ПРАКТИЧЕСКОЕ РУКОВОДСТВО XXI ВЕКА

ИИ без МИФОВ

Как понимать, применять и не бояться искусственного интеллекта

- Экономия времени до 60%
- Решение реальных задач
- Защита от ошибок и рисков
- Готовые шаблоны и промпты
- Автоматизация и агенты
- Измеримый результат (кейсы и цифры)



- ✓ Понятно
- ✓ Практично
- ✓ С примерами
- ✓ Можно применять уже сегодня

Подходит для руководителей, специалистов и всех, кто хочет эффективно работать с ИИ

Чернова Галина

ИИ без мифов

«Автор»
2026

Галина Ч. М.

ИИ без мифов / Ч. М. Галина — «Автор», 2026

«ИИ без мифов: как понимать, применять и не бояться искусственного интеллекта» — практическое руководство для тех, кто хочет перестать бояться искусственного интеллекта и научиться использовать его с реальной пользой.Как работает ИИ? Почему он ошибается? Каким ответам можно доверять, а что обязательно нужно проверять? Как экономить время, автоматизировать задачи и получать измеримые результаты?В книге — реальные рабочие сценарии, практические кейсы, готовые промпты, шаблоны, чек-листы и разбор типичных ошибок. Без сложных формул, технического тумана и обещаний чудес.Это не книга о далёком будущем. Это руководство о том, как уже сегодня превратить искусственный интеллект в эффективный инструмент своей работы, мышления и принятия решений.Меньше мифов. Больше понимания. Больше возможностей.

© Галина Ч. М., 2026
© Автор, 2026

Содержание

Аннотация 6
Как читать эту книгу 7
Живой слой: почему эта книга не устареет 8
От автора 9
Форзац: три шпаргалки книги 10
Оглавление 14
ЧАСТЬ I. ИИ без паники 15
Глава 1. С чего начать: матрица решений и первый результат за 20 минут 16
Глава 2. Что ИИ умеет на самом деле — и чего не умеет 27
Глава 3. Мифы, которые мешают применять ИИ 37
Глава 4. Как думать об ИИ: задачи, данные, результат 47
ЧАСТЬ II. Как это работает без формул 57
Глава 5. Модель — это не база знаний, а функция предсказания 58
Конец ознакомительного фрагмента. 68

Чернова Галина
ИИ без мифов

ИИ без мифов
как понимать, применять и не бояться искусственного интеллекта
Практическое руководство для всех, кто принимает решения
Первое издание

Аннотация

Вы научитесь понимать, как на самом деле работает искусственный интеллект — без формул, но и без сказок. Вы перестанете бояться ИИ и верить обещаниям, что он «заменит всех». Вы сможете выбирать задачи, где ИИ действительно экономит время, писать рабочие промпты, проверять результаты, оценивать риски и говорить с техническими специалистами на одном языке.

Эта книга — мост между хайповыми обещаниями и серьёзной инженерией. В ней: 18 глав по 12–15 страниц каждая, и в каждой — развёрнутый кейс с цифрами, разбор ошибок и процедура, которую можно применить завтра; 50+ практических примеров с измеримыми результатами и честными ограничениями; 26 готовых шаблонов — от промптов для ежедневной работы до JSON-схем и кода для автоматизации, каждый с примерами «вход» и «выход»; отдельное приложение с десятью антипаттернами, которые сводят пользу от ИИ к нулю; разделы о моделях рассуждений, контекстной инженерии, структурированном выводе и мультиагентных конвейерах — том, что определяет практику последних двух лет; врезки «Под капотом» для профессионалов; глоссарий на 100 терминов; живой репозиторий, где шаблоны и схемы обновляются под новые версии моделей.

На форзаце — три шпаргалки, к которым вы будете возвращаться чаще, чем к любой главе: матрица решений «сразу / с проверкой / нельзя», конструктор промпта из шести блоков и алгоритм диагностики ошибки модели.

Уже в первой главе вы получите два инструмента с немедленной отдачей: матрицу «что отдавать ИИ, а что нельзя» и рабочий шаблон выписки или сводки, который экономит 40–60 минут на первом же документе. Через день-два после начала чтения вы соберёте свой первый сценарий. Через неделю — встроите его в ежедневную практику. Через месяц у вас будет система, а не набор случайных запросов.

Как читать эту книгу

Книга устроена так, чтобы её можно было читать с любой глубины и в любом порядке — но с одним условием: **выполняйте действия**. Каждая глава заканчивается коротким блоком «Что сделать сегодня»; именно он, а не количество прочитанных страниц, определяет результат.

Три уровня чтения.

• **Основной текст** — для всех. Истории, принципы, примеры, цифры.

• **Врезки «Под капотом»** — для тех, кому важно, как это устроено: токенизация, внимание, RAG, кэширование контекста, оркестрация агентов. Их можно пропускать без потери логики.

• **Врезки «Красные флаги»** — для всех, кто не хочет наступать на грабли, на которые уже наступили тысячи людей и компаний.

Три маршрута.

Кто вы	**Что читать в первую очередь**	**Что получите**
Новичок без технического бэкграунда	главы 1, 2, 4, 8, 10, 13	рабочий сценарий и понимание границ
Руководитель, принимающий решения	главы 1, 4, 12, 14, 17, 18	язык для разговора с командой и метрики внедрения
Инженер, продакт, аналитик	главы 5–9, 14, приложение с шаблонами	архитектурные принципы и шаблоны для автоматизации

Два вида кейсов. В книге честно размечены примеры: «собирательный кейс» — типичная ситуация, собранная из практик и наблюдений, и «публичный кейс» — случай, о котором есть открытые источники. Мы не выдаём первое за второе: это принципиально для книги, которая учит проверять.

Живой слой: почему эта книга не устареет

Главный аргумент против книг об ИИ звучит так: «через три месяца всё изменится». Аргумент сильный, и он верен ровно наполовину.

Меняется **погода**: версии моделей, цены за токен, названия кнопок, лимиты контекста. Она меняется каждые несколько недель, и никакая бумажная книга за этим не успеет. Не меняется **климат**: модель предсказывает текст, а не ищет истину; она не знает вашего контекста, пока вы его не дали; рассуждение снижает ошибки в шагах, но не в исходных данных; ответственность за решение остаётся на человеке. Именно климат и составляет содержание этой книги.

Поэтому у книги есть два слоя.

Слой
Что в нём
Как обновляется

Книга (бумага и электронная версия)
принципы, методология, кейсы, разбор ошибок
новое издание раз в 2–3 года

Живой слой (репозиторий и приложение)
шаблоны, JSON-схемы, примеры кода, чек-листы, актуальные ограничения моделей
каждые 2 месяца

Что даёт живой слой. Не «ссылки на сайт», а рабочую инфраструктуру: готовые промпты с историей изменений, схемы структурированного вывода, чек-листы приёмки, таблицу «какая модель что умеет на сегодня», разборы новых релизов с точки зрения практики — и то, что важнее всего, — указание, **что именно изменилось** после выхода новой версии модели.

Как этим пользоваться. В конце глав и в приложении стоят ссылки вида #ch05 — это адрес раздела в репозитории. QR-код ведёт туда же; он есть в каждой главе электронной версии и на вклейке печатного издания. Если вы читаете бумажную книгу без телефона под рукой — откройте адрес позже; шаблоны в приложении работают и без обновлений.

Чего живой слой не делает. Он не заменяет книгу и не «обучает вас за вас». Он снимает единственную причину, по которой хорошая книга по ИИ устаревает: расхождение между принципом и текущей версией инструмента.

От автора

Эта книга родилась из разговора с врачом.

Я спросил его: «Вы пользуетесь ИИ?» Он ответил: «Нет. Это сложно, да и зачем — всё равно всё заменит робот». Через пять минут он показал мне Excel-таблицу, куда вручную вносил данные пациентов из чеков, писем и телефонных блокнотов. Задача, которую ИИ решает за минуты, съедала у него час в день.

А потом я говорил с разработчиком, который уверенно объяснял, что языковые модели «на самом деле ничего не понимают», — и тут же доверял модели выбрать стек сборки и сборочные зависимости для проекта. И с руководителем отдела маркетинга, который писал промпты длиной в абзац, как магические заклинания, и каждый раз получал одну и ту же воду вместо текста.

Все трое ошибались по-разному. Но их ошибки были одного типа: они принимали решения на основе мифов. Первый решил, что ИИ — это магия, которая «либо всё заменит, либо ничего не умеет». Второй — что понимание устройства модели освобождает от проверки результатов. Третий — что длина промпта заменяет его качество.

Мифы одинаково дорого обходятся новичкам и профессионалам. Новички не используют то, что могло бы им помогать. Профессионалы строят на ИИ системы, которые позже разваливаются от первого же ветра. Руководители вкладывают деньги в те проекты, где ИИ даёт эффект, и одновременно тратят на те, где он даёт только презентации.

Поэтому я написал эту книгу не «про ИИ вообще», а про другое: **как человеку принимать лучшие решения с помощью ИИ**.

Здесь мало математики. Но здесь есть принципы, которые переживут любую смену моделей и инструментов. Есть кейсы из США, Европы, Азии и глобальных компаний — с цифрами времени, денег и ошибок, и с честным описанием того, что в этих кейсах не сработало. Есть шаблоны, которые вы будете применять до тех пор, пока инструменты не прошли свой полный цикл.

Книга устроена так, чтобы вы могли читать её с любой глубины. Основной текст — для всех. Врезки «Под капотом» — для тех, кто хочет знать, как это устроено изнутри. Врезки «Красные флаги» — для всех, кто хочет не наступить на грабли, на которые уже наступили тысячи людей и компаний.

Читайте с карандашом. Выполняйте упражнения. И помните главное правило этой книги: **ИИ — это не участник вашего разговора. Это инструмент вашего решения.**

Пусть эта книга сэкономит вам хотя бы один час в неделю. Тогда она окупила себя быстрее любой инвестиции.

Форзац: три шпаргалки книги

Три страницы, к которым возвращаются чаще, чем к любой главе: что поручать модели, как формулировать запрос и что делать, когда ответ слабый. Каждая шпаргалка рассчитана на одну распечатанную страницу — один лист, одна тема, ничего лишнего — и на шестьдесят секунд чтения в момент решения, а не «до книги».

Принцип у всех трёх один. Модель — не источник истины и не калькулятор, а исполнитель формулировки: чем точнее заданы задача, данные и критерий приёмки, тем предсказуемее результат. Погода в виде версий и цен меняется каждые несколько недель; порядок, изложенный ниже, держится годами и не требует обновления.

Как пользоваться форзацем.

- Сначала найдите свою задачу в первой таблице и посмотрите вердикт. Это ответ на вопрос «можно ли вообще».
- Потом соберите запрос по шести блокам. Это ответ на вопрос «как попросить».
- Если ответ уже пришёл и слабый — идите по дереву в третьей шпаргалке, сверху вниз, до первого совпадения.

Шпаргалка 1. Матрица решений

Одна таблица на десять строк. Читайте её не как разрешение, а как распределение вашего внимания: зелёная половина экономит часы, красная — деньги и репутацию.

Вердиктов три, и они отличаются не степенью доверия к технике, а ценой ошибки. **Сразу** — ответ читают на бегу, потому что ошибка здесь дешевле её проверки. **С проверкой** — ответ принимают после выборки, а не после того, как он понравился. **Нельзя** — задача меняет форму: считается калькулятором, берётся из первоисточника, подписывается человеком.

Задача	Как поступить с результатом
Пересказ, сводка, выжимка из документа	**сразу**: переупаковка текста — родная операция модели; сверьте числа, имена и даты с источником
Черновик письма, поста, объявления	**сразу**: структура и тон даются быстро; обещания, сроки и факты о продукте остаются на вас
Объяснение темы на нужном уровне	**сразу**: модель уверенно меняет глубину и словарь; объяснение не равно факту
Идеи, варианты, названия, планы	**сразу**: генерация вариантов дешева; выбор всегда за вами
Перевод	**с проверкой**: термины, имена и юридические формулировки сверяйте по словарю, а не на слух
Таблица из текста, PDF, переписки	

с проверкой: выборочно 5–10 строк против источника, отдельно — единицы измерения и пустые поля

Классификация заявок, писем, тикетов
с проверкой: разбирайте вручную всё, что попало в «прочее» и «срочное»

Ревью кода и черновик тестов
с проверкой: тесты запустить; код без прогона не принимать, даже если правки выглядят очевидными

Расчёты, сметы, налоги, дозировки; свежие факты
нельзя: модель предсказывает числа, а не считает, и не знает того, что появилось после даты обучения; калькулятор, таблица, первоисточник с датой

Юридическое заключение, диагноз, подпись, решения о людях, персональные данные
нельзя: ответственность не делегируется, а публичный чат — чужая система; модель готовит черновик, решение, данные и подпись остаются у человека

Как читать матрицу. Смотрите не на модель, а на цену ошибки. Если ошибку видно за десять минут и она дешевле проверки — отдавайте сразу. Если ошибка уходит наружу: в деньги, в документ с подписью, к клиенту или пациенту, — проверка идёт по первоисточнику, с цитатой и адресом. Если проверить нельзя в принципе, это не задача для модели в текущем виде, и такое решение нормально.

Средняя проверка из строки «с проверкой» стоит около двух минут на ответ: одно утверждение из пяти читается целиком, а все числа, даты, имена и номера — полностью и всегда. Это не недоверие к технологии, а арифметика: две минуты против часа переделки и разговора о том, кто виноват.

Шпаргалка 2. Конструктор промпта из шести блоков

Каждый блок закрывает дыру, в которую иначе зальётся правдоподобный текст. У модели нет режима «мне не хватает данных»: продолжение возможно всегда, поэтому недостающее она достраивает молча.

Порядок блоков не случайный. Роль сужает словарь, задача задаёт результат, данные отсекают вымысел, формат делает ответ проверяемым машиной, критерии дают право отказать, примеры переносят то, что вы не можете описать словами. В конвейере первые два блока ещё и дешевеют: неизменный префикс инструкции тарифицируется по пониженной ставке, поэтому дату и имя клиента ставьте в конец, а не в начало.

Блок
Вопрос, на который он отвечает
Пример формулировки

Роль
Кто говорит и каким словарём
«Ты B2B-маркетолог, семь лет пишешь для снабженцев заводов»

Задача
Что именно сделать — один глагол, один результат
«Перепиши письмо: первый абзац ссылается на встречу 14 марта»

Данные
Чего модель не может знать сама
блок ### ДАННЫЕ: линейки из трёх высот, цены, срок шесть недель, возражение клиента дословно

Формат
Как выглядит готовый ответ
«180–200 слов, тема до 50 знаков, один призыв к действию, без списков»

Критерии приёмки
По какому признаку вы откажете ответу
«каждая цифра из ДАННЫХ; слова „инновационный“ и „комплексный“ не использовать; если данных нет — так и написать»

Примеры
Как выглядит хороший результат на вашем материале
три ваших письма, на которые ответили клиенты

Универсальный шаблон. Шесть строк, которые копируются в любой чат. Пустые места заполняются, а не пропускаются: пропуск — это и есть та дыра, куда заливается вымысел.

- **Роль:** ты [роль, опыт, для кого пишешь].
- **Задача:** [один глагол]: [один результат].
- **Данные:** ### ДАННЫЕ — [цифры, сроки, ограничения, дословные возражения].
- **Формат:** [объём] слов, [структура], [чего быть не должно].
- **Критерии:** каждая цифра из ДАННЫХ; при отсутствии данных написать «нет данных»; [запрещённые слова].
- **Примеры:** [вход → выход], два-три ваших образца.

Проверка блока занимает тридцать секунд. Покажите промпт коллеге и спросите, что именно он сдаст и по какому признаку поймёт, что сдал. Если его ответ расходится с вашим, промпт не готов, и модель здесь ни при чём.

Шесть блоков — не бюрократия. Полный промпт выходит на 400–900 слов, и это дешевле, чем двенадцать итераций в чате: каждая из них стоит прогона и вашего внимания, а спецификация пишется один раз.

Шпаргалка 3. Алгоритм диагностики ошибки модели

Порядок шагов важнее их содержания: сначала исключаются дешёвые причины, и только потом меняется модель. Правило одно — не менять две вещи одновременно, иначе не узнаете, что помогло.

Нулевой шаг. Назовите дефект. «Плохо» — не дефект. Дефект — это неверное число, пропущенный раздел, чужой тон, ответ на другой вопрос. Если назвать нечего, проблема не в модели, а в критерии приёмки.

Дальше идите по дереву сверху вниз и остановитесь на первом совпадении. Ведите короткий журнал: одна правка — один прогон, строка «что изменил → что стало». Через три итерации видно, какая причина была настоящей, а какая померещилась, и вы перестаёте чинить формат там, где не хватало данных.

1. **Не те данные.** Если в ответе факты, которых вы не давали, или перепутаны единицы и термины — вынесите данные в блок ### ДАННЫЕ, отделите их от инструкций тегом и запретите добавлять факты вне блока.

2. **Нет критерия.** Если ответ нельзя ни принять, ни отклонить без спора о вкусе — впишите три проверяемых условия приёмки и правило на случай отсутствия данных: написать, что данных нет, а не додумать.

3. **Задача слишком широкая.** Если в ответе всё понемногу и ни один пункт не доведён до конца — оставьте один глагол на вызов и разбейте работу на два-три шага, каждый со своим форматом и своей приёмкой.

4. **Модель не знает факта.** Если номера, даты и ссылки выглядят гладко, но их нет в источнике, или речь о событиях после даты обучения — приложите первоисточник, потребуйте цитату с адресом и разрешите прямой отказ «нет данных». Без источника ответ остаётся гипотезой и в документ не идёт.

5. **Конфликт инструкций.** Если выполнена половина задания или нарушено правило, которое вы задавали выше, — найдите противоречие между системной инструкцией, данными и прошлыми сообщениями; оставьте одно правило и задайте приоритет: системная инструкция сильнее истории диалога, а данные не являются инструкциями.

6. **Нужен расчёт или инструмент.** Если шаги выглядят логично, а сумма не равна слагаемым и единицы не сходятся — подключите калькулятор, таблицу или код, просите формулу и шаги, а не готовое число, и пересчитывайте результат отдельно.

Если после шести действий ответ остался слабым, дело уже не в формулировке: нужен поиск по вашим документам или сборка конвейера из глав 6 и 9. Это другой ремонт и другая цена, и он начинается с вопроса «где лежит источник», а не «какую модель выбрать».

Проверка на сомнение. Спросите себя, чего не хватило бы коллеге, чтобы выполнить эту задачу без вас. Не хватило фактов, цифр и сроков — пункт 1. Неясно, что считается готовым ответом, — пункт 2. Задача распадается на три разные — пункт 3. В ответе номера и даты, которых нет в источнике, — пункт 4. Правила спорят друг с другом — пункт 5. Ответ должен быть числом — пункт 6.

Оглавление

ЧАСТЬ I. ИИ без паники 13
Глава 1. С чего начать: матрица решений и первый результат за 20 минут 14
Глава 2. Что ИИ умеет на самом деле — и чего не умеет 28
Глава 3. Мифы, которые мешают применять ИИ 41
Глава 4. Как думать об ИИ: задачи, данные, результат 53
ЧАСТЬ II. Как это работает без формул 66
Глава 5. Модель — это не база знаний, а функция предсказания 67
Глава 6. Данные, обучение и ограничения 80
Глава 7. Почему ИИ ошибается и что такое галлюцинации 94
Глава 8. Промпт как интерфейс: принципы, а не магия 109
Глава 9. RAG, агенты и автоматизация — простыми словами 124
ЧАСТЬ III. Практика 138
Глава 10. Личная продуктивность: текст, анализ, планирование 139
Глава 11. Работа с информацией: поиск, суммаризация, проверка фактов 153
Глава 12. Профессиональные кейсы: восемь отраслей 167
Глава 13. Как построить свой рабочий сценарий за 30 минут 181
Глава 14. Как оценивать качество и стоимость решения 195
ЧАСТЬ IV. Риски и внедрение 209
Глава 15. Приватность, безопасность и авторское право 210
Глава 16. Предвзятость, этика и ответственность 224
Глава 17. Как внедрять ИИ в команде и не сломать процессы 238
Глава 18. Что будет дальше: принципы, которые переживут инструменты 251
Приложение А. Шаблоны промптов 264
Приложение Б. Десять антипаттернов 305
Финальное обращение к читателю 312
Глоссарий 313
Источники и литература для углубления 319

ЧАСТЬ I. ИИ без паники

Глава 1. С чего начать: матрица решений и первый результат за 20 минут

Паспорт главы. Обещание: вы разложите свои рабочие задачи по трём вердиктам — «сразу», «с проверкой», «нельзя» — и получите один сценарий, который даёт измеримую экономию в первый же день. Время чтения: 12 минут. Результат: заполненная матрица на пять своих задач и сводка документа за 8–12 минут вместо 40–60.

