Зубков Андрей

ИИ на пальцах

как работает нейросеть и как ее использовать

Зубков Андрей

ИИ на пальцах: как работает нейросеть и как ее использовать

https://litres.ru/74057609
SelfPub; 2026

Аннотация

"ИИ на пальцах: как работает нейросеть и как ее использовать" — это увлекательный самоучитель по искусственному интеллекту. Чтобы понять эту книгу и научиться работать с нейросетями, не нужно быть экспертом. Она написана максимально простым языком и ориентирована на непрофессионалов.

Вы узнаете:

- Что такое ИИ, как он устроен, за счет чего он обучается и принимает решения

- Какие ошибки в работе с нейросетями допускают люди и как их избежать

- Как именно вам использовать нейросети для упрощения вашей жизни

Вы научитесь использовать нейросети для работы, решения бытовых задач и даже для творчества. А также получите крепкую

теоретическую базу, которая позволит легко разбираться в современных ИИ-моделях и особенностях их работы.

Если для вас важно не просто понять технологию, а научиться применять нейросети, то эта книга станет отличным помощником.

Содержание

Зубков Андрей
ИИ на пальцах: как работает нейросеть и как ее использовать

Глава 1. Зачем вам разбираться в ИИ

Вы, скорее всего, уже встречались с ИИ так: нужно быстро придумать идеи для поста, написать письмо клиенту, составить план обучения или разобраться в новой теме. Вы открываете чат, вводите запрос, получаете ответ — и дальше начинается сомнение. Можно ли этому верить? Почему ответ звучит уверенно, но местами странно? Что уточнить, чтобы стало лучше? И где граница: когда ИИ помогает, а когда только добавляет путаницы?

Типичные задачи, где ИИ уже полезен новичку, обычно простые и «текстовые».

Во-первых, поиск идей: варианты тем, заголовков, примеры формулировок, список шагов.

Во-вторых, черновики текстов: письмо, резюме, описание товара, план статьи — не как финал, а как заготовка.

В-третьих, обучение: попросить объяснить понятие простыми словами, сравнить два подхода, составить мини-план занятий и контрольные вопросы.

Во всех этих ситуациях ИИ экономит время, но только если вы умеете направлять его и проверять результат.

Ключевой принцип простой: ИИ стоит воспринимать как инструмент, который помогает думать и оформлять мысли, а не как «кнопку правильного ответа». Он дает полезные варианты, но не гарантирует точность и не понимает вашу задачу без контекста. Поэтому важнее не умение один раз нажать кнопку, а умение задавать рамки, уточнять и проверять.

Как это работает на уровне логики пользователя. Когда вы «просто нажимаете кнопку», вы отдаете управление двум вещам: случайности формулировки и недостатку исходных данных. ИИ отвечает на то, что вы написали, а не на то, что вы имели в виду.

Если запрос расплывчатый, ответ будет либо общим, либо уверенно неправильным в деталях. Если вы не проверяете,

вы рискуете принять красивый текст за факт. Если не уточняете, вы получаете «среднюю температуру по больнице», а не решение вашей ситуации.

Понимание принципов работы нужно не для теории ради теории, а чтобы решать три практические проблемы.

Первая проблема — ожидания. Многие ждут от ИИ «как от эксперта»: точных фактов, ссылок, ответственности за результат. Но ИИ устроен иначе: он хорошо продолжает текст и предлагает вероятные варианты, а не «знает истину» по умолчанию.

Когда вы это понимаете, вы перестаете удивляться ошибкам и начинаете строить запрос так, чтобы снизить их вероятность: просите ограничения, формат, уточняющие вопросы.

Вторая проблема — качество результата. Без понимания принципа «ИИ не читает мысли» легко получить длинный ответ, который не подходит по тону, объему или цели. С пониманием вы начинаете управлять входом: добавляете контекст (кто вы, для кого текст, какая цель), задаете формат (список, таблица, письмо), указываете ограничения (без сложных терминов, до 120 слов, с примерами). Это не «магия промптов», а обычная постановка задачи.

Третья проблема — безопасность и ответственность. Даже в бытовых задачах важно помнить: ИИ может придумать несуществующие факты, перепутать даты, уверенно сослаться на «исследования», которых нет. Он также может неудачно обобщать или выдавать спорные советы.

Понимание границ помогает выбрать правильную роль ИИ: не «реши за меня», а «предложи варианты», «помоги составить план», «проверь на логические дыры», «составь список вопросов, которые нужно уточнить у специалиста».

Как понять, что вы усвоили:

— Вы можете объяснить одной фразой, почему ответ ИИ нельзя принимать как факт без проверки.

— Вы отличаете «полезный черновик» от «готового решения», которое требует ответственности.

Теперь — один цельный сценарий, как применять этот принцип в реальной задаче. Представим, вам нужно написать письмо: попросить коллегу помочь с задачей и согласовать сроки. Вы открываете ИИ и не пишете «составь письмо», потому что это почти гарантирует слишком общее и не в вашем стиле. Вместо этого действуете пошагово.

Шаг 1. Формулируете цель и контекст: кто пишет, кому,

зачем, какой тон.

Шаг 2. Задаете формат и ограничения: длина, структура, что обязательно упомянуть.

Шаг 3. Просите ИИ сначала задать вопросы, если данных не хватает.

Шаг 4. Получив черновик, проверяете: факты, тон, конкретика, нет ли лишних обещаний.

Шаг 5. Уточняете и доводите до финала.

