12+

Андрей Ладыгин

ПРОМПТ-ИНЖИНИРИНГ

Как ставить задачи ИИ
и проверять результат.
Практикум для новичков

Андрей Ладыгин

Промпт-инжиниринг. Как ставить задачи ИИ и проверять результат. Практикум для новичков

«Издательские решения»

Ладыгин А.

Промпт-инжиниринг. Как ставить задачи ИИ и проверять результат. Практикум для новичков / А. Ладыгин — «Издательские решения»,

Промпт-инжиниринг — это не «магические фразы», а умение превращать задачу в точное ТЗ для ИИ и проверять результат. Практическое руководство для новичков без технического бэкграунда: 14 глав, 30 готовых промптов, 15 чек-листов, 6-недельный мини-курс, три сквозных кейса и финальный проект — собственный ИИ-ассистент под вашу задачу. С упражнениями и разбором типичных ошибок.

© Ладыгин А.
© Издательские решения

Содержание

ПРЕДИСЛОВИЕ 6
ВВЕДЕНИЕ 7
РАЗДЕЛ I. ФУНДАМЕНТ 12
ГЛАВА 1. КАК РАБОТАЕТ LLM БЕЗ ФОРМУЛ 13
ГЛАВА 2. АНАТОМИЯ ПРОМПТА: TASK, CONTEXT, FORMAT, CONSTRAINTS 20
ГЛАВА 3. ZERO-SHOT, FEW-SHOT И РОЛЬ ПРИМЕРОВ 27
РАЗДЕЛ II. ИНСТРУМЕНТЫ 34
ГЛАВА 4. ФРЕЙМВОРК CO-STAR И ДРУГИЕ СИСТЕМЫ 35
Конец ознакомительного фрагмента. 36

Промпт-инжиниринг
Как ставить задачи ИИ и проверять результат. Практикум для новичков

Андрей Ладыгин

Иллюстрации ИИ GPT Image 2
Обложка ИИ GPT Image 2

© Андрей Ладыгин, 2026

ISBN 978-5-0071-7967-6
Создано в интеллектуальной издательской системе Ridero

ПРЕДИСЛОВИЕ

Эта книга выросла из одного наблюдения. Люди делают вывод о возможностях ИИ по первому неудачному запросу. Написали «напиши пост» — получили «средний текст по рынку» — закрыли вкладку с мыслью «это не для моей работы». Проблема почти никогда не в модели. Она в том, что запрос не содержит задачи.

Книга учит одному: превращать «хотелку» в техническое задание, а результат — в проверяемый артефакт. Не «заклинать нейросеть», а ставить задачу так, как вы поставили бы её сотруднику: что сделать, для кого, в каком виде, чего не делать.

Три вещи, которые стоит знать до начала.

Первое. Это практикум, а не сборник шаблонов. В каждой главе есть практикум с критерием успеха и чек-лист самопроверки. Если вы читаете без практики, книга останется теорией. Если выполняете — к концу у вас будет личная библиотека промптов и собственный ИИ-ассистент. Поэтому я прошу: не пропускайте практикумы, даже если глава кажется простой.

Второе. Книга упрощает — сознательно. Объяснения того, как работает языковая модель, пошаговое рассуждение и визуальные модели даны на уровне, достаточном для практики, а не для исследования. Специалист заметит упрощения: механизм вероятностного предсказания описан без математики, Chain-of-Thought — без обзора исследований, визуальные модели — без архитектурных деталей. Это сделано намеренно: цель — рабочий навык, а не учебник по машинному обучению. Полные объяснения — в специализированной литературе, и приложение 5 подскажет, где искать.

Третье. Эта книга — стартовая точка, а не финиш. Инструменты меняются быстрее, чем выходят книги. Поэтому книга учит принципам — структуре задачи, роли, проверке результата, — а не кнопкам. Принципы переживут интерфейсы. Глава 14 объясняет, как оставаться актуальным, не заучивая шаблоны.

Книга создана с участием инструментов ИИ: они использовались для черновиков, структурирования и редакционной проверки. Финальный текст прошёл человеческую редактуру, фактчекинг и юридическую проверку. Все примеры вымышлены или обезличены. Все правовые утверждения даны как «общая практика» и требуют проверки специалистом. Подробнее — в выходных данных.

Если вы готовы не просто прочитать, а сделать — начните с диагностики во введении. Она займёт пять минут и покажет, с какой главы вам лучше стартовать.

[Имя автора]

[Год]

ВВЕДЕНИЕ

Зачем вам промпт-инжиниринг и с чего начать

Цель введения

Настроить читателя на практику, развеять пять мифов о работе с ИИ, дать диагностику уровня и объяснить, как устроена книга и что вы будете уметь к концу.

Ключевые тезисы

1. Промпт-инжиниринг — это практика формулирования задач для генеративных моделей так, чтобы результат был предсказуемым и проверяемым. Это навык ставить задачу, а не коллекция «волшебных фраз».

2. Качество ответа ИИ ограничено качеством вашего ТЗ, а не «силой модели».

3. Новичку не нужен код, чтобы получить рабочий результат — нужна дисциплина формулировки.

4. Шаблоны устаревают, методология — нет.

5. К концу книги у вас будет личная библиотека промптов, собственный ИИ-ассистент и не менее 30 готовых промптов в приложении 2 и 15 — в главах.

Проблема, с которой вы, скорее всего, пришли

Вы открыли ChatGPT, написали «напиши пост про наш продукт» — и получили текст, который стыдно показать коллегам. Второй раз попробовали «сделай лучше» — стало ещё хуже. Третий раз закрыли вкладку с мыслью: «это не для моей работы».

Знакомо? В моей практике обучения это самая частая точка входа. Люди делают вывод о возможностях ИИ по первому неудачному запросу. При этом проблема почти никогда не в модели. Она в том, что запрос не содержит задачи.

Сравните: «напиши пост» — это не задача, это направление. Задача звучит так: «Напиши пост о запуске курса по промпт-инжинирингу. Аудитория — руководители отделов обучения в компаниях от 500 человек. Цель — собрать заявки на пилот. Формат — 900 знаков, без эмодзи, первый абзац с конкретной болью, последний — с призывом записаться на вебинар 12 марта. Ограничения: не использовать слова „инновационный“, „экосистема“, „трансформация“; не обещать трудоустройство».

Разница между этими двумя запросами — и есть содержание этой книги.

«Напиши пост» против задачи

Плохой промпт

«Напиши пост про наш продукт»

↓

«В современном мире каждый хочет… уникальная возможность… не упустите шанс…»

× Усреднённый текст «по рынку». Перепишешь три раза — будет то же самое.

Хороший промпт

Задача: пост о запуске курса

Аудитория: руководители отделов обучения

Цель: заявки на вебинар 12 марта

Формат: 900 знаков, 4 абзаца

Запрещено: «уникальный», «инновационный», эмодзи

↓

Текст, который можно отправить: конкретная аудитория, конкретная цель, конкретный формат.

✓ Результат, пригодный без перделки

Разница — не в «волшебных словах», а в задаче: что сделать, для кого, в каком виде, чего нельзя.

Пять мифов, которые стоит закрыть сразу

Миф 1. «Нужно уметь программировать». Нет. Промпт-инжиниринг на уровне уверенного пользователя — это работа с текстом, логикой и структурой. Программирование понадобится, если вы пойдёте в разработку ИИ-приложений. Это другая профессия и другая книга.

Миф 2. «Есть секретные слова, которые всё меняют». Есть работающие приёмы: структура, роли, примеры, пошаговое рассуждение, ограничения. Но каждый из них — не заклинание, а инструмент постановки задачи. Приём без понимания задачи даёт убедительный на вид, но бесполезный результат.

Миф 3. «Модель сама поймёт, что я имею в виду». Не поймёт. Модель предсказывает вероятное продолжение текста, а не читает ваши мысли. Чем меньше контекста вы дали, тем больше модель достроит за вас — и тем дальше результат будет от вашего замысла.

Миф 4. «Один раз настроил промпт — и он работает всегда». Работает до первого изменения задачи или модели. Поэтому книга учит не шаблонам, а способу их собирать и отлаживать. Шаблон вы получите в приложении — но пользоваться им без метода не сможете.

Миф 5. «ИИ заменит специалиста». Заменяет не ИИ, а специалист, который умеет ставить ему задачи. Разница между двумя сотрудниками с одинаковым доступом к ChatGPT — это разница в качестве ТЗ и в умении проверять результат.

Как устроена книга

Четыре раздела идут по нарастающей.

Раздел I. Фундамент (главы 1—3) — как работает модель, из чего состоит промпт, зачем нужны примеры. Здесь вы перестанете угадывать.

Раздел II. Инструменты (главы 4—7) — фреймворки, роли, пошаговое рассуждение, ограничения. Здесь вы соберёте арсенал.

Раздел III. Практика (главы 8—11) — текст, данные, изображения, автоматизация. Здесь вы перенесёте навык на свою работу.