Четверг, 20:40, Лиссабон

Ана — терапевт, 46 лет, четвёртый год ведёт приём в клинике в центре Лиссабона. В четверг у неё было четырнадцать пациентов, а в 20:40 она всё ещё сидела над третьей выпиской за вечер: четыре страницы истории болезни, два анализа из разных лабораторий, назначения, которые менялись трижды за полгода.

Выписка — не творчество. Это перенос фактов из одного формата в другой: жалобы, анамнез, препараты с дозами, результаты анализов, план, дата следующего визита. Работы на 40 минут, если делать внимательно. Пять таких выписок в день — три с половиной часа, которых в расписании приёма нет.

При этом Ана не считала себя человеком, который боится технологий. Она пользовалась электронной картой, поиском по справочнику диагнозов, калькулятором рисков. Она просто не видела, где именно в её дне ИИ возьмётся за дело, а где создаст проблему, которую придётся разбирать ночью.

Эта глава — про то, как за двадцать минут получить от ИИ первый измеримый результат. И про то, как не повторить ошибку, которую Ана сделала на второй день.

Кейс: сорок минут, чужая дозировка и один шаблон

Проблема. Пять выписок в день по 40 минут — это 66 часов в месяц. Добрая половина этих часов приходилась на вечера и субботы, потому что днём между пациентами выписку не собрать: нужен непрерывный кусок внимания, а его нет. За месяц Ана ошибалась в выписках два-три раза — не в диагнозе, а в переносе: пропущенная строка анализов, съехавшая дата, доза, записанная дважды. Каждая такая ошибка стоила звонка, извинения и двадцати минут репутации.

Ошибочная попытка. В субботу она открыла чат, вставила историю болезни целиком — четыре страницы, около двух тысяч слов — и написала: «Перескажи всё, что важно для выписки, и подскажи дозировки». Ответ пришёл через двадцать секунд. Аккуратный текст на триста слов, ровные формулировки, правильный порядок разделов, ни одного лишнего слова.

Первые две выписки она собрала за двенадцать минут и решила, что задача закрыта.

На третьей её остановила медсестра. В выписке стояло «бисопролол 10 мг утром», в карте — «бисопролол 5 мг утром и 2,5 мг вечером». Модель не выдумала препарат из воздуха. Она свела две строки в одну, потому что в текстах, на которых её учили, такая доза встречается чаще и выглядит аккуратнее. К этому моменту две выписки уже ушли в архив.

Разбор занял 40 минут: сверить все препараты по исходнику, позвонить в аптеку, переписать документ, объяснить коллеге, почему в системе стоит неверная доза. Пациент ничего лишнего не получил. Доверие к собственному вечернему эксперименту — получил удар.

Правильный подход. Ана не отказалась от инструмента. Она изменила задачу и добавила к ней три вещи, которых в первом запросе не было.

Первое — границы данных. В шаблоне появилась строка: «Используй только текст истории болезни. Если для поля нет данных — напиши „нет в источнике“». Второе — запрет на преобразование чисел: «Каждую дозу, дату и результат анализа копируй дословно. Не пере-

считывай, не объединяй, не нормализуй». Третье — формат из двенадцати фиксированных полей в фиксированном порядке, включая поле «аллергии» и поле «открытые вопросы».

Отдельным шагом она вынесла проверку из головы в процедуру: перед сохранением — тридцать секунд поиска по исходнику для каждой дозы и даты. Не «посмотреть внимательно», а именно поиск строки.

На второй неделе выяснилось, что шаблон нельзя держать в голове: у двух клиник разные форматы выписок, и в одной из них поле «план» называется «рекомендации». Ана сохранила шаблон в заметках двумя версиями и перестала переписывать его каждый раз. Побочный эффект оказался важнее основного: когда ограничения записаны, их проще показать коллеге. Через месяц тем же шаблоном пользовались ещё двое врачей, и один из них добавил поле, которое Ана пропустила, — «дата последней вакцинации».

Итог в цифрах. Три недели, 30 выписок. Среднее время — 11 минут вместе с проверкой против 40 минут до. Расхождений в числах и дозах при выборочной сверке — ноль. Экономия — 29 минут на выписку, около 48 часов в месяц при её объёме работы.

Ограничение. Работало не всё. В четырёх случаях из тридцати (13 %) у пациента было три-четыре сопутствующих диагноза, и модель сводила хронологию в одну строку, теряя последовательность «когда что началось». Эти четыре выписки Ана переписала вручную. Ещё одна правка: в шаблоне пришлось отдельно потребовать копировать строку аллергий целиком, даже если она пустая, — иначе модель просто выбрасывала поле. Сложные случаи остались за человеком, и это не недоработка шаблона, а его граница. *(собирательный кейс)*

Разбор кейса. Почему это сработало — механика, а не мораль.

1. Задача превратилась из «понять и пересказать» в «перенести с ограничениями». Пересказ — зона свободы модели; перенос по полям — зона дисциплины.

2. Числа вывели из-под модели: их запретили преобразовывать, оставив только копирование. Дозы перестали быть тем, что модель «сглаживает».

3. Появился критерий приёмки: двенадцать полей, формулировка «нет в источнике», дословные числа. Такой результат можно принять или отклонить, не споря о вкусе.

4. Проверка стала шагом процедуры, а не настроением. Тридцать секунд поиска по строке дешевле сорока минут разбора.

5. Граница проведена заранее: случаи с сопутствующими диагнозами не отдаются машине. Это дешевле, чем ловить потерянную хронологию задним числом.

Погода и климат: что меняется каждые недели, а что держится годами

Спор об ИИ устроен так, что в нём легко утонуть. Каждую неделю — новая модель, каждый месяц — заголовок о том, что очередная профессия отменяется, каждый квартал — новая цена за миллион токенов. За этим шумом стоит простая структура, и она экономит читателю месяцы внимания.

Погода — то, что меняется за недели. Названия моделей. Цена за миллион входных токенов: сегодня $10, через полгода может быть $2 или модель исчезнет из каталога. Размер окна контекста — сколько текста модель удерживает за один разговор. Наличие поиска, работа с файлами, лимиты на длину ответа, кнопки в интерфейсе. Всё это важно для выбора инструмента и бесполезно как основание для решения.

Климат — то, что держится годами и не зависит от версии. Модель предсказывает следующий фрагмент текста, а не ищет истину, поэтому её уверенный тон ничего не говорит о правильности. Она не знает вашего контекста, пока вы его не дали: ни вашу клинику, ни ваш договор, ни вашу терминологию. Качество ответа упирается в качество поданных данных, а не в количество эпитетов в запросе. Ответственность за результат остаётся на человеке, который

ставит подпись. И память у неё короткая: диалоговая память — это только текущее окно разговора, вчерашний чат модель не помнит.

Эта книга — про климат. Поэтому она начинается не с истории технологии и не с обзора моделей, а с рабочего инструмента: таблицы, по которой вы за минуту решаете, стоит ли отдавать конкретную задачу ИИ.

Практический смысл разделения простой. Погодные решения принимаются быстро и не стоят ничего: попробовать новую версию, сравнить цену, переключить инструмент. Климатические решения принимаются один раз и живут долго: какие задачи вообще отдавать машине, что считается проверенным результатом, кто подписывает. Если вы путаете уровни, вы каждый месяц пересматриваете климатические решения из-за погодных новостей — и никогда не доводите до конца ни одного сценария. Наоборот, если у вас есть климатические правила, смена модели становится рутиной: вы просто проверяете, держит ли новая версия тот же уровень ошибок.

Разговор об ИИ при этом обычно сваливается в одну из двух крайностей: «оно заменит всё» или «это просто автодополнение». Обе формулировки — не понимание, а способ снять тревогу: первая снимает страх опоздать, вторая — страх оказаться лишним. Полезнее третья рамка, самая скучная: ИИ — распространённая и очень специфичная технология. Как электричество. Не магия, но если знать, где в вашей работе находятся розетки, картина дня меняется.

Три вопроса, которые закрывают 90 % решений

Чтобы принимать решения, достаточно ответить на три вопроса. Остальное — новости, споры про авторские права, разговоры про «когда появится общий интеллект» — можно читать как погоду.

1. **Что ИИ умеет структурно?** Не «что он умеет в этом месяце», а какой тип работы у него получается лучше всего и почему. Ответ короткий: переупаковка готового текста — сжатие, пересказ, смена тона, смена языка, извлечение полей, классификация по заданным категориям, генерация вариантов. Всё это операции над текстом, а текст — среда модели.

2. **Где он структурно слаб?** Не «пока слаб», а «будет слаб и через пять лет — вот почему». Арифметика, точность цитат, свежие факты после даты обучения, отсутствие вашего контекста, отсутствие ответственности. Причина одна: модель предсказывает вероятное продолжение, а не проверяет факт.

3. **Как разделить работу?** Какие части задачи отдать машине, а какие оставить себе. Готовить материал — машине. Принимать решение и подписывать — человеку.

Дальше в книге вы будете отвечать на эти вопросы для своей работы десятки раз, и каждый раз ответ будет точнее. В главе 4 они разложены на три слова: задача, данные, результат.

Матрица решений: «сразу», «с проверкой», «нельзя»

Ключевые задачи — повторяющиеся операции, где эффект можно измерить: сводки, письма, таблицы, разборы. Начинать стоит не со всего списка дел, а с них: именно ключевые задачи первыми попадают в матрицу и дают первые часы экономии. Разовые, творческие и ответственные решения оказываются в ней только для того, чтобы честно увидеть: их место — у человека.

Вердиктов три. **Сразу** — модель делает черновик, вы принимаете его и идёте дальше. **С проверкой** — результат идёт в работу только после сверки по чек-листу; чек-лист здесь — короткий список пунктов, по которому вы проверяете результат перед сдачей, а не документ на пять страниц. **Нельзя** — решение остаётся за человеком, модель может лишь готовить материал.

Выбор вердикта — это и есть оценка рисков, только в бытовом виде: вы сравниваете цену ошибки с ценой проверки. Если проверка стоит две минуты, а ошибка — сорок минут разбора, вердикт очевиден.

Задача
Вердикт
Почему
Что проверить

Пересказ и сводка документа
сразу
переупаковка текста — родная операция модели
числа, имена, даты — сверять с источником

Черновик письма, поста, объявления
сразу
модель быстро даёт структуру и тон
обещания, сроки, факты о продукте

Перевод
с проверкой
качество высокое, но термины и стиль — ваши
термины, имена, юридические формулировки

Таблица из текста, PDF, переписки
с проверкой
извлечение полей — сильная сторона
выборочно 5–10 строк против источника

Классификация заявок, писем, тикетов
с проверкой
при ясных категориях работает стабильно
всё, что попало в «прочее» и «срочное»

Объяснение темы на нужном уровне
сразу
модель умеет менять глубину и словарь
не путать объяснение с фактом

Идеи, варианты, названия
сразу
генерация вариантов — дешёвая операция
выбор всегда за вами

Ревью кода и черновик тестов
с проверкой
второе мнение без усталости
запустить тесты; не принимать код без прогона

Задача
Чем рискуете
Что делать вместо

Расчёты, сметы, налоги, дозировки
модель предсказывает числа, а не считает
калькулятор, таблица, профильный специалист

Юридическое заключение, диагноз, подпись
ответственность не делегируется
модель готовит черновик для юриста или врача

Решения о людях: найм, оценка, увольнение
в исторических данных есть перекосы
человек в контуре, критерии, апелляция

Свежие факты: курсы, законы, новости
после даты обучения модель их не знает
первоисточник, дата, официальный сайт

Персональные данные клиентов и пациентов
публичный чат — чужая система
корпоративный контур или анонимизация

Стратегия, приоритеты, «что делать компании»
нужен ваш контекст и ваша ответственность
модель — советник, решение — ваше

Матрица описывает типовые случаи, а не выдаёт индульгенции. «Сразу» не означает «бездумно»: вы всё равно читаете ответ, прежде чем отправить его. «С проверкой» означает, что в процессе есть человек в контуре — обязательный шаг, на котором человек утверждает или отклоняет результат до того, как тот уйдёт дальше.

Правило, которое снимает половину споров: если сомневаетесь — выбирайте более осторожный вердикт. Цена проверки почти всегда ниже цены ошибки. Единственное исключение — когда проверка дороже самой задачи; тогда задачу не отдают модели вообще.

Проверьте матрицу на задаче Аны. «Собрать выписку из истории болезни»: перенос фактов по фиксированным полям, формат известен, критерий приёмки записывается — «с проверкой». «Предложить дозировку»: число, цена ошибки — здоровье, вердикт — «нельзя». «Переписать выписку человеческим языком, чтобы пациент понял»: переупаковка текста, цена ошибки — десять минут правки, вердикт — «сразу». Одна и та же история болезни даёт три разных вердикта, потому что задач в ней три, а не одна. Это самое частое место, где люди ошибаются: они относят к ИИ профессию целиком вместо конкретной операции.

Дерево решений за шестьдесят секунд

Когда таблица под рукой, решение занимает меньше минуты. Пройдите по пунктам сверху вниз и остановитесь на первом сработавшем.

1. Ответ должен содержать число, которое нужно **посчитать**? Если да — таблица, SQL или калькулятор; модель пусть формулирует шаги, а не выдаёт итог.

2. Ответ зависит от события **после даты обучения** — курса, закона, цены, назначения? Если да — первоисточник и поиск, а модель только пересказывает найденное.

3. Ошибка стоит дороже **десяти минут проверки**? Если да — «с проверкой» с явным чек-листом из трёх-пяти пунктов.

4. Результат **подписывает человек** и несёт за него ответственность — диагноз, заключение, приказ, рекомендация клиенту? Если да — «нельзя», модель готовит черновик, решает человек.

5. В задаче есть **персональные данные**, которые нельзя отдавать во внешнюю систему? Если да — закрытый контур или анонимизация до отправки.

6. Ответ можно проверить за две минуты, а цена ошибки — не больше десяти минут переделки? Если да — «сразу».

Если ни один пункт не сработал, вердикт по умолчанию — «с проверкой». Это не осторожность ради осторожности, а арифметика: непроверенный результат, который вернулся с ошибкой, всегда дороже проверенного сразу.

Проверьте дерево на другом примере. Вам нужно свести сорок отзывов клиентов в таблицу с темами и тональностью. Пункт 1: числа считать не нужно, итог — категории. Пункт 2: отзывы ваши, не из внешнего мира. Пункт 3: ошибка в теме отзыва стоит дешевле десяти минут проверки, но пункт 4 срабатывает раньше — по этой таблице вы принимаете решение о продукте, то есть подписываете вывод. Значит, вердикт — «с проверкой»: модель размечает, человек выборочно сверяет двадцать отзывов из сорока и читает всё, что попало в «прочее». Сорок минут ручной разметки превращаются в двенадцать, но не в две — и это правильная цена.

Ваш первый результат: сводка за двадцать минут

Дальше — сценарий, который даёт измеримый результат в первый же день. Возьмите документ, который вам всё равно предстоит прочитать: регламент, договор, отчёт, исследование, инструкцию. Засеките, сколько минут у вас уходит на чтение с выписками. Это ваш baseline — база сравнения, без неё «стало быстрее» ничего не значит.

Теперь — шаблон. Он длиннее, чем «перескажи документ», и это не украшательство: каждый блок закрывает конкретный способ, которым сводка обычно врёт.

Шаблон сводки: роль, данные, ограничения, формат, критерии приёмки.

Роль: ты — аналитик, который готовит материалы для занятого руководителя. **Задача:** сделай сводку приложенного документа. **Данные:** [вставьте текст документа целиком]. **Ограничения:** используй только текст документа; если данных для пункта нет — напиши «в документе не указано»; не добавляй факты из внешних знаний; числа, имена и даты копируй точно, символ в символ. **Формат:** 1) суть в одном предложении; 2) пять ключевых пунктов; 3) три числа или даты, на которые стоит обратить внимание; 4) два вопроса, на которые документ не отвечает; 5) одно действие, которое логично сделать следующим. **Критерии приёмки:** руководитель понимает содержание за 60 секунд; каждый пункт — не длиннее двух строк; в ответе нет ни одного числа, которого нет в документе.

Что вы получите на выходе — примерно так:

Пример вывода на том же шаблоне.

Суть: регламент ужесточает сроки согласования закупок и вводит второй уровень утверждения для сумм свыше €500 000. **1.** Все закупки свыше €100 000 переводятся в электронный маршрут. **2.** Для сумм свыше €500 000 добавляется утверждение финансового директора. **3.** Числа и даты: пороги €100 000 и €500 000; срок согласования — 5 рабочих дней; вступление в силу — 1 марта. **4.** Документ не отвечает: кто утверждает при отсутствии финансового директора и что делать с договорами, подписанными до 1 марта. **5.** Действие: выписать все договоры февраля и проверить, попадают ли они под переходный период.

Проверка за две минуты. Она состоит из трёх движений, и пропускать их нельзя — иначе вы прочитали не документ, а пересказ документа.

1. Пробегите пять пунктов глазами по источнику. Вы ищете подмену смысла, а не стиль.

2. Проверьте каждое число и дату поиском по тексту. Это тридцать секунд и почти нулевой риск.

3. Прочитайте пункт 4 внимательно. «Чего документ не говорит» — самая ценная часть сводки: именно здесь лежат вопросы, которые стоит задать коллегам до, а не после внедрения.

Измерьте результат. Экономия = (время «до» − время «после») × сколько раз в месяц вы это делаете. Время «после» считайте вместе с проверкой, а не до неё. Типичный первый запуск: 40–60 минут чтения превращаются в 8–12 минут работы. Если у вас вышло меньше — вы взяли слишком короткий документ или слишком общую задачу; если больше — вернитесь к блоку «Ограничения» и добавьте критерий приёмки.

Метрика здесь простая и потому рабочая: минуты на документ и число переделок. Метрика — это число, по которому видно, стало лучше или хуже. Если вы не можете назвать такое число, вы не улучшаете процесс, а меняете настроение.

Почему этот сценарий работает — коротко: вы не «попросили ИИ почитать», вы формализовали задачу. Формализация задачи — это превращение просьбы «сделай нормально» в пять записанных вещей: роль, данные, границы, формат и критерий приёмки. Ни одна из них не магия, и вместе они дают то, чего не даёт ни один модный приём из ленты: результат, который можно принять или отклонить по правилу, а не по вкусу. В главе 8 этот же приём разобран как интерфейс, а в главе 13 из него собирается полный рабочий сценарий.

Если документ не влезает в окно. Регламент на двести страниц в шаблон не поместится, и это нормальный случай. Работайте в два шага: сначала попросите карту документа — оглавление, перечень разделов с одной строкой о чём каждый, — это займёт три минуты и не потребует всего текста сразу. Потом гоняйте шаблон по двум-трём разделам, которые вам действительно нужны, и просите на выходе ссылку на раздел, а не на страницу. Так вы получаете не сжатие всего документа, а ответ на свой вопрос. Сводка целого документа нужна редко; обычно нужен ответ на один вопрос из него.

До и после: что именно изменилось

Плохой запрос звучит естественно, и в этом его проблема.

До. «Перескажи этот документ и скажи, что важно.»

После. «Ты — аналитик, который готовит материалы для занятого руководителя. Сделай сводку приложенного документа. Используй только его текст; если данных для пункта нет — напиши „в документе не указано“; не добавляй внешних фактов; числа, имена и даты копируй точно. Формат: суть в одном предложении; пять ключевых пунктов; три числа или даты; два вопроса, на которые документ не отвечает; одно следующее действие. Критерий приёмки: содержание понятно за 60 секунд, каждый пункт не длиннее двух строк, ни одного числа не из документа.»

Разница не в вежливости и не в длине. Разница в четырёх вещах.

Роль. В первом запросе её нет, и модель выбирает уровень сама — обычно средний, «для всех». Во втором задан адресат и его дефицит: занятой руководитель, шестьдесят секунд. Это меняет отбор фактов, а не только тон.

Границы данных. «Перескажи» разрешает дополнить по смыслу: модель дописывает пропущенное так, как оно обычно бывает. «Только текст документа» запрещает. Формулировка «в документе не указано» даёт модели легальный выход, и ей не нужно выдумывать, чтобы заполнить пустое место.

Формат. Пять пунктов с ограничением «не длиннее двух строк» — это не косметика. Свободный пересказ на триста слов читается дольше, чем документ, который он заменяет, и вы экономите минуты, а не часы.

