🤖 Prompting Playbook: Отладка и создание промптов

🎯 Prompting Playbook: Отладка и создание эффективных промптов

Ключевые тезисы:

- Промптинг — критический навык для создания эффективных AI-систем.
- Основные сценарии: **поддержка/миграция существующего промпта** и **создание нового агента с нуля**.
- Основа для любой работы — **строгие эвалюации (оценки)**, чтобы объективно измерять влияние изменений.
- Улучшение часто идёт по пути: **общая очистка → целевое исправление конкретных ошибок**.
- Инструкции не добавляют модели новых способностей; для сложных задач нужны **инструменты (tools)**.
- Для сложных задач **агентский подход** (разделение на этапы) может быть эффективнее одного сложного промпта.

📊 Фундамент: Эвалюации (Evaluations)

Перед любыми изменениями нужна система оценки. Эвал-сьют должен включать три типа тест-кейсов:

- **Контрольный случай**: Простой, однозначный запрос, который должен всегда проходить.
- **Пограничные случаи (Edge cases)**: Ситуации, где модель ранее ошибалась.
- **Проверка границ возможностей**: Понимает ли модель, когда нужно передать задачу человеку или отказаться?

Пример из кейса поддержки телеком-компании:

1. Контрольный: «Каков лимит данных в базовом тарифе?»
2. Пограничный: Расчёт пропорционального счёта при смене тарифа.
3. Проверка эскалации: Перевод к специалисту при ошибке в биллинге.
4. Проверка утаивания: Не скрывает ли модель информацию, к которой имеет доступ.

🔧 Сценарий 1: Поддержка и миграция промпта

Цель: Улучшить работающий промпт, который начал давать сбои (например, после миграции на новую модель).

✅ Шаг 1: Общая очистка и «гигиена»

Перед точечными исправлениями приведите промпт в порядок:

- **Уберите ложные утверждения** (например, «ты — человек»).
- **Удалите лишнюю информацию**, скопированную с сайтов (упоминания картинок, cookies).
- **Добавьте чёткую структуру** с использованием XML-тегов для разделения роли, политик, гайдлайнов и тона.

> 💡 *Правило: Если вы не можете отличить гайдлайны от политик и данных, то, скорее всего, и модель не может.*

Результат: Часто уже это даёт заметный прирост качества на эвалюациях.

✅ Шаг 2: Чёткий контракт на вывод (Output Contract)

Для сложных форматов вывода (JSON, XML) явно укажите ожидаемую структуру.

- Используйте **стоп-последовательности (stop sequences)** в API, чтобы обрезать лишний вывод.
- Для очень сложных схем рассмотрите **структурированные выходы (structured outputs)**.

🎯 Шаг 3: Целевое исправление ошибок (Failure Modes)

Исправляйте проблемы по одной, проверяя результат через эвалюации.

🔍 **Проблема 1: Модель утаивает доступную информацию**

Симптом: Вместо того чтобы сказать, что у клиента 5 ГБ hotspot-данных (это есть в контексте), модель отсылает его проверять данные в личном кабинете.

Причина: В промпте был «заплаточный» пункт для старой модели: «Никогда не давай клиенту неверные данные о тарифе. Вместо этого направляй его по URL». Новая модель следует этому слишком буквально.

Решение: Сбалансируйте инструкцию. Укажите, что данные в контексте клиента — это источник истины, и их можно сообщать.

> ⚠️ ***Урок:** Используйте контроль версий для промптов, чтобы отслеживать, зачем были добавлены такие «защитные» инструкции.*

🔍 **Проблема 2: Модель не может выполнить точный расчёт**

Симптом: Модель рассуждает о пропорциональном платеже, но даёт размытый ответ без конкретной суммы.

Причина: Инструкция «Критически важно всегда правильно рассчитывать суммы» не даёт модели способности делать точные вычисления.

*Решение:** **Дайте модели инструмент (tool)**. Опишите в промпте и API схему инструмента `calculate_proration` и реализуйте его логику.

> 💡 ***Ключевой вывод:** Инструкции не добавляют возможности. Чтобы модель надёжно выполняла сложные задачи (математика, поиск), давайте ей инструменты.*

🔍 **Проблема 3: Модель не эскалирует проблему к человеку**

Симптом: При биллинговой ошибке модель пытается сама диагностировать проблему, а не передаёт специалисту.

Причина: В промпте был перекос: «Избегай эскалации, так как это стоит $8». Модель оптимизировала цель минимизации затрат.

*Решение:** **Дайте обе стороны trade-off**. Добавьте: «Эскалация стоит $8, но если ошибиться, это приведёт к возврату средств и потере доверия клиента».

> 🎯 ***Урок:** Современные умные модели лучше справляются с компромиссами, если им дают полную картину, а не односторонние указания.*

🏗️ Сценарий 2: Создание нового агента с нуля

Кейс: Агент для составления недельного графика работы сотрудников с учётом ограничений.

Подход: Нужно экспериментировать по трём направлениям: **модель, промпт, архитектура (harness)**.

📈 Сравнение подходов:

1. **Базовая модель (Sonnet) + простой промпт:** ❌ Все тесты провалены. Много токенов, плохие результаты.
2. **Более мощная модель (Opus) + тот же промпт:** 📉 Нарушений стало меньше, но все тесты ещё провалены.
3. **Opus + Adaptive Thinking:** ✅ Все тесты пройдены. Но ⚠️ **высокая стоимость (в 3 раза больше токенов) и задержка (латентность)**.
4. **Sonnet + улучшенный промпт** (с инструкцией «проверь свою работу»): ➖ Лучше, но нестабильно. Упирается в лимит токенов.
5. **Агентский цикл «Сгенерировать-Оценить-Исправить» (Generate-Evaluate-Repair):**
    - **Генератор:** Создаёт черновик графика.
    - **Оценщик (LLM-судья):** Проверяет черновик на нарушения правил и формирует отчёт.
    - **Исправитель:** Вносит целевые правки на основе отчёта.
    - **Итог:** ✅ Все тесты пройдены. **Оптимальный баланс токенов и латентности.**

🚀 Преимущества агентского подхода:

- **Гибкость:** Позволяет добавлять «мягкие» ограничения (например, «Гарри не любит работать с Салли») прямо в промпт оценщика на лету, без изменения кода.
- **Разделение ответственности:** Каждый этап решает свою чёткую задачу, что упрощает отладку и улучшение.

🎓 Выводы

1. **Эвалюации — это основа** для объективной оценки любых изменений в промпте или модели.
2. **Начинайте с «гигиены»:** Структурирование и очистка промпта часто дают быстрый прирост качества.
3. **Избегайте длинных «запретительных списков»** и однобоких инструкций. Давайте сбалансированные указания.
4. **Инструкции ≠ возможности.** Для сложных задач (расчёты, поиск) предоставляйте модели **инструменты (tools)**.
5. **Для сложных use-case рассмотрите агентскую архитектуру.** Разделение на этапы (генерация, оценка, исправление) может быть эффективнее и гибче, чем один монолитный промпт.