Раздел IV. Ответственность (главы 12—14) — отладка, этика, безопасность, горизонт. Здесь вы научитесь не наступать на грабли.

Каждая глава построена одинаково: цель → ключевые тезисы → объяснение с примерами «плохо → почему → хорошо» → врезка «Что пошло не так» → практикум с критерием успеха → чек-лист самопроверки → промпт «ИИ-критик», которым вы проверяете свой результат. В конце — связка со следующей главой.

Карта книги: 4 раздела — 14 глав

Раздел I. Фундамент (главы 1—3)

1. Как работает LLM
2. Анатомия промпта (TCFC)
3. Zero-shot, Few-shot, примеры

Раздел II. Инструменты (главы 4—7)

4. CO-STAR и фреймворки
5. Роли и персоналии
6. Chain-of-Thought
7. Negative Prompting

Раздел III. Практика (главы 8—11)

8. Текст и контент
9. Анализ данных

10. Генерация изображений
11. Автоматизация

Раздел IV. Ответственность (главы 12—14)

12. Отладка промптов
13. Этика и безопасность
14. Будущее поля

Сквозной путь: введение → 14 глав → заключение. К концу — личная библиотека промптов и собственный ИИ-ассистент.

Как читать

Три режима на выбор.

Режим «за неделю». Читаете подряд, выполняете только практикумы в главах 2, 5, 8 и итоговый проект. Подходит, если нужно быстро закрыть базовый уровень.

Режим «по методу». Пять глав в неделю по 45—60 минут — шесть недель с учётом практикумов и итогового проекта. Если темп не ваш — две-три главы в неделю, десять недель.

Режим «справочник». Пропускаете теорию, идёте в приложение 2 за библиотекой промптов, возвращаетесь в нужную главу, когда промпт не сработал. Рабочий вариант для занятых.

Что бы вы ни выбрали, заведите дневник промптов — таблицу с колонками: задача → промпт → результат → что улучшить. Без дневника вы будете повторять одни и те же ошибки, не замечая их.

Диагностика: на каком вы уровне

Ответьте «да» или «нет». Считайте количество ответов «да».

1. Я формулирую задачу до того, как открыть чат с ИИ.
2. Я указываю в промпте, для кого и зачем нужен результат.
3. Я задаю формат ответа (список, таблица, JSON, длина).
4. Я использую ограничения — что модель делать не должна.
5. Я даю примеры, когда нужен конкретный стиль.
6. Я прошу модель рассуждать пошагово для сложных задач.
7. Я проверяю факты из ответа ИИ.
8. Я храню удачные промпты и переиспользую их.
9. Я умею найти причину, почему промпт не сработал.
10. Я могу объяснить коллеге, как строится промпт.

0—2 «да». Вы в начале. Книгу читайте подряд, начиная с главы 1, и не пропускайте практикумы.

3—5 «да». У вас есть интуиция, но нет системы. Обратите особое внимание на главы 2, 4 и 12.

6—8 «да». Вы уверенный пользователь. Начните с главы 7, затем 9—13; главы 1—3 просмотрите по диагонали.

9—10 «да». Вам нужны главы 12—14 и приложение 2. Основную часть используйте как справочник.

Практикум введения

Задача. Сделать первый замер и первый артефакт.

Шаги.

1. Пройдите диагностику и запишите результат.

2. Возьмите одну реальную рабочую задачу, которую вы решаете регулярно (письмо клиенту, сводка встречи, план занятия).

3. Напишите к ней промпт так, как написали бы сейчас, не читая книгу.

4. Сохраните его в дневник промптов — Notion, Google Sheets или таблица в любом редакторе — под номером 1.

5. В конце каждой главы возвращайтесь к этому промпту и переписывайте его с учётом нового навыка. К главе 14 у вас будет наглядная история роста.

Пример заполнения дневника:

Задача | Промпт | Результат | Что улучшить

Сводка встречи | «Суммируй текст…» | 8 пунктов, вода | Задать формат: 3 пункта, до 10 слов

Письмо клиенту | «Напиши письмо о задержке» | Общие извинения | Добавить контекст и реальные сроки

План занятия | «Составь план на 90 минут» | 5 блоков теории | Задать баланс теории и практики

Критерий успеха. У вас есть файл с диагностикой и первый промпт, сохранённый в неизменном виде.

Чек-лист самопроверки

☐ Я могу назвать пять мифов и объяснить, почему они неверны.
☐ Я понимаю разницу между «напиши пост» и полноценным ТЗ.
☐ Я выбрал режим чтения.
☐ Я создал дневник промптов.
☐ Я прошёл диагностику и записал результат.
☐ Я сохранил первый промпт без правок.
☐ Я знаю, что итоговый проект — ИИ-ассистент под мою задачу.

Разбор типичных ошибок

Ошибка 1. «Прочитаю теорию, практику сделаю потом». Практика потом не наступает. Каждая глава даёт один артефакт; без него следующая глава ложится на пустое место.

Ошибка 2. «Сразу перейду к продвинутым техникам». Chain-of-Thought не спасает промпт без задачи и контекста. Порядок глав неслучаен.

Ошибка 3. «Буду хранить промпты в голове». Через две недели вы не вспомните, какой из трёх вариантов дал лучший результат. Дневник — не бюрократия, а инструмент обучения.

Ошибка 4. «Оценю книгу по первому результату». Первый промпт почти всегда слабый. Сравнивайте не с идеалом, а с собственной версией недельной давности.

Ошибка 5. «Пропущу главу про этику и безопасность». Именно там — ответы на вопросы, которые возникают в реальной работе: можно ли отправлять в чат данные клиента, нужно ли раскрывать, что текст сгенерирован, что делать со стереотипами в ответах.

ИИ-критик

Вставьте в чат:

Ты — критик-редактор. Оцени мой первый промпт по критериям: есть ли задача, контекст, формат, ограничения; понятно ли, для кого результат; можно ли по этому промпту получить предсказуемый ответ. Не хвали ради вежливости. Дай оценку по 10-балльной шкале, три слабых места и одну правку «было → стало». В конце задай мне один вопрос, который поможет улучшить промпт.

Мой промпт: [вставить]

Связка со следующей главой

Прежде чем учиться формулировать, разберёмся, с чем именно мы разговариваем. Глава 1 объясняет без формул, как модель выбирает следующее слово, почему она уверенно ошибается и как это знание меняет ваши промпты.

РАЗДЕЛ I. ФУНДАМЕНТ

Цель раздела: снять магию вокруг нейросетей и дать базу для осознанного промптинга. Три главы — как работает модель, из чего состоит промпт, зачем нужны примеры.

ГЛАВА 1. КАК РАБОТАЕТ LLM БЕЗ ФОРМУЛ

Цель главы

Понять, что происходит между вашим сообщением и ответом модели: как она выбирает слова, почему уверенно ошибается и как это знание меняет формулировку промпта. К концу главы вы сможете объяснить своими словами, откуда берутся галлюцинации, и предсказать, где модель ошибётся.

Ключевые тезисы

1. Модель не ищет ответ — она продолжает текст, выбирая вероятное следующее слово.
2. Она видит не слова и не буквы, а токены — куски текста.
3. У модели есть контекстное окно — конечная «оперативная память», и она заканчивается.
4. Поиск, генерация и рассуждение — три разных режима, и их путают чаще всего.
5. Галлюцинация — не сбой, а побочный эффект устройства: модель обязана продолжить текст, даже когда не знает.

1.1. Что происходит, когда вы нажимаете «Отправить»

Представьте, что вы просите коллегу дописать фразу, которую он видит впервые в жизни, но должен сделать это уверенно и без пауз. Ваш коллега не идёт в библиотеку и не звонит эксперту. Он опирается на то, что видел раньше в похожих текстах, и подбирает продолжение, которое чаще всего встречалось в подобных ситуациях.

Так работает большая языковая модель — LLM (large language model). Ваш промпт попадает в модель, она превращает его в последовательность чисел и на каждом шаге решает одну задачу: какое следующее слово (точнее, токен) вероятнее всего идёт дальше. Выбрала — добавляет к тексту — повторяет. Слово за словом, токен за токеном, пока не решит остановиться.

Из этого устройства следуют почти все практические свойства, с которыми вы будете работать: скорость, отсутствие настоящей проверки фактов, чувствительность к формулировке, «усталость» на длинных текстах. Не нужно знать математику, чтобы этим пользоваться. Нужно держать в голове три вещи: токены, вероятности, контекст.

Модель выбирает следующее слово — по вероятности

Вы: «На встрече мы…» → На | встрече | мы |?

Модель не видит слова — она видит токены и выбирает следующий.

Кандидаты на следующий токен:

обсудили — 42% (выбирает модель)
провели — 18%
свернули — 9%
запланировали — 7%
прочие — 24%

Выбрала → добавила к тексту → повторила. Так, пока модель не решит остановиться.

1.2. Токены: модель не видит слов

Модель не работает с буквами и не работает со словами целиком. Она работает с токенами — кусками текста, которые могут быть словом, частью слова, знаком препинания или пробелом с буквой.