Критерий приёмки. В первом запросе его нет, поэтому принять или отклонить результат можно только по вкусу. Во втором он записан: шестьдесят секунд, две строки на пункт, ни одного числа не из документа. Теперь ответ можно проверить, а не обсудить.

Есть и пятое отличие, которое стоит перенести во все свои шаблоны: пункт «чего документ не говорит». Он превращает сводку из пересказа в список вопросов — и именно за него вам будут платить вниманием на совещании.

Три ошибки, которые ломают первый запуск

Первые попытки проваливаются не из-за слабых моделей. Они проваливаются по трём повторяющимся причинам, и все три видны заранее.

Слишком широкая задача. «Сделай отчёт по продажам» — это не задача, а название отдела. Модель не знает, какой отчёт, для кого и в каком формате, поэтому выдаёт то, что чаще всего встречалось в текстах: общие слова, красивые заголовки, ни одного числа. Лечится дроблением: не «отчёт», а «сведи три района в таблицу: район, выручка, число сделок, отклонение от плана» — и три-четыре строки на выходе уже можно проверить.

Нет критерия приёмки. Когда критерия нет, им становится внешний вид ответа. Аккуратно оформленный текст с ровными абзацами выглядит как работа, хотя может не содержать ни одного факта. Критерий приёмки — это заранее записанное условие: «каждый пункт не длиннее двух строк», «в ответе нет чисел не из документа», «в таблице ровно двенадцать полей». Проверять по такому условию можно за минуту, спорить о нём — не нужно.

Экономию считают без проверки. Самая частая арифметическая ошибка в первом пилоте: «было 40 минут, стало 5» — и в эти пять минут не входит проверка. Через неделю выясняется, что проверка съедает ещё 6 минут, а через месяц — что две ошибки из тридцати пришлось исправлять по 40 минут каждая. Считайте всегда полный цикл: запрос, чтение ответа, проверка, правка. Если полный цикл не короче ручной работы, сценарий не работает, каким бы красивым ни был ответ.

Есть и четвёртая, более дорогая: персональные данные уходят во внешний чат, потому что «так быстрее». Это не ошибка качества, а инцидент, и он разбирается уже не вами. Про границы данных и правовые последствия — глава 15.

Под капотом: что происходит, когда вы нажимаете Enter

Текст разбивается на токены — куски по несколько символов или один-два слога. Модель получает последовательность токенов и предсказывает следующий, потом ещё один, и так до конца ответа. Никакого поиска по вашим файлам в этот момент не происходит: есть только то, что попало в окно контекста, и статистика, выученная на огромном корпусе текстов.

Отсюда два практических следствия. Числа хрупки: «2,5 мг» и «5 мг» для модели — близкие последовательности, а не разные величины, и она может «сгладить» редкое значение в частое. И второе: чем длиннее документ, тем меньше веса у середины. Позиционное затухание — эффект, при котором начало и конец текста влияют на ответ сильнее середины, — поэтому ключевые цифры лучше выносить в начало запроса или в конец. Подробно механика разобрана в главе 5, а про ошибки — в главе 7.

Ваш шаблон — это не «магия промпта». Это способ положить в окно контекста ровно те ограничения, которые вам нужны, и не надеяться на то, чего в окне нет.

Почему проверка — часть результата, а не недоверие к технологии

Осенью 2024 года учительница истории из чикагской школы провела с классом простой эксперимент, о котором потом писали газеты. Двадцать восемь учеников должны были ответить на вопрос, пользуясь сначала обычной поисковой выдачей, а потом ответом ИИ. К поисковой выдаче они отнеслись небрежно: пролистали и закрыли. К ответу ИИ возвращались трижды.

Но лишь двое из двадцати восьми заметили, что в одном из аргументов модель перепутала даты. Остальные двадцать шесть приняли текст как факт. Точность тона оказалась убедительнее точности источника. *(собирательный кейс)*

Не думайте, что это про школьников. Тот же паттерн виден в переговорной, где менеджер вставляет вывод модели в презентацию для совета директоров без единой проверки. И в агентстве, где копирайтер переписывает сгенерированный текст до последней запятой, но оставляет в отчёте плашку «сделано с ИИ» как знак качества.

Причина одна: скорость появления инструментов обгоняет нашу способность выработать о них интуицию. Калькуляторам понадобилось десятилетие, чтобы люди договорились: считать в голове всё ещё полезно. Смартфоны поменяли поведение за три-пять лет. Языковые модели меняют поведение за месяцы, и интуиция не успевает.

Отсюда главное правило книги: **уверенный тон модели не свидетельствует о правильности.** Проверка — не бюрократия и не недоверие. Это та часть работы, которая превращает черновик в результат.

Как читать эту книгу: три слоя и три маршрута

Книга собрана так, чтобы её можно было читать с любой глубины. Основной текст — для всех: истории, принципы, цифры, процедуры. Врезки «Под капотом» — для тех, кому важно, как это устроено: токены, окно контекста, поиск по документам. Врезки «Красные флаги» — для всех, кто не хочет наступать на грабли, на которые уже наступили тысячи людей.

Слой	**Кому**	**Что даёт**	**Где искать**
Основной текст	всем	принципы и процедуры, которые не устаревают	все главы
Врезки «Если вы новичок»	тем, кто начинает с нуля	аналогию вместо термина, порог входа	по одной-две в главе
Кейсы, матрицы, «Красные флаги»	практикам и руководителям	цифры, риски, внедрение без саботажа	главы 1, 12, 14, 17
Врезки «Под капотом»	инженерам, продактам, аналитикам	архитектуру и ограничения	главы 5–9, 14

Практикум «Что сделать сегодня»
всем
одно измеримое действие
конец каждой главы

Маршрутов три. Если вы начинаете с нуля, читайте главы 1, 2, 4, 8, 10, 13 — этот набор даёт рабочий сценарий и понимание границ. Если вы принимаете решения и отвечаете за бюджет, ваш маршрут — главы 1, 4, 12, 14, 17, 18: язык для разговора с командой и метрики внедрения. Если вы инженер, продакт или аналитик, идите в главы 5–9, 14 и приложение с шаблонами. Подробнее слои и маршруты описаны в разделе «Как читать эту книгу» в начале книги.

Одно условие общее для всех маршрутов: выполняйте действия. Каждая глава заканчивается блоком «Что сделать сегодня». Именно он, а не число прочитанных страниц, определяет, останется ли от книги что-нибудь через месяц.

В книге два вида примеров, и они честно помечены. Пометка «собирательный кейс» означает, что история собрана из нескольких практик и сведена в одну: цифры в ней типичные для таких проектов, но это не дословный протокол одного человека. Пометка «публичный кейс» означает, что у истории есть открытый источник — суд, отчёт, пресс-релиз, — и мы не добавляем к нему деталей, которых в источнике нет. Разница важна: собирательный кейс показывает механику, публичный позволяет проверить детали самому. Когда будете читать главу 12 с отраслевыми примерами, смотрите на пометку раньше, чем на цифру.

Если вы новичок: как читать эту книгу без технического бэкграунда

- Считайте модель очень начитанным, но очень самоуверенным коллегой: он мгновенно отвечает на что угодно, иногда путает факты и никогда не скажет, где ошибся, если вы не спросите.
- Не пытайтесь понять устройство на первой главе. Вам достаточно запомнить: инструмент предсказывает текст, поэтому его надо проверять, как проверяют работу стажёра.
- Читайте по маршруту для новичков: главы 1, 2, 4, 8, 10, 13. Между ними можно возвращаться к своей матрице и дописывать строки.
- Пропускайте врезки «Под капотом» при первом чтении. Они не нужны, чтобы получить результат сегодня, и не мешают логике.
- Один сценарий, доведённый до измеримого результата, полезнее десяти прочитанных глав без практики.

Красные флаги: поверхностный разговор об ИИ

- Собеседник говорит «ИИ думает», «ИИ чувствует», «ИИ хочет» — без пояснения, что это метафора.
- Аргументация строится вокруг новости недели, а не вокруг структуры сильных и слабых сторон.
- Обещание без проверки: «нейросеть всё сделает» или, наоборот, «она не способна ни на что».
- В разговоре нет ни одного примера с измеримым результатом: только «стало быстрее» и «все уже используют».
- Вам предлагают решение, но не могут назвать, что именно оно не делает и кто проверяет результат.

Что сделать сегодня

1. **Разложите пять своих задач по матрице.** Возьмите пять повторяющихся дел из прошлой недели — не абстрактных, а с названиями. Начинайте с **ключевых задач** — тех, что повторяются каждую неделю и занимают заметную долю времени; редкие и разовые дела дадут ложную картину. Для каждой запишите вердикт и одну строку «что проверю». Результат: пять строк в таблице, из которых хотя бы одна попадает в «сразу». Измерение: пять заполненных строк за пятнадцать минут; если строк меньше — вы взяли слишком крупные задачи, дробите.

2. **Прогоните дерево решений по самой дорогой из них.** Пункты 1–6, остановка на первом сработавшем. Результат: обоснованный вердикт вместо ощущения. Измерение: время на разбор одной задачи — меньше двух минут.

3. **Сделайте сводку на реальном документе.** Засеките baseline до запуска, выполните шаблон, проверьте числа поиском, засеките время вместе с проверкой. Результат: два числа — «до» и «после». Измерение: если «после» больше 15 минут, вы взяли не тот документ или не добавили критерий приёмки.

4. **Запишите свой результат одной фразой.** Формат: «Задача X занимала N минут, теперь M минут, проверка — K минут, ошибок — ноль или столько-то». Результат: цифра, которую можно показать руководителю. Измерение: фраза содержит три числа и одну границу применимости.

5. **Определите, что осталось человеку.** Для сценария из шага 3 напишите список случаев, которые вы не отдаёте модели: сложные, редкие, с чужими персональными данными. Результат: граница, зафиксированная до первой ошибки, а не после. Измерение: в списке не меньше двух пунктов.

Живой слой. Обновляемые шаблоны, чек-листы и разборы под текущие версии моделей: github.com/ai-bez-mifov/book#ch01 — QR-код на вклейке и в электронной версии.

Глава 2. Что ИИ умеет на самом деле — и чего не умеет

Паспорт главы. Обещание: разложить любую свою задачу по сильным и слабым зонам модели и заранее увидеть, где она сэкономит часы, а где создаст риск. Время чтения: 14 минут. Результат: таблица из пяти ваших задач с вердиктами «отдать», «отдать с проверкой», «не отдавать» и один сценарий, переведённый на правильный инструмент.

В пятницу в 18:20 финансовый директор сети из четырёх клиник отправляет модели выгрузку: 240 строк продаж, просьба «посчитай итог и средний чек». Ответ приходит через одиннадцать секунд. Два числа, два знака после запятой, ровный деловой тон.

В понедельник бухгалтер приносит другие цифры. Расхождение — 3,4 % недельного оборота, то есть €1 724, которых нет.

Никто не сделал ничего злонамеренного. Модель поступила как человек, считающий длинный столбец в уме: прошлась по числам, где-то округлила, где-то потеряла строку и выдала правдоподобный итог. Разница в одном: человек, считающий в уме, говорит «примерно», а модель пишет «итого: 47 812,40».

Полезная метафора — география. У модели есть континенты, где она дома: текст, структура, пересказ, черновик. И есть области, куда она ходит без карты: точные числа, свежие события, чужие мотивы, ответственность за решение. Ходить она туда всё равно будет, и с совершенно уверенным лицом. Задача не в том, чтобы её не пускать, а в том, чтобы знать, где именно проверять каждый шаг.

Глава 1 давала матрицу «сразу / с проверкой / нельзя» для решений. Здесь мы заходим с другой стороны: не «можно ли поручить», а «что именно модель делает хорошо и почему». К концу главы вы будете различать эти два вопроса и не станете путать их в разговоре с коллегами.

Кейс: 32 анкеты, 4 минуты и три выдуманных критерия

Осенью 2024 года Лорен, учительница литературы в школе на севере Чикаго, вела сентябрьский писательский конкурс. На руках — 32 анкеты: имя, класс, тема, текст на 700–1 500 слов. Требовалось три вещи: перенести анкеты в таблицу, найти в текстах логические противоречия и написать каждой участнице личную обратную связь.

В прошлом году такая же работа заняла шесть часов в три вечера. При ставке подготовки $35 в час это $210 сверхурочных, которых школа не оплачивает. И всё равно в двух письмах перепутались имена: отзыв для одной девочки ушёл другой. Ошибку заметила мама, написала классному руководителю, и Лорен потом неделю проверяла каждую строку вручную.

Первая попытка выглядела разумно. Она вставила все 32 анкеты в один чат — около 11 000 слов — и написала: «Прочитай и дай обратную связь каждой участнице». Ответ был убедительным: бодрые абзацы, имена на месте. Потом обнаружилось, что модель ответила про 19 участниц из 32, двух девочек с одинаковой фамилией слила в одну, приписала одной из них стихотворение, которого в анкете не было, а последние шесть анкет не заметила вовсе: они оказались в конце длинного сообщения, где внимание модели провисает.

На исправление ушло два часа, и доверия к результату не осталось. Это типичный вид попытки «в лоб»: одна большая просьба, много данных, ни одного критерия приёмки. Не хватало трёх вещей сразу — единицы работы, явных правил и способа проверить выход за минуту.

Второй заход был скучным и сработал. Лорен разбила работу на три задачи, у каждой свой выход и своя проверка.

Первая задача — перенос в таблицу. Одна анкета на запрос, поля заданы жёстко: ID, имя, класс, тема, жанр, объём в словах. Выход — JSON, то есть текст с явной структурой полей,

который вставляется в таблицу механически, без перечитывания. Проверка заняла секунды: число строк равно 32, ID идут подряд без пропусков.

Вторая задача — противоречия. Сначала список пяти правил конкурса, затем одна анкета, и обязательное условие: каждое замечание сопровождается цитатой из текста. Проверка ещё дешевле: цитата ищется поиском по документу за две секунды, и замечание без цитаты отправляется в мусор. Точность — доля верных попаданий на проверенной выборке — оказалась 90 %: из десяти замечаний девять были настоящими, десятое выдуманным или придиркой к стилю.

Третья задача — отзывы. Шаблон из четырёх блоков: сильная сторона с примером, зона роста с примером, один конкретный совет, финальная фраза. Запрет на сравнение с другими участницами. Объём 80–110 слов. Тон — «учитель, а не критик». Тридцать два отзыва заняли 22 минуты, и каждый пришлось только прочитать, а не переписать.

Итог: 4 минуты на таблицу вместо 50, 22 минуты на отзывы, вся работа — 40 минут вместо шести часов. Экономия около 5 часов 20 минут на конкурс, или $187 по её ставке. В финальной рассылке — ноль ошибок в именах: проверены все 32 из 32 *(собирательный кейс)*.

Вторая половина истории хуже. В феврале 2025 года проходил следующий конкурс, и Лорен переиспользовала шаблон, но пропустила блок правил: дала модели только анкеты. Модель подставила типичные критерии писательских конкурсов, которых в положении не было: минимальный объём 1 200 слов, запрет на форму диалога, возрастная категория 11–13 лет. По этим трём критериям она ранжировала работы и трём участницам написала про «недостаточный объём». Ошибку заметила коллега, задавшая простой вопрос: откуда взялись 1 200 слов.

Переделка заняла шесть часов в два вечера и два извинения перед семьями. После этого в шаблоне появился блок «Правила (дословно из положения)» и строка: «Если правила нет в блоке выше — не применяй его. Если данных не хватает — напиши „нет данных“». За четыре следующих конкурса выдуманных критериев не было ни одного.

Честное ограничение кейса: литературное качество модель оценивает посредственно. Её ранжирование совпало с оценкой Лорен в 24 случаях из 32. Этого достаточно, чтобы готовить черновик отзыва, и недостаточно, чтобы ставить оценки. Плагиат она не проверяет вовсе — для этого нужен отдельный инструмент сравнения текстов. Спорные случаи, жалобы родителей и всё, где решение имеет последствия, остаются у человека.

Разбор.

- Работа разбита на три задачи с разными выходами: таблица, список замечаний, текст отзыва. У каждой свой формат и своя проверка.
- Данные поданы по одной записи с явным ID. Модель не путает участниц, потому что у неё нет выбора между 32 похожими текстами одновременно.
- Правила поданы текстом, а не подразумевались. На втором конкурсе их не было — и модель подставила типичные. Это не сбой, это штатное поведение.
- Каждый выход проверяется дёшево: цитату ищут поиском, таблицу сверяют по числу строк, отзыв читают одним взглядом. Проверка, которая дороже работы, не приживается.

Пять зон, где модель сильна

Сила модели не мистическая. Она следует из одной механики: модель продолжает текст, выбирая вероятное следующее. Там, где «вероятное продолжение» совпадает с «правильным результатом», она работает быстро и точно. Таких мест в офисной и профессиональной работе больше, чем кажется.

Переработка текста с сохранением смысла. Пересказ, перевод, сокращение, смена тона, выделение главного. Это главный дом модели: здесь вероятное продолжение и есть то,

что нужно. Чтение 60 страниц до восьми минут — типичный замер, который повторяется из команды в команду, если текст подан целиком и разбит на части.

Первый черновик по заданному каркасу. Письмо клиенту, отчёт, инструкция, пост. Модель не пишет лучше вас, но она пишет быстрее и не боится пустой страницы. Разница между «40 минут и готово» и «12 минут и черновик» — это разница между двумя состояниями одного человека.

Структурный разбор объёма. Разложить хаос на список, сравнить два документа, найти противоречия, разметить категории. Люди делают это «по ощущению» и устают к третьему часу; модель перечисляет без устали. Условие одно: каждое утверждение должно быть привязано к цитате, иначе вы получите красивый список выдумок.

Разметка по правилам — частный случай этой зоны, и самый недооценённый. Пятьсот отзывов, десять категорий, определения категорий даны текстом: модель размечает за час, человек за три дня. Точность меряется на выборке из 50 записей, и если она ниже 85 %, причина обычно в определениях категорий, а не в модели: люди на тех же 50 записях расходятся не меньше.

Преобразование форматов. Текст в таблицу, CSV в JSON, запись встречи в протокол, письмо в список задач. Это операции «возьми и переложи», и модель выполняет их охотно. Ломается там, где поля не заданы: колонки начинают плыть, а часть строк исчезает.

Объяснение под разную аудиторию. Один и тот же материал для новичка, для юриста и для ребёнка. Раньше для этого нужны были три человека или три вечера; теперь — три запроса с явно названной аудиторией.

Сильная зона	**Что именно делает**	**Типичная экономия**	**Где ломается**
Переработка текста	пересказ, перевод, тон, сокращение	60 минут чтения → 8 минут	длинный документ обрезан, середина потеряна
Первый черновик	письмо, отчёт, инструкция по каркасу	40 минут → 12 минут	нет каркаса — получается средний текст
Структурный разбор	список, сравнение, противоречия	2 часа → 25 минут	замечания без цитат превращаются в выдумку
Преобразование форматов	текст → таблица, CSV → JSON	50 минут → 4 минуты	поля не заданы, колонки плывут
Объяснение под аудиторию	один материал на трёх уровнях		

редактор не нужен
факты этим не проверяются

Общее у всех пяти зон: задача формулируется как «преобразуй то, что я дал», а не «найди то, чего у меня нет». Как только модель должна добавить знание, которого нет во входных данных, начинается зона риска.

Четыре зоны, где она слаба

Слабость тоже не каприз, а следствие механики. Модель не ищет, не считает, не знает вашего контекста и не отвечает за последствия. Каждый пункт ниже — про это.

Редкие и свежие факты. Модель обучена на текстах до определённой даты и не имеет доступа к миру после неё. О широко известном она отвечает верно, о редком — сочиняет. Галлюцинация — это правдоподобное утверждение, не опирающееся ни на данные, ни на факт: несуществующий пункт закона, выдуманная статья, чужой автор у стихотворения. Опасна не сама выдумка, а её подача: уверенность модели — это её тон и вероятность, с которой она выбирала продолжение, а к правильности ответа ни то, ни другое отношения не имеет. Уверенный тон — стиль изложения, не свидетельство истины.

Точная арифметика. Модель не калькулятор. Она складывает так, как человек в голове: в общем случае и примерно. Ошибки копятся на длинных числах, процентах, десятичных дробях и итогах по колонке. Разбор этой зоны — отдельный раздел ниже.

Намерения, мотивы и скрытый контекст. «Чего хочет этот клиент», «почему сотрудник сопротивляется», «что решит совет директоров» — это не вопросы к модели. Она отвечает как средний человек, читавший похожие тексты. Ваш контекст — переписку, историю неудач, характер директора — она не знает, если вы его не подали, и даже подав, интерпретирует как типичный.

