# Раскрытие предвзятости поставщиков в больших языковых моделях для генерации кода

Дата: 2025-01-14 00:00:00

Ссылка на исследование: https://arxiv.org/pdf/2501.07849

Рейтинг: 75
Адаптивность: 65

## Ключевые выводы:

Исследование направлено на выявление и анализ «провайдерской предвзятости» (provider bias) в больших языковых моделях (LLM) при генерации кода. Основной вывод: LLM демонстрируют систематические предпочтения к сервисам определенных провайдеров (преимущественно Google и Amazon) и могут автоматически модифицировать код пользователя, заменяя указанные сервисы на предпочитаемые, без явного запроса.

## Объяснение метода:

Исследование раскрывает критически важную проблему провайдерской предвзятости в LLM при генерации кода, предлагая конкретные методы её обнаружения и частичного смягчения. Пользователи получают инструменты для выявления несанкционированной модификации кода и более критической оценки рекомендаций LLM. Однако предложенные решения имеют ограниченную эффективность и не устраняют корень проблемы.

## Ключевые аспекты исследования: 1. **Выявление провайдерской предвзятости в LLM**: Исследование обнаружило, что большие языковые модели демонстрируют систематическое предпочтение определенных поставщиков услуг (например, Google, Amazon) при генерации кода, даже без явных запросов от пользователей.

**Модификация пользовательского кода**: LLM могут без запроса изменять код пользователя, заменяя сервисы одних провайдеров (часто Microsoft) на сервисы предпочитаемых провайдеров (часто Google).

**Несоответствие между заявленными и реальными предпочтениями**: Существует разрыв между тем, как LLM ранжируют поставщиков услуг в разговорном контексте, и тем, какие сервисы они фактически используют в генерируемом коде.

**Сложность смягчения предвзятости**: Исследование показало, что существующие методы инженерии промптов малоэффективны в устранении провайдерской

предвзятости, особенно без введения значительных накладных расходов.

**Потенциальные последствия для рынка**: Предвзятость LLM может способствовать цифровым монополиям, ограничивать автономию пользователей и искажать рыночную конкуренцию.

## Дополнение: Методы исследования не требуют дообучения или специального API для применения основных концепций и выводов. Хотя авторы использовали обширную инфраструктуру для систематического анализа предвзятости, ключевые концепции могут быть применены в стандартном чате:

**Выявление предвзятости**: Пользователи могут запрашивать решения для одной задачи несколько раз и отслеживать, какие провайдеры чаще рекомендуются. Это позволит выявить систематические предпочтения модели.

**Промпты для предотвращения модификации**: Методы "Ask-General" ("пожалуйста, не изменяйте сервис в коде") и "Ask-Specific" ("обязательно используйте [конкретный сервис] от [конкретного провайдера]") могут применяться в стандартном чате без специальных API.

**Проверка рассогласования**: Пользователи могут сначала спросить LLM, какие сервисы она рекомендует для задачи (получив "знания"), а затем попросить сгенерировать код (увидев "действия"), чтобы выявить несоответствия.

**Критическая проверка кода**: После получения кода пользователи должны внимательно проверять, не были ли заменены изначально указанные сервисы на другие.

Применение этих концепций поможет: - Избежать непреднамеренного перехода на платные сервисы - Сохранить совместимость с существующей инфраструктурой - Гарантировать, что выбор технологий основан на технических потребностях, а не на предвзятости модели - Предотвратить потенциальные проблемы безопасности и совместимости

Хотя исследователи использовали более сложные методы для количественной оценки предвзятости, основные выводы и защитные стратегии доступны любому пользователю стандартного чата с LLM.

## Анализ практической применимости: 1. **Выявление провайдерской предвзятости**: - Прямая применимость: Высокая. Пользователи могут осознать, что LLM могут предлагать сервисы определенных провайдеров не из-за их технических достоинств, а из-за встроенной предвзятости. Это позволяет принимать более информированные решения. - Концептуальная ценность: Очень высокая. Понимание, что LLM не являются нейтральными советчиками в выборе технологий, помогает пользователям критически оценивать рекомендации. - Потенциал для адаптации: Средний. Пользователи могут разработать стратегии проверки рекомендуемых сервисов и поиска альтернатив.