Для нас важно одно следствие: модель «считает» не так, как вы. Короткое слово может быть одним токеном, длинное — тремя. По распространённой оценке, русский язык «дороже» английского: на ту же мысль уходит больше токенов. Отсюда два практических вывода.

Вывод первый: длина — это ресурс. Когда вы пишете «сделай красиво и подробно», вы расходуете токены на инструкции, которые ничего не уточняют. Когда вы пишете «три пункта, каждый до 12 слов, без вводных конструкций» — вы тратите те же токены, но на управление результатом.

Вывод второй: незнакомые конструкции модель «достраивает». Если в вашем запросе есть опечатка или необычное сокращение, модель не спросит, что вы имели в виду, — она подставит наиболее вероятное прочтение. Иногда правильное. Иногда нет, и вы этого не заметите.

Пример.

Плохо: «Сделай мне норм текст про наш вебинар».

Почему плохо: «норм» — оценочное слово без критерия; «текст» — не формат; «про вебинар» — не задача. Модель заполнит все пробелы своим наиболее вероятным вариантом: получится усреднённый рекламный абзац.

Хорошо: «Напиши приглашение на вебинар о промпт-инжиниринге для HR-специалистов. Формат: 3 абзаца по 2—3 предложения. Первый — боль: тратят часы на типовые тексты. Второй — что даст вебинар. Третий — дата, время, ссылка и одно действие: зарегистрироваться. Без эмодзи и без слов „уникальный“, „инновационный“».

1.3. Вероятности: почему это не поиск

Самое частое заблуждение новичка: «модель где-то нашла этот ответ». Нет. Модель не хранит документы и не выдаёт ссылки. Она хранит статистические закономерности того, как устроены тексты, и на их основе строит продолжение.

Разница принципиальная, и вот как она проявляется на практике.

Поиск (поисковая система) возвращает то, что существует в источниках, и может дать ссылку. Если источника нет — он честно скажет «не найдено».

Генерация (LLM в обычном режиме) возвращает то, что похоже на правильный ответ. Если правильного ответа в обучающих данных не было — модель всё равно его построит, потому что её задача — продолжить текст, а не подтвердить факт. Ссылку она тоже может «построить»: красивую, правдоподобную и несуществующую.

Рассуждение — это когда модель строит цепочку шагов. Она может прийти к верному выводу, но каждый шаг остаётся вероятностным. Цепочка может быть логичной и при этом опираться на выдуманную посылку.

Правило: генерация — это не проверка. Любой факт, цифра, дата, имя и ссылка из ответа модели требуют отдельного подтверждения. Это не недоверие к инструменту, это понимание его природы. Вы же не требуете от калькулятора, чтобы он объяснил смысл числа, — вы требуете от него правильного вычисления. От модели вы требуете хорошего текста, а факты проверяете сами.

Пример.

Плохо: «Дай мне статистику по рынку онлайн-образования в России за 2024 год».

Почему плохо: модель поставит в текст числа, которые выглядят как статистика. Часть может оказаться верной, часть — смешанной из разных лет, часть — выдуманной. Проверить это внутри чата невозможно.

Хорошо: «Мне нужна структура обзора рынка онлайн-образования в России: какие 5 метрик обычно приводят в таких обзорах и какие источники их публикуют. Числа не приводи — я подставлю их сам из первоисточников». Модель даёт вам каркас, вы — данные.

1.4. Контекстное окно: память, которая заканчивается

У модели есть контекстное окно — объём текста, который она «держит в голове» в рамках одного диалога. В окно входит всё: ваши сообщения, её ответы, вставленные документы, системная инструкция.

Для работы с длинными текстами важны три следствия.

1) Окно конечно. Когда оно заполняется, ранние части разговора «выпадают» или сжимаются. Модель может перестать учитывать инструкцию, которую вы дали в самом начале. Это не каприз, это механика.

2) Чем длиннее контекст, тем ниже точность. Как правило, даже в пределах окна внимание модели распределяется неравномерно. Ключевая инструкция, спрятанная в середине большого документа, может «потеряться». Практический приём: важные требования ставьте в начало и повторяйте в конце.

3) Контекст — это не память между чатами. Новый чат начинается с чистого окна. Всё, что вы обсуждали вчера, модель не помнит, если вы не перенесли это в новый запрос.

Пример.

Плохо: в одном чате на 40 сообщений сначала попросить «пиши в официально-деловом стиле», а через двадцать сообщений удивляться, что стиль съехал.

Почему плохо: инструкция вытеснена или ослаблена объёмом последующего текста.

Хорошо: перед новой задачей в том же чате повторить рамку: «Возвращаемся к правилам: официально-деловой стиль, без эмодзи, до 150 слов. Задача:…». Ещё лучше — для каждой крупной задачи открывать новый чат и переносить туда только нужный контекст.

1.5. Три режима, которые путают

Большинство неудачных промптов — это не плохие формулировки, а неправильно выбранный режим. Вы просите модель рассуждать там, где нужен поиск. Или просите найти там, где нужна генерация.

Что вам нужно | Режим | Как формулировать | Где ловушка

Узнать существующий факт | Поиск (модель с доступом к источникам) или внешний поиск | «Найди и дай ссылку на первоисточник» | Модель без доступа к сети отвечает по памяти и ошибается

Получить текст, структуру, идеи | Генерация | «Напиши… Формат… Ограничения…» | Принимаете правдоподобное за верное

Решить задачу с шагами | Рассуждение | «Рассуждай пошагово: шаг 1… шаг 2…» | Каждый шаг вероятностный; ошибка в начале рушит вывод

Практический вывод: сначала решите, какой режим вам нужен, и только потом пишите промпт. Если нужен факт — просите источник или проверяйте сами. Если нужен текст — задавайте формат и ограничения. Если нужна логика — просите пошаговое рассуждение и проверяйте посылки.

Кейс 1. Маркетинг. Неправильный режим — самая частая причина провала.

Маркетолог просит: «Найди, какие посты про промпт-инжиниринг сейчас заходят лучше всего». Формально это запрос на поиск. Но модель без доступа к сети не ищет — она генерирует правдоподобное описание «типичных заходящих постов». Результат выглядит как аналитика, но это не аналитика.

Правильная декомпозиция — два шага в разных режимах. Шаг первый, генерация: «Опиши 7 форматов постов, которые обычно дают высокую вовлечённость в профессиональных блогах про ИИ, и для каждого укажи, какую боль аудитории он закрывает». Шаг второй, поиск во внешнем инструменте: маркетолог сам смотрит статистику в своих источниках и проверяет, какие из форматов работают у его аудитории.

Модель дала каркас гипотез. Проверку сделал человек. Именно так выглядит рабочее разделение труда.

Кейс 2. Обучение. Рассуждение там, где нужен формат.

Методист просит: «Подумай, как лучше структурировать курс по промпт-инжинирингу для новичков». Модель начнёт рассуждать — и выдаст общие, гладкие, но бесполезные принципы: «начните с основ, добавьте практику, используйте примеры».

Проблема в режиме. Методисту нужен не ход рассуждений, а структура под конкретные ограничения. Правильная формулировка: «Структурируй курс на 6 недель по 4 часа. Аудитория — маркетологи без техбэкграунда. Выходной результат — каждый участник собирает библиотеку из 10 промптов под свои задачи. Формат: таблица, колонки — неделя, тема, практикум, критерий успеха». Модель выдаст таблицу, которую можно править, а не рассуждение, с которым можно только согласиться или нет.

1.6. Галлюцинация: уверенность без знания

Галлюцинация — это когда модель выдаёт неверную информацию уверенно, без оговорок и в том же тоне, что и верную. Это опасное свойство LLM в рабочем использовании и частое основание для разочарования.

Почему это происходит, если понимать устройство: модель обязана продолжить текст. У неё нет встроенного «стоп, я не знаю». Современные модели обучены отказываться, но отказ тоже вероятностен — он появляется, только если в похожих ситуациях такая фраза встречалась достаточно часто.

Что пошло не так.

Пример, иллюстрирующий механизм. Менеджер попросил: «Дай три исследования о влиянии микробучения на удержание знаний, с авторами и годами». Модель выдала три аккуратных пункта с фамилиями, годами и журналами. Два из трёх оказались несуществующими — фамилии правдоподобные, журналы реальные, статьи нет. Ошибка была не в модели, а в задаче: менеджер попросил готовые ссылки, а не способ их найти. Правильная формулировка: «Назови 3 направления исследований по микробучению и удержанию знаний и укажи, по каким ключевым словам и в каких базах искать первоисточники». Модель дала каркас поиска, менеджер нашёл реальные статьи сам.

Как снижать риск галлюцинаций в промпте:

— Просите источник, а не утверждение. «Дай ссылку» вместо «Дай факт».

— Вводите маркировку уверенности. «Для каждого утверждения помечай: подтверждено источником / общая практика / предположение».

— Запрещайте достраивать. «Если данных нет — напиши „данных нет“, не подставляй похожие».