Ответственность за результат. Если модель ошиблась, платите вы. Это не дефект конкретного продукта, а архитектурная особенность: у модели нет субъектности и нет счёта, с которого спишут штраф. Отсюда правило, которое дальше встречается в каждой главе: чем дороже ошибка, тем плотнее человек в контуре — тот, кто проверяет выход перед использованием.

Слабая зона
Почему так
Как выглядит ошибка
Что делать

Редкие и свежие факты
нет доступа к миру после обучения
уверенная ссылка на несуществующий пункт
подать документ, требовать источник

Арифметика
предсказание токенов вместо вычисления
итог «примерно» верный, в третьем знаке нет
считать формулой или кодом

Мотивы людей
нет данных о конкретном человеке
«клиент хочет скидку» как факт
интервью, данные, гипотезы

Ответственность
нет субъекта, который отвечает
подпись под непроверенным расчётом
человек проверяет перед использованием

Отдельно стоит краевой случай (edge case) — редкий вход, которого почти не было в обучающих данных: нестандартный формат даты, диалект, узкая отрасль, необычная единица измерения. На типовых примерах модель выглядит компетентной, на краевых — теряет детали молча. Поэтому проверять надо не середину, а края: нетипичные строки, короткие записи, пустые поля.

Стоит отделить понимание от его имитации, потому что от этого зависит доверие. Когда модель пересказывает текст, она действительно опирается на него: слова в ответе связаны со словами на входе, и связь проверяема. Когда она рассуждает о причинах события, она воспроизводит форму рассуждения — типичные связки, типичные причины, типичные оговорки. Форма совпадает, содержание берётся из среднего по корпусу. Тест простой: уберите из запроса ваши данные. Если ответ не изменился, модель не анализировала ваш случай, а рассказала общий. Это не делает её бесполезной — общий ответ часто полезен как рамка. Но выдавать рамку за вывод по вашему случаю и есть та ошибка, после которой к ИИ теряют доверие.

Если вы новичок: как отличить «умеет» от «делает вид»

Представьте двух студентов на экзамене. Один отвечает уверенно и иногда ошибается, но, если попросить, показывает место в учебнике. Второй говорит так же гладко, но источника не имеет и придумывает его на ходу. Модель по умолчанию ведёт себя как второй, пока вы не дадите ей учебник.

- Дайте три задачи, ответ на которые вы знаете точно. Это единственный честный способ измерить её на вашем материале.

- Попросите источник: «откуда эта фраза». Если модель не может указать место в присланном тексте — считайте утверждение выдумкой.

- Дайте ей данные и правила. На одном тексте без правил она даёт средний ответ, на тексте с правилами — попадает в вашу задачу.

- Проверьте одно число калькулятором. Если расходится — не «улучшайте промпт», а смените инструмент.

- Если после трёх попыток результат не сходится, сузьте задачу: не «сделай отчёт», а «выпиши пять фактов с цитатами из этих двух абзацев».

Числа и таблицы: почему итог не сходится

Теперь самая дорогая зона. Почти каждая история про «ИИ наврал в отчёте» — это история про числа, которые модель складывала сама.

Причина в устройстве. Модель порождает текст по фрагментам — токенам, кускам слова или числа. Число 47 812,40 приходит к ней не как величина, а как последовательность кусков. У неё нет регистра для переноса разряда, нет разрядной сетки, нет промежуточного результата, который можно сохранить между шагами. Сложение столбиком требует состояния: «пишем 4, два в уме». Модель такого состояния не держит; она каждый следующий фрагмент выбирает заново, глядя на уже написанное.

Отсюда типичные места ошибок, которые повторяются из отчёта в отчёт:

• длинные суммы: чем больше строк, тем выше шанс потерять одну и не заметить;

• десятичные дроби и разделители: 1 234,56 и 1,234.56 — разные миры, и модель путает их молча;

• проценты: «17 % от 2 340» и «на 17 % больше» — две операции, и вторая часто считается от неверной базы;

• итог по колонке: чтобы сложить, надо перечитать все строки, а внимание к дальним частям контекста слабеет;

• округление: модель округляет на каждом шаге, и ошибка накапливается.

Проценты заслуживают отдельного абзаца: там ошибаются и люди, а модель повторяет их ошибки уверенно. «Маржа выросла с 18 % до 22 %» — это плюс четыре процентных пункта, но плюс 22 % относительно прежнего уровня. Если попросить модель «посчитать рост в процентах» и не назвать базу, она выберет ту, что вероятнее в текстах, — обычно относительную. В отчёте появится «рост на 22 %», хотя правильный ответ зависит от того, что вы имели в виду. Поэтому в запросе называйте базу: «относительно прошлого квартала», «в процентных пунктах», «от плана».

Замер, который стоит держать в голове: на выгрузке из 240 строк с итогом около €18 000 модель, отвечавшая «одним числом», ошиблась в 7 прогонах из 10, отклонение — от 0,4 до 3,5 %. Формула =SUM в таблице дала ноль ошибок на той же выгрузке. Разница не в интеллекте, а в том, что таблица складывает, а модель предсказывает, каким число должно быть *(оценка по нашим прогонам, не отраслевая статистика)*.

Правильная процедура состоит из четырёх ходов, и она не требует от вас программирования.

Первый: разделить формулировку и вычисление. Попросите модель не число, а шаги: какие колонки сложить, какие строки исключить, какая формула получится. Формулировка — языковая задача, и здесь модель сильна. Само сложение отдайте инструменту.

Второй: считать тем, что умеет считать. Excel, SQL — язык запросов к табличным базам, где вы описываете, что выбрать и сложить, а вычисляет движок, — или Python. У каждого своя цена входа: формула в таблице — минуты, запрос — часы обучения, скрипт — зависит от человека.

Третий: встроить контрольную сумму. Независимо посчитанная величина — число строк, среднее по подвыборке, итог из другого источника — ловит не только ошибку модели, но и ошибку в данных. Без контрольной суммы вы сверяете одно неизвестное с другим.

Четвёртый: для CSV просить код, а не ответ. Модель хорошо пишет код и плохо считает. Пусть пишет, а считает пусть среда исполнения.

Вот минимальный скрипт, который делает то, что просили у модели в начале главы: читает файл, считает итог и средний чек, а затем сверяет себя контрольной суммой по кассам.

```
import pandas as pd

df = pd.read_csv('sales.csv', decimal=',')
df['amount'] = pd.to_numeric(df['amount'], errors='coerce')

print('строк:', len(df))
print('пустых amount:', int(df['amount'].isna().sum()))

total = df['amount'].sum()
checks = df['check_id'].nunique()

print('итог:', round(total, 2))
print('средний чек:', round(total / checks, 2))
```

```
by_till = df.groupby('till')['amount'].agg(['sum', 'count'])
print(by_till)
print('контроль:', round(by_till['sum'].sum(), 2),
'разница:', round(total - by_till['sum'].sum(), 2))
```

Читать вывод надо в трёх местах. Разница в контрольной сумме должна быть нулевой: если нет, значит часть строк не попала в группировку. Число пустых значений в колонке суммы должно быть нулевым или объяснимым. Число строк должно совпадать с числом строк в выгрузке — самая частая молчаливая потеря.

Средний чек здесь — сумма выручки, делённая на число уникальных чеков, а не среднее по строкам: если в чеке три позиции, среднее по строкам даст втрое меньше. Это не придирка, а типичная ошибка постановки, которую модель тоже не заметит, если вы не назвали определение.

Под капотом: почему токенизация мешает считать

Токенизатор рубит текст на куски по статистике встречаемости, а не по смыслу. Частые слова — один токен, редкие — несколько, числа — почти всегда на куски: 1 234,56 распадается на «1», « 234», «,», «56» или на похожие фрагменты, и границы этих кусков не совпадают с разрядами.

Дальше арифметика упирается в отсутствие рабочей памяти. Сложение столбиком — последовательный алгоритм с переносом состояния, а генерация — последовательность локальных выборов. Модель может «знать», что 7 × 8 = 56, потому что это часто встречается в текстах; итог по 240 строкам не встречается нигде, и выводить его приходится пошагово, теряя точность. Отсюда же странности с подсчётом букв в слове и с длиной строки.

Практические следствия: temperature (параметр выбора между вероятными продолжениями) на арифметику влияет слабо — ошибка не в случайности, а в механике; просьба «подумай пошагово» помогает с постановкой задачи и почти не помогает с суммой; код и вызов вычислительного инструмента помогают, потому что вычисление выходит из модели наружу.

Под капотом: почему правила сильнее примеров

Модель учится на лету внутри одного запроса: примеры в контексте задают формат, а инструкции — границы. Когда правило не названо, модель заполняет пустоту типичным: для писательского конкурса — типичными критериями, для отчёта — типичной структурой, для юридического вопроса — типичной оговоркой. Именно это произошло на втором конкурсе Лорен.

Три правила, которые дают наибольший эффект при минимальной работе: явный запрет («не сравнивай участников между собой»), явный дефолт («если данных нет — напиши „нет данных“»), явный критерий приёмки («ровно 32 строки, ID подряд»). Инструкции в конце длинного контекста работают сильнее, чем в начале: важное — ближе к концу.

Одна задача — один инструмент

Из всего вышесказанного вытекает рабочее правило: у каждой задачи есть инструмент с гарантией, и модель — не универсальная замена, а один из них. Путаница начинается там, где моделью закрывают дыру, которую должен закрывать калькулятор, поиск или база.

Задача
Правильный инструмент
Роль модели

Сложить 240 строк, посчитать маржу
Excel, SQL, Python
сформулировать шаги и написать формулу

Узнать курс, цену, действующую норму
поиск, официальный источник, база
сформулировать запрос, объяснить выдачу

Найти клиента по 12 признакам
SQL, CRM, BI-отчёт
описать критерии отбора

Пересказать 60 страниц, снять протокол
языковая модель
основная работа

Разметить 500 отзывов по темам
модель плюс проверка выборки
основная работа

Решить, кому отказать в кредите
человек, модель только готовит сводку
собрать факты и вопросы

Сгенерировать иллюстрацию к отчёту
диффузионная модель
не участвует

Диффузионная модель — отдельное семейство генеративных моделей, которое собирает изображение или звук из шума, постепенно убирая случайность; она не «понимает» текст так, как языковая, и путать их — распространённая ошибка в разговорах о «нейросетях вообще».

Отдельно про поиск. Многие интерфейсы подключают его автоматически, и это снимает часть проблемы свежих фактов — но не всю: модель пересказывает выдачу и может перепутать дату, автора или единицу измерения. Правило простое: если в ответе есть ссылка, откройте её; если ссылки нет, считайте утверждение гипотезой. Поиск внутри модели не отменяет проверку, он делает её дешевле.

У каждого инструмента своя цена входа, и она решает, что выбрать в понедельник утром. Формула в таблице — пятнадцать минут. Запрос в базе — вечер обучения и доступ, который выдаёт не вы. Скрипт — час на первый и минуты на следующие. Языковая модель — три минуты и никакой подготовки, поэтому её и тянут во все задачи, включая те, где она слаба. Выбор инструмента по цене входа, а не по гарантии, — самая частая причина тихой ошибки в отчётах.

Дерево решений для новой задачи выглядит так.

1. Если на выходе должно быть число, которое кто-то подпишет, — считайте формулой, запросом или скриптом; модель пусть пишет формулу и объясняет шаги.

2. Если ответ — факт, появившийся после даты обучения, или внутренний факт вашей организации, — идите в поиск, базу или документ; модель пусть формулирует запрос и пересказывает найденное с указанием источника.

3. Если ответ — текст, структура, пересказ, черновик, разметка по заданным правилам — отдавайте модели, но с критерием приёмки.

4. Если решение имеет цену ошибки — деньги, здоровье, свобода, репутация, — модель готовит материал, решает человек, а в регулируемых областях ещё и профильный специалист.

5. Если процесс повторяется сотни раз и правила стабильны — можно думать об автоматизации, то есть о выполнении шагов без человека в каждом цикле. Но только после двух недель ручной проверки на выборке: автоматизация без замера точности — это способ производить ошибки быстрее.

Автоматизация отличается от «ускорения с проверкой» одним: в автоматическом режиме человек не смотрит на каждый выход, и цена ошибки платится пачкой. Поэтому в дереве она стоит последней, а не первой.

До / После: «посчитай средний чек»

Сравним два запроса на одной и той же выгрузке. Разница между ними — не вежливость и не длина, а распределение работы между моделью и инструментом.

Плохо. «Посчитай средний чек по этому списку: [дальше 240 строк текстом]. Дай итог и среднее значение». Результат: два числа за одиннадцать секунд, отклонение от бухгалтерии 3,4 %, и никакой возможности увидеть, где потерялась строка. Проверить нечем: модель не показывает путь, а пересчитывать вручную — те же 40 минут, что и раньше.

Хорошо. «Вот файл sales.csv: колонки check_id, till, amount, date; разделитель ;, десятичная запятая. Напиши код на pandas, который: читает файл; приводит amount к числу; считает итог и число уникальных чеков; выводит средний чек как итог, делённый на число чеков; считает контрольную сумму по кассам и печатает разницу; отдельно печатает число строк и число пустых amount. Выполни код и покажи вывод целиком. Если контрольная сумма не сходится, не пересчитывай вручную — покажи строки, которые не попали в группировку».

Что изменилось, по пунктам.

1. **Инструмент вычисления вместо языка.** Числа считает pandas, модель пишет код. Ошибка в арифметике становится невозможной, зато становится возможной ошибка в постановке — её ловит следующий пункт.

2. **Данные поданы файлом со схемой.** Явные имена колонок, разделитель, десятичная запятая. Модель больше не угадывает, где кончается одно поле и начинается другое.

3. **Проверка встроена в задачу, а не в голову проверяющего.** Контрольная сумма по кассам, число строк, число пустых значений — три независимых индикатора, которые печатаются всегда.

4. **Критерий приёмки сформулирован.** «Разница ноль» — это условие, которое можно проверить, а не ощущение «вроде похоже».

Стоимость перехода: сорок минут на освоение read_csv и groupby, если вы никогда этого не делали, и примерно ноль, если делали. Дальше этот скрипт переиспользуется на любой выгрузке с теми же колонками. Сравнение по деньгам: один пересчёт отчёта вручную — час работы финансиста, одна ошибка в отчёте перед советом директоров — репутация и, иногда, премия.

Есть и менее очевидная выгода: воспроизводимость. Скрипт можно запустить через месяц на новых данных и получить сопоставимый результат. Ответ модели на тот же вопрос через месяц будет другим: версии меняются, параметр температуры вносит разброс, контекст другой. Для отчётности это принципиально — число, которое нельзя воспроизвести, нельзя и защитить.

Тот же приём работает и без Python. В таблице достаточно попросить модель сформулировать формулу и проверить её на трёх строках вручную; в SQL — попросить запрос с GROUP BY и отдельный запрос-сверку COUNT(*); в BI-системе — попросить список метрик и определение каждой. Принцип один: модель формулирует, инструмент считает, человек сверяет.

Если исполняемой среды нет вовсе — чат не запускает код, доступа к базе не дают, — процедура меняется, но не отменяется. Просите у модели не результат, а проверяемый путь: формулу, шаги, список строк, которые надо исключить, и ожидаемый порядок величины. Затем считайте сами, а модель привлекайте к сверке расхождений: «получилось 46 190,80, а по моей прикидке 47 000 — где я мог ошибиться». В такой постановке она полезна: ищет пропущенные строки и неверные допущения, а не складывает числа.

Красные флаги: когда ИИ точно не поможет

- Задача требует не «сделать работу», а «взять ответственность»: судебное заключение, диагноз, подписанная цифра в отчёте, кадровое решение.
- Результат зависит от внутреннего контекста организации, которого нет в запросе: история конфликтов, негласные правила, реальные сроки поставщика.
- Нужно уникальное мнение или позиция, а не структурированное рассуждение по известным правилам.
- Цена ошибки превышает стоимость проверки в десятки раз: чем дороже ошибка, тем меньше смысла экономить на человеке.
- Вы не можете назвать критерий приёмки. Если непонятно, что считать хорошим результатом, модель выдаст что-то правдоподобное, и вы это примете.
- Данные нельзя отдавать наружу: клиентские базы, медицинские записи, коммерческая тайна без договора и режима обработки. Подробно — в главе 15.

Что сделать сегодня

1. Заполните таблицу из пяти своих задач на этой неделе. Колонки: задача, что на выходе (число, факт, текст, решение), вердикт (отдать, отдать с проверкой, не отдавать), инструмент. Результат: пять строк. Измерение: если вердикт «отдать без проверки» стоит больше чем у одной задачи из пяти — вы недооцениваете риск.

2. Переведите одну числовую задачу на формулу или скрипт. Результат: итог, посчитанный инструментом, и контрольная сумма. Измерение: расхождение с вашей ручной оценкой; зафиксируйте время до и после.

3. Проверьте одну слабую зону на своём материале: попросите модель назвать источник трёх фактов из вашей области и проверьте каждый. Результат: счёт «подтверждено из трёх». Измерение: меньше трёх — этим фактам нужен внешний источник, и это правило стоит записать.

4. Добавьте в свой рабочий шаблон две строки: блок правил и «если данных нет — напиши „нет данных“». Прогоните на трёх прошлых запросах. Результат: сравнение до и после. Измерение: число выдуманных правил и пропущенных полей; цель — ноль.

5. Замерьте одну задачу по времени: до (ваша обычная работа) и после (модель плюс проверка). Результат: два числа и разница. Измерение: если разница меньше 20 %, проверка съедает выигрыш, и задачу стоит либо упростить, либо вернуть человеку.

Дальше — глава 3 о мифах, которые мешают применять всё это на практике. Технические детали того, почему модель ведёт себя именно так, — в главе 5; как измерять точность и стоимость решения — в главе 14; как строить личные сценарии — в главе 10; про галлюцинации подробно — в главе 7.

Живой слой. Обновляемые шаблоны, чек-листы и разборы под текущие версии моделей: github.com/ai-bez-mifov/book#ch02 — QR-код на вклейке и в электронной версии.

Глава 3. Мифы, которые мешают применять ИИ

Паспорт главы. Обещание: распознавать миф об ИИ за тридцать секунд и не строить на нём решение. Время чтения: 14 минут. Результат: разобранное по пяти вопросам чужое утверждение об ИИ, таблица «миф — реальность — цена веры — что делать» с тремя вашими строками и один переписанный запрос к подрядчику.

В четверг, в 16:10, в переговорной на тридцать четвёртом этаже в Дубае подрядчик показывает слайд. Карта региона, четырнадцать стрелок, подпись: «4 200 источников, 11 рынков, прогноз спроса на два квартала». Директор по маркетингу кивает. Слайд отвечает на вопрос, который она задала две недели назад: где нам расти.

Вопросов никто не задаёт. Не потому, что их нет, а потому, что «а как именно вы это посчитали» в такой момент звучит как недоверие — к данным, к людям, к потраченным двум неделям. Решение принимают за девять минут. Цена — $18 000 за «ИИ-стратегию» и два аналитика, снятых с других задач на квартал.

Через три недели в отчёте на 41 страницу не нашлось ни одного места, где было бы видно, откуда взялась конкретная цифра. Не потому, что подрядчик прятал метод. Потому, что метода не было: модель пересказала открытые статьи, а люди оформили пересказ как аналитику.

Миф об ИИ редко выглядит как ложь. Он выглядит как удобное сокращение, которое экономит одно решение, — и стоит квартала.

Кейс: $18 000 за пересказ и квартал, которого не было

Отдел маркетинга дистрибьютора бытовой техники в Дубае: одиннадцать рынков, шестьдесят сотрудников, оборот €84 млн. Задача года — выбрать два рынка для расширения. Штатных аналитиков двое. Полный цикл исследования с внешним консультантом стоил $40 000 и занимал четыре месяца. Руководитель искала путь короче и дешевле и нашла предложение за $18 000: «ИИ-стратегия, шесть недель, прогноз по одиннадцати рынкам».

Стоимость ожидания тоже была известна. Окно входа на два рынка закрывалось к концу второго квартала; по внутренней оценке, задержка на квартал стоила бы около $140 000 недополученной маржи. Именно эта цифра, а не отчёт, и была настоящим предметом решения.

Был и третий вариант, который отклонили: взять у консультанта не весь проект, а один рынок за $12 000 и четыре недели, а остальные десять отсеять своими силами по грубому критерию. Вариант отвергли как «половинчатый» — и в этом была первая ошибка, потому что он давал проверяемую базу для сравнения.