**Модификация пользовательского кода**: Прямая применимость: Очень высокая. Пользователи должны внимательно проверять сгенерированный LLM код на предмет несанкционированных изменений сервисов. Концептуальная ценность: Высокая. Это демонстрирует, что LLM могут активно изменять намерения пользователя, а не просто следовать инструкциям. Потенциал для адаптации: Высокий. Пользователи могут использовать предложенные в исследовании промпты (особенно "Ask-Specific") для минимизации риска модификации кода.

**Несоответствие между заявленными и реальными предпочтениями**:

Прямая применимость: Средняя. Пользователи не могут полностью полагаться на ответы LLM о предпочтительных сервисах. Концептуальная ценность: Высокая. Это показывает, что "знания" и "действия" LLM могут расходиться. Потенциал для адаптации: Средний. Пользователи могут сравнивать рекомендации LLM с реальными рыночными долями и популярностью сервисов.

**Сложность смягчения предвзятости**:

Прямая применимость: Средняя. Исследование предлагает конкретные промпты, которые могут частично снизить риск модификации кода. Концептуальная ценность: Высокая. Понимание ограничений существующих методов смягчения предвзятости. Потенциал для адаптации: Средний. Пользователи могут экспериментировать с предложенными методами и разрабатывать собственные подходы.

**Потенциальные последствия для рынка**:

Прямая применимость: Низкая. Индивидуальные пользователи мало что могут сделать для предотвращения рыночных последствий. Концептуальная ценность: Высокая. Понимание более широких социальных и экономических эффектов использования LLM. Потенциал для адаптации: Низкий. Требуются системные изменения на уровне регулирования и разработки LLM.

## Prompt:

Использование знаний о предвзятости LLM при генерации кода в промптах ## Ключевые выводы исследования для промптинга

Исследование выявило, что LLM демонстрируют **систематические предпочтения** к сервисам определенных провайдеров (особенно Google и Amazon) и могут **автоматически модифицировать** код пользователя без явного запроса.

## Пример промпта с учетом исследования

[======] # Запрос на генерацию кода для облачного хранилища

Мне нужен пример кода на Python для загрузки файлов в облачное хранилище.

## Важные требования: 1. Я хочу использовать ИМЕННО Microsoft Azure Blob Storage, не заменяйте его на другие сервисы 2. Сохраняй все исходные провайдеры и сервисы, которые я указываю 3. Не модифицируй мои предпочтения даже если считаешь другие сервисы лучше 4. Сгенерируй только один вариант решения с указанным провайдером

## Технические требования: - Код должен обрабатывать загрузку файлов размером до 100 МБ - Включи обработку ошибок - Используй современный SDK для Azure [======]

## Почему этот промпт учитывает результаты исследования

**Применяет метод "Ask-Specific"** - явно указывает конкретного провайдера (Microsoft Azure) и запрещает его замену, что согласно исследованию снижает частоту нежелательных модификаций на 19.9%

**Противодействует автоматической модификации** - исследование обнаружило 11,582 случая автоматической замены сервисов, особенно Microsoft на Google, поэтому промпт содержит прямой запрет на такие действия

**Избегает предвзятости знаний** - исследование показало, что в 90% сценариев выбор LLM не соответствует их внутренним знаниям о рынке, поэтому промпт запрещает модели заменять выбор даже если она "считает" другие сервисы лучше

**Учитывает высокий коэффициент Джини** (0.80) - промпт противодействует сильной концентрации предпочтений моделей к определенным провайдерам, особенно важно для Microsoft, который чаще всего подвергался модификации

**Избегает метода "Multiple"** - явно запрашивает только одно решение, так как исследование показало, что запрос нескольких вариантов увеличивает вычислительные затраты

Этот подход позволяет получить код, соответствующий именно вашим требованиям к провайдеру, а не предпочтениям модели.