ComplexFuncBench: Изучение многошагового и ограниченного вызова функций в условиях длинного контекста

Дата: 2025-01-17 00:00:00

Ссылка на исследование: https://arxiv.org/pdf/2501.10132

Рейтинг: 65
Адаптивность: 70

Ключевые выводы:

Исследование направлено на оценку способностей больших языковых моделей (LLM) выполнять сложные вызовы функций в реальных сценариях. Авторы создали бенчмарк Complex-FunC-Bench для оценки многошаговых и ограниченных вызовов функций в контексте длиной 128k токенов. Результаты показали, что даже современные LLM имеют значительные недостатки в обработке сложных вызовов функций, особенно в определении правильных значений параметров.

Объяснение метода:

Исследование предоставляет ценные концептуальные знания о слабых местах LLM при вызове функций, включая детальную классификацию ошибок и их распределение по типам. Это помогает пользователям адаптировать стратегии взаимодействия и диагностировать проблемы. Однако практическое применение требует технических знаний и значительной адаптации для неспециалистов.

Ключевые аспекты исследования: 1. **Complex FuncBench** - новый бенчмарк для оценки сложных вызовов функций в LLM, который включает многошаговые и ограниченные вызовы функций в сценариях с длинным контекстом (128k). Охватывает 5 реальных доменов (отели, авиабилеты, достопримечательности, аренда автомобилей, такси).

Многомерная оценка вызова функций - авторы предлагают ComplexEval, автоматическую систему оценки, которая использует три подхода для определения эквивалентности вызовов функций: точное соответствие, сопоставление по ответу API и сопоставление на основе LLM.

Методология создания и аннотирования данных - исследование описывает трехэтапный процесс: предварительная генерация с помощью GPT-4o, тщательное аннотирование экспертами и масштабирование датасета с 100 до 1000 примеров.

Выявление слабых мест современных моделей - исследование показывает, что даже передовые модели (GPT-4o, Claude-3.5) имеют значительные проблемы с параметризацией вызовов функций, особенно при работе с многошаговыми вызовами и длинными контекстами.

Анализ ошибок по типам - авторы классифицируют ошибки вызова функций на 5 категорий (func_error, param_missing, hallucination, value_error, stop_early) и анализируют их распространенность в разных моделях.

Дополнение:

Применимость в стандартном чате без дообучения и API

Исследование ComplexFuncBench в основном ориентировано на оценку моделей, которые имеют встроенные возможности вызова функций через API. Однако многие концепции и подходы можно адаптировать для использования в стандартном чате без специального API или дообучения.

Концепции, применимые в стандартном чате:

Структурирование многошаговых запросов Пользователи могут разбивать сложные задачи на последовательность простых шагов Можно явно указывать модели порядок выполнения действий, имитируя многошаговый вызов функций Пример: "Сначала найди информацию о X, затем используй эту информацию для Y"

Обработка ограничений

Исследование показывает, что модели затрудняются с параметрами фильтрации и ограничениями Пользователи могут формулировать ограничения более явно и структурированно Можно использовать маркированные списки для перечисления ограничений

Работа с памятью контекста

Исследование показывает проблемы с извлечением информации из длинного контекста Пользователи могут периодически просить модель резюмировать ключевую информацию Важную информацию можно повторять в последующих запросах

Предотвращение ранней остановки

Исследование выявило, что модели часто прекращают выполнение задачи преждевременно Пользователи могут явно просить модель проверить, все ли требования выполнены Можно использовать чек-листы для отслеживания прогресса

Проверка параметров

Большинство ошибок связано с неправильными значениями параметров Пользователи могут просить модель обосновывать свои выводы и значения Можно использовать пошаговое рассуждение для проверки правильности извлеченных данных #### Ожидаемые результаты при адаптации:

• Повышение точности выполнения сложных многошаговых задач

• Улучшение способности модели работать с ограничениями и фильтрами

• Более эффективное использование контекста в длинных беседах

• Снижение частоты преждевременного завершения задач

• Повышение точности извлечения и использования параметров

Несмотря на отсутствие формального API для вызова функций, эти подходы позволяют имитировать многие аспекты функциональности, исследуемой в ComplexFuncBench, и значительно повысить эффективность взаимодействия с LLM в стандартном чате.