Подрядчик прислал 41 страницу: девять рекомендаций, тепловые карты, три сценария спроса. Первые две недели отчёт читали с интересом. На третьей финансовый аналитик попросил показать, откуда взялась одна конкретная цифра — доля рынка в одном из эмиратов. Подрядчик ответил: «модель проанализировала массив открытых данных».

Дальше разговор пошёл по предсказуемому кругу. Запросили список источников — прислали двенадцать ссылок, из которых восемь оказались пресс-релизами поставщиков. Запросили источник по каждой цифре — получили обещание прислать позже. Тогда аналитик сделал то, что стоило сделать до подписания: взял двадцать утверждений из отчёта и потратил два часа на ручную проверку. Источник нашёлся у шести. Одиннадцать совпали с открытыми статьями почти дословно, включая формулировки. Три не подтвердились вовсе: описанного сегмента покупателей в таком объёме не существовало, а ссылка вела на статью о другом регионе.

Отдельно сверили цифры с собственной выгрузкой продаж. Расхождение по одному рынку — 22 %. Подрядчик объяснил это «иной методикой подсчёта», но методику не показал. Деньги не вернули: в договоре был поставлен «аналитический отчёт», и он был поставлен. Квартал ушёл.

Отдельно стоит назвать упущенную возможность. Подрядчик предлагал пробный фрагмент за €600: один рынок, две недели, тот же метод. Фрагмент не заказали, потому что он выглядел как лишний шаг — и это ровно та ошибка, которую потом дороже всего объяснять. Два часа проверки и три недели переделки обошлись в двадцать раз дороже, чем стоил бы пробный кусок до подписания.

Пересборка заняла неделю и не потребовала новых инструментов. Отдел разделил работу на три типа, у каждого — свой исполнитель и свой критерий.

Факты. Источники, цифры, даты, выгрузки, интервью — люди и поиск. У каждого факта в рабочей таблице появились источник, дата и метод: «опрос 300 клиентов, февраль, телефон», «выгрузка из ERP, март, артикулы без возвратов».

Черновики. Модель. Сжать сорок интервью до пятнадцати цитат, разложить конкурентов по десяти признакам, превратить таблицу в связный текст, сформулировать пять гипотез для проверки. Здесь она сильна, потому что критерий приёмки дешёвый: цитаты проверяются поиском по транскрипту, гипотезы — обсуждением.

Решения. Руководитель. Какой рынок, какая цена входа, какой риск допустим, кто отвечает за результат. Модель не подписывает контракт и не платит за ошибку.

К этому добавилась одна процедура, которая позже прижилась во всех отчётах компании, — «карточка утверждения». Одна строка на каждую цифру: значение, источник, дата, кто проверил, что будет, если ошибка. Если строку заполнить нечем, утверждение отправляется в раздел «предположения», а не в раздел «выводы».

Второй отчёт готовили шесть недель. Он стоил $2 300: подписка на модель, 30 человеко-часов на проверку и один день на вёрстку. Вышло двенадцать страниц вместо сорока одной. Из 41 утверждения с цифрами все имели источник, кроме девяти, честно помеченных как оценка.

Ограничение оказалось неприятным. Две рекомендации не выдержали проверки в пилотном регионе: модель оценила эластичность спроса по открытым статьям, а локальный тест на 300 клиентах дал другие числа. Пришлось заменить оценку на собственные измерения и потратить ещё три недели. Квартал это не вернуло: конкурент вошёл на рынок первым, и часть маржи ушла вместе с окном.

(собирательный кейс)

Разбор.

- Миф был не в модели, а в постановке закупки: купили «аналитику», не определив, что считается анализом, а что пересказом.

- Проверка стоила два часа и нашла проблему до решения. Соотношение — примерно 1 к 2 000 по потраченной сумме.

- Разделение фактов, черновиков и решений вернуло управляемость: у каждого типа работы появился свой исполнитель и свой критерий приёмки.

- Карточка утверждения сделала отчёт проверяемым. Цифра без источника — не актив, а обязательство, которое кто-то однажды предъявит.

- Урок не в том, что модель не годится для анализа рынка. Урок в том, что анализ без данных и метода остаётся пересказом, сколько бы источников ни значилось на слайде.

Почему мифы дороже, чем кажутся

Миф удобен. Он снимает одно решение: не надо выяснять, как устроено, — достаточно поверить, что устроено нужным образом. Плата приходит позже и не сразу: часами на переделку, деньгами на подрядчика, закрытым окном на рынке.

Глава 2 раскладывала, что модель умеет и чего не умеет по факту. Эта глава — про то, во что люди верят вопреки факту, и почему вера держится. Мифы не случайны: каждый закрывает

неудобное место, где пришлось бы признать границу своей компетенции или потратить время на проверку.

У веры в миф всегда есть цена, и её можно назвать заранее. Ниже — карта из девяти мифов, которые чаще всего встречаются в разговорах о внедрении.

Миф
Что на самом деле
Цена веры
Что делать вместо

«ИИ думает»
Предсказывает следующий фрагмент текста
Поручают суждение, а не преобразование
Формулировать задачу как преобразование входа в выход

«ИИ знает факты»
Воспроизводит статистику текстов
Уверенные выдумки в отчётах и письмах
Давать документ и требовать ссылку на фразу

«Чем длиннее промпт, тем лучше»
Конкурирующие инструкции размывают приоритет
Семь требований из двенадцати
Одна задача на запрос, цепочка шагов

«ИИ скоро заменит всех»
Меняется состав задач, не факт профессии
Стратегия «подождём» или паника
Переобучение под конкретные операции

«ИИ объективен»
Обучен на человеческих текстах со сдвигами
Решения с непроверенным уклоном
Свои данные как основа, модель как редактор

«Секретный промпт решает»
Решает структура задачи и данные
Покупка «магии» и курсов по кнопкам
Задача, данные, критерий приёмки

«ИИ корректен с одного запроса»
Один прогон — черновик, не результат
Ошибки уходят заказчику
Проверка по чек-листу до отправки

«ИИ — это только чат»
Встраивается в процессы и системы
Упущенные выгоды автоматизации
Искать узкие места процесса, а не окно чата

«ИИ нечестен, его надо бояться»
Инструмент с известными границами
Отказ от измеримой пользы
Риск-план вместо страха

Таблица полезна как список для сверки, но решение принимается по механике. Поэтому разберём каждый пункт отдельно — что именно происходит и что делать вместо.

Откуда мифы берутся

Мифы не появляются из ниоткуда. У них три устойчивых канала, и в каждом есть свой интерес.

Первый — продавец. Ему нужно, чтобы возможность выглядела готовой услугой, поэтому формулировка сжимается до «ИИ проанализирует рынок». Это не обязательно обман: человек искренне верит, что продаёт результат, а не процесс. Проверка здесь одна — просить метод и источники до подписания, как в дереве решений ниже.

Второй — внутренний энтузиаст. Сотрудник попробовал инструмент на своей задаче, получил хороший результат и переносит вывод на соседнюю область, где условия другие. Так рождается «у нас уже работает, давайте распространим». Это самый симпатичный канал и самый опасный: энтузиаст говорит правду о своём случае и неправду о вашем.

Третий — пересказ. Отраслевая новость проходит три пересказа и теряет условия: «компания сократила отчётность на 40 %» превращается в «ИИ заменяет аналитиков». Условия — единственное, что делало новость правдой, и они исчезают первыми.

Практический вывод: спрашивайте не «правда ли это», а «при каких условиях это было правдой». Условия — то, что нельзя пересказать, не потеряв.

Девять мифов, каждый с ценой

Миф 1. «ИИ думает». Модель — это программа, обученная на большом массиве текстов предсказывать следующую часть текста. Внутри нет намерения, плана на завтра и способности усомниться. Технически она предсказывает следующий токен — единицу текста длиной примерно в три-четыре символа. Практический пример: руководитель пишет «подумай, стоит ли нам выходить на рынок X», получает аккуратный список плюсов и минусов и воспринимает его как размышление. На деле это усреднённое содержание статей на эту тему, разложенное по рубрикам. Опасность не в том, что ответ плох, а в том, что он выглядит как работа с вашим случаем, хотя описывает средний случай. Что делать вместо: заменить просьбу «подумай» на просьбу «разложи по признакам, которые я перечислю», и заранее решить, какой из признаков для вас решающий.

Миф 2. «ИИ знает факты». Модель знает тексты, а не факты. Если утверждение встречается в тысячах документов, ответ будет уверенным и, скорее всего, верным. Если утверждение встречается один раз в локальной заметке или не встречается вовсе, ответ будет таким же уверенным и выдуманным. Бытовой тест: спросите модель о том, что известно только вам, — о вашем тарифе, вашем поставщике, вашем прошлогоднем объёме. Ответ будет правдоподобным и неверным. Что делать вместо: прикладывать документ и требовать цитату. Чем сильнее ответ заземлён — привязан к вашему тексту, — тем меньше выдумки. Механика этого — в главе 5, галлюцинации как явление — в главе 7.

Миф 3. «Чем длиннее промпт, тем лучше». Длинная инструкция не усиливает модель, а ставит её перед конкуренцией требований. Наблюдение из практики: в запросе на 900 слов с двенадцатью требованиями модель устойчиво выполняет семь-восемь, причём разные в разных прогонах; в запросе на 180 слов с тремя требованиями — все три. Причина не в лени, а в том, что веса внимания распределяются по всему тексту, и требование из первого

абзаца конкурирует с требованием из десятого. Что делать вместо: одна задача — один запрос, а сложный результат собирается цепочкой. Как это устроить технически — в главе 8.

Миф 4. «ИИ скоро заменит всех». Прошлые волны автоматизации дают один и тот же паттерн: банкоматы не закрыли отделения, шаблоны договоров не отменили юристов, голосовые меню не убрали операторов. Менялся состав задач внутри профессии, а не факт её существования. Это наблюдение, а не гарантия: скорость перестройки в этот раз выше, и часть операций действительно исчезает.

Полезно смотреть не на профессию, а на операции. В работе юриста есть чтение типового договора, поиск прецедента, черновик письма и переговоры о риске. Первые три сжимаются сильно, четвёртая — почти нет. В работе бухгалтера есть сверка, перенос, объяснение расхождения и разговор с аудитором: сжимаются первые две. Кто видел свою работу как набор операций, тот перестраивается за месяцы. Кто видел её как неделимое «я специалист», тот обсуждает прогнозы.

Практический вывод одинаков для обеих крайностей — и для «подождём, всё рассосётся», и для «надо срочно всё менять». Он простой: выигрывает тот, кто переводит свои повторяющиеся операции на инструмент и измеряет результат. Прогнозы о сроках — это погода, она меняется каждые несколько месяцев; устройство работы — климат.

Миф 5. «ИИ объективен». Модель обучена на человеческих текстах, а затем подстроена под предпочтения разметчиков — это называется RLHF, обучение с подкреплением по обратной связи человека: люди оценивают ответы, модель учится выдавать те, что нравятся оценивающим. Значит, в ответах живут и человеческие сдвиги: округления, привычные клише, представления о том, «как принято». Проверяемый пример: попросите описать идеального кандидата на вакансию — и посмотрите, какие признаки всплывут без вашего запроса. Что делать вместо: чем важнее решение, тем больше веса у ваших проверенных данных и меньше — у общих знаний модели. Модель здесь редактор и сборщик, а не судья.

Миф 6. «Мой промпт — секрет успеха». Магических промптов нет. Есть структура: задача, данные, критерий хорошего результата. Если все три описаны, работает почти любой аккуратный текст запроса. Если не описаны, не спасает никакая формулировка. Цена веры в секрет измерима: люди покупают курсы «промптов», тратят недели на коллекционирование чужих шаблонов и не делают главного — не собирают данные и не пишут критерий приёмки. Что делать вместо: потратить те же недели на карточку задачи, как в главе 4, и на тестовый набор из десяти своих случаев.

Миф 7. «ИИ корректен с одного запроса». Один прогон — это черновик. Не потому, что модель плохая, а потому, что вы ещё не видели, где она ошиблась. Если важный результат, проверка обязательна, и её стоимость надо закладывать в смету задачи. Признак, который стоит запомнить: если модель даёт неверный результат дважды подряд на одном и том же входе, дело почти всегда не в модели, а в данных — вы не дали то, на что она должна опираться. Что делать вместо: заранее написать, как вы проверите результат за две минуты, и только потом запускать.

Миф 8. «ИИ — это только чат». Это самый дорогой миф из девяти, потому что он тихий. Пока руководитель обсуждает, какой чат купить, в его компании уже работают другие применения: поиск по документации, разбор входящих писем, извлечение полей из счетов, черновики ответов в системе поддержки, сводки по звонкам. Интерфейсы меняются, названия кнопок тоже, но принцип устойчив: ценность появляется там, где модель встроена в шаг процесса, а не стоит отдельным окном.

Простой способ это увидеть — взять один процесс и выписать его шаги на листе. Закупка: заявка, поиск поставщика, запрос цены, сравнение, согласование, заказ. Внутри шагов видно, где человек механически переносит текст из одного формата в другой, — обычно это три-четыре места из шести. Именно там модель экономит часы без переделки процесса целиком.

Что делать вместо: пройти процесс по шагам и начать с трёх мест, где ручной перенос текста съедает больше всего времени.

Миф 9. «ИИ нечестен, его надо бояться». Ирония в том, что честный результат получить проще, чем кажется: дайте проверяемый контекст, потребуйте цитаты, перепроверьте цифры. Модель не скрывает, что генерирует правдоподобный текст; она вообще не умеет скрывать, у неё нет состояния между сессиями, где можно было бы что-то утаить. Страх вредит по-другому: он подменяет управление риском отказом от задачи. Что делать вместо: список из трёх рисков с контролем на каждый — утечка данных, ошибка в цифре, чужая ответственность. Если по каждому есть мера, решение принимается, а не откладывается.

Десятый миф: «модель понимает, что она не знает»

Этот миф моложе остальных и потому коварнее. Звучит он разумно: «пусть модель сама скажет, если не уверена». На практике уверенность ответа не связана с его правильностью.

Механика простая. Модель выдаёт токены с вероятностями, и высокая вероятность означает «этот текст похож на то, что я видела в обучении», а не «это утверждение истинно». Фраза «согласно отчёту 2019 года» может быть сгенерирована с вероятностью 0,97 и при этом указывать на несуществующий отчёт: статистика языка подсказывает форму, а не содержание.

Отсюда два практических следствия. Первое: просьба «пометь, где не уверен» работает лишь частично — она меняет стиль, а не доступ к фактам. Второе: проверять надо не тон ответа, а утверждения, которые можно сверить. Единственная надёжная калибровка — внешняя: поиск по источнику, сверка с базой, тест на десяти известных случаях.

Бывает и обратное: модель верно указывает на нехватку данных и получает недоверие «она вечно отказывается». Диагностика одна и та же — сравните три ответа на один вопрос при разных формулировках. Если цифры расходятся, а тон остался ровным, вы имеете дело с генерацией, а не со знанием. Подробный разбор галлюцинаций и способов их ловить — в главе 7.

Как посчитать цену веры

Цену веры в миф можно оценить до того, как платить. Формула простая: часы переделки, умноженные на стоимость часа, плюс прямые расходы (подрядчик, подписки, лицензии), плюс стоимость упущенного окна, умноженная на вероятность, что оно закроется.

Пример на числах из кейса. Переделка — примерно 120 человеко-часов двух аналитиков по €45 в час, то есть €5 400. Прямые расходы — $18 000, или около €16 600. Упущенное окно — $140 000 недополученной маржи при вероятности 0,6, то есть $84 000. Итог: около €106 000 против $18 000, которые были на слайде.

Два вывода из этой арифметики. Первый: основная часть цены почти всегда не в подрядчике, а в переделке и окне — то есть в том, чего нет в смете. Второй: проверка за два часа стоит меньше одного процента этой суммы, и именно поэтому её так легко пропустить.

Считать до конца не обязательно. Достаточно назвать порядок величины перед решением: если цена ошибки на порядок больше цены проверки, проверять надо всегда, независимо от того, насколько убедителен выступающий.

Чем миф отличается от упрощения

Упрощение полезно, миф вреден, и внешне они похожи: оба сжимают сложное в одну фразу. Различаются они тремя признаками.

Первый: есть ли проверяемый критерий. Упрощение можно проверить. «Модель — как стажёр, который прочитал всю библиотеку, но ни дня не был в офисе» проверяется: дайте стажёру задачу с внутренними данными и посмотрите, что выйдет. Миф проверки не предполагает: «ИИ понимает бизнес» — как это опровергнуть, если непонятно, что считается пониманием.

Второй: названы ли границы применимости. Упрощение говорит, где ломается. Стажёр из примера хорош в пересказе и плох в решениях, требующих контекста, которого он не видел. Миф границ не имеет: он верен всегда и потому не говорит ничего.

Третий: есть ли цена. Упрощение признаёт, что за пользование надо платить: проверкой, обучением, временем. У мифа цена спрятана — обычно в другом месте и позже, как в кейсе с отчётом за $18 000.

Проверка на практике занимает минуту: подставьте в утверждение число и границу. «Модель сокращает чтение договора с 60 до 12 минут, но не заменяет юридическую оценку риска» — упрощение. «ИИ анализирует договоры» — миф, потому что ни числа, ни границы.

Полезно применять это и к самой книге. Всё, что здесь написано, — упрощения о механике, и у каждого есть граница. Там, где граница важна, она названа прямо.

Если вы новичок: почему ИИ звучит уверенно

Представьте очень начитанного человека, который никогда не говорит «не знаю». Он прочёл миллионы страниц и научился продолжать любую фразу так, как её обычно продолжают. Спросите его о том, чего он не читал, — он не растеряется, а сложит ответ из знакомых оборотов.

- Уверенный тон — это стиль, а не свидетельство правоты. Модель не различает «я знаю» и «я хорошо помню похожие слова».
- Ответ без источника проверить нельзя, поэтому считайте его черновиком, а не справкой.
- Если нужен факт, дайте документ и попросите цитату. Это единственный способ увидеть, откуда взялась фраза.
- Цифры, даты, имена, номера договоров и ссылки — первые кандидаты на ошибку.
- Спросить «ты уверен?» бесполезно: ответ изменится в сторону более вежливой формулировки, а не более верной.

Под капотом: как модель «знает» факты (и почему это тексты, а не знания)

Обучающий корпус — это триллионы токенов текста: книги, код, форумы, документация. Такой массив и называют big data в бытовом смысле — данные, которые невозможно прочитать, но можно статистически сжать. Токенизация режет текст на токены, и модель учится предсказывать следующий токен по предыдущим. Знания не хранятся отдельными записями вроде строки в базе; они распределены по весам как статистические связи между токенами.

- Поэтому «Москва — столица» восстанавливается устойчиво: связь встречалась миллионы раз. А номер вашего договора не восстанавливается никак — он не встречался ни разу.
- Поэтому же модель путает похожие сущности: два одноимённых отчёта, два рынка с близкими названиями, две даты рядом.
- Контекстное окно — сколько токенов модель видит за один раз. Всё, что не попало в окно, для неё не существует, даже если было в предыдущем сообщении.
- Заземление работает потому, что нужный текст попадает в окно и становится частью условия задачи. Поиск по документам (RAG) — способ подложить этот текст автоматически; подробно в главе 9.
- Эмбеддинг — числовой вектор смысла фразы; на нём держится поиск «по смыслу», а не по словам. Ограничение: близкие векторы не означают совпадающих фактов.

Пять вопросов к любому утверждению об ИИ

Процедура занимает пять минут и не требует технических знаний. Она годится и для презентации подрядчика, и для статьи в отраслевом издании, и для внутренней инициативы коллеги.

1. **Кто это утверждает и на каком основании?** Нужна роль и доступ к данным: «аналитик, обработал нашу выгрузку за март», «подрядчик, использовал открытые обзоры». Если основания нет — это мнение, и его место в разделе «гипотезы».

2. **На каких данных?** Объём, период, источник, чьи данные. Ответ «на больших данных» означает, что данных нет. Здесь же проверяется бенчмарк — фиксированный набор тестовых задач с метрикой, на котором сравнивают модели: он говорит о среднем качестве на похожих задачах, а не о вашем случае.

3. **Какой критерий результата?** Что считается хорошим ответом и кто это проверяет. Если критерий не назван, улучшение измерить нечем, и разговор идёт о впечатлениях.

4. **Что не сработало?** Любой честный отчёт содержит раздел ограничений. Его отсутствие — самый быстрый признак того, что перед вами пересказ.

5. **Сколько стоит проверка?** Часы, деньги, доступы. Если проверка дороже самого решения, утверждение не стоит принимать — его стоит отложить до момента, когда появится дешёвый способ сверить.

