🧠 Введение в эвалы для AI-агентов

🎯 Введение в эвалы (Evals) для AI-агентов

Ключевые тезисы:

- ✅ **Эвалы** — это систематические тесты для измерения производительности AI-системы в конкретной области.
- 🔄 Они превращают субъективные ощущения ("вибсы") в **действенные данные**.
- 🛠️ Эвалы — это мост между утверждениями "вроде работает" и точным пониманием, что и как улучшать.
- 🎪 Готовые общие бенчмарки (SWE-Bench, OSWorld) полезны, но для вашего кейса **нужно строить свои эвалы**.

📖 Что такое эвалы?

Эвалы — это систематические тесты, которые измеряют, насколько хорошо AI-система справляется с конкретной задачей или доменом. Они дают информацию о качестве результатов, сильных и слабых сторонах системы и показывают пути для улучшений.

Эвалы состоят из **задач**, которые моделируют определённые сценарии, и **логики оценивания**, которая кодирует ожидания от системы. Если эвал падает — вы сразу знаете, что агент ведёт себя не так, как задумано.

🔥 Зачем нужны свои эвалы?

Без эвалов вы оказываетесь в **реактивном цикле**:

- ❌ Вы ловите проблемы только в продакшене.
- ❌ Сложно отличить полезный фидбэк от шума.
- ❌ Невозможно проверить, улучшили ли вы систему или ухудшили её после изменений.
- ❌ Риск, что исправление одной проблемы сломает что-то другое.

Эвалы дают ясность и делают процесс управления агентом проактивным:

- ✅ **Формализуют ожидания**: чтобы построить эвал, нужно чётко определить, что такое успех.
- ✅ **Позволяют итерировать**: можно тестировать разные конфигурации агента, промпты и модели.
- ✅ **Ускоряют adoption новых моделей**: есть чёткие метрики для сравнения.
- ✅ **Выявляют проблемы до запуска**.

🧑‍⚖️ Типы "градеров" (оценщиков)

1. Код-градеры (Code-based graders)

Похожи на юнит-тесты в разработке.

- **Как работают**: строгое сравнение (string match, regex, проверка кода).
- **Плюсы**: быстрые, дёшевые, детерминированные.
- **Минусы**: хрупкие (`brittle`), не улавливают нюансов качества.
- **Пример для агента слайдов**: подсчёт количества слайдов, подсчёт эмодзи.

2. Модель-градеры (Model-based graders)

Используют LLM для оценки по заданным критериям (рубрикам).

- **Как работают**: LLM оценивает выход агента по заданным критериям (качество текста, вёрстки и т.д.).

- **Методы**:
  - **Рубричное оценивание**: "Оцени когерентность текста от 1 до 5".
  - **Парное сравнение (Pairwise comparison)**: "Какой из двух выводов лучше и почему?".
  - **Консенсус нескольких судей (Multi-judge consensus)**: несколько LLM-оценок, побеждает мнение большинства.
- **Плюсы**: гибкие, масштабируемые, учитывают нюансы.
- **Минусы**: недетерминированные, дороже, требуют калибровки.

3. Человек-градеры (Human graders)

- **Плюсы**: самое высокое качество оценки, максимально nuanced.
- **Минусы**: очень дорого, медленно.
- **Применение**: A/B-тестирование, выборочная проверка.

🧪 Практический кейс: Агент для генерации презентаций

Цель: Показать цикл "эвал → инсайт → улучшение агента → новый эвал".

🚀 Шаг 1: Базовый агент

- **Промпт**: "Ты агент для генерации слайдов. Создай PowerPoint-файл по заданной теме".
- **Результат**: Слайды созданы, но качество низкое (мелкий шрифт, нагромождение, эмодзи).
- **Построенные эвалы (примеры)**:
  - *Код-градеры*: количество слайдов, количество слайдов с картинками, количество "загруженных" слайдов, количество слайдов с мелким шрифтом, подсчёт эмодзи.
  - *Модель-градеры*: оценка цвета, компоновки, текста, изображений по шкале от 0 до 5.

🔧 Шаг 2: Итерация на основе эвалов

- **Инсайты из эвалов**: много эмодзи, мелкий шрифт, плохая вёрстка.
- **Действие**: Уточняем системный промпт, добавляя конкретные инструкции по типографике, layout и запрету на "AI-признаки" (например, декоративные эмодзи).
- **Результат**: Слайды стали визуально лучше и последовательнее.

🖼️ Шаг 3: Добавление нового требования

- **Новое требование**: "Каждый слайд должен содержать хотя бы одну сгенерированную диаграмму".
- **Действие**: Обновляем промпт и запускаем эвалы снова.
- **Результат**: Агент начал добавлять графики, что улучшило восприятие.

🔁 Шаг 4: Внедрение QA-цикла

- **Идея**: Добавить второго агента-критика, который ищет ошибки в работе первого.
- **Промпт для критика**: "Подходи к QA как к охоте на баги, а не как к шагу подтверждения. Предполагай, что проблемы есть — ищи их".
- **Результат**: Качество слайдов снова возросло, т.к. агент сам себя проверял и исправлял в несколько итераций.

🧠 Шаг 5: Переход на более умную модель

- **Действие**: Смена модели с Sonnet на Opus (более мощная) **без изменения промпта**.
- **Результат**: Качество слайдов значительно выросло "из коробки". Модель сама избегала типичных ошибок (эмодзи, мелкий шрифт).
- **Важный инсайт**: Эвалы помогли **объективно** зафиксировать и измерить этот скачок в качестве.

⚠️ Ключевые вызовы и лучшие практики

1. **Эвалы — это "живой артефакт"**. Их нужно постоянно пересматривать и калибровать. Риск **насыщения эвалов** — когда они перестают давать полезную информацию.

2. **Калибровка модель-градеров критически важна и сложна.**
    - 💡 Давайте градеру **примеры плохого и хорошего** для [illegible] шкалы.
    - ⚠️ **Порядок в промпте имеет значение!** Сначала попросите LLM перечислить *все "за" и "против"*, а уже потом на основе этого списка выставить итоговую оценку. Если сначала попросить оценку, LLM будет подгонять аргументы под неё.
3. **Для сложных задач используйте несколько техник**:
    - Консенсус нескольких судей (multi-judge consensus).
    - Парное сравнение (pairwise comparison).
    - Циклы с adversarial-агентами (один создаёт, другой критикует).

🎯 Выводы

- Эвалы — это **фундаментальный инструмент** для ответственной и эффективной разработки AI-агентов.
- Они переводят разработку из режима **гадания и реактивных правок** в режим **измерения, анализа и проактивных улучшений**.
- **Начинайте с малого**: определите 1-2 ключевые метрики успеха для вашего агента, постройте под них простые эвалы и запустите цикл итераций.
- Процесс бесконечен: **строим эвалы → получаем данные → улучшаем агента → перепроверяем эвалы**.