Готовый запрос, который можно копировать:

«Помоги составить письмо коллеге. Контекст: я — менеджер проекта, пишу разработчику. Цель: попросить помочь с задачей X и согласовать срок. Тон: уважительно, без давления, по делу. Обязательные детали: что нужно сделать (2 пункта), почему это важно (1 предложение), предложить 2 варианта сроков, попросить подтвердить. Ограничения: до 120 слов, без канцелярита. Если информации не хватает — сначала задай до 3 уточняющих вопросов».

Дальше вы получаете текст и делаете короткую проверку. Совпадает ли задача X с реальностью? Нет ли «лишних» деталей, которых вы не говорили? Есть ли конкретные варианты сроков? Тон не слишком жесткий или, наоборот, слишком мягкий?

Если что-то не так, вы не начинаете заново — вы уточняете: «сделай тон более нейтральным», «убери фразу про срочность», «добавь, что я могу помочь с тестированием».

Как понять, что вы усвоили:

— Вы умеете превратить «напиши письмо» в запрос с целью, контекстом и ограничениями.

— Вы после ответа делаете минимум одну проверку, а не сразу отправляете текст.

Чтобы чтение этой книги было полезным, поставьте себе цель не «узнать все про ИИ», а научиться стабильно получать от него результат в типовых задачах и понимать, где он может ошибаться. Удобная цель для новичка звучит так: «Я умею выбрать подходящий ИИ-инструмент под задачу, сформулировать понятный запрос и проверить ответ перед использованием».

Измерить прогресс можно простыми признаками, без тестов и математики:

— Вы можете объяснить своими словами, почему ИИ иногда ошибается, и что вы делаете, чтобы это заметить.

— Вы умеете написать запрос из 4 частей: цель, контекст, формат, ограничения — и получаете более предсказуемый результат.

— Вы используете ИИ хотя бы в одной бытовой или ра-

бочей задаче как черновик и доводите итог сами, а не копируете ответ вслепую.

Глава 2. Что такое ИИ простыми словами

Частая ситуация: вы слышите «у нас в продукте ИИ», видите кнопку «Сделать с помощью нейросети», пробуете — и не понимаете, что именно там «умного». Это отдельная программа? Это ChatGPT внутри? Это просто набор правил? Из-за этого сложно выбрать инструмент под задачу и легко поверить обещаниям, которые звучат красиво, но не дают понятного результата.

Ключевой принцип простой: «ИИ» — это не одна вещь, а ярлык для разных уровней: общая идея, конкретная модель, сервис вокруг модели и готовое приложение. Если вы научитесь мысленно раскладывать «ИИ» на эти уровни, станет ясно, что именно вам предлагают и чего от этого ждать.

Термин «модель» здесь означает обученную систему, которая по входным данным выдает результат по шаблону, выученному на примерах. Она не «понимает мир» как человек.

Как это работает на практике — разложим по слоям смысла, без технических деталей.

ИИ как общая идея — это подход: заставить компьютер решать задачи не только по заранее прописанным правилам, а по «опыту», полученному из данных. Данные — это примеры (тексты, картинки, таблицы, записи), на которых система учится находить закономерности.

Модель — это «сжатый опыт» из этих данных. Она не хранит весь интернет как папку с файлами, а запоминает, какие ответы обычно подходят к похожим входам. Поэтому модель хорошо делает то, что похоже на ее обучающие примеры, и хуже — то, что выходит за рамки.

Сервис — это упаковка модели в удобный инструмент. В сервисе есть интерфейс, ограничения, фильтры безопасности, иногда — подключение к интернету, к вашим документам, к таблицам, история запросов. Две разные компании могут использовать похожую модель, но сервисы будут вести себя по-разному из-за настроек и дополнительных функций.

Приложение — это конкретная кнопка или функция в знакомой программе: «суммировать письмо», «улучшить текст», «сгенерировать картинку», «подобрать товары». Внутри может быть реальная модель, а может быть более простой механизм. Пользователю важно не название, а что именно функция делает, на каких данных и с какими ограничениями.

Как понять, что вы усвоили:

— Вы можете объяснить разницу между «моделью» и «приложением» на примере любой кнопки «AI».

— Вы понимаете, что слово «ИИ» само по себе ничего не гарантирует без уточнения уровня.

Теперь — про задачи: где ИИ обычно сильнее человека, где слабее, а где вообще не должен быть единственным решением.

ИИ лучше человека в рутинной обработке больших объемов однотипной информации. Например: быстро сделать черновик письма в нужном тоне, кратко пересказать длинный текст, предложить несколько вариантов заголовков, разложить список идей по категориям, найти типичные формулировки, привести текст к единому стилю. Тут ценность в скорости и количестве вариантов.

ИИ хуже человека там, где важны точные факты, ответственность и понимание контекста «в реальном мире». Модель может звучать уверенно и при этом ошибаться: перепутать даты, придумать источник, неверно трактовать условия.

Также ей трудно удерживать ваши скрытые цели и ограничения, если вы их не написали. Например, что письмо нельзя

отправлять без согласования, что цифры должны совпадать с отчетом, что нельзя раскрывать персональные данные.

Есть задачи, которые ИИ не решает «сам по себе», даже если выглядит убедительно. Он не может гарантировать правду без проверки, не может принимать решения за вас в ситуациях с риском, не может «узнать» то, чего нет в его данных или в предоставленных вами материалах.

ИИ не является свидетелем событий, не имеет доступа к вашей реальности, если вы явно не дали ему данные. Поэтому «проверь, правда ли это» без источников превращается в угадывание.

Как понять, что вы усвоили:

— Вы можете назвать одну задачу, где ИИ полезен как черновик, и одну, где нужен человек как финальный контролер.