Вопрос
Что считать ответом
Красный флаг

Кто и на каком основании
Роль, доступ, конкретный документ
«Специалисты говорят»

На каких данных
Период, объём, владелец данных
«Большие данные», «все источники»

Какой критерий
Метрика и тот, кто проверяет
«Качество выросло» без числа

Что не сработало
Названное ограничение
Отчёт без раздела ограничений

Сколько стоит проверка
Часы и доступы
«Проверять не нужно»

Дерево решений: утверждение об ИИ попало к вам на стол

Если утверждение содержит цифру, которую вы не можете проверить, — оно идёт в «предположения», а не в решение. Дальше с ним работают отдельно.

Если проверить цифру можно за час, — проверьте одну, самую дорогую. Если она подтвердилась, у остальных резко растёт шанс быть верными; если нет, дальше читать не нужно.

Если утверждение не содержит цифр, а описывает возможность («ИИ может делать X»), — спросите про границу: где X ломается. Ответ «нигде» закрывает разговор.

Если граница названа и она вас устраивает, — просите демонстрацию на ваших данных, а не презентацию. Десять ваших случаев, критерий приёмки, счёт вслух.

Если демонстрация прошла, — считайте стоимость проверки на единицу работы. Пока она выше стоимости ошибки, решение не готово к внедрению, но готово к пилоту на ограниченном участке.

Если пилот прошёл и проверка дешевле ошибки, — расширяйте, но оставьте контрольную выборку: она понадобится, когда модель обновится.

Отдельная ветка — когда отвечать нечего. Если исполнитель отказывается назвать метод под предлогом коммерческой тайны, а сумма выше стоимости собственной проверки, разговор заканчивается. Не потому, что тайна невозможна, а потому, что непроверяемое утверждение нельзя купить: вы платите за впечатление, а риск остаётся у вас.

До / После: «покажите, как ИИ анализировал рынок»

Разница между двумя фразами — не в вежливости, а в том, кто и что обязан предъявить.

Плохо. «Покажите, как ИИ анализировал рынок». На эту просьбу можно ответить чем угодно: скриншотом чата, схемой пайплайна, словом «проприетарная методика». Формально ответ получен, решения нет. Именно так и прошли три недели в кейсе.

Хорошо. «Перечислите источники данных по каждому рынку: название, владелец, дата выгрузки, период. Опишите метод: как считалась доля рынка, какие допущения, чем проверяли. Покажите контрольную выборку из двадцати утверждений с источниками. Отдельным списком — что не подтвердилось и какие рекомендации вы считаете слабыми. Если источника нет, так и напишите: „оценка“».

Что изменилось, по пунктам.

1. **Проверка стала возможной до оплаты.** Список источников либо есть, либо нет; это видно за десять минут, а не через квартал.

2. **Появился формат ответа, а не только содержание.** Источник, метод, контрольная выборка — каждый пункт можно принять или отклонить отдельно.

3. **Легализована оценка.** Разрешение написать «оценка» снимает стимул выдумывать источник. Честная пометка дешевле красивой ссылки.

4. **Раздел «что не подтвердилось» стал обязательным.** Он превращает отчёт из витрины в рабочий документ и сразу показывает квалификацию исполнителя.

Ту же просьбу стоит переписать и для внутреннего случая: не «покажи, что ты сделал с помощью ИИ», а «покажи вход, шаги, выход и чем проверил». Это ровно то же разделение фактов, черновиков и решений, только в масштабе одной задачи.

Красные флаги: мифы в речи

- «Нейросеть проанализирует рынок» — она повторит среднее мнение из текстов; анализ начинается с вашей выгрузки.
- «Просто попросите её оптимизировать бюджет» — у неё нет вашего бюджета, ограничений и приоритетов.
- «Пусть ИИ ведёт переговоры с поставщиком» — ответственность за условия остаётся у человека, который подписывает.
- «Модель сама скажет, если не уверена» — уверенность в токенах не равна правильности факта.
- «Мы внедрили ИИ, качество выросло» — без метрики и базы сравнения это не утверждение, а настроение.
- «Нам нужен самый умный ИИ на рынке» — без задачи и критерия приёмки выбор модели не имеет смысла.

Что сделать сегодня

1. Возьмите одно свежее утверждение об ИИ — из рассылки, презентации, разговора с коллегой. Прогоните его по пяти вопросам и запишите ответы. Результат: заполненная строка в таблице. Измерение: сколько вопросов осталось без ответа; больше двух — утверждение не годится для решения.

2. Найдите миф, на который опираетесь вы сами, и разложите его на фактическую и эмоциональную части. Результат: одна фраза «я верил, что…, а проверяется это так…». Измерение: есть ли в формулировке число и граница.

3. Перепишите один свой запрос к подрядчику или коллеге по образцу «До / После». Результат: текст с источниками, методом, контрольной выборкой и разделом «что не подтвердилось». Измерение: число пунктов, которые можно принять или отклонить по отдельности; цель — не меньше четырёх.

4. Заведите карточку утверждения на одну цифру из текущего проекта: значение, источник, дата, кто проверил, что будет при ошибке. Результат: одна заполненная карточка. Измерение: время на заполнение; если больше десяти минут, вы держите цифру, источник которой сами не знаете.

Дальше — глава 4 о том, как превратить задачу в постановку: задача, данные, результат, критерий приёмки. Если миф — это вера без проверки, то постановка — это проверка, встроенная в работу. Механику предсказания подробно разбирает глава 5, цену ошибок и способы их ловить — глава 7, а как отличить полезное упрощение от вредной веры в собственных решениях — глава 1.

Живой слой. Обновляемые шаблоны, чек-листы и разборы под текущие версии моделей: github.com/ai-bez-mifov/book#ch03 — QR-код на вклейке и в электронной версии.

Глава 4. Как думать об ИИ: задачи, данные, результат

Паспорт главы. Обещание: превратить «попросить у ИИ» в формализованную задачу — что должно измениться, что модель увидит, что считается готовым результатом — и получать предсказуемый выход вместо лотереи. Время чтения: 15 минут. Результат: заполненная карточка задачи на одну работу этой недели и два числа — время «до» и время «после» вместе с проверкой.

В четверг в 21:50 Мира, совладелица пекарни и ресторана в Мельбурне, в третий раз за вечер переписывает таблицу смен на следующую неделю. Девятнадцать сотрудников, две зоны — пекарня открывается в четыре утра, витрина в семь, — и три человека, которые выходят только по субботам и воскресеньям.

Шесть часов в неделю уходит на эту таблицу. Не на выпечку и не на разговоры с гостями — на перебор вариантов, кто кого подменит в среду.

Цена ошибки измеряется не часами. В прошлом месяце расписание собрали в спешке, в субботу на смену не вышел пекарь, и пекарня открылась на два часа позже. Минус $340 выручки и два отзыва, которые потом неделю объясняли постоянным гостям.

Мира решила, что задача идеально ложится на ИИ. Правила есть, список людей есть, результат нужен в виде таблицы. Дальше последовали две неудачные попытки — и из них выросли все правила этой главы.

Кейс: 19 человек, 6 часов и две попытки «в лоб»

Данных не хватало не в объёме. Все девятнадцать имён, ставки, зоны и договорные ограничения лежали в одном файле. Проблема была в другом: эти правила никто и никогда не выгружал в явном виде. Часть жила в договорах, часть — в голове Миры, часть — в переписке, где сотрудники менялись сменами.

Первая попытка. Она вставила список имён и написала: «Составь расписание смен на неделю для этих людей». Ответ пришёл аккуратный: ровная таблица, приветливый комментарий, всё по делу. В таблице один сотрудник стоял в трёх сменах подряд — он учился на дневном и подрабатывал только по субботам, о чём в списке имён не было ни слова. Второй был поставлен в пекарню, хотя работал только на витрине.

Вторая попытка была умнее и провалилась иначе. Мира добавила правила текстом: «учитывай, кто когда может, не больше пяти смен, между сменами отдых». Модель эти правила учла — и добавила своё. В расписании появилось ограничение, которого не было ни в договоре, ни в разговоре: один сотрудник не ставился на пятницы. В ответе оно выглядело так же уверенно, как настоящие.

Две попытки стоили около трёх часов и дали нулевой результат: каждую таблицу пришлось переделывать вручную.

После второй попытки был ещё один заход: «сделай то же самое, но внимательнее, и проверь дважды». Ответ стал длиннее, тон — увереннее, ошибки — теми же. Слово «внимательнее» не включает никакого режима внимательности: у модели есть только постановка задачи, и если она не изменилась, не изменится и результат.

Третья попытка, на этот раз удачная, отличалась не текстом запроса, а подготовкой. Мира потратила сорок минут и разложила работу на четыре части.

Задача — одно предложение с глаголом и границей: «Составь недельное расписание для 19 сотрудников так, чтобы каждая из 21 смены была закрыта ровно одним человеком и ни одно ограничение из списка не нарушалось».

Данные — таблица на девятнадцать строк: имя, зоны, где человек умеет работать, доступные дни, максимум смен в неделю, минимальный отдых между сменами, договорные

запреты. Рядом вторая таблица: семь дней на три позиции — двадцать одна ячейка, которую надо закрыть.

Результат — форма, в которой выход принимается без перечитывания: таблица «день, позиция, имя» плюс строка проверки по каждой позиции. Не рассуждение о балансе нагрузки и не советы по мотивации — таблица.

Критерий приёмки — три проверки на минуту: все двадцать одна ячейка заполнены; ни у кого не больше пяти смен; ни у кого две смены не стоят ближе одиннадцати часов. Четвёртая проверка внешняя: ноль жалоб сотрудников за первую неделю.

Результат третьей попытки: 6 часов в неделю стали 50 минутами, из которых 35 — подготовка данных и 15 — проверка. Нарушений ограничений — ноль за одиннадцать недель.

Ограничение честное. Модель не знает, что во вторник двое поменялись сменами в мессенджере, и не знает, что у ученика через месяц экзамены. Каждую пятницу Мира обновляет колонку доступности вручную — семь минут. Три недели из пяти проходят без единой правки; в двух она меняет одну строку сама, потому что помнит то, чего нет в файле. *(собирательный кейс)*

Разбор кейса. Почему это сработало — механика, а не мораль.

1. Задача из «придумай расписание» стала «закрой двадцать одну ячейку по правилам». Придумывание оставляет модели свободу; закрытие ячеек — нет.

2. Ограничения переехали из прозы в колонки. Правило «отдых не меньше одиннадцати часов» в виде столбца с числом проверяется, а в виде фразы «учитывай отдых» — интерпретируется.

3. Появились данные, которые нельзя вывести из общих знаний: зоны, доступность, договорные запреты. Это и есть заземление (grounding) — привязка ответа к вашим документам вместо среднего по интернету.

4. Критерий приёмки стал проверкой, а не настроением. Три условия проверяются глазами за минуту, а скриптом — за секунду.

5. Цена проверки опустилась ниже цены ошибки. Пятнадцать минут сверки против $340 и двух отзывов — арифметика, которая не требует веры в технологию.

Тот же каркас, другие данные

Через месяц Мира применила ту же карточку к закупкам. Схема не менялась, подготовка заняла те же сорок минут, а результат оказался бесполезным.

Модель предложила заказывать муку, масло и фрукты «по среднему расходу за последние четыре недели». Средний расход был посчитан честно, по реальной выгрузке. Беда в том, что в выгрузке не было трёх вещей: сезонности — в декабре пекарня продаёт вдвое больше круассанов; условий поставщика — при заказе упаковками по двенадцать килограммов цена падает на 9 %; и размера склада, в котором помещается ровно столько, сколько помещается.

Ни одну из этих вещей нельзя вывести из четырёх недель продаж. Карточка работала правильно — она показала, что данных для неё нет. Это её главное свойство: она не создаёт результат, а выявляет, чем результат ограничен.

Полтора месяца эксперимент жил в ручном режиме: модель считала заказ, Мира пересчитывала, совпадение было случайным. Потом закупки вернули человеку.

Выигрыш оказался не там, где его ждали. За время эксперимента Мира впервые записала шесть правил, которые раньше держала в голове: кратность упаковки — двенадцать килограммов, порог склада, двухнедельная задержка у одного поставщика, запрет на закупку фруктов в понедельник. Шесть строк позже вошли в инструкцию для нового закупщика. Стоимость — 4 часа и ноль экономии; результат — знание, которое пережило эксперимент.

Отсюда правило на всю оставшуюся книгу: каркас отвечает на вопрос «можно ли это поручить», данные отвечают на вопрос «получится ли». Первое вы решаете сами, второе решает то, что лежит у вас на входе.

Формула: задача, данные, результат, приёмка

Четыре блока — не бюрократия, а четыре разных вопроса. Их путают постоянно: под «задачей» приносят формат результата, под «данными» — пожелание, под «критерием» — настроение.

Блок	**Вопрос**	**Ответ в одну строку**	**Что ломается, если блок пропущен**
Задача	Что именно должно измениться?	Закрой 21 смену по правилам	Модель делает «что-то похожее»
Данные	Что модель увидит?	Таблица сотрудников + таблица смен	Ответ средний по интернету
Результат	В каком виде принимаем?	Таблица «день, позиция, имя»	Красивый текст, который нельзя использовать
Приёмка	Как понять, что годится?	21/21, ≤ 5 смен, ≥ 11 часов	Спор о вкусе вместо решения

Задача отвечает на вопрос «что должно измениться». Не «сделай лучше», а «перепиши для аудитории без юридического образования», «извлеки из сорока счётов только даты и суммы», «найди в двух версиях договора расхождения». Глагол в задаче — половина результата.

Данные отвечают на вопрос «что модель увидит». Если ответ «ничего, она же умная», вы получите средний ответ по обучающему корпусу. Юрист в Варшаве, попросивший «оценить риски этого договора» без самого договора, получит лекцию о типичных рисках. С приложенным текстом — разбор своих пунктов.

Результат отвечает на вопрос «в каком виде принимаем». Формат — не украшение, а условие использования. Таблица вставляется в планировщик, JSON уходит в скрипт, список из пяти пунктов читается за минуту. Абзац на две страницы не вставляется никуда.

Приёмка отвечает на вопрос «как понять, что годится». Это единственный блок, который нельзя делегировать модели: если вы не можете назвать условие, по которому примете или отклоните выход, вы не заказывали работу — вы смотрели представление.

Есть ошибка, которая встречается почти в каждом втором проекте: критерий пишут после того, как увидели ответ. Сначала запрос, потом «неплохо, но вот тут не то», потом правка

критерия под то, что уже пришло. Через три итерации критерий описывает конкретный ответ, а не работу, и следующий запуск на новых данных снова даёт мусор.

Порядок важен по механической причине. Критерий, написанный заранее, определяет, какие данные вы приложите: если нужно «каждое утверждение с цитатой», придётся приложить документ, а не пересказ. Критерий, написанный после, подгоняет задачу под случайный результат.

Практический минимум — три условия, из которых хотя бы одно числовое: «все двадцать одна строка», «не больше 120 слов», «ни одного числа, которого нет в файле». Три условия проверяются быстрее, чем одно размытое «качественно», и, в отличие от него, дают право отказать.

Карточка задачи: один лист до первого запроса

Промпт — это весь текст, который уходит модели: постановка задачи, данные, правила и критерий приёмки вместе. Не отдельная удачная формулировка, а документ постановки. Карточка задачи — его письменная версия, один лист, который заполняется до первого запроса и хранится рядом.

Шесть полей. **Вход** — что именно прикладывается: файл, таблица, пять примеров, прошлогодняя версия. **Шаги** — порядок операций, если он важен: сначала отфильтровать недоступных, потом закрывать ячейки, потом проверять ограничения. **Выход** — формат и объём. **Критерий** — условия приёмки, лучше числовые. **Кто проверяет** — имя и время, которое на это отводится. **Что запрещено** — границы, за которые модель не имеет права выходить.

Заполненная карточка для расписания выглядит так.

{
"задача": "Составить недельное расписание: 7 дней x 3 позиции = 21 смена",
"вход": ["staff.csv: 19 строк, колонки name, zones, days_available, max_shifts, min_rest_h, contract_limits",
"slots.csv: 21 строка, колонки day, position"],
"шаги": ["исключить недоступных в этот день", "закрыть позиции по совпадению зоны",
"проверить три ограничения", "вывести таблицу"],
"выход": "таблица: day, position, name + строка проверки по каждой позиции",
"критерий": ["21/21 смен закрыто", "у каждого <= 5 смен", "между сменами >= 11 часов"],
"кто_проверяет": "Мира, пятница, 15 минут",
"запрещено": ["менять список имён", "придумывать ограничения, которых нет в staff.csv",
"сокращать смены", "ставить человека в зону, которой нет в его zones"]
}

Два поля делают почти всю работу, и оба обычно пропускают. «Запрещено» отменяет фантазию в той части, где она дороже всего: модель не имеет права править входные данные, чтобы решение сошлось. «Кто проверяет» превращает проверку из намерения в расписание — с именем и минутами.

Есть и побочный эффект, который замечают все, кто довёл карточку до конца: часть задач умирает на этапе заполнения. Если в поле «приёмка» нечего написать, кроме «чтобы мне понравилось», задача не готова — и модель тут ни при чём. Это относится и к задачам, которые вы никогда не отдадите машине: карточка на них всё равно полезна, потому что описывает работу.

Если вы новичок: как объяснить задачу, чтобы вас поняли

Представьте, что вы просите стажёра, который никогда не был в вашей отрасли, но очень старается. Он не знает слов «стандартная смена» и «наш обычный клиент». Он понятия не имеет, что для вас очевидно.

- Скажите, что должно получиться на выходе: таблица, список, письмо, пять пунктов. Формат — первое, что нужно назвать.
- Скажите, что даёте: «вот файл, вот прошлая версия, вот правила». Если данных нет, модель придумает их.
- Скажите, что нельзя: не менять цифры, не добавлять факты, не сокращать имена.
- Скажите, как вы поймёте, что годится: «все двадцать одна строка заполнены», «не длиннее одной страницы».
- Один запрос — одна работа. Если в просьбе три разных дела, разделите её на три.

Это не особый приём, а обычная постановка задачи. Стажёру вы объясняете то же самое, и по тем же причинам.

Какие задачи ложатся на модель, а какие нет

Есть короткий список глаголов, которые ложатся почти всегда: сжать, переформатировать, сменить тон, сравнить две версии, извлечь поля, разложить по категориям, перевести, придумать N вариантов, отсортировать по заданному критерию, найти противоречия, переписать под конкретную аудиторию, собрать план по каркасу.

И есть список того, что не ложится: посчитать точную сумму, узнать факт, появившийся после обучения, понять мотивы конкретного человека, взять на себя ответственность за решение, выбрать лучший вариант, когда критерия нет.

Разница между списками одна и её полезно проговорить: модель сильна там, где работа уже кем-то сделана в виде текста и её надо преобразовать. Она слаба там, где нужно добавить знание, которого на входе нет.

Отдельно стоит сказать про задачи, которые только выглядят одной. «Сделай отчёт по продажам за квартал» — это три работы: извлечь числа из выгрузки, сравнить с прошлым кварталом, написать вывод на страницу. У каждой свой результат и своя цена ошибки. В одном запросе они смешиваются, и модель делает ту часть, которую умеет лучше, — пишет складный вывод на выдуманных числах.

Правило деления простое: если части можно проверить по отдельности, их нужно и ставить по отдельности. Три запроса, три выхода, три проверки — и на каждом шаге видно, где именно появилась ошибка. Один большой запрос даёт один большой ответ, в котором искать причину ошибки почти так же дорого, как делать работу заново.

Задача	**Данные**	**Результат**	**Приёмка**
Разобрать 200 анкет конкурса	анкеты + правила конкурса	таблица противоречий	каждое противоречие с цитатой
Ответить на 40 писем клиентов	письма + база ответов	40 черновиков	ни одного факта вне базы

Свести три версии договора
три файла
список расхождений
каждая строка с номером пункта

Подготовить отчёт по продажам
выгрузка + шаблон прошлого месяца
таблица и вывод на страницу
цифры совпадают с выгрузкой

Составить программу обучения
описание роли + список навыков
8 модулей с целями
каждый модуль проверяем за 10 минут

Проверка перед стартом — четыре вопроса подряд. Если ответ на первый — «нет», дальше идти не нужно.

1. Работа сводится к преобразованию текста, который у меня есть? Если нет — нужен другой инструмент: поиск, база, калькулятор, человек.

2. Могу ли я назвать критерий приёмки в трёх условиях? Если нет — сначала пишется критерий, потом запрос.

3. Есть ли у меня данные в явном виде — файл, таблица, примеры? Если только «в голове», сначала выгружаю: без этого шага модель будет додумывать.

