновения ошибки - Более релевантные предложения по исправлению - Более контекстно-зависимые решения, учитывающие архитектуру проекта

Хотя полная эффективность HAFix не может быть достигнута без специальной инфраструктуры, даже частичное применение этих концепций может значительно улучшить качество взаимодействия с LLM при решении задач программирования.

Анализ практической применимости: **1. HAFix: использование исторического контекста** - **Прямая применимость**: Средняя. Пользователи могут адаптировать идею использования исторических данных в своих запросах к LLM, особенно имена измененных файлов (FLN-all), показавшие лучший результат. Однако требуется

доступ к истории репозитория и инструменты для извлечения этих данных. - **Концептуальная ценность**: Высокая. Понимание важности исторического контекста для LLM существенно улучшает взаимодействие, показывая, что модель может лучше понимать причины ошибок, имея доступ к эволюции кода. - **Потенциал для адаптации**: Высокий. Принцип обогащения запросов историческим контекстом применим к различным задачам, включая генерацию кода и документации.

2. Семь исторических эвристик - **Прямая применимость**: Средняя. Пользователи могут вручную извлекать некоторые из этих данных для своих запросов, особенно имена измененных файлов, но полная автоматизация требует технических навыков. - **Концептуальная ценность**: Высокая. Понимание того, какие типы исторической информации наиболее полезны, помогает формулировать более эффективные запросы к LLM. - **Потенциал для адаптации**: Высокий. Эвристики могут быть адаптированы для различных языков программирования и типов проектов.

3. HAFix-Agg: агрегированный подход - **Прямая применимость**: Низкая. Требует выполнения множественных запросов с различными эвристиками и агрегации результатов, что затруднительно для ручного исполнения. - **Концептуальная ценность**: Средняя. Демонстрирует ценность комбинирования различных подходов, но сложен для реализации обычными пользователями. - **Потенциал для адаптации**: Средний. Принцип агрегации может быть применен к другим задачам, но требует технической реализации.

4. Анализ стилей запросов - **Прямая применимость**: Очень высокая. Пользователи могут немедленно применять стиль Instruction в своих запросах, который показал наилучшие результаты. - **Концептуальная ценность**: Высокая. Понимание влияния структуры запроса на качество ответа LLM критически важно для эффективного использования. - **Потенциал для адаптации**: Высокий. Принципы структурирования запросов применимы к широкому спектру задач.

5. Анализ компромисса между производительностью и затратами - **Прямая применимость**: Средняя. Стратегии ранней остановки могут быть адаптированы пользователями для оптимизации использования LLM. - **Концептуальная ценность**: Высокая. Понимание компромиссов между качеством, временем и стоимостью важно для эффективного использования LLM. - **Потенциал для адаптации**: Средний. Принципы оптимизации затрат применимы к различным сценариям, но требуют технической реализации.

Prompt:

Использование исследования HAFix в промптах для GPT ## Ключевые знания из исследования для улучшения промптов

Исследование HAFix показывает, что добавление исторического контекста из репозиториев кода значительно улучшает способность языковых моделей исправлять ошибки в коде (до 45% улучшения). Особенно эффективными оказались:

Имена измененных файлов (FLN-all) - улучшение на 10% **Агрегированный подход (HAFix-Agg)** - улучшение на 45% **Стиль промпта с явными инструкциями** - наиболее эффективный формат ## Пример эффективного промпта на основе исследования

[======] # Задача: исправь ошибку в коде

Контекст ошибки Файл: user_authentication.py Ошибка в строке 45: *user_data = process_credentials(username, password, None)* Сообщение об ошибке: TypeError: process_credentials() takes 2 positional arguments but 3 were given

Исторический контекст (на основе HAFix) Связанные файлы, которые недавно менялись: - auth_utils.py - credential_processor.py - session_manager.py

Последние изменения в модуле аутентификации: - Три дня назад была обновлена сигнатура функции process_credentials() - Был добавлен новый класс SessionManager для управления сессиями

Инструкция 1. Проанализируй ошибку, используя предоставленный исторический контекст 2. Предложи исправление кода 3. Объясни, почему это исправление решает проблему [======]

Почему это работает

Данный промпт использует три ключевых принципа из исследования HAFix:

Включает имена связанных файлов (эвристика FLN-all), что дает модели более широкий контекст для понимания взаимосвязей в коде **Предоставляет историю изменений**, что помогает модели понять происхождение ошибки (изменение сигнатуры функции) **Использует формат с явными инструкциями**, который, согласно исследованию, показал наилучшие результаты Такой подход к составлению промптов может значительно повысить качество исправления ошибок в коде, особенно для сложных случаев, где контекст текущего файла недостаточен для полного понимания проблемы.