— Вы понимаете, что уверенный тон ответа не равен точности.

Остается третий важный навык: отличать «маркетинговый ИИ» от реального.

«Маркетинговый ИИ» — это когда слово «ИИ» используют как украшение, но вам не объясняют, что именно делает

функция и как проверить результат. Признаки обычно такие: обещают «решит все», не говорят про ограничения, не дают примеров входа и выхода, невозможно повторить результат, нет понятного способа оценить качество.

Реальный сервис с моделью обычно можно распознать по более приземленным вещам:

— Есть ясное описание задач: что именно делает функция (например, «суммирует текст до 5 пунктов», «генерирует варианты ответа в заданном стиле»), а не «повышает эффективность».

— Есть управляемость: вы можете задать контекст, формат, ограничения и получить результат, который меняется предсказуемо от ваших уточнений.

— Есть границы: указано, что возможны ошибки, что нужен контроль фактов, что есть ограничения по данным (например, длина текста, типы файлов, язык).

— Есть проверяемость: результат можно сравнить с исходными данными, попросить привести источники (если сервис их реально умеет), получить несколько вариантов и выбрать.

Важно: даже «настоящий» ИИ не обязан быть идеальным. Разница не в том, ошибается он или нет, а в том, можете ли вы понять, что он делает, и контролировать качество.

Один сценарий, который помогает собрать все вместе.

Вы выбираете инструмент для задачи: «Нужно подготовить короткое описание продукта для сайта и 5 вариантов заголовка». В рекламе сервиса написано: «ИИ сделает продающий текст за секунды».

Шаг 1. Разложите обещание по уровням. Спросите себя: это модель, сервис или просто кнопка в редакторе? Если это кнопка, что она делает конкретно: пишет с нуля или перефразирует ваш текст?

Шаг 2. Проверьте, есть ли управляемость. Вставьте минимальный контекст и требования. Готовый запрос, который можно копировать:

«Сделай черновик описания продукта для сайта. Продукт: [что это]. Для кого: [аудитория]. Главная выгода: [1–2 пункта]. Ограничения: не обещать того, чего нет; без сложных терминов; 600–800 знаков. Формат: 1 абзац + 5 вариантов заголовка.»

Шаг 3. Оцените, где ИИ силен, а где нужен контроль. ИИ быстро даст варианты формулировок — это его сильная сторона. Но вы проверяете факты: нет ли выдуманных характеристик, не нарушены ли ограничения, не появились ли «гарантии», которых вы не даете.

Шаг 4. Отделите реальность от маркетинга по результату. Если после уточнений текст становится ближе к вашим требованиям и вы можете повторить процесс на другом продукте — это похоже на реальный инструмент. Если ответы хаотичны, не держат ограничения, а «умность» сводится к красивым словам — перед вами либо слабая реализация, либо просто маркетинговая наклейка.

Как понять, что вы усвоили:

— Вы можете описать любой «AI-инструмент» через четыре уровня: идея → модель → сервис → приложение.

— Вы умеете заранее определить, где ИИ поможет (черновик, варианты, структура), а где нужна проверка человеком (факты, ответственность, контекст).

— Вы знаете признаки, по которым «ИИ» в описании превращается из слова в проверяемую функцию.

Что унести из этой главы:

— Слово «ИИ» полезно только тогда, когда вы уточнили: какая модель, какой сервис и какая конкретная функция вам нужна.

— Оценивайте ИИ не по обещаниям, а по управляемости и проверяемости: можете ли вы задать требования и получить результат, который можно контролировать и проверять.

— Держите простое правило: ИИ хорош как помощник

для черновиков и обработки информации, но не как единственный источник истины и решений.

Глава 3. От данных к решению: общий цикл работы ИИ

Обычно новичок видит ИИ как кнопку: ввёл запрос — получил ответ. Но дальше начинаются вопросы. Почему один и тот же запрос сегодня работает, а завтра — хуже? Кто «решил», что показывать в рекомендациях? И где в этом всём место человека: он просто пользователь или от него что-то зависит?

Ключевая идея простая: любой ИИ-сервис работает по одному и тому же циклу. Сначала собирают данные, потом на них обучают модель, а потом эту модель используют для получения результата в конкретной ситуации.

Этот цикл можно представить как цепочку из четырёх звеньев.

Первое звено — данные. Данные — это примеры из реального мира, на которых ИИ учится: тексты, картинки, клики, оценки, истории покупок, записи разговоров, ответы людей. Важно понимать: данные не «истина», а просто следы того, что уже происходило. Если в данных есть перекосы (например, одни темы представлены, другие почти нет), модель по-

том будет повторять этот перекос.

Роль человека на этом этапе обычно самая большая, хотя её не видно. Люди решают, какие данные собирать и можно ли их собирать вообще. Настраивают правила: что считать «хорошим» примером, что удалить, как обезличить (убрать привязку к конкретному человеку), как пометить ошибки. И даже если данные собираются автоматически, человек задаёт рамки: какие события логировать, какие кнопки считать «успехом», какие — «ошибкой».

Как понять, что вы усвоили:

— вы можете ответить, какие данные нужны сервису, чтобы он работал (хотя бы на уровне «клики/тексты/оценки»);

— вы понимаете, что качество результата ограничено качеством данных.

Второе звено — обучение модели. Модель — это программа, которая находит закономерности в данных и учится делать предсказание: какое слово дальше, какой товар предложить, какой результат показать выше. Обучение — это настройка внутренних параметров модели так, чтобы она чаще угадывала правильно на примерах из данных и реже ошибалась.