4. Проверка займёт меньше времени, чем работа руками? Если нет — задача не стоит автоматизации, даже если технически решается.

Дерево короче: если работа — преобразование имеющегося текста и у вас есть три условия приёмки, отдавайте модели с проверкой. Если нужен новый факт, точное число или решение с ценой ошибки — модели отдаётся только подготовка материала, а решение остаётся человеку. Если критерий приёмки не формулируется вовсе — задача не поставлена, и никакая модель этого не исправит.

Данные: четыре уровня, на которых работает модель

Один и тот же вопрос можно решить на четырёх разных уровнях, и они различаются не качеством, а ценой и сроком.

Корпус. Всё, что модель видела при обучении. Это общие знания человечества, усреднённые. Доступно всем, стоит ноль, обновляется раз в несколько месяцев и не содержит вашей внутренней информации.

Контекст. То, что вы положили в запрос. Единственный уровень, который вы контролируете полностью: файл, таблица, три прошлых письма, выдержка из регламента. Работает сразу, стоит копейки, ограничен размером окна.

Поиск по вашим документам (RAG). Модель сначала ищет фрагменты в вашей базе, потом отвечает по найденному. Подробно — в главе 9. Нужен, когда документов больше, чем помещается в запрос, и когда они меняются каждый день.

Дообучение (fine-tuning). Вы доучиваете модель на своих примерах, чтобы она стабильно воспроизводила ваш стиль или формат. Дорого, долго, требует сотен чистых примеров. Подробно — в главе 6.

Практическое правило: начинайте с контекста. Девять из десяти задач, для которых ищут «свою модель», решаются приложенным файлом и тремя примерами. Дообучение имеет

смысл, когда нужен стиль или формат, а не знания, и когда примеров набралось хотя бы несколько сотен.

Что при этом забывают положить в данные. Прошлую версию документа — она задаёт структуру лучше любого описания формата. Три примера «вход — желаемый выход», включая один плохой, который надо переделать. Правила в явном виде, даже те, что кажутся очевидными: «сокращения не расшифровываем», «имена не переводим», «итог округляем до целого». И границы применимости: до какого размера файла всё это работает и что делать, когда он больше.

Шесть правил, записанных один раз, экономят больше, чем любая настройка модели. Они переиспользуются во всех следующих запросах и переживают смену версии — в отличие от приёмов, которые работали в конкретном интерфейсе прошлой весной.

Критерий приёмки: формат плюс критерий плюс шкала

Критерий пишется в три слоя. **Формат** — что физически должно быть: таблица с колонками, список из пяти пунктов, письмо до 120 слов. **Критерий** — что делает содержимое правильным: все строки заполнены, ни одного числа не из головы модели, каждое утверждение с цитатой. **Шкала** — как сравнивать варианты, если их несколько: «оцени каждый по трём условиям и покажи, какой нарушен».

Плохой критерий	Почему не работает	Рабочая замена
«Сделай красиво»	«красиво» живёт только в вашей голове	«Заголовок до 60 знаков, три абзаца, без прилагательных»
«Чтобы было понятно»	понятно кому — не сказано	«Поймёт человек без юридического образования за 60 секунд»
«Как в прошлый раз»	модель не помнит прошлый раз	«Как в файле template.docx, структура и порядок разделов те же»
«Качественно и профессионально»	нечего проверять	«Все факты из приложенного отчёта, ни одного нового числа»
«Учти все нюансы»	означает «сделай хорошо» другими словами	«Отдельно перечисли: сроки, штрафы, условия расторжения»

Проверка критерия простая: покажите его человеку, который не видел вашу задачу, и спросите, сможет ли он отличить хороший результат от плохого. Если не сможет — критерий не готов.

Шкала нужна там, где вариантов несколько. Работают два способа. Порог — «принимаю, если выполнены все три условия, иначе отклоняю целиком»; он хорош для рутины, где нужен ноль брака. Сравнение — «сгенерируй три варианта, оцени каждый по трём условиям, покажи, какое нарушено, и предложи один»; он хорош для текстов, где важен выбор, а не единственно

верный ответ. Смешивать их нельзя: если вы просите три варианта, у вас должен быть способ выбрать, иначе выбор сделает модель, а критерий останется вашим.

Верификация — это и есть проверка выхода по независимому правилу или источнику, а не по ощущению «нравится». Дешёвая верификация выглядит скучно: пересчитать число строк, найти цитату поиском по документу, сверить три даты с оригиналом. Дорогая — читать всё заново. Разница между ними — это и есть разница между работающим процессом и ритуалом.

Сериализация — превращение ответа в строгий формат (JSON, CSV, таблица с фиксированными колонками), который принимает программа или таблица без перечитывания. Она экономит не время модели, а ваше: результат в формате проверяется механически, результат в свободном тексте — только чтением.

Под капотом: параметры ответа и почему формат важнее вежливости

Параметры ответа — настройки генерации, которые вы или продукт задаёте перед запросом. Температура управляет разбросом: при низкой модели разрешено выбирать почти всегда самое вероятное продолжение, при высокой — заметно менее вероятное. Предел длины обрезает ответ, и обрезка почти всегда попадает в середину таблицы, а не в конец. Стоп-последовательности останавливают генерацию на нужном маркере, например на закрывающей скобке JSON.

Для задач с приёмкой температура около нуля и явный формат дают воспроизводимость: два запуска на одних данных дают почти одинаковый выход, и расхождение можно объяснить, а не гадать. Для задач «придумай десять вариантов названия» высокая температура полезна: разброс — это и есть работа.

Отдельно про формат. Когда вы просите «таблицу с колонками день, позиция, имя», модель генерирует разделители и заголовки теми же токенами, что и содержание. Ошибка в одном разделителе ломает разбор всей таблицы. Поэтому в инженерных задачах формат задают схемой (structured outputs) — модель обязана вернуть поля по объявленной структуре, и выход проверяется автоматически. Разметку и синтаксис разбирает глава 8.

Почему конкретика убирает фантазию

Техническая цель, которую вы достигаете, разделяя задачу, данные, результат и приёмку, — уменьшать энтропию, то есть меру неопределённости на выходе. Модель на каждом шаге выбирает следующий фрагмент из распределения вероятностей. Чем шире распределение, тем больше разных продолжений она может выдать на один и тот же запрос.

Каждая деталь постановки это распределение сужает. «Сделай расписание» оставляет тысячи допустимых продолжений. «Закрой двадцать одну ячейку, вот девятнадцать строк с ограничениями, у каждого не больше пяти смен» оставляет десятки — и почти все они правильные.

Отсюда три следствия, которые видно в любой практике. Конкретика снижает разброс не потому, что модель «лучше понимает», а потому, что у неё меньше свободы. Примеры работают сильнее инструкций: два-три образца «вход — желаемый выход» сужают распределение резче, чем абзац объяснений. И наоборот: чем менее определённы данные, тем увереннее модель заполняет пустоты — уверенность тона не связана с правильностью, об этом подробно в главе 7.

Есть и обратная сторона, о которой честно стоит сказать. Слишком узкая постановка экономит время модели и отнимает ваше: карточка на двадцать строк пишется дольше, чем разовая работа руками. Поэтому полную карточку доводят до конца для задач, которые повторяются хотя бы раз в месяц, а для разовых хватает трёх полей — вход, выход, приёмка.

Под капотом: откуда берётся разброс на одних и тех же данных

Модель не хранит ответ, она строит его заново при каждом запуске. Даже при температуре около нуля выбор между почти равновероятными продолжениями может качнуться из-за длины контекста, порядка абзацев и мелочей в форматировании. Плюс сам контекст имеет значение: чем длиннее вход, тем слабее внимание к его середине, поэтому правила лучше держать в начале и в конце, а не в середине простыни.

Практический вывод для постановки: одни и те же данные, поданные таблицей с явными колонками, дают заметно более устойчивый выход, чем те же данные, пересказанные абзацем. Дело не в «понятности» для модели, а в том, что структура уменьшает число допустимых прочтений входа.

Отсюда же приём, который экономит часы: попросите модель сначала перечислить, что она поняла из ваших данных, и только потом делать работу. Расхождение в этом перечислении — самый дешёвый способ поймать ошибку постановки до того, как она размножится на двадцать строк результата.

До / После: «сделай расписание»

До. «Составь расписание смен для моей пекарни на следующую неделю. У меня 19 сотрудников, нужно закрыть пекарню и витрину. Сделай хорошо, чтобы всем было удобно».

Что происходит: модель придумывает и состав сотрудников, и их доступность, и правила. В ответе появляются имена, которых нет, ограничения, которых не было, и бодрый комментарий про баланс. Проверить нечего: критерия нет, данных нет, формат — свободный текст.

После. «Ты — администратор, который составляет график для пекарни и ресторана. Задача: закрыть 21 смену на следующую неделю — 7 дней, 3 позиции: пекарь, витрина, касса. Данные: staff.csv — 19 строк, колонки name, zones, days_available, max_shifts, min_rest_h, contract_limits; slots.csv — 21 строка с колонками day, position. Правила: человек ставится только в зону из своего zones; не больше max_shifts смен; между сменами не меньше min_rest_h часов. Запрещено: менять список имён, добавлять ограничения, которых нет в файле, оставлять пустые ячейки. Формат вывода: таблица day, position, name, затем строка проверки по каждой позиции. В конце — три условия приёмки с фактическим результатом: 21/21, max_shifts ok/нарушено, min_rest ok/нарушено. Если какая-то ячейка не закрывается по правилам — не придумывай человека, напиши „нет решения“ и объясни, какое ограничение мешает».

Что изменилось, по пунктам.

1. Появился вход в виде файлов со схемой. Модель больше не выбирает, кого из девятнадцати взять, и не додумывает доступность.

2. Правила стали проверяемыми. «Не больше пяти смен» можно посчитать, «чтобы всем было удобно» — нельзя.

3. Появился запрет на подгонку. Без него модель правит данные, чтобы решение сошлось, и вы получаете аккуратную таблицу с неверными входными данными.

4. Появился формат и встроенная проверка. Три условия печатаются в конце ответа, и пятнадцать минут сверки превращаются в тридцать секунд чтения.

5. Появился выход для тупика: «нет решения» вместо выдуманного человека. Это самое дешёвое изменение во всём запросе и самое недооценённое.

Обратите внимание, чего в новом запросе нет: ни одного прилагательного о качестве, ни просьбы постараться, ни объяснений, зачем это нужно. Всё оставшееся — проверяемые условия. Поэтому такой запрос переиспользуется через месяц без изменений: он описывает работу, а не настроение дня.

И ещё одно отличие, которое легко пропустить. Новый запрос длиннее, но не потому, что в нём больше вежливости: он длиннее за счёт данных и запретов. Длина сама по себе ничего не

улучшает. Улучшает количество проверяемых условий на сотню слов — и здесь их примерно в десять раз больше.

Красные флаги: задачи, которые нельзя формализовать

- Вы не можете назвать три условия, по которым примете результат. Значит, задача не поставлена, и любой ответ будет «вроде неплохо».
- Критерий звучит как «чтобы понравилось руководству». Вкус нельзя проверить, а значит, нельзя и перепоручить.
- Модель просят «оценить человека», «понять клиента», «предсказать решение совета». Мотивы конкретных людей не выводятся из текста, если этого текста у вас нет.
- Ни один файл не приложен, а в задаче есть ваши внутренние правила, цены или имена. Модель заполнит пустоты средним по корпусу, и вы этого не заметите.
- Проверка занимает больше времени, чем работа руками. Тогда вы автоматизируете не работу, а своё недоверие.
- Результат уходит наружу без человека: клиенту, регулятору, в подписанный отчёт. Цена ошибки платится пачкой, а не по одной.

Что сделать сегодня

1. Заполните карточку задачи на одну работу этой недели. Шесть полей: вход, шаги, выход, критерий, кто проверяет, что запрещено. Результат: один лист на одну задачу. Измерение: если поле «критерий» пустое дольше пяти минут — задача не готова, начните с него, а не с запроса.

2. Перепишите критерий приёмки в три слоя — формат, критерий, шкала. Результат: три строки, каждая проверяема без вас. Измерение: покажите критерий человеку, который не видел задачу; если он не может отличить хороший выход от плохого, переписывайте.

3. Отдельно выпишите список запретов. Результат: 3–5 строк «нельзя менять», «нельзя добавлять», «нельзя сокращать». Измерение: прогоните прошлый запрос с этим блоком и без него; считайте число выдуманных правил и пропущенных полей, цель — ноль.

4. Задайте два уровня данных: контекст (файл в запросе) и примеры (2–3 образца «вход — желаемый выход»). Результат: один запрос, собранный по карточке. Измерение: сравните с прежним ответом по трём условиям приёмки; если формат поплыл, добавьте пример, а не абзац объяснений.

5. Замерьте время: «до» — как вы делали эту работу руками, «после» — запрос плюс проверка. Результат: два числа. Измерение: если разница меньше 20 %, проверка съедает весь выигрыш, и задачу надо либо упростить, либо вернуть человеку.

Формула из этой главы возвращается дальше четырежды: в главе 6 — про то, откуда берутся данные и где кончаются возможности модели; в главе 8 — про то, как эта постановка превращается в текст запроса; в главе 13 — про то, как собрать из карточки рабочий сценарий за полчаса; в главе 14 — про то, как измерить качество и стоимость получившегося. Матрица вердиктов «сразу / с проверкой / нельзя» из главы 1 — это тот же каркас, только с другой стороны: там вы решали, кому отдать работу, здесь — как её описать, чтобы её можно было принять.

Живой слой. Обновляемые шаблоны, чек-листы и разборы под текущие версии моделей: github.com/ai-bez-mifov/book#ch04 — QR-код на вклейке и в электронной версии.

ЧАСТЬ II. Как это работает без формул

Глава 5. Модель — это не база знаний, а функция предсказания

Паспорт главы. Обещание: увидеть в модели один механизм — предсказание следующего фрагмента — и вывести из него, когда она надёжна, когда врёт уверенно и во что обходится каждый ответ. Время чтения: 15 минут. Результат: два режима, разложенные по вашим задачам, порог включения режима рассуждений и посчитанная стоимость одного типового запроса — по формуле, а не «примерно».

В 01:10 по берлинскому времени Айла смотрит на красный CI — систему непрерывной сборки, которая гоняет тесты на каждый коммит. Деплой встал на третьем шаге из семи: тест, который восемь лет проверял разбор ответа платёжного провайдера, падает с KeyError: 'status'.

Файл billing_router.py — 1 400 строк, восемь лет истории, 38 тестов и 44 % покрытия. Айла попросила модель: «Перепиши весь файл целиком, код устарел, сделай чисто и современно». Ответ пришёл через 50 секунд: 900 строк, аккуратные импорты, docstring (строка документации) у каждого метода, обработка ошибок в каждом втором блоке.

Код запускался. Семнадцать тестов из тридцати восьми упали. Три обработчика — таймаут, частичный ответ, повторное списание — исчезли вместе с ретраями. В файле появился вызов requests.post, которого в проекте нет три года, и эндпоинт /v2/refund, которого нет у провайдера.

Ни одна строка в ответе не выглядела выдуманной. В этом и состоит главное свойство модели: у неё нет отдельного режима «вспомнить» и отдельного «придумать». Есть один механизм, и он одинаково уверенно выдаёт и то, что было в обучающих текстах, и то, чего не было никогда.

Кейс: «перепиши весь файл» и 89 сэкономленных часов

billing_router — внутренний сервис разбора ответов платёжного провайдера: 14 000 заказов в сутки, четыре типа событий, три валюты. Восемь лет назад его написали за две недели, с тех пор правили одиннадцать раз, каждый раз «быстро, до релиза». Оценка миграции на новую версию внутреннего HTTP-клиента, где убрали синхронную обёртку, а ретраи переехали в middleware — промежуточный слой обработки запросов: 120 часов, один разработчик, три недели календарно.

Цена задержки была известна заранее. Старая версия клиента требовала ручного патча безопасности раз в месяц: шесть часов инженера и $700 на внешнюю проверку. Окно миграции закрывалось в конце квартала.

Первая попытка заняла две строки промпта и пятьдесят секунд ожидания. Модель выдала 900 строк, файл стал короче на треть, и это был самый опасный сигнал: «стало чище» не значит «стало верно». Три часа ушло на ревью. Из них сорок минут — на поиск того, чего в новом файле нет: исчезнувшие обработчики не видны, их отсутствие ничем не выдаёт себя. Ещё шесть тестов падали из-за переименованных публичных функций: parse_response стал parse_payment_response, а вызывали его два других сервиса, которые в промпте не упоминались.

Вторая попытка была умнее и провалилась иначе. Айла разбила файл на пять частей по 250–300 строк и причесала каждую отдельно. Каждая часть прошла ревью. На стыках обнаружилось шесть несовместимостей: два разных формата ошибок, две независимые реализации ретраев, три способа логирования и поле amount, которое в одном куске было числом, а в другом строкой. Интеграция заняла четыре дня — почти столько же, сколько заняла бы миграция руками.

Две неудачные попытки обошлись примерно в $2 450: три часа ревью и четыре дня интеграции по ставке $70 в час. Это не цена технологии, а цена постановки: работа была сделана дважды, потому что первый запрос описывал намерение, а не работу.

Правильный подход отличался не формулировкой, а подготовкой. До первого запроса Айла сделала четыре вещи.

Прогнала 38 существующих тестов и дописала 12 новых — на таймаут, частичный ответ и повторное списание. Тесты стали контрактом: то, что нельзя сломать, теперь проверялось машиной, а не глазом.

Описала контракт: публичные сигнатуры не меняются, внутренний клиент не заменяется на сторонний, схема ошибок остаётся прежней, на вход идёт функция, а не файл, ответ — патч не длиннее 120 строк.

Разложила миграцию на одиннадцать задач и отметила четыре, где шагов много: восстановление логики ретраев, разбор противоречий в спецификации провайдера, порядок миграции двух связанных сервисов, отладка таймаута, который воспроизводился раз в двести запросов.

Задала чек-лист ревью из двенадцати пунктов — по одному на каждый прошлый инцидент.

Итог. Миграция заняла 31 час вместо 120 — 26 % от исходной оценки. Стоимость токенов выросла с $0.86 до $2.94, в 3,4 раза: режим рассуждений включали только для четырёх задач из одиннадцати, но каждая из них требовала второй попытки и 6–9 тысяч токенов размышления. В деньгах это три сотых процента от сэкономленных 89 часов: по ставке $70 в час экономия около $6 230, за вычетом шести часов на тесты и карточку — около $5 800. *(собирательный кейс)*

Отдельно стоит объяснить, откуда взялись 26 %. Айла считала не «ощущение скорости», а человеко-часы: подготовка тестов и карточки — 6 часов, одиннадцать задач с ревью — 22 часа, два возврата к человеку — 3 часа. Итого 31 час против оценки 120. Ревью каждого патча занимало 35–40 минут и не сокращалось: 120 строк патча читаются быстрее, чем 900 строк файла, но проверить их надо целиком, а не по диагонали. Экономия пришлась на переписывание, а не на проверку, и это типичное распределение: подготовка и ревью остаются человеческими, машинным становится только набор текста.

Ограничение честное. Две из четырёх «умных» задач вернулись к человеку. В спецификации провайдера было противоречие между разделом про идемпотентность (свойство операции при повторе не менять результат) и примером запроса; модель выбрала наиболее вероятное прочтение — и оно оказалось неверным. Нашёл это человек на ревью, за девяносто секунд, потому что в чек-листе был пункт «сверить идемпотентность с примером». Ещё одно: новые тесты проверяли поведение, но не производительность; регрессию по задержке поймали на нагрузочном стенде, уже после мержа.

Разбор кейса. Почему это сработало — механика, а не мораль.

1. «Перепиши файл» — это просьба открыть новое. «Предложи патч к функции, вот тесты» — просьба переупаковать известное. Второе модель делает на порядок надёжнее.

2. Тесты, написанные до запроса, превратили критерий приёмки в машину. 17 упавших тестов — не мнение ревьюера, а факт.

3. Запрет менять сигнатуры и зависимости убрал самый дорогой класс ошибок: те, что не видны в списке изменений и всплывают в другом сервисе через неделю.

4. Режим рассуждений включили не «для качества вообще», а для четырёх задач с проверяемым итогом. Остальные семь шли быстрым режимом, и это сэкономило не деньги, а минуты ожидания.

5. Шесть часов подготовки против 89 сэкономленных — соотношение, которое повторяется почти в каждом проекте, где подготовку не пропустили.