— Разделяйте факт и интерпретацию. «Сначала факты с источниками, потом твоя интерпретация, отдельным блоком».

Ни один из этих приёмов не даёт гарантии. Гарантию даёт только внешняя проверка. Но вместе они резко снижают долю выдумок.

Как это выглядит в чувствительных темах.

В медицине и праве галлюцинация — не «неудобный текст», а риск для человека. В медицинской теме модель уверенно назовёт дозировку, противопоказание или «исследование», которого нет. Она делает это не из злого умысла: в обучающих данных множество текстов, где дозировки и противопоказания обсуждаются в утвердительном тоне. Модель воспроизводит тон, а не проверяет факт.

В юридической теме тот же механизм даёт «номера статей» и «судебную практику», которые выглядят корректно и не существуют. Формулировка «согласно статье…» звучит убедительнее, чем «я не уверен», поэтому модель выбирает первую.

Дисклеймер. Материалы этой книги не являются медицинской или юридической консультацией. Для тем, где цена ошибки высока, ИИ допустим только как инструмент черновика и структурирования, а решение принимает профильный специалист. В промпте это фиксируется прямо: «Не приводи дозировки и номера статей. Дай список вопросов, которые нужно проверить у специалиста, и ключевые слова для поиска первоисточников».

Это не отказ от инструмента. Это точное понимание, где он полезен: в подготовке вопросов, структуры, черновика и списка того, что нужно проверить. Не в финальном решении.

Уверенный, но неверный ответ

Вы: «Дай три исследования о влиянии микробучения, с авторами и годами»

Модель:

1. Смирнова А. (2021) — «Микробучение и удержание», Journal of Learning
2. Петров И. (2019) — «Короткие сессии и память», Applied Cognition
3. Соколова Е. (2022) — «Интервальное повторение в корпоративе», HBR

Проверка в первоисточниках:

✓ 1 — найдена, данные совпадают

× 2 — статья не существует

× 3 — фамилия и журнал — реальные, статьи нет

Галлюцинация = уверенность без знания

Модель обязана продолжить текст. У неё нет кнопки «я не знаю», если так не учили.

Как снизить риск: просить способ найти первоисточник, а не готовые «факты». Проверка — всегда снаружи.

1.7. Что это меняет в ваших промптах

Пять правил, которые следуют из устройства модели.

1. **Задача важнее вежливости.** «Пожалуйста, напиши красиво» не добавляет информации. «Три пункта до 12 слов» добавляет.

2. **Формат — это управление.** Указав формат, вы сужаете пространство вероятных ответов и получаете предсказуемость.

3. **Контекст нужно давать, а не подразумевать.** Модель не знает вашу компанию, продукт, аудиторию и историю.

4. **Ограничения работают лучше просьб.** «Не используй слово X» надёжнее, чем «пиши профессионально».

5. **Факты проверяйте отдельно.** Всегда. Это часть профессии, а не признак недоверия.

Пять правил — пять промптов:

Правило | Промпт-пример

Задача важнее вежливости | «Сократи текст ниже до 3 пунктов, каждый до 12 слов, без вводных конструкций»

Формат — это управление | «Верни результат таблицей: колонки „Проблема“, „Частота“, „Рекомендация“»

Контекст давайте, а не подразумевайте | «Аудитория — руководители отдела продаж, читают с телефона, не любят термины»

Ограничения работают лучше просьб | «Не используй слова „инновационный“, „экосистема“; не пиши больше 150 слов»

Факты проверяйте отдельно | «Для каждого утверждения укажи: подтверждено источником / общая практика / предположение»

Эти пять строк — не шаблон, а проверочный список. Если в вашем промпте нет хотя бы трёх из пяти элементов, результат будет случайным.

Практикум. «Угадай следующий токен»

Задача: на собственном опыте увидеть, что ответ модели — это выбор вероятного продолжения, а не поиск истины.

Шаги.

1. Откройте чат с любой текстовой моделью.

2. Отправьте: «Продолжи фразу одним словом, без пояснений: «На встрече мы обсудили…«». Повторите 5 раз в новых чатах.

3. Запишите 5 вариантов. Посмотрите, сколько раз ответ совпал и какие слова повторяются.

4. Теперь отправьте: «Продолжи фразу одним словом, без пояснений, но так, чтобы слово было неожиданным и не встречалось в типичных бизнес-текстах: «На встрече мы обсудили…«». Повторите 5 раз.

5. Сравните два набора вариантов.

6. Затем отправьте: «Назови три факта о компании [вымышленное название] с источниками». Посмотрите, что произойдёт, когда фактов не существует.

Критерий успеха. Вы можете сформулировать своими словами: (а) почему ответы в шаге 3 однотипны; (б) почему ограничение в шаге 4 меняет распределение; (в) как модель ведёт себя в шаге 6, когда знания нет. Если на шаг 6 модель честно ответила «данных нет» — это хорошая модель; если выдала «факты» — вы увидели галлюцинацию в чистом виде.

Дополнительно. Занесите в дневник промптов один пример из шага 6 с пометкой «галлюцинация: что именно выдумано».

Чек-лист самопроверки

☐ Я могу объяснить, что такое токен, не используя слово «токен».

☐ Я понимаю, почему модель не «ищет ответ», а «продолжает текст».

☐ Я знаю, что контекстное окно конечно и инструкция в начале длинного чата может ослабнуть.

☐ Я различаю три режима: поиск, генерация, рассуждение.

☐ Я могу назвать три причины галлюцинаций.

☐ Я знаю минимум три приёма снижения галлюцинаций в промпте.

☐ Я выполнил практикум и записал результат в дневник.

Разбор типичных ошибок

Ошибка 1. «Модель знает, что я имею в виду». В чём ошибка: ожидание, что контекст подразумевается. Модель достроит пробелы наиболее вероятным способом — и это не ваш способ. Как исправить: перед отправкой перечитайте промпт и найдите все места, где вы «держите в голове» важную деталь. Перенесите её в текст.

Ошибка 2. «Попрошу проверить факты у самой модели». В чём ошибка: просьба «проверь себя» не создаёт доступа к источникам. Модель подтвердит то, что уже сказала, или

сгенерирует новую версию. Как исправить: проверяйте во внешних источниках. В промпте просите ключевые слова и базы для поиска, а не подтверждение.

Ошибка 3. «Один длинный чат на всю рабочую неделю». В чём ошибка: инструкции вытесняются, ранние договорённости теряются, стиль съезжает. Как исправить: новая крупная задача — новый чат. В начало — рамка: роль, формат, ограничения, аудитория. Ключевое — повторить в конце.

Ошибка 4. «Если ответ плохой, значит модель плохая». В чём ошибка: отсутствие диагностики. Чаще всего проблема в отсутствии задачи, контекста или формата, а не в модели. Как исправить: задайте себе три вопроса. Есть ли в промпте задача? Есть ли контекст? Задан ли формат? Если хотя бы один ответ «нет» — правьте промпт, а не меняйте инструмент.

Ошибка 5. «Чем больше инструкций, тем лучше». В чём ошибка: противоречивые и избыточные инструкции снижают качество. Модель пытается выполнить всё сразу и усредняет. Как исправить: не больше 5—7 ключевых требований на промпт. Если требований больше — разбейте задачу на два шага.

ИИ-критик

Вставьте в чат:

Ты — критик промптов. Оцени мой промпт ниже по пяти критериям: (1) есть ли явная задача; (2) дан ли контекст — для кого и зачем; (3) задан ли формат результата; (4) есть ли ограничения; (5) не перегружен ли промпт противоречивыми требованиями. Для каждого критерия поставь 0/1 и объясни одной фразой. Затем дай общую оценку по 10-балльной шкале, укажи три слабых места и одну правку «было → стало». Не хвали ради вежливости. В конце задай мне один вопрос, ответ на который улучшит промпт.

Мой промпт: [вставить]

Связка со следующей главой

Мы разобрались, с чем разговариваем: с вероятностной моделью, у которой есть конечная память и склонность достраивать недостающее. Теперь перейдём к тому, что зависит от вас полностью, — к структуре промпта. Глава 2 даёт четыре элемента (TCFC), из которых собирается любое техническое задание для ИИ, и учит переписывать «плохие» запросы за две минуты.

ГЛАВА 2. АНАТОМИЯ ПРОМПТА: TASK, CONTEXT, FORMAT, CONSTRAINTS

Цель главы

Научиться собирать любой промпт из четырёх элементов и за две минуты находить, чего не хватает в запросе, который даёт плохой результат. К концу главы вы переписываете пять «плохих» промптов по схеме TCFC и понимаете, когда эта схема избыточна.

Ключевые тезисы

1. Плохой промпт — это не «неправильные слова», а отсутствие одного или нескольких элементов ТЗ.
2. **Task** — что именно сделать, с каким результатом на выходе.
3. **Context** — для кого, зачем и на основе каких данных.
4. **Format** — в каком виде вернуть результат: структура, длина, носитель.
5. **Constraints** — что делать нельзя: слова, объём, тон, факты, которые нельзя трогать.
6. TCFC — не догма. Для простых задач хватает двух элементов, для сложных — четыре обязательны.