Анализ практической применимости: 1. **Complex FuncBench как бенчмарк**: - Прямая применимость: Ограниченная для обычных пользователей, так как сам бенчмарк предназначен для исследователей и разработчиков LLM. - Концептуальная ценность: Высокая, поскольку выявляет типичные проблемы при вызове функций в сложных сценариях, что помогает пользователям лучше понимать ограничения моделей. - Потенциал для адаптации: Средний - пользователи могут использовать описанные сценарии как шаблоны для тестирования возможностей вызова функций в своих приложениях.

Многомерная оценка вызовов функций: Прямая применимость: Низкая для обычных пользователей, поскольку требует технических знаний и доступа к API моделей. Концептуальная ценность: Высокая, демонстрирует, что эквивалентность вызовов функций не сводится к точному соответствию строк. Потенциал для адаптации: Средний - идея многомерной оценки может быть упрощена для пользовательских проверок работы функций.

Методология создания и аннотирования данных:

Прямая применимость: Низкая для большинства пользователей, но полезна для разработчиков и исследователей. Концептуальная ценность: Средняя, демонстрирует важность качественного аннотирования данных. Потенциал для адаптации: Средний - подход к разрешению неоднозначности может быть полезен при разработке пользовательских инструкций для LLM.

Выявление слабых мест современных моделей:

Прямая применимость: Высокая, помогает пользователям осознать ограничения

моделей при вызове функций и адаптировать свои запросы. Концептуальная ценность: Высокая, позволяет понять, какие типы задач вызывают наибольшие трудности у LLM. Потенциал для адаптации: Высокий - пользователи могут разрабатывать стратегии для обхода выявленных ограничений.

Анализ ошибок по типам:

Прямая применимость: Высокая, помогает пользователям идентифицировать и исправлять типичные ошибки при работе с вызовами функций. Концептуальная ценность: Высокая, дает понимание когнитивных ограничений LLM при обработке сложных задач. Потенциал для адаптации: Высокий - пользователи могут разрабатывать проверки и подсказки для минимизации каждого типа ошибок.

Prompt:

Использование знаний из исследования ComplexFuncBench в промптах для GPT ## Ключевые выводы для применения в промптах

Исследование ComplexFuncBench показывает, что даже современные LLM имеют трудности с обработкой сложных вызовов функций, особенно в определении правильных значений параметров и работе с длинным контекстом. Эти знания можно использовать для создания более эффективных промптов.

Пример промпта с применением выводов исследования

[======] # Задание: Бронирование авиабилета

Контекст Мне нужно найти авиабилет из Москвы в Барселону на период с 15 по 22 июля. Бюджет до 30000 рублей. Предпочитаю прямые рейсы, вылет в первой половине дня.

Инструкции для выполнения задачи: 1. Разбей задачу на последовательные шаги (исследование показывает, что разбиение сложных задач на шаги повышает точность) 2. Для каждого параметра поиска: - Четко выдели значение параметра - Проверь соответствие значения ограничениям (важно: 78.8% ошибок в исследовании связаны с неправильными значениями параметров) - Убедись, что параметр соответствует требованиям API 3. После формирования запроса, проведи самопроверку всех параметров перед финальным вызовом функции 4. Структурируй вывод в формате JSON для вызова search_flights API

Ожидаемый формат вывода: [======]json { "origin": "строка", "destination": "строка", "departure_date": "YYYY-MM-DD", "return_date": "YYYY-MM-DD", "max_price": число, "direct_only": логическое значение, "preferred_departure_time": "строка" } [======]

Пожалуйста, подробно объясни каждый шаг твоего рассуждения. [======]

Объяснение применения выводов исследования

Разбиение на шаги: Исследование показало, что модели часто требуют больше шагов для выполнения задач. Промпт структурирует задачу по шагам, что снижает вероятность ошибок.

Фокус на параметрах: Поскольку 78.8% ошибок связаны с неправильными значениями параметров, промпт специально требует выделять, проверять и валидировать каждый параметр.

Самопроверка: Добавлен этап самопроверки перед финальным вызовом функции, что помогает избежать ошибок в цепочке рассуждений.

Структурирование информации: Для работы с потенциально длинным контекстом информация в промпте четко структурирована, что облегчает модели извлечение нужных значений.

Четкий формат вывода: Предоставлен точный шаблон JSON для вызова API, что снижает вероятность ошибок в структуре ответа.

Такой подход к составлению промптов учитывает выявленные в исследовании ComplexFuncBench слабые места LLM при работе со сложными вызовами функций и помогает минимизировать эти проблемы.