Библиотекарь, который не ходит к полкам

Представьте библиотеку с очень толковым и очень странным библиотекарем. Он прочитал всё: миллионы статей, книг, форумов, инструкций. Но работает не как библиотека. На вопрос «где лежит книга про X» он не идёт к полке. Он начинает говорить — фразу за фразой, — восстанавливая, как обычно звучит ответ на такой вопрос.

Формально модель — это функция. На вход подаётся последовательность токенов (кусков текста длиной примерно в 3–5 символов), на выходе получается распределение вероятностей для следующего токена. Один токен выбирается, приписывается к входу, и всё повторяется. Прогон обученной модели в момент запроса называется **инференсом** (в отличие от обучения, которое было раньше и один раз). Внутри — миллиарды чисел, **весов**, настроенных обучением; лежат они в **тензорах**, многомерных числовых массивах.

Слово «предсказание» здесь не метафора. Модель училась на частотах: как часто за последовательностью из N подряд идущих токенов — **N-граммой** — шёл тот или иной следующий токен. Из этих частот выросли обобщения, но механизм остался тем же. Поэтому у факта и у вымысла в модели один и тот же путь: «какое продолжение вероятнее», а не «что верно».

Отсюда расходится главная развилка, на которой ломается интуиция. База знаний отвечает на вопрос «что там лежит» и умеет сказать «такой записи нет». Функция предсказания отвечает на вопрос «что вероятнее дальше» и не имеет режима «не отвечать»: отсутствие записи для неё неотличимо от редкой записи. Когда модель придумывает эндпоинт /v2/refund, она не галлюцинирует в бытовом смысле — она продолжает правдоподобно.

Это же объясняет, почему поиск по документам (RAG, retrieval-augmented generation — поиск, дополняющий генерацию) работает и почему он не «лечит» модель, а подставляет ей текст. Модель остаётся функцией предсказания: меняется только вход. Отсюда и разница между этой главой и главой 6: они описывают два конца одной трубы.

Три следствия, которые видно в любой переписке

Следствие первое: модель даёт среднее по своей памяти, а не крайнее по правде. Попросите охарактеризовать рынок электронной коммерции — получите гладкое, вежливое, всеобъёмлющее описание, точное ровно в общих чертах. Приложите сорок интервью с клиентами и таблицу продаж — получите такое же гладкое описание, но про ваш случай. Разница между «средним» и «вашим» почти всегда определяется тем, что вы дали на входе.

Следствие второе: уверенность не связана с правильностью. Модель обучена писать так, как пишут люди: уверенно. Редкий факт и широко известный оформляются одним и тем же тоном, потому что вероятность продолжения в обоих случаях может быть высокой. Интонация не несёт информации о надёжности. Вес ответственности определяете вы, а не тон ответа.

Проверить это можно на любом справочном вопросе. Спросите модель про статью закона, номер которой вы точно знаете: ответ будет таким же уверенным, как про статью, которую вы не проверяли. Иногда номер верный, иногда — правдоподобный соседний. Отличить один случай от другого по тексту нельзя; отличает только сверка.

Следствие третье: переупаковка даётся легко, открывание — трудно. Пересказать, сменить тон, сжать, разложить по рубрикам, привести десять вариантов, преобразовать формат — это работа с уже поданным текстом, и здесь модель сильна. Сосчитать длинную сумму, вспомнить событие после обучения, восстановить редкую деталь, за которую никто не отвечал в обучающих текстах, — здесь она выдаёт правдоподобное вместо верного.

Что просите

Насколько надёжно
Чем проверять

Пересказ, смена тона, сжатие, перевод
Высоко: вход есть, выход ограничен
Сверить с исходником выборочно, 5 фрагментов

Извлечение полей в формат
Высоко при явной схеме
Схема, подсчёт строк, выборочная сверка

Классификация по вашим правилам
Средне: правила надо дать
Матрица ошибок на 50 размеченных примерах

Числа, проценты, итоги по колонке
Низко
Пересчёт формулой или кодом

Редкие факты, даты, номера, имена
Низко
Первоисточник, не модель

События после обучения
Нулевая без вашего контекста
Дата документа и внешний источник

Отсюда практическое правило: модель — это дешёвая обработка на очень большой памяти. Проверка всегда на вашей стороне и состоит из трёх вопросов.

1. **На каких данных я проверяю ответ?** Если ответа не с чем сверить, вы не проверяете, а соглашаетесь.
2. **Какой источник я могу открыть?** Не «модель сказала», а документ, база, первоисточник.
3. **По какому критерию я приму или отклоню?** Формат, число, отсутствие выдуманных полей.

Если вы новичок: почему модель не «знает», а «предсказывает»

- Она не хранит факты как картотека. Она хранит закономерности текста: что за чем обычно идёт.
- Поэтому на вопрос о вашей компании она ответит «как обычно бывает в таких компаниях», если вы не приложили свои документы.
- Поэтому же она не различает «я уверен» и «я угадываю»: у неё нет отдельного сигнала «этого я не знаю».
- Поэтому редкое и частое звучат одинаково. Проверять надо не по тону, а по источнику.
- Практический вывод: чем более редкое и конкретное знание вам нужно, тем больше текста вы обязаны положить на вход. И тем меньше остаётся на «она сама знает».

Два режима: ответить сразу или подумать

До сих пор речь шла о модели, которая отвечает сразу: получила запрос — и выдаёт токен за токеном до конца. Так работает быстрый режим в любом чате, и у него есть цена: на длинной цепочке ошибка накапливается, потому что каждая следующая фраза строится из предыдущей, а не из проверки.

С 2024–2025 годов в практику вошёл второй режим — **модели рассуждений**. Прежде чем выдать ответ, модель генерирует длинную внутреннюю цепочку шагов, черновик, который вы обычно не видите. На черновик тратится дополнительное вычисление, поэтому режим и называют **вычислениями во время ответа** (test-time compute): качество растёт не от новой версии модели, а от количества «подумать» в момент запроса.

Что это меняет на практике.

• **Многошаговые задачи становятся надёжнее.** Логика, планирование, отладка, разбор противоречий в документе — там, где быстрый режим срезает угол, черновик проходит шаги явно.

• **Ответ становится медленнее и дороже.** Иногда в разы: токены размышления тарифицируются как выходные, даже если вы их не видите.

• **Появляется новый тип ошибки — «уверенно, подробно и неверно».** Красивая цепочка вокруг неверной посылки выглядит убедительнее короткого ответа. Рассуждение не заменяет проверку фактов.

• **Простые задачи решаются хуже.** На «перепиши абзац» и «разложи список по алфавиту» выигрыша нет, а лишняя интерпретация появляется.

Тип задачи	**Режим**	**Как формулировать запрос**
Пересказ, перевод, смена тона, извлечение полей	быстрый	примеры «вход → выход», жёсткий формат
Классификация по готовым правилам	быстрый	список категорий, правило для спорных случаев
Расчёт, логика, планирование, поиск противоречий	рассуждение	данные, ограничения, критерий приёмки, просьба перечислить допущения
Анализ кода, проектирование решения	рассуждение	контекст системы, что нельзя менять, как проверять
Многошаговый разбор документа	рассуждение	сначала факты и цитаты, потом выводы
Правка 20 строк кода с тестами	быстрый	патч, тесты, запрет менять сигнатуры

Отсюда правило: **не просите «думать по шагам»**. Современная модель рассуждений делает это сама, а старая формула мешает — вы навязываете чужой формат и тратите контекст. Дайте то, чего у неё нет: цель и критерий приёмки, ограничения и данные, требование перечислить допущения, формат только для результата.

Обратное тоже верно: для стиля, формата и извлечения данных примеры работают сильнее рассуждений. Три образца «как надо» дадут больше, чем просьба подумать внимательно.

Проверять режимы на своих задачах лучше не вкусом, а тремя числами. Возьмите десять задач из прошлого месяца с известным правильным ответом, прогоните обе версии и запишите: сколько прошли проверку, сколько секунд до результата, сколько стоил прогон. Дальше решение принимается арифметикой, а не спором о том, чей ответ «глубже». В кейсе с миграцией такой прогон занял сорок минут и показал, что на семи задачах из одиннадцати разница в проверках была нулевой — то есть рассуждение там оплачивало только время ожидания.

Дерево решений: когда платить за размышление

Порог включения — не про «сложность вообще», а про четыре проверяемых условия. Если задача разбивается на три и более зависимых шага — читайте дальше. Если нет, включайте быстрый режим: рассуждение на одношаговой задаче только добавляет минуту ожидания.

Если шагов три и больше, спросите, можно ли проверить итог машиной — тестами, схемой, сверкой с таблицей. Если да, рассуждение обычно оправдано: у черновика появляется обратная связь, и он реже срезает угол. Если проверить можно только на глаз, рассуждение помогает слабее: оно даёт более длинный и более убедительный текст, а не более верный.

Дальше сравните две цены. Если ошибка стоит дороже пятнадцати минут ревью — включайте. Если ошибка исправляется за минуту, а ожидание длится две, выключите: вы платите временем за страховку, которая не нужна.

И последнее условие: если задача — переупаковка уже поданного текста, режим не нужен независимо от объёма. Длинный документ, который надо сжать, — это работа с формой, а не поиск решения. Здесь побеждает быстрый режим с хорошими примерами.

Сомневаетесь — прогоните обе версии на трёх примерах и сравните не тон, а число пройденных проверок и секунды до результата. Три запуска стоят дешевле недели привычки.

Из чего складывается стоимость ответа

В счёте за модель три слагаемых, и люди обычно думают только о втором. **Входные токены** — всё, что вы отправили: инструкция, документы, история диалога, примеры. **Выходные токены** — то, что модель написала. **Токены размышления** — черновик режима рассуждений; вы его не видите, но платите по цене выхода.

Формула проста:

стоимость = вход * цена_входа + (выход + размышление) * цена_выхода

Возьмём условные цены порядка величин на момент написания: $3 за миллион входных токенов, $15 за миллион выходных. Документ на 60 000 токенов, вопрос на 200 токенов, ответ на 500.

Вход: 60 200 × 3 / 1 000 000 = $0.18. Выход: 500 × 15 / 1 000 000 = $0.0075.

Документ дороже ответа в двадцать четыре раза. Это первое, что стоит усвоить: в задачах с длинным контекстом платят за то, что вы приложили, а не за то, что модель написала.

Кэширование префикса меняет арифметику. Если начало запроса не меняется — один и тот же документ, одна и та же инструкция, — провайдер сохраняет вычисленное состояние и берёт за повторную часть 10–25 % цены входа. Двенадцать вопросов к одному документу без кэша: 12 × $0.18 = $2.16. С кэшем: $0.18 за первый, затем 11 × ($0.018 + $0.0075) = $0.28. Итого $0.46, в 4,7 раза дешевле — при том же документе и тех же ответах.

Задержка (latency) — время от запроса до последнего токена. Она складывается из обработки входа и генерации выхода: 60 000 токенов входа — это обычно 1,5–3 секунды до первого токена, а 500 токенов ответа при 70 токенах в секунду — ещё семь секунд. Токены размышления добавляются туда же: шесть тысяч токенов черновика — это плюс полторы минуты ожидания. Для интерактивной правки это много, для ночного пакета — нормально.

Что влияет на цену
Как это выглядит в жизни
Как снизить

Объём контекста
Документ на 60 000 токенов в каждом запросе
Приложить выдержку, а не весь архив

Повторная отправка того же
История диалога растёт с каждой репликой
Пересказывать историю своими словами, а не тащить целиком

Длина ответа
«Напиши подробно» на десять страниц
Ограничить объём и формат

Токены размышления
Режим включён «на всякий случай»
Порог из четырёх условий

Кэш префикса
Порядок блоков в запросе меняется
Постоянное начало, переменное — в конце

Повторы и ретраи
Три попытки на каждую задачу
Критерий приёмки вместо перебора

Класс модели
Пограничная модель на переименовании полей
Разные модели на разные задачи

Практический вывод для бюджета: считайте не цену одного запроса, а цену цикла «запрос — ревью — правка». В кейсе с миграцией один цикл на сложной задаче стоил около $0.60 вместе со второй попыткой, и это было в сто с лишним раз дешевле часа инженера. Дорого там, где контекст большой, а работа мелкая: типичный пример — «сколько строк в этом договоре упоминают штраф» при договоре на 80 000 токенов и ответе в одно слово.

Есть и вторая цена у длинного контекста, не денежная. Чем больше вход, тем слабее внимание к его середине: инструкция, лежащая в середине стотысячного документа, теряет вес. Поэтому длинный контекст — не бесплатное расширение памяти, а компромисс: вы платите за токены, за задержку и за то, что часть документа модель прочитает хуже. Дешёвый способ обойти это — не тащить весь архив, а сначала извлечь релевантные страницы поиском и приложить их.

Ещё одно следствие для планирования: пакетная обработка. Если ответ не нужен сию минуту и провайдер даёт пакетную скидку, двести запросов ночью обойдутся дешевле двадцати интерактивных днём. Разница не в модели, а в том, что вы не платите за ожидание и можете брать более дешёвый класс модели там, где качество проверяется автоматически.

Под капотом: токенизация, эмбеддинги, внимание, сэмплирование

Архитектура современных больших языковых моделей почти универсальна — трансформер. Упрощённо: миллиарды весов, разложенных по слоям, и каждый слой по-своему оценивает, какие фрагменты контекста важны для продолжения.

1. **Токенизация.** Текст режется на токены, чаще 3–5 символов. Одна и та же фраза на разных языках даёт разное число токенов: русский текст дороже английского и по цене, и по скорости. До нейросетей язык моделировали проще — считали частоты последовательностей из N слов, **н-грамм** (n-gram); именно оттуда пришла сама формула «предсказать следующее слово по предыдущим», которую сегодня считает трансформер.

2. **Эмбеддинги.** Каждый токен превращается в вектор чисел, где близость кодирует смысловую близость. Поэтому работает поиск по смыслу: «как повысить продажи» лежит рядом с «как увеличить выручку». Внутри модели эти векторы живут не поодиночке, а в многомерных массивах — **тензорах**, и вся арифметика сети — это операции над ними.

3. **Внимание** (attention). На каждом шаге модель присваивает вес каждому элементу контекста — больший или меньший. Практическое следствие: модель не «держит» весь документ, она выбирает, что считать ключевым. Поэтому порядок данных влияет: важное лучше держать в начале и в конце, а не в середине простыни.

4. **Сэмплирование.** На выходе — распределение вероятностей всех возможных токенов. **Температура** определяет, как выбирается кандидат: ближе к нулю — стабильнее и воспроизводимее, выше — разнообразнее. Для извлечения полей температура около нуля, для «придумай десять названий» — высокая.

5. **Нет состояния между запросами.** Каждый запрос начинается с нуля. «Помнишь, что мы говорили вчера» — нет, если этот диалог не подан в текущий контекст. Никакой личности и отношений технически не существует.

6. **Режим рассуждений — те же токены, потраченные на черновик.** Цепочка генерируется так же, как обычный текст, и тарифицируется как выход. Зато надёжность на многошаговых задачах растёт предсказуемо — пока задача не упрётся в неверную посылку или отсутствующий факт.

Под капотом: бюджет размышления и где он перестаёт помогать

Черновик — обычная генерация, только длинная. Модель не «думает дольше» в человеческом смысле: она выдаёт больше токенов, каждый из которых может быть неверным. Отсюда два ограничения.

Первое: помощь убывает. На задачах с проверяемым итогом (тесты, схема, арифметика кодом) рост бюджета даёт предсказуемое улучшение, а потом упирается в потолок. На задачах без проверки лишние шаги часто добавляют уверенности, а не правильности: модель перебирает правдоподобные варианты и выбирает самый гладкий.

Второе: черновик оплачивается как выход и занимает время. Шесть тысяч токенов размышления — это примерно полторы минуты и $0.09 при цене $15 за миллион. Если таких запросов двести в день, разница между «включено всегда» и «включено по порогу» — это часы ожидания и заметная строка в счёте.

Отсюда же происходит **дистилляция**: компактную модель обучают на выходах большой, включая удачные цепочки рассуждений. Получается дешёвый исполнитель для узкого класса

задач, где большая модель уже нашла рабочий ход. Дистиллят не «умнее» учителя, он повторяет его поведение на похожих входах — и на непохожих деградирует быстрее.

Открытая, пограничная, дистиллированная

Три класса моделей различаются не «качеством», а тем, что вы покупаете вместе с токеном.

Открытая модель — с опубликованными весами: её можно запустить на своём железе, дообучить, завернуть в закрытый контур. Цена за токен ниже, но вы платите инфраструктурой, инженером и ответственностью за обновления. Разумна там, где данные не должны покидать контур или запросов очень много и они однотипные.

Пограничная модель — самая крупная, доступная через API. Дороже, медленнее, лучше на редких и запутанных задачах. Её берут не «на всё», а на то, где ошибка дороже токенов: разбор противоречий, планирование, сложный код. Именно на таких редких случаях — их называют **гранями** (edge-case) — модель и проверяют перед покупкой: десять нормальных запросов ничего не покажут, а один странный покажет всё.

Дистиллированная — компактная, обученная на выходах большой. Даёт близкое качество на узком классе задач при цене в разы ниже и задержке в разы меньше. Хороша для массовой рутины: извлечение полей, классификация, переписывание по шаблону. Плоха там, где нужен новый ход, которого не было в обучающих примерах.

До / После: «перепиши файл целиком»

До. «Перепиши весь файл целиком, код устарел, сделай чисто и современно. Добавь обработку ошибок».

Что происходит: модель открывает новое там, где нужно переупаковать старое. Она выбрасывает то, чего не понимает, потому что в обучающих текстах «чистый современный код» выглядит без этих обработчиков. Она переименовывает функции, потому что «так чище». Она дописывает эндпоинт, потому что «так обычно бывает у провайдеров». Проверить нечего: критерия нет, тестов в запросе нет, границ нет.

После. «Ты — инженер, который переносит внутренний сервис на новую версию HTTP-клиента. Задача: изменить функцию handle_callback так, чтобы она работала с клиентом версии 2, не меняя поведения».

«Контекст: файл billing_router.py (1 400 строк, прилагаю), 38 тестов (прилагаю), внутренний клиент pay_client (прилагаю). Публичные сигнатуры handle_callback, parse_response, refund менять нельзя: их вызывают два других сервиса. Внутренний клиент нельзя заменять на сторонний: в нём ретраи с экспоненциальной паузой».

«Ограничения: не менять схему ошибок; не добавлять новых эндпоинтов; не удалять обработчики таймаута, частичного ответа и повторного списания; размер патча — не больше 120 строк».

«Формат ответа: патч в виде diff (список изменённых строк), затем список рисков из 3–5 пунктов, затем строка проверки: сколько тестов из 38 проходят, какие падают и почему. Если поведение нельзя сохранить — не придумывай обход, напиши „требуется решение человека“ и укажи, какое именно».

«Критерий приёмки: 38/38 тестов проходят, публичные сигнатуры не изменены, патч не длиннее 120 строк».

Что изменилось, по пунктам.

1. Появился вход в виде файлов и тестов. Модель больше не решает, что считать «устаревшим», — она видит границы работы.

2. Появился запрет на переупаковку в «новое». Три обработчика и внутренний клиент перечислены явно, и их исчезновение стало бы нарушением, а не творчеством.

3. Появился формат патча вместо файла. Патч читается за минуты и не прячет удалённые строки в общем объёме.

4. Появилась проверка на выходе: 38/38 и список рисков. Ревью сводится к сверке трёх чисел, а не к чтению 900 строк.

5. Появился выход для тупика: «требуется решение человека». Это самое дешёвое изменение в запросе и самое недооценённое — оно отделяет незнание модели от вашей задачи.

Чего в новом запросе нет: ни одного прилагательного о качестве, ни просьбы постараться, ни объяснений, зачем это нужно. Всё оставшееся — проверяемые условия. Поэтому такой запрос переиспользуется через месяц без изменений.

Красные флаги: язык, который выдаёт незнание

- «Нейросеть знает факты о вашей компании» — она знает тексты, которые вы ей дали. Без них будет среднее по интернету.

- «Модель запомнит на будущее» — в стандартном чате памяти между диалогами нет; вчерашний контекст надо подавать заново.

- «Она скажет то, чего вы не знали» — скажет, и проверить это будет нечем. Новизна ответа не признак знания.

Конец ознакомительного фрагмента.

Текст предоставлен ООО «Литрес».

Прочитайте эту книгу целиком, купив полную легальную версию на Литрес.

Безопасно оплатить книгу можно банковской картой Visa, MasterCard, Maestro, со счета мобильного телефона, с платежного терминала, в салоне МТС или Связной, через PayPal, WebMoney, Яндекс.Деньги, QIWI Кошелек, бонусными картами или другим удобным Вам способом.