2.1. Почему «напиши пост» не работает

Возьмите фразу «напиши пост про наш продукт». В ней есть глагол «напиши» и объект «пост». Формально это указание. Содержательно — пустая форма, которую модель заполнит наиболее вероятным способом.

Что модель может решить за вас: для кого пост (для коллег, для клиентов, для инвесторов?), какой длины (200 знаков или 2000?), в каком тоне (деловом, дружеском, ироничном?), с какой целью (продать, объяснить, пригласить?), какие слова использовать нельзя (у вас же есть запрещённые — «инновационный», «уникальный»), где факты (название продукта, цена, дата).

Каждое из этих решений — развилка. Модель проходит все развилки, выбирая самый частый вариант в обучающих данных. Это значит, что вы получаете средний пост по рынку, а не пост для вашей задачи.

Первое рабочее правило: если вы не приняли решение за модель, она примет его за вас. Промпт-инжиниринг — это в первую очередь дисциплина принятия решений до отправки запроса.

2.2. Task: что именно сделать

Task — это глагол действия плюс результат, который вы хотите получить. Не «про пост», а «напиши пост». Не «поработай с текстом», а «сократи текст до 300 слов, сохранив три ключевых аргумента».

Три требования к хорошему Task.

Глагол должен быть операциональным. «Напиши», «сократи», «переформулируй», «сравни», «извлеки», «составь список», «проверь на противоречия». Плохие глаголы: «проработай», «оптимизируй», «улучши», «сделай красиво». Они не описывают операцию — они описывают ваше ощущение от результата.

Результат должен быть назван. Не «напиши письмо», а «напиши письмо клиенту с предложением перенести встречу на следующую неделю, с новым вариантом даты и просьбой подтвердить». Модель не должна догадываться, что именно должно получиться.

Один промпт — одна задача. Если вам нужно и письмо, и список аргументов для звонка, и три варианта темы письма — это три промпта или один с явной структурой из трёх частей. Смешанная задача размывает результат.

Пример.

Плохо: «Оптимизируй мой текст».

Почему плохо: «оптимизируй» — не операция. Непонятно, что оптимизировать: длину, читаемость, тон, структуру? Модель выберет что-то одно, и это будет не то, что вам нужно.

Хорошо: «Сократи текст ниже с 850 до 350 слов. Сохрани: три аргумента, одну цитату клиента, финальный призыв. Убери: вводные конструкции, повторы, обращения к читателю».

2.3. Context: для кого, зачем и на основе чего

Context — самый недооценённый элемент. Context новички почти всегда пропускают, а потом удивляются, что «модель не поняла». Модель не может «не понять» то, чего вы не написали.

Context отвечает на три вопроса.

Для кого. Аудитория: возраст, роль, уровень знаний, отношение к теме. «Для HR-директоров крупных компаний» и «для студентов первого курса» — это два разных текста, даже если тема одна.

Зачем. Цель результата: что должно произойти после того, как читатель это прочитает. Записаться на вебинар? Согласовать бюджет? Просто узнать? Цель определяет призыв и структуру.

На основе чего. Источник: ваш текст, данные клиента, фрагмент документа, ваш прошлый пост. Если нужен ответ «по материалу» — материал нужно вложить в промпт. Если ответ «из общих знаний» — это стоит сказать явно, чтобы модель не смешивала.

Пример.

Плохо: «Напиши письмо про задержку проекта».

Почему плохо: нет ни аудитории, ни цели, ни фактов. Модель напишет вежливое письмо «в воздух».

Хорошо: «Напиши письмо заказчику — руководителю IT-отдела, который уже дважды переносил сроки приёмки. Цель — сообщить о задержке на 5 рабочих дней и предложить план компенсации: два дополнительных дня поддержки бесплатно. Тон — деловой, без извинений в каждом абзаце, без обещаний, которые мы не можем выполнить. Факты: новый срок сдачи — 24 марта; причина — задержка поставки оборудования у подрядчика; контакт для вопросов — [имя]».

2.4. Format: в каком виде вернуть результат

Format — это управление формой. Он не улучшает содержание, но повышает предсказуемость. Без формата модель, как правило, выберет формат по умолчанию — чаще всего связный текст с вводным абзацем, который вам не нужен.

Что стоит задавать в Format.

Структура. Список, таблица, нумерованные пункты, JSON, три абзаца, эссе. Если результат должен быть вставлен в другой документ — формат нужно задать точно: «колонки: проблема, частота, рекомендация».

Длина. В словах, знаках или пунктах. «До 150 слов» работает лучше, чем «коротко». «Ровно 5 пунктов» работает лучше, чем «несколько».

Носитель. Пост, письмо для email, тезисы для слайда, сообщение в мессенджер. Носитель диктует длину и тон.

Что не включать. «Без вводного абзаца», «без итогового резюме», «без пояснений после таблицы». Раньше модели часто добавляли «Надеюсь, это поможет»; сейчас реже, но проверять стоит — если этого не нужно, запрещайте.

Пример.

Плохо: «Дай мне список идей».

Почему плохо: непонятно, сколько идей, какой длины, в каком виде, с пояснениями или без.

Хорошо: «Дай 7 идей для постов о промпт-инжиниринге. Формат: нумерованный список, каждая идея — одно предложение до 15 слов, без пояснений и без вводного абзаца. В конце — одна строка „Выбрать можно любые три“».

2.5. Constraints: что делать нельзя

Constraints — ограничения. Работают надёжнее просьб, потому что сужают пространство вариантов, а не описывают желаемое. «Пиши профессионально» — просьба. «Не используй слова „инновационный“, „экосистема“, „трансформация“» — ограничение.

Что стоит ограничивать.

Запрещённые слова и клише. Ваш личный стоп-лист: «уникальный», «лидирующий», «в современном мире». Это быстрый способ убрать «ИИ-глянец».

Объём. Верхняя и нижняя границы. «Не меньше 3 и не больше 5 пунктов».

Факты, которые нельзя менять. Цитаты, цифры, имена, даты из исходника. «Не меняй цитату клиента, даже если она стилистически неудобна».

Факты, которых нельзя добавлять. «Не добавляй статистику, которой нет в исходном тексте. Если для аргумента нужна цифра — оставь место [ЦИФРА]».

Тон, который запрещён. «Без сарказма», «без панибратства», «без мотивационного тона».

Форматы, которые запрещены. «Без эмодзи», «без маркированных списков», «без заголовков».

Пример.

Плохо: «Напиши серьёзно и по делу».

Почему плохо: «серьёзно» и «по делу» — оценочные слова. У модели своя шкала серьёзности.

Хорошо: «Тон — деловой. Запрещено: эмодзи, восклицательные знаки, обращения „друзья“, „коллеги“, слова „крутой“, „мощный“, „прорывной“. Максимум 200 слов».

2.6. TCFC в сборе: пять разборов

Ниже — пять промптов из разных рабочих ситуаций. Для каждого: что было, почему не сработало, как выглядит версия по TCFC.

Разбор 1. Маркетинг. Пост для соцсетей.

Плохо: «Напиши пост про наш новый курс».

Почему плохо: нет аудитории, цели, формата и ограничений. Модель выдаст усреднённый рекламный пост.

По TCFC:

— **Task:** напиши пост о запуске курса «Промпт-инжиниринг для менеджеров».

— **Context:** аудитория — руководители отделов обучения в компаниях от 500 человек. Цель — собрать регистрации на бесплатный вебинар 12 марта. Боль аудитории: сотрудники тратят часы на типовые тексты и не умеют ставить задачи ИИ.

— **Format:** 900—1100 знаков, 4 абзаца по 2—3 предложения. Первый — боль. Второй — что изменится после курса. Третий — кому подойдёт. Четвёртый — дата, время, ссылка и одно действие: зарегистрироваться.

— **Constraints:** без эмодзи; без слов «уникальный», «инновационный», «трансформация»; не обещать трудоустройство; не упоминать конкурентов.

Разбор 2. Менеджмент. Письмо клиенту.

Плохо: «Напиши письмо клиенту, что мы задерживаем проект».

Почему плохо: непонятно, какой клиент, какая история отношений, что можно обещать в компенсацию.

По TCFC:

— **Task:** напиши письмо клиенту о задержке проекта на 5 рабочих дней.

— **Context:** клиент — руководитель IT-отдела, уже дважды переносил приёмку. Отношения рабочие, но напряжённые. Причина задержки — поставка оборудования у подрядчика. Мы готовы дать 2 дополнительных дня поддержки бесплатно.

— **Format:** email, 180—220 слов. Тема письма отдельной строкой. Структура: факт задержки и новый срок → причина без перекладывания вины → компенсация → контакт для вопросов.

— **Constraints:** без извинений в каждом абзаце (одно — в начале); без обещаний, которых нет в контексте; тон деловой, без канцелярита.

Разбор 3. Обучение. План занятия.

Плохо: «Составь план занятия по промпт-инжинирингу».

Почему плохо: неизвестна длительность, аудитория, уровень, оборудование, цель.

По TCFC:

— **Task:** составь план занятия на 90 минут.

— **Context:** аудитория — маркетологи без техбэкграунда, 12 человек, уровень «пробовали ChatGPT пару раз». Цель занятия — к концу каждый участник умеет переписать свой промпт по TCFC.

— **Format:** таблица с колонками: время, блок, что делает преподаватель, что делают участники, результат блока. Плюс список материалов.

— **Constraints:** не больше 20 минут теории подряд; минимум 40 минут практики; никаких слайдов с кодом.

Разбор 4. Аналитика. Сводка отзывов.

Плохо: «Проанализируй отзывы клиентов и скажи, что не так».

Почему плохо: не указан источник, объём, формат вывода, критерии «что не так».

По TCFC:

— **Task:** проанализируй 20 отзывов ниже и выдели главные проблемы.

— **Context:** отзывы с маркетплейса за последний месяц по категории «товары для дома». Цель — понять, что чинить в первую очередь.

— **Format:** таблица с колонками: проблема, частота упоминаний (посчитай по тексту), цитата-пример, рекомендация. После таблицы — один абзац «главный вывод».

— **Constraints:** не выдумывай проблемы, которых нет в тексте; цитаты — дословно; если проблем меньше трёх, так и напиши.

Разбор 5. Личная продуктивность. План недели.

Плохо: «Помоги мне спланировать неделю».

Почему плохо: нет ни списка задач, ни приоритетов, ни ограничений по времени.

По TCFC:

— **Task:** составь план недели из списка задач ниже.

— **Context:** 5 рабочих дней, по 8 часов. Три встречи с фиксированным временем (вторник 11:00, среда 15:00, четверг 10:00). Дедлайн по отчёту — пятница 18:00. Есть две задачи, которые нельзя делать в один день (правка договора и согласование с юристом).

— **Format:** таблица «день — задачи — время на задачу». После таблицы — три риска, которые могут сломать план.

— **Constraints:** не ставить больше 3 крупных задач на день; оставить 1 час буфера ежедневно; не переносить отчёт с пятницы.

2.7. Врезка «Что пошло не так»

Что пошло не так.

Пример, иллюстрирующий механизм. Менеджер по продукту просил модель «написать описание для лендинга». Получил текст, который ему не понравился, и отправил правку: «перепиши лучше». Второй вариант оказался ещё хуже. Третий — тоже. После четвёртой итерации менеджер решил, что «модель слабая». На самом деле в промпте отсутствовали все четыре элемента: не было сказано, для кого лендинг, какая цель, какой формат блока, что нельзя использовать. Каждая итерация «перепиши лучше» заставляла модель заново угадывать те же развилки — только теперь с учётом предыдущего текста, который ей тоже не нравился. Через 15 минут менеджер переписал промпт по TCFC, и первый же ответ подошёл на 80%. Время ушло не на итерации, а на то, чтобы сформулировать задачу.

2.8. Когда TCFC избыточен

Схема из четырёх элементов — инструмент, а не ритуал. Есть задачи, где хватит двух элементов.

Простые операции. «Переведи этот абзац на английский» — Task и контекст (сам абзац) есть, Format и Constraints не нужны. Добавлять «в деловом стиле, без эмодзи» здесь бессмысленно.

Уточняющие вопросы. «Что ты имел в виду под „оптимизировать структуру“?» — это короткий запрос, которому TCFC не нужен.

Итерации внутри готового диалога. Если вы уже в контексте и правила заданы, короткие команды «сократи до 200 слов», «убери второй абзац» работают в уже настроенном контексте, где правила заданы, лучше, чем полный TCFC каждый раз.

Правило: TCFC нужен там, где результат должен быть предсказуемым с первого раза. Для черновика хватит Task. Для рабочего документа — все четыре.

Четыре элемента TCFC

T — Task (задача)

Что сделать — и с каким результатом. Пример: «Составь план занятия на 90 минут».

C — Context (контекст)

Для кого, зачем, на основе чего. Пример: «Маркетологи без техбэкграунда, 12 человек».

F — Format (формат)

Структура, длина, носитель. Пример: «Таблица: время — блок — результат».

C — Constraints (ограничения)

Что делать нельзя. Пример: «Не больше 20 минут теории подряд».

Промпт = Task + Context + Format + Constraints — задайте все четыре, и результат станет предсказуемым.

Практикум. «Перепиши пять промптов по TCFC»

Задача: взять пять своих реальных промптов и переписать их по схеме TCFC.

Шаги.

1. Откройте дневник промптов. Возьмите пять записей — свои или из примеров главы.
2. Для каждого промпта определите, каких элементов TCFC не хватает. Запишите в таблицу: Task — есть/нет, Context — есть/нет, Format — есть/нет, Constraints — есть/нет.
3. Перепишите каждый промпт по TCFC. Не сокращайте — пишите полностью.
4. Отправьте старую и новую версии в модель. Сравните результаты по трём критериям: попал ли в задачу, нужен ли формат, пришлось ли переделывать.
5. Запишите в дневник, какая версия сработала лучше и почему. Если версия по TCFC хуже — запишите гипотезу: какой элемент добавлен формально, а не по делу.

Таблица для работы:

№ | Промпт (старый) | Чего не хватало | Версия по TCFC | Результат: лучше / хуже / так же
1 | | | |
2 | | | |
3 | | | |
4 | | | |
5 | | | |

Критерий успеха. В четырёх из пяти случаев версия по TCFC даёт результат, который можно использовать без полной переделки. Если в трёх и более случаях результат не улучшился — проверьте, не добавили ли вы формальные элементы без содержания (например, Format «в виде текста» — это не формат).

Дополнительно. Возьмите промпт, который уже работал хорошо, и разберите его по TCFC. Скорее всего, вы интуитивно использовали все четыре элемента — теперь вы знаете, какие именно.

Чек-лист самопроверки

☐ Я могу назвать четыре элемента TCFC и объяснить каждый одной фразой.
☐ Я понимаю разницу между операциональным глаголом и оценочным.
☐ Я знаю, на какие три вопроса отвечает Context.
☐ Я задаю Format в словах, знаках или пунктах, а не «коротко».
☐ Я формулирую Constraints как запреты, а не как просьбы.
☐ Я могу за две минуты найти, чего не хватает в плохом промпте.
☐ Я понимаю, когда TCFC избыточен, и переписал пять промптов.

Разбор типичных ошибок

Ошибка 1. Формальный Format. В чём ошибка: в промпте написано «Format: текст». Это не формат — это отсутствие формата. Как исправить: задайте структуру (сколько абзацев, что в каждом), длину (в словах или знаках) и что не включать («без вводного абзаца»).

Ошибка 2. Context, который описывает вас, а не аудиторию. В чём ошибка: «Мне нужно, чтобы было убедительно» — это ваше состояние, а не контекст. Модель не знает, что для вас «убедительно». Как исправить: описывайте аудиторию и цель: «для скептически настроенного финансового директора, цель — согласовать бюджет».

Ошибка 3. Constraints-просьбы вместо Constraints-запретов. В чём ошибка: «пиши профессионально» — просьба. Модель понимает «профессионально» по-своему. Как исправить: переведите просьбу в запрет. «Профессионально» → «без эмодзи, без сленга, без обращений „друзья“».

Ошибка 4. Слишком много Constraints. В чём ошибка: 15 запретов превращаются в противоречивую систему. Модель пытается выполнить всё и усредняет. Как исправить: не больше 5—7 ограничений на промпт. Если ограничений больше — разбейте задачу на два промпта.

Ошибка 5. TCFC ради TCFC. В чём ошибка: вы добавляете четыре элемента к запросу «переведи на английский» и тратите время на ритуал. Как исправить: для простых операций хватит Task и контекста. Полный TCFC — для задач, где результат должен быть предсказуемым с первого раза.

ИИ-критик

Вставьте в чат:

Ты — критик промптов. Проверь мой промпт по схеме TCFC: (1) Task — операциональный ли глагол, назван ли результат; (2) Context — есть ли аудитория, цель, источник; (3) Format — задана ли структура, длина, носитель; (4) Constraints — есть ли запреты, а не просьбы. Для каждого элемента поставь «есть / частично / нет» и объясни одной фразой. Затем укажи, какой элемент критичнее всего добавить, и покажи правку «было → стало» именно для него. Не переписывай весь промпт — только проблемный элемент. В конце задай мне один вопрос, ответ на который улучшит Context.

Мой промпт: [вставить]

Связка со следующей главой

Теперь у вас есть каркас из четырёх элементов. Следующий вопрос — что делать, если модель всё равно уходит от нужного стиля или формата. Ответ: показать ей примеры. Глава 3 объясняет, когда Zero-shot достаточно, а когда без Few-shot не обойтись, и как подбирать примеры так, чтобы они не путали модель.