Роль человека здесь — выбрать цель обучения и крите-

рий «правильно/неправильно». Например, в рекомендациях можно оптимизировать «чтобы кликали», а можно «чтобы досматривали», а можно «чтобы возвращались через неделю». Это разные цели, и они дадут разные модели даже на одних и тех же данных. Человек также решает, когда остановиться. Если модель слишком «подгоняется» под прошлые примеры, она может хуже работать на новых ситуациях. Это называют переобучением: модель запомнила частные случаи вместо общего правила.

Как понять, что вы усвоили:

— вы можете объяснить, что модель не «понимает мир», а учится по примерам;

— вы можете назвать, кто задаёт цель обучения (не модель, а люди и компания).

Третье звено — использование модели, или инференс. Инференс — это момент, когда уже обученную модель применяют к новой ситуации: к вашему запросу, вашей фотографии, вашему профилю, вашему текущему контексту. Здесь модель не «учится заново» (в базовом смысле), а быстро считает ответ на основе того, чему научилась раньше.

Роль человека на этом этапе двоякая. Со стороны сервиса люди решают, как именно применять модель: какой контекст ей дать, какие ограничения поставить, как фильтровать

опасные ответы, как объединить несколько моделей в одну систему. Со стороны пользователя человек формулирует задачу и даёт входные данные: запрос, параметры, уточнения. Чем точнее вход, тем выше шанс получить полезный результат.

Как понять, что вы усвоили:

— вы различаете «модель обучают заранее» и «модель используют сейчас»;

— вы понимаете, что ваш запрос — это часть входных данных для инференса.

Четвёртое звено — результат. Это то, что вы видите: список ссылок, лента рекомендаций, ответ чат-бота, подсказка в письме, распознавание речи. Но результат почти никогда не равен «чистому ответу модели». Обычно есть слой правил вокруг: форматирование, ранжирование (порядок выдачи), фильтры, ограничения, подсказки, иногда — проверка другими алгоритмами.

Роль человека здесь снова ключевая. Создатели сервиса решают, как показывать результат и как измерять качество: по кликам, по жалобам, по времени на странице, по оценкам. Пользователь отвечает за применение: проверить важные факты, не передавать лишнее, не воспринимать ответ как гарантию. И если результат используется для решения

(например, что купить, что прочитать, как ответить клиенту), ответственность за финальный выбор остаётся у человека.

Как понять, что вы усвоили:

— вы можете отделить «ответ модели» от «как сервис его показал»;

— вы помните, что решение принимает человек, даже если подсказку дал ИИ.

Теперь посмотрим, как этот цикл проявляется в привычных сервисах — одним цельным сценарием, чтобы увидеть ход событий.

Представьте, вы открыли приложение с видео и видите ленту рекомендаций.

Шаг 1: сервис собирает данные — какие ролики вы смотрели, сколько секунд, что пропустили, что лайкнули, на что подписались. Плюс данные о самих роликах: тема, длительность, язык, популярность, жалобы.

Шаг 2: на этих данных обучают модель предсказывать, что вы с большей вероятностью посмотрите дальше. Цель задают люди: например, «удержание» или «удовлетворённость», и от этого меняется поведение рекомендаций.

Шаг 3: когда вы открываете ленту, происходит инференс: модель получает ваш недавний контекст (последние про-

смотры, время суток, устройство) и выдаёт список кандидатов.

Шаг 4: сервис формирует итоговую ленту: часть роликов может быть добавлена по правилам (новости, подписки), часть — убрана фильтрами, порядок — переставлен. Вы видите результат и решаете, что смотреть. Ваши действия снова становятся данными — цикл замыкается.

То же самое, только с другими данными и целями, происходит в поиске, рекомендациях товаров и чат-ботах. В поиске данные — это тексты страниц и поведение пользователей, результат — ранжированный список. В магазине данные — покупки и просмотры, результат — «вам может понравиться». В чат-боте данные — тексты, на которых его обучали, инференс — генерация ответа на ваш запрос, результат — текст, который вы должны оценить и при необходимости проверить.

После этой главы стоит унести две вещи. Во-первых, когда вы видите «умный» ответ или рекомендацию, мысленно раскладывайте её на цикл: какие данные могли быть использованы, чему модель училась, что ей дали на вход сейчас, как сформировали итоговый результат. Во-вторых, помните про роль человека: создатели задают цели и правила, а пользователь отвечает за корректную постановку задачи и за финальное решение, особенно там, где цена ошибки высокая.

Глава 4. Данные: из чего «кормят» ИИ

Обычно знакомство с ИИ начинается с вопроса: «Откуда он вообще это знает?» Кажется, что модель «читала интернет» и теперь умеет отвечать на любые темы. Но потом вы просите что-то простое — и получаете странную ошибку, уверенный вымысел или ответ с перекосом. В этот момент полезно понять не «как устроены нейроны внутри», а из чего ИИ учится — то есть какие данные ему дают.

Ключевой принцип простой: модель учится на данных и становится похожей на них. Данные — это примеры из реального мира, которые показывают, что бывает «входом» (что мы подаём) и «выходом» (что хотим получить). Если данных мало, они однообразные или в них много мусора, модель будет ошибаться именно в этих местах. Если данные перекошены в одну сторону, модель тоже будет перекошена.

Данные бывают разными по форме, и это важно, потому что форма определяет, чему можно научить модель.

Текст — это письма, статьи, чаты, инструкции, отзывы.

Большие языковые модели в основном питаются текстом: они учатся продолжать фразы и подбирать слова по контексту. Поэтому они сильны в объяснениях, черновиках, резюме, но могут «уверенно сочинять», если в данных не хватило точных примеров или если вопрос требует свежих фактов.

Картинки — это фотографии товаров, снимки документов, изображения с подписями, картинки с объектами (например, «кот», «машина»). На картинках учат распознавать, что изображено, находить объекты, улучшать качество, генерировать новые изображения по описанию.

Звук — это речь, музыка, шумы. На аудиоданных учат распознавание речи (транскрибацию), синтез речи, определение звуков (например, «звонок», «аплодисменты»). Тут важно качество записи: шум, разные микрофоны и акценты сильно влияют на результат.

Числа — это таблицы, измерения, показатели продаж, температуры, время доставки, результаты опросов. На числовых данных учат прогнозы, поиск закономерностей, выявление аномалий. Если в таблице пропуски или разные форматы (например, даты вперемешку), модель будет «спотыкаться».

Действия пользователей — это клики, просмотры, покуп-

ки, лайки, время чтения, маршруты. На таких данных учат рекомендации: «похожим людям нравится…», «после этого обычно выбирают…». Это не про «истину», а про вероятные предпочтения, поэтому рекомендации легко уводят в сторону того, что чаще выбирали раньше.

Теперь важное различие: размеченные и неразмеченные данные.

Размеченные данные — это когда к каждому примеру добавили понятный «ярлык», то есть правильный ответ. Бытовой пример: у вас есть папка с фотографиями, и на каждой подпись: «кот» или «собака». Или таблица заявок, где рядом с текстом обращения стоит метка: «срочно» / «не срочно». Разметка отвечает на вопрос: «Как правильно?» Благодаря этому модель учится делать конкретное предсказание: по входу выдавать нужную метку, число или текст.

Неразмеченные данные — это когда есть только сами примеры без готовых ярлыков. Бытовой пример: у вас просто куча отзывов клиентов без отметки «позитивный/негативный». Или набор фотографий без подписей. Тогда модель учится находить структуру сама: группировать похожее, выделять темы, замечать повторяющиеся шаблоны. Это полезно, когда вы не знаете заранее, какие категории нужны, или когда разметка слишком дорогая и долгая.

Как понять, что вы усвоили:

— Размеченные данные отвечают на «что правильно», неразмеченные — на «что похоже на что».

— Разметка — это не магия, а человеческая подпись к примеру.

Дальше — то, что сильнее всего влияет на качество ответов: качество и разнообразие данных.

Качество — это насколько данные точные, чистые и соответствуют задаче. Если в данных много ошибок, дубликатов, случайных фраз, неправильных подписей, модель выучит эти ошибки как норму. Например, если часть писем в обучающей выборке помечена «спам», хотя это обычные рассылки, фильтр начнёт выкидывать полезное. Если в картинках «кошка» иногда подписана как «собака», распознавание будет путаться.

Разнообразие — это насколько данные покрывают разные случаи, а не только «средний» вариант. Если модель училась на фотографиях товаров только при хорошем освещении, она начнёт ошибаться на снимках из квартиры вечером. Если распознавание речи училось в основном на одном акценте, оно будет хуже понимать другой. Если рекомендации обучались на поведении узкой группы людей, они будут на-

вязывать её вкусы всем остальным.

Из качества и разнообразия вытекают две типичные проблемы: ошибки и предвзятость.

Ошибки появляются там, где данных не хватило или они «грязные». Модель не понимает редкий случай, путает похожие вещи, уверенно продолжает текст не туда. Это не «плохой характер ИИ», а следствие того, что в данных не было достаточного числа похожих примеров или правильных ответов.

Предвзятость — это устойчивый перекос в сторону одной группы, стиля или точки зрения. Она возникает, когда в данных одна сторона представлена сильнее, чем другая, или когда разметка делалась людьми с одним взглядом. В быту это видно так: модель чаще предлагает «типичные» варианты и хуже справляется с тем, что встречалось реже. Важно понимать: предвзятость может появиться даже без злого умысла — просто из-за перекоса в том, что собирали и что считали «нормой».

Как понять, что вы усвоили:

— Если модель систематически ошибается в одном типе случаев, ищите «дыру» в данных: там мало примеров или они плохие.

— Если ответы перекошены в одну сторону, скорее всего, перекошены были и данные.

Представьте простой сценарий. Вам нужно настроить ИИ-помощника для поддержки клиентов: он должен читать входящее сообщение и предлагать черновик ответа. Вы собираете данные: прошлые письма клиентов (текст) и ответы операторов (тоже текст).

Дальше вы решаете, будет ли разметка. Если вы хотите, чтобы система сначала определяла тему обращения, вы добавляете метки: «доставка», «оплата», «возврат», «техподдержка». Это размеченные данные: у каждого письма есть ярлык. Если меток нет, вы можете начать с неразмеченного подхода: сгруппировать письма по похожести и посмотреть, какие темы получаются естественно.

Потом вы проверяете качество: убираете письма с персональными данными, удаляете дубли, исправляете явные ошибки в метках (например, «возврат» помечен как «оплата»). И отдельно проверяете разнообразие: есть ли письма короткие и длинные, спокойные и раздражённые, с опечатками, с разными формулировками одной проблемы, из разных каналов (почта, чат).

Если вы обучите помощника только на «идеальных» пись-

мах без эмоций и без опечаток, он станет теряться на реальных сообщениях. Если в данных почти нет случаев «сложного возврата», модель будет уверенно отвечать шаблоном и ошибаться именно там, где важна точность.

Что унести из этой главы:

— ИИ учится на данных и повторяет их свойства: сильные стороны и слабые места.

— Данные бывают разными (текст, картинки, звук, числа, действия), и это определяет, чему модель вообще может научиться.

— Размеченные данные — с «правильными ответами», неразмеченные — без них; качество и разнообразие данных напрямую влияют на ошибки и перекосы в ответах.

Глава 5. Как ИИ «видит» информацию

Обычно путаница начинается в момент, когда вы слышите фразу «модель обучилась на данных». Кажется, что ИИ «смотрит» на текст как человек, «видит» картинку как человек и «понимает» ваши действия в приложении как человек.

А потом вы сталкиваетесь с неожиданным: два почти одинаковых сообщения дают разные ответы, картинка с очевидным котом распознаётся странно, рекомендации в сервисе выглядят неуместно. Возникает вопрос: что именно ИИ получает на вход и почему он так реагирует?

Ключевая мысль простая: ИИ не видит мир напрямую — он работает с признаками, то есть с числами, которые описывают объект, текст или действие в удобном для модели виде. Признак — это измеримая «деталь», которую можно записать числом: длина текста, наличие слова, цвет пикселя, время клика, категория товара.

Модель не «читает» и не «смотрит» как человек; она получает набор чисел и учится находить в них закономерности.

Как понять, что вы усвоили:

— Вы можете закончить фразу: «Признаки — это числа, которые…»

— Вы понимаете, что один и тот же объект можно описать разными наборами признаков.

Чтобы превратить реальный объект в признаки, всегда происходит один и тот же переход: «вход → кодирование → числа». Кодирование — это способ перевода информации в числовой вид.

Дальше модель уже не различает, откуда пришли числа: из текста, изображения или истории кликов. Для неё это просто набор значений, по которым она делает вывод.

У признаков есть три важных свойства, которые влияют на результат.

Первое — что именно вы решили измерять. Если вы описываете текст только его длиной, вы теряете смысл. Если изображение описываете только средним цветом, вы теряете форму. Признаки — это выбор того, какие детали «разрешено» видеть модели.

Второе — насколько признаки похожи на задачу. Для разных задач полезны разные описания. Если вы хотите отли-

чать «спам» от «не спама», важны слова, ссылки, повторяемость. Если хотите предсказывать «понравится ли фильм», важны жанры, актёры, история просмотров. Один и тот же набор признаков может быть отличным для одной цели и бесполезным для другой.

Третье — качество и устойчивость признаков. Хороший признак стабилен: он измеряется одинаково сегодня и завтра и не ломается от мелочей. Плохой признак случайный или «шумный»: он меняется из-за деталей, которые не должны влиять на ответ (например, лишние пробелы, формат даты, случайная опечатка), и модель начинает учиться на ерунде.

Как понять, что вы усвоили:

— Вы можете назвать три свойства: «что измеряем», «насколько связано с задачей», «насколько устойчиво».

— Вы понимаете, что признаки — это не “истина”, а выбранный способ описания.

Теперь примеры, чтобы стало видно, как это выглядит в разных типах данных.

В тексте признаки могут быть очень простыми: количество слов, наличие конкретных слов («скидка», «срочно»), язык текста, наличие ссылок, доля заглавных букв. Могут быть и более смысловыми: какие темы встречаются, ка-

кие слова стоят рядом, какой тон (нейтральный/эмоциональный).

Но в любом случае текст сначала превращают в числа. Даже если вы общаетесь с чат-моделью, внутри она работает не с «буквами», а с токенами — кусочками текста, которым соответствуют числовые коды. Дальше модель опирается на вероятности: какие продолжения чаще встречались в похожих числовых контекстах.

В изображениях признаки начинаются с пикселей: каждый пиксель — это числа, которые задают яркость и цвет. Но «кот» как идея не лежит в пикселях готовым. Поэтому полезными оказываются признаки, которые отражают формы и структуры: края, контуры, текстуры, сочетания линий, расположение объектов.

Чем ближе признаки к тому, что нужно распознать (например, «есть ли круглая морда и уши»), тем легче модели учиться. Если же оставить только грубые числа вроде «средний цвет картинки», модель будет уверенно ошибаться: средний цвет у кота и у дивана может совпасть.

В пользовательских действиях признаки — это следы поведения, которые можно измерить: что вы открывали, что искали, на что нажимали, сколько времени смотрели, что до-

бавляли в корзину, в какое время суток активны, с какого устройства заходите. Часто добавляют признаки контекста: город (в грубом виде), язык интерфейса, тип подписки, категория товара.

На основе таких чисел системы рекомендаций пытаются угадать, что вам будет полезно дальше.

Как понять, что вы усвоили:

— Вы можете привести по 2–3 примера признаков для текста, изображения и действий.

— Вы видите, что «признак» — это не обязательно “умная” штука; иногда это простые счётчики и отметки.

Самая неприятная часть — когда признаки выбраны плохо. Тогда модель может выдавать ответы, которые выглядят «странно», хотя формально она делает всё правильно: она честно опирается на те числа, которые ей дали.

Плохой выбор признаков приводит к трём типичным проблемам.

Первая — модель учится на совпадениях, а не на смысле. Например, если в обучающих данных «срочные письма» были короткими, а «несрочные» — длинными, и вы дали модели только длину текста, она начнёт считать короткие письма

срочными. Смысл письма при этом вообще не учитывается, и результат будет раздражающе неверным.

Вторая — модель цепляется за «подсказки», которые случайно оказались рядом с правильным ответом. Допустим, на фото с собаками был зелёный газон, а на фото с кошками — домашний интерьер. Если признаки слишком грубо описывают фон, модель может начать «распознавать» не животных, а фон. Тогда собака на диване внезапно станет «кошкой», потому что числа больше похожи на «домашнюю сцену».

Третья — модель становится чувствительной к мелочам. Если признаки устроены так, что лишний пробел, другая раскладка, другой формат даты сильно меняют числовое представление, ответы будут «прыгать». Вам будет казаться, что ИИ непредсказуем, хотя на самом деле вы каждый раз подаёте на вход заметно разные числа.

Один цельный сценарий, который можно повторять в жизни, — проверка «на какие признаки я надеюсь» перед запросом к чат-модели.

Представьте, вы хотите, чтобы ИИ помог составить письмо клиенту, который недоволен задержкой. Вы пишете: «Составь ответ клиенту про задержку». Часто получается либо

слишком сухо, либо слишком оправдательно, либо без конкретики.

Почему так? Потому что по вашему короткому тексту модель видит мало полезных чисел: нет признаков про тон, нет признаков про контекст, нет признаков про ограничения.

Действия по шагам:

Шаг 1. Добавьте признаки контекста (что произошло и какие факты важны).

Пример запроса:

«Составь письмо клиенту. Ситуация: заказ №123, задержка 3 дня из-за сбоя на складе, новая дата доставки — 15 июня. Клиент уже писал один раз.»

Шаг 2. Добавьте признаки цели и тона (каким должен быть результат).

«Цель: сохранить доверие и предложить компенсацию. Тон: спокойный, уважительный, без оправданий.»

Шаг 3. Добавьте признаки формата (как ответ должен выглядеть).

«Формат: 6–8 предложений, сначала извинение, затем факт, затем план, затем компенсация, затем контакт для уточнений.»

Шаг 4. Добавьте ограничения (что нельзя делать).

«Не упоминать внутренние процессы компании, не обвинять клиента, не обещать невозможного.»

После этого ответы обычно становятся заметно полезнее не потому, что модель «стала умнее», а потому что вы дали ей больше правильных числовых сигналов: про задачу, про стиль, про структуру. Вы как бы помогли модели «увидеть» то, что для вас важно.

Как понять, что вы усвоили:

— Вы можете объяснить, какие «признаки» добавились в запросе: факты, цель, тон, формат, ограничения.

— Вы понимаете, почему короткий запрос даёт расплывчатый результат: мало информации для кодирования в числа.

Запомнить стоит следующее. ИИ работает с признаками — числовым описанием входа, а не с «пониманием» как у человека. Для текста, изображений и действий признаки выглядят по-разному, но логика одна: сначала перевод в числа, потом поиск закономерностей.

Если признаки бедные или случайные, ответы будут странными: модель начнёт опираться на длину, фон или ме-

лочи вместо смысла. Попробуйте в своих запросах добавлять признаки контекста, цели, формата и ограничений — это простой способ сделать результат стабильнее и полезнее.

Глава 6. Модель: упрощенная схема реальности

Обычно новичок представляет ИИ как «умную программу», которая где-то внутри хранит правильные ответы. Из-за этого возникают три частых вопроса. Первый: почему модель иногда ошибается в очевидных вещах, если «она же должна знать»? Второй: чем это отличается от обычной программы, где всё заранее прописано? И третий, практический: как так получается, что одну и ту же модель используют и в чате, и в переводчике, и в помощнике для писем — это разные ИИ или один?

Ключевой принцип такой: модель — это упрощённая схема реальности, то есть шаблон, который по примерам из данных учится предсказывать результат. «Упрощённая» означает, что модель не хранит мир целиком и не понимает его как человек. Она выделяет повторяющиеся закономерности из того, что видела, и использует их, чтобы угадать наиболее подходящий ответ для нового случая.

Как это работает на уровне идеи. У модели есть вход и выход. На вход вы даёте описание ситуации: текст письма, набор признаков товара или фразу на другом языке. На вы-

ходе модель выдаёт предсказание: следующее слово, метку «спам/не спам», вероятный перевод, рекомендацию товара и так далее.

Чтобы научиться, модели нужны данные — примеры из прошлого. В данных есть то, что подаётся на вход, и то, что считается правильным результатом. Во время обучения модель много раз сравнивает свои ответы с правильными и подстраивает свои внутренние настройки так, чтобы ошибаться реже.

Эти настройки можно представить как «ручки», которые подкручиваются, пока ответы в среднем не станут лучше. Важно: модель не запоминает один-единственный правильный ответ на каждую ситуацию. Она учится общему шаблону: какие входы обычно ведут к каким выходам.

Как понять, что вы усвоили:

— вы можете сказать фразу «модель не знает, она предсказывает по примерам» и объяснить её на бытовом примере;

— вы понимаете, что «упрощённая схема» означает неизбежные ошибки и неточности.

Теперь про отличие от программы с жёстко прописанными правилами. В обычной программе разработчик заранее

описывает, что делать: «если A — сделай B, иначе сделай C». Такой подход хорошо работает там, где правила можно перечислить и они не меняются слишком: посчитать НДС, отсортировать список, проверить формат номера телефона.

Модель устроена иначе: вместо списка правил она получает примеры и сама «выводит» правила в виде настроек. Поэтому модель полезна там, где правила трудно выписать вручную. Например, можно попытаться написать строгие правила для «вежливого тона письма» или для «похоже ли это сообщение на спам», но быстро выяснится, что исключений слишком много. Модель же учится на большом количестве примеров и начинает ловить тонкие признаки, которые сложно формализовать.

Отсюда следуют практические различия:

— программа обычно ведёт себя одинаково и предсказуемо при одинаковом вводе, потому что правила фиксированы;

— модель может давать разные формулировки и иногда ошибаться, потому что она выбирает наиболее вероятный вариант, а не выполняет строгий алгоритм;

— чтобы улучшить программу, вы переписываете правила; чтобы улучшить модель, вы меняете данные, цель обучения или саму модель (а не добавляете ещё сто «если»).

Как понять, что вы усвоили:

— вы можете отличить задачу «где правила можно выписать» от задачи «где проще показать примеры»;

— вы понимаете, почему модель может быть полезной даже без стопроцентной точности.

Остаётся вопрос: как одна и та же модель может использоваться в разных приложениях. Здесь помогает простая мысль: модель — это «двигатель», а приложение — это «кузов и приборная панель». Двигатель один и тот же, но вокруг него можно построить разные сценарии использования.

Одна и та же языковая модель, например, умеет продолжать текст. Это базовое умение можно «упаковать» по-разному:

— как чат: приложение добавляет историю диалога, роль «ассистента» и удобный интерфейс вопросов-ответов;

— как переводчик: приложение каждый раз подаёт модели текст и просит выдать перевод в нужном формате;

— как помощник для писем: приложение добавляет шаблоны («тема письма», «тон», «адресат») и просит модель написать черновик;

— как инструмент для резюме: приложение просит кратко пересказать длинный текст и ограничивает длину.

В каждом случае модель внутри делает похожую вещь — предсказывает следующий фрагмент текста. Но приложение

задаёт контекст, ограничения и формат. Поэтому иногда кажется, что «это разные ИИ», хотя на деле это один и тот же базовый шаблон, применённый к разным задачам.

Один цельный сценарий. Представьте, что у вас есть одна языковая модель, а вам нужно решить три задачи на работе: ответить клиенту, сделать краткое резюме встречи и подготовить вариант перевода абзаца.

Шаг 1. Вы формулируете вход и желаемый выход для первой задачи.

Запрос (можно копировать):

«Ты — помощник по деловой переписке. Контекст: клиент недоволен сроками. Вставляю письмо клиента ниже. Сформулируй ответ: вежливо, без оправданий, предложи 2 варианта решения, длина до 120 слов. Письмо клиента: …»

Шаг 2. Ту же модель вы используете для резюме, но меняете упаковку задачи.

Запрос:

«Сделай краткое резюме текста ниже в 5 пунктах: решения, сроки, ответственные, риски, следующий шаг. Текст: …»

Шаг 3. Для перевода вы снова не меняете «двигатель», меняете требования.

Запрос:

«Переведи на английский, стиль нейтрально-деловой. Сохрани смысл, не добавляй фактов. Текст: …»

Шаг 4. Вы сравниваете: модель одна, но результаты разные, потому что вы каждый раз задаёте другой формат выхода и другой контекст. Если где-то ответ «плывёт», вы не переписываете программу, а уточняете вход: добавляете ограничения, пример желаемого тона или просите 2 варианта.

Что унести из этой главы:

— Модель — это шаблон, который учится на данных предсказывать результат; это упрощённая схема, поэтому ошибки возможны.

— Модель отличается от программы тем, что правила не прописаны вручную: она извлекает закономерности из примеров.

— Одна и та же модель может работать в разных приложениях: приложение задаёт контекст, формат и ограничения, а «двигатель предсказаний» остаётся тем же.

Глава 7. Обучение модели: как она «учится на примерах»

Новичок представляет обучение нейросети так: «Ей объяснили правила, и она запомнила». А потом возникает путаница. Если правила не писали вручную, то что именно происходит во время обучения? Почему нужны «тонны данных»? И почему говорят, что обучать модель дорого и долго, а пользоваться потом — быстро?

Ключевая идея простая: модель учится на примерах. Ей много раз показывают пару «вход → правильный ответ», а она подстраивается так, чтобы ошибаться всё реже.

Под «входом» здесь понимается то, что вы подаёте модели: текст, картинку, таблицу, звук — в зависимости от задачи. «Правильный ответ» — это то, что считается верным результатом для этого входа. Например, для письма это может быть следующий фрагмент текста, для картинки — подпись «кот/не кот», для таблицы — нужная категория или число. Такой подход называют обучением с учителем: есть примеры и есть правильные ответы, по которым можно проверять себя.

Как это работает по шагам.

Шаг 1: модель получает вход и выдаёт свой вариант ответа. На старте он обычно плохой, потому что модель ещё «не настроена».

Шаг 2: этот ответ сравнивают с правильным. Разница между ними — это ошибка. Ошибка не обязательно «неправильно/правильно». Чаще это степень несовпадения: насколько модель промахнулась.

Шаг 3: модель чуть-чуть меняет свои внутренние настройки (их называют весами — это числа, которые определяют, как сильно модель реагирует на разные детали входа). Изменение маленькое, чтобы не «сломать» то, что уже начало получаться.

Шаг 4: то же самое повторяют на следующем примере. И так — много раз.

Важно понимать логику: модель не запоминает каждый пример как в тетрадке «вопрос–ответ». Она ищет такие внутренние настройки, при которых в среднем по множеству примеров ошибка становится меньше. Поэтому качество растёт постепенно: не потому что модель «поняла смысл», а потому что она стала лучше угадывать правильный ответ по похожим входам, опираясь на то, что встречалось в данных.

Как понять, что вы усвоили:

— вы можете своими словами объяснить, что обучение

— это повтор «предсказал → сравнил с правильным → чуть подстроился»;

— вы понимаете, что «правильный ответ» нужен, чтобы измерять ошибку и знать, куда подстраиваться.

Теперь про ресурсы и скорость. Обучение занимает много времени и вычислений по трём причинам.

Первая: примеров очень много. Чтобы модель