🤖 Выбор LLM-модели: от бенчмарков к успешному результату

🎯 Выбор правильной модели: от теории к практике

Ключевые тезисы:

- ✅ Публичные бенчмарки дают лишь общее направление, но не заменяют собственные эвалюации (evals) под конкретный кейс.
- 💰 Правильная модель — не та, что дешевле за токен, а та, что дешевле за **успешный результат**.
- 🎛️ Используйте «ручки» (thinking, effort, prompt caching, контекстный инжиниринг) для тонкой настройки компромисса между стоимостью, качеством и латенцией.

📊 Три столпа выбора модели

При выборе модели для продукта нужно оценивать три ключевых параметра:

1. **Качество модели** — точность и процент успешного выполнения задачи.
2. **Латентность** — критично для пользовательских (customer-facing) сценариев.
3. **Стоимость** — ключевой фактор для многих проектов.

🔬 Создание собственной эвалюации (Eval)

Публичные бенчмарки (SWE-bench, MMLU) не отражают специфику вашей рабочей нагрузки. Необходимо строить приватные эвалюации.

Эвалюация — это набор задач (atomic unit), каждая из которых содержит:

- Входные данные (input)

- Критерии успеха (success criteria)

Аналогия: Эвалюация как школьный экзамен по математике. Важен не только правильный ответ, но и **ход решения**. Для агентских задач нужно проверять и конечный результат, и правильность промежуточных шагов.

Способы проверки:

- **LLM как судья (judge):** Гибкая проверка итогового ответа или корректности шагов (например, SQL-запросов).
- **Детерминированные проверки (code-based):** Точная проверка обязательных действий (например, вызов конкретного инструмента).

> 💡 *Создание репрезентативного набора тестовых данных — одно из самых эффективных вложений человеческого времени в мире автоматизации с помощью ИИ.*

⚠️ Типичные ошибки при построении эвалюаций

1. **Шум вместо сигнала:** Если результаты сильно «плавают» при повторных запусках, возможно, задача плохо определена или критерии оценки не выровнены.
2. **Инфраструктурные сбои:** Падение метрик может быть вызвано ошибками API или инструментов, а не плохой работой модели. **Важно анализировать логи (transcripts)**, чтобы отделить инфраструктурные проблемы от проблем модели.
3. **«Тихое насыщение» (Silent saturation):** Набор данных должен отражать реальные запросы из продакшена. Необходимо создавать петлю обратной связи: собирать трейсы, анализировать ошибки и добавлять их в эвалюацию.
4. **Особенности моделей:** Каждая модель (даже разные версии Claude) имеет нюансы. При смене модели необходимо **читать руководства по промптингу** и корректировать промпты.

🔍 Важность анализа логов (Transcripts)

Обязательно настройте удобную observability (LangSmith, BrainTrust и др.), чтобы видеть:

- Системные промпты

- Вызовы инструментов агентом
- Результаты инструментов
  Только «закопавшись» в логи, можно обнаружить реальные паттерны и ошибки (например, модель, подсматривающая ответ из истории предыдущих попыток).

🎛️ Инструменты для управления компромиссами

Используйте параметры Claude для тонкой настройки:

- ***Thinking*** (Мышление): Даёт модели «черновик» для размышлений перед действием (System 2 thinking). Может быть адаптивным (модель решает сама) или фиксированным.
- ***Effort*** (Усилие): Указывает модели, сколько «работы» вложить в задачу (влияет на длину reasoning, tool calls и ответов).

Контр-интуитивный вывод: Более умные модели (Opus) могут выполнять задачи **быстрее** и с **меньшим числом токенов**, так как действуют стратегически и делают меньше «лишних» шагов.

🚀 Сдвиг кривой эффективности

Можно не просто двигаться по кривой «качество-стоимость», а сдвинуть её целиком.

1. 📦 **Кэширование промптов (Prompt Caching):**
   - Сохраняемый префикс промпта стоит **в 10 раз дешевле**.
   - Позволяет получить качество Opus по цене Sonnet, а Sonnet — по цене Haiku.
   - **Стратегия:** Используйте подход «append-only». Системный промпт должен быть **неизменным (immutable)**, динамические данные добавляются только в конец.

2. 🧹 **Контекстный инжиниринг (Context Engineering):**
    - Оптимизируйте ответы инструментов, которые передаются модели.
    - **Примеры оптимизаций:**
        - Используйте Markdown вместо JSON.
        - Упрощайте форматы дат.
        - Добавляйте полезные метаданные (день недели).
        - Дедуплицируйте данные (например, статьи из веб-поиска).
    - **Результат:** Сокращение токенов на 65-77%, снижение стоимости и часто — **повышение точности** модели за счёт более чистых данных.

📈 Практический воркшоп: запуск sweep-эвалюации

Цель: Запустить эвалюацию (на примере Tao Bench для авиа-агента) с разными конфигурациями:

- **Модели:** Haiku, Sonnet, Opus
- **Параметры:** thinking on/off, разный уровень effort

Результаты (на примере прогона) наглядно показывают на графиках:

- Pass rate vs. выходные токены
- Pass rate vs. стоимость
- Pass rate vs. латентность

Это позволяет принять **взвешенное решение**, основанное на данных, а не на интуиции.

🎯 Ключевые выводы

1. **Инвестируйте в эвалюации:** Небольшой, но хорошо спроектированный приватный eval даст больше, чем любой публичный бенчмарк.
2. **Оптимизируйте за успех:** Выбирайте модель с минимальной стоимостью **успешного исхода** (cost per successful outcome).
3. **Используйте все «ручки»:** Комбинируйте выбор модели, thinking, effort, prompt caching и контекстный инжиниринг для точного позиционирования на границе Парето или её сдвига. Это открывает доступ к более умным моделям и новым сценариям в рамках бюджета.