ГЛАВА 3. ZERO-SHOT, FEW-SHOT И РОЛЬ ПРИМЕРОВ

Цель главы

Понять, когда в промпте нужны примеры, а когда они только вредят. К концу главы вы умеете выбрать режим — без примеров или с примерами — и подобрать примеры так, чтобы модель не начала копировать не то, что нужно.

Ключевые тезисы

1. **Zero-shot** — запрос без примеров. Работает для типовых задач и понятных форматов.
2. **Few-shot** — запрос с примерами «вот как надо». Работает, когда стиль или формат трудно описать словами.
3. Больше примеров — не всегда лучше. Три хороших примера сильнее десяти случайных.
4. Примеры должны быть репрезентативными: показывать разброс, а не один и тот же случай.
5. Примеры не заменяют Task и Context. Если нет задачи, примеры только запутают.
6. Few-shot — это не обучение модели. Это подсказка в рамках одного запроса.

3.1. Что такое Zero-shot и когда его хватает

Zero-shot — это промпт, в котором вы описываете задачу словами, без единого примера «вот как надо». Вы говорите, что сделать, для кого, в каком формате, с какими ограничениями — и ждёте результат.

Для большинства типовых задач этого достаточно. «Переведи этот абзац на английский», «сократи текст до 300 слов», «составь список из 7 идей для постов», «выдели три главные проблемы из отзывов» — всё это Zero-shot. Модель видела миллионы подобных задач в обучающих данных и понимает, что от неё хотят.

Когда Zero-shot работает хорошо:

— задача типовая (перевод, суммаризация, переформулировка);

— формат стандартный (список, абзац, таблица с очевидными колонками);

— стиль нейтральный или легко описываемый словами («деловой», «без эмодзи», «до 150 слов»);

— у вас нет образца, который можно показать.

Пример.

Плохо (но это не про Zero-shot): «Напиши что-нибудь про наш продукт».

Почему плохо: нет задачи, контекста и формата. Zero-shot не спасает от пустого промпта.

Хорошо: «Напиши три варианта заголовка для лендинга курса по промпт-инжинирингу. Аудитория — менеджеры среднего звена. Формат: три строки, каждая до 60 знаков, без вводных слов. Ограничения: без слов „уникальный“ и „инновационный“».

Здесь Zero-shot достаточен: задача понятная, формат задан, стиль описан.

3.2. Где Zero-shot ломается

Zero-shot перестаёт работать, когда результат зависит от стиля, который трудно описать словами. Вы можете сказать «пиши в стиле Apple» — и получите усреднённое представление модели о стиле Apple, а не то, что вы имеете в виду. Вы можете сказать «в нашем корпоративном тоне» — и модель придумает этот тон сама.

Второй случай — нестандартный формат. Если вам нужна таблица с неочевидными колонками или структура документа, которую проще показать, чем описать, словами вы будете объяснять дольше, чем приведёте пример.

Третий случай — специфическая логика. Например, вы хотите, чтобы модель классифицировала обращения клиентов по вашей внутренней системе категорий. Категории можно описать словами — но без примеров модель будет трактовать границы категорий по-своему.

Во всех трёх случаях на помощь приходит Few-shot.

3.3. Few-shot: примеры как инструкция

Few-shot — это промпт, в котором вы показываете модели несколько примеров «вход → выход» и просите сделать то же самое для нового входа.

Структура Few-shot промпта:

1. **Task —** что нужно сделать (как в TCFC).
2. **Примеры** — 1—5 пар «вход → выход», оформленных единообразно.
3. **Новый вход —** то, что нужно обработать.
4. **Format и Constraints** — как в обычном промпте, если нужно.

Пример. Few-shot для классификации обращений.

Задача: классифицируй обращение клиента по одной из четырёх категорий: «Оплата», «Доставка», «Качество», «Другое». Верни только название категории, без пояснений.

Пример 1.

Вход: «Списали деньги дважды, верните вторую сумму».

Выход: Оплата

Пример 2.

Вход: «Курьер не приехал в указанное время, перенесите на завтра».

Выход: Доставка

Пример 3.

Вход: «Товар пришёл с царапиной на корпусе».

Выход: Качество

Пример 4.

Вход: «Как поменять адрес в личном кабинете?»

Выход: Другое

Новый вход: «Заказ должен был прийти вчера, но статус не меняется уже два дня. Что происходит?»

Здесь Few-shot решает задачу, которую Zero-shot решил бы «примерно»: без примеров модель могла бы отнести обращение к «Доставке» или «Другому», а с примерами видит границу — статус заказа и сроки относятся к «Доставке».

Zero-shot против Few-shot

Zero-shot — без примеров

Задача: «Классифицируй обращение по категориям: Оплата, Доставка, Качество, Другое».

Модели нужно описать каждую категорию словами. На пограничных случаях — угадает.

Результат: «примерно», зависит от удачи формулировок.

Few-shot — с примерами

Та же задача + примеры «вход → выход»:

1. «Списали дважды…» → Оплата
2. «Курьер не приехал…» → Доставка
3. «Как поменять адрес…» → Другое

Модель видит границу категорий по примерам — пограничные случаи попадают туда, куда надо.

Результат: попадает в стиль и в категории.

Рабочий минимум — 3 примера: типичный случай, пограничный случай и случай-исключение.

3.4. Сколько примеров нужно

Частое заблуждение: «чем больше примеров, тем лучше». Это не так. Работает качество и репрезентативность, а не количество.

Один пример — минимальный Few-shot. Показывает формат, но не показывает разброс. Подходит, когда задача узкая и вариантов мало.

Три примера — рабочий минимум для большинства задач (рекомендация, не строгая норма). Три примера позволяют показать: типичный случай, пограничный случай и случай, который легко спутать с другим.

Пять примеров — имеет смысл, если у вас действительно разные категории или стили, и каждый пример добавляет новую грань.

Больше пяти — редко оправдано. Примеры занимают место в контекстном окне, а выигрыш от шестого и седьмого примера обычно нулевой. Хуже того, лишние примеры могут сузить модель: она начнёт копировать не только формат, но и содержание.

Практическое правило: начинайте с трёх. Если результат уходит от нужного стиля или категории — добавьте четвёртый и пятый, но не больше. Если и это не помогло, проблема не в количестве примеров, а в их качестве или в задаче.

Важно: примеров — не больше пяти. Ограничений в промпте — не больше 5—7. Это разные счётчики.

Пример.

Плохо: десять примеров одного и того же типа поста — все про запуск продукта.

Почему плохо: модель решит, что «правильный» пост — это пост про запуск, и будет копировать эту структуру даже там, где она не подходит.

Хорошо: три примера: пост-анонс, пост-разбор и пост-приглашение. Модель видит разброс и понимает, что формат зависит от типа.

3.5. Репрезентативность: главное требование к примерам

Репрезентативность — это когда примеры показывают разброс, а не один и тот же случай в десяти вариациях.

Что должно быть в наборе примеров:

— типичный случай — то, что встречается чаще всего;

— пограничный случай — то, что легко спутать с другой категорией;

— исключение — то, что не похоже на остальные, но всё равно относится к задаче.

Чего быть не должно:

— все примеры одинаковой длины и структуры — модель решит, что длина и структура обязательны;

— все примеры на одну тему — модель сузит область;

— примеры, которые противоречат заданию (см. 3.6);

— примеры с ошибками в форматировании — модель скопирует и ошибки.

Пример. Few-shot для генерации заголовков.

Плохо: три примера заголовков, все начинаются с глагола и заканчиваются вопросом. Модель решит, что «правильный» заголовок — это глагол плюс вопрос, и будет делать так всегда.

Хорошо: три примера разной структуры: заголовок-вопрос, заголовок-утверждение, заголовок-число. Модель увидит, что структура может быть разной, и выберет подходящую под содержание.

3.6. Ошибка, которая ломает Few-shot: примеры противоречат заданию

Частая и незаметная ошибка — примеры, которые не соответствуют тому, что вы просите в Task или Constraints. Модель видит противоречие и выбирает примеры, а не слова. Как правило, примеры сильнее инструкций, потому что они конкретны.

Что пошло не так.

Пример, иллюстрирующий механизм. Маркетолог просил: «Напиши три коротких поста для Twitter, до 140 знаков, без эмодзи». И приложил три примера «хороших постов» — длинных, по 300 знаков, с эмодзи. Модель выдала три длинных поста с эмодзи. Маркетолог удивился: «Я же написал „коротких“ и „без эмодзи“». Формально — да. Фактически — примеры сказали модели обратное: «вот как надо». Модель последовала примерам.

Правило: если примеры противоречат инструкции, модель выберет примеры. Поэтому примеры должны быть такими же, как требования: та же длина, тот же стиль, те же ограничения.

Как проверить себя: прогоните каждый пример через чек-лист Constraints. Если пример нарушает хотя бы одно ограничение — он не годится.

3.7. Few-shot — это не обучение модели

Это два разных процесса.

Few-shot в промпте — это подсказка в рамках одного запроса. Модель не «учится» на ваших примерах. Она использует их как образец, чтобы понять, что вы имеете в виду. В следующем чате, без этих примеров, она не вспомнит их.

Дообучение (fine-tuning) — это отдельный процесс, при котором модель переобучается на большом наборе данных. Это другая область, требующая других навыков и ресурсов. В этой книге мы её не касаемся.

Путаница между ними приводит к завышенным ожиданиям: «я дал модели три примера, почему она не запомнила их навсегда?». Не запомнила, потому что Few-shot — это не память, а подсказка. В новом чате примеры нужно давать заново.

3.8. Конструктор Few-Shot промпта

Собрать Few-shot промпт можно по шаблону. Заполните поля:

[Task]
Задача: [что сделать]
Формат: [структура, длина, носитель]
Ограничения: [что нельзя]
[Примеры]
Пример 1.
Вход: [текст / данные]
Выход: [результат]
Пример 2.
Вход: […]
Выход: […]

Пример 3.
Вход: […]
Выход: […]
[Новый вход]
Вход: [то, что нужно обработать]
Выход:

Три правила заполнения:

1. **Единообразие.** Все примеры оформлены одинаково: «Вход:» / «Выход:», без лишних пояснений.

2. **Репрезентативность.** Типичный, пограничный, исключение. Разная длина, если задача не требует обратного.

3. **Соответствие ограничениям.** Каждый пример проходит чек-лист Constraints.

3.9. Эксперимент: один запрос с 0, 1, 3 и 5 примерами

Чтобы увидеть эффект на практике, проведите эксперимент.

Задача: сгенерировать заголовок для статьи о том, как компании внедряют промпт-инжиниринг.

Шаг 1. Zero-shot. Промпт: «Придумай заголовок для статьи о том, как компании внедряют промпт-инжиниринг. Формат: одна строка до 70 знаков». Запишите результат.

Шаг 2. Один пример. Добавьте один пример «хорошего» заголовка: «Как мы сэкономили 200 часов: опыт внедрения промпт-инжиниринга». Запишите результат.

Шаг 3. Три примера. Добавьте три примера разной структуры: заголовок-вопрос, заголовок-утверждение, заголовок-число. Запишите результат.

Шаг 4. Пять примеров. Добавьте ещё два — с другим тоном (более деловым и более разговорным). Запишите результат.

Что сравнивать:

— Попал ли заголовок в тему статьи.
— Похож ли он на примеры (и не слишком ли).
— Можно ли его использовать без правки.

Ожидаемый результат: Zero-shot даст усреднённый заголовок. Один пример сузит стиль до одного варианта. Три примера дадут разброс. Пять — возможно, ещё больше разброса, но не всегда лучше трёх. Если пять примеров дали результат хуже трёх — вы увидели, что «больше» не значит «лучше».

Каждый шаг — в новом чате, чтобы примеры не смешивались.

3.10. Врезка «Что пошло не так»

Что пошло не так.

Пример, иллюстрирующий механизм. Методист готовил курс и просил модель «написать описание модуля в нашем стиле». Приложил три примера описаний — все из одного курса, все в одном формате, все про технические темы. Модель выдала описание нового модуля, идеально похожее на примеры — но новый модуль был про Soft Skills, и описание получилось неживым: технический тон, сухие формулировки, ни одного человеческого примера. Методист решил, что «модель не умеет в Soft Skills». На самом деле примеры сказали модели: «вот как надо» — и она сделала так. Ошибка была в репрезентативности: три примера из одной области не показывали разброс. Исправление: добавить один пример из другого курса, с другим тоном и другой структурой.

Практикум. «Три варианта email с разным количеством примеров»

Задача: сгенерировать email-приглашение на вебинар, используя 0, 1 и 3 примера, и сравнить результаты.

Шаги.

1. Откройте чат с моделью. Отправьте Zero-shot промпт: «Напиши email-приглашение на вебинар о промпт-инжиниринге для HR-специалистов. Формат: тема письма +3 абзаца по 2—3 предложения. Ограничения: без эмодзи, без слов „уникальный“ и „инновационный“». Сохраните результат как **Вариант 0**.

2. Откройте новый чат. Отправьте тот же промпт, добавив **один пример** хорошего email-приглашения (свой или вымышленный). Сохраните как **Вариант 1**.

3. Откройте новый чат. Отправьте тот же промпт, добавив **три примера** разной структуры: официальный, дружеский, с акцентом на боль. Сохраните как **Вариант 3**.

4. Сравните три варианта по критериям: попал ли в аудиторию (HR-специалисты); есть ли структура (тема +3 абзаца); соблюдены ли ограничения (без эмодзи, без запрещённых слов); можно ли использовать без переделки.

5. Запишите в дневник промптов: какой вариант лучше и почему. Если Вариант 1 лучше Варианта 3 — запишите гипотезу, почему (например, три примера дали слишком большой разброс и модель растерялась).

Критерий успеха. Вы можете объяснить, почему один из вариантов с примерами лучше Zero-shot, и почему больше примеров не всегда лучше. Если все три варианта оказались примерно одинаковыми — вероятно, задача слишком типовая для Few-shot, и это тоже полезный вывод.

Дополнительно. Возьмите свой рабочий промпт, который даёт «средний» результат, и попробуйте улучшить его через Few-shot: добавьте три репрезентативных примера. Сравните с исходным.

Чек-лист самопроверки

☐ Я понимаю разницу между Zero-shot и Few-shot.

☐ Я знаю, когда Zero-shot достаточно, а когда нужны примеры.

☐ Я могу назвать три требования к хорошим примерам: репрезентативность, единообразие, соответствие ограничениям.

☐ Я знаю, почему больше примеров не всегда лучше.

☐ Я понимаю, что Few-shot — это не обучение модели, а подсказка в рамках запроса.

☐ Я могу проверить примеры на противоречие заданию.

☐ Я провёл эксперимент с 0, 1, 3 и 5 примерами и записал результат.

Разбор типичных ошибок

Ошибка 1. Примеры противоречат заданию. В чём ошибка: вы просите «коротко и без эмодзи», а в примерах — длинные тексты с эмодзи. Модель выберет примеры. Как исправить: прогоните каждый пример через чек-лист Constraints. Если пример нарушает ограничение — замените его.

Ошибка 2. Все примеры одинаковые. В чём ошибка: три примера одной структуры и одного тона. Модель решит, что «правильно» — только так. Как исправить: добавьте разброс: типичный, пограничный, исключение. Разная структура, разный тон, разная длина.

Ошибка 3. Слишком много примеров. В чём ошибка: десять примеров занимают контекст, сужают модель и не дают выигрыша. Как исправить: начните с трёх. Добавляйте по одному, только если результат уходит от нужного. Не больше пяти.

Ошибка 4. Примеры без Task. В чём ошибка: вы дали три примера «хороших постов», но не сказали, что именно нужно сделать с новым входом. Модель копирует примеры вместо того, чтобы применить их к задаче. Как исправить: сначала Task и Format, потом примеры, потом новый вход. Примеры — это иллюстрация к задаче, а не замена ей.

Ошибка 5. Ожидание, что модель «запомнит» примеры. В чём ошибка: вы дали примеры в одном чате и ждёте, что в новом чате модель их вспомнит. Как исправить: Few-shot работает в рамках одного запроса. В новом чате примеры нужно давать заново. Если нужна память — это другая технология (fine-tuning), и она вне рамок этой книги.

ИИ-критик

Вставьте в чат:

Ты — критик Few-shot промптов. Проверь мой промпт по пяти критериям: (1) есть ли Task и Format; (2) сколько примеров и достаточно ли их для задачи; (3) репрезентативны ли примеры — есть ли типичный, пограничный и исключение; (4) единообразно ли оформлены примеры; (5) не противоречат ли примеры ограничениям из Constraints. Для каждого критерия поставь «ок / есть проблема» и объясни одной фразой. Затем укажи самое слабое место и покажи правку «было → стало» именно для него. Не переписывай весь промпт. В конце задай мне один вопрос, ответ на который улучшит набор примеров.

Мой промпт: [вставить]

Связка со следующей главой

Теперь у вас есть три инструмента: структура TCFC, режим Zero-shot и режим Few-shot с примерами. Следующий вопрос — можно ли собрать из них фреймворк, который работает для целых классов задач, а не для одной. Глава 4 даёт CO-STAR и другие системы, учит выбирать фреймворк под тип задачи и показывает, где фреймворк начинает мешать.

РАЗДЕЛ II. ИНСТРУМЕНТЫ

Цель раздела: собрать арсенал проверенных подходов — фреймворки, роли, пошаговое рассуждение, ограничения — и научиться выбирать инструмент под задачу, а не применять всё сразу.

ГЛАВА 4. ФРЕЙМВОРК CO-STAR И ДРУГИЕ СИСТЕМЫ

Цель главы

Научиться выбирать фреймворк под тип задачи и не путать фреймворк с ритуалом. К концу главы вы умеете применять CO-STAR, сравнивать его с TCFC и понимать, когда фреймворк начинает мешать, а не помогать.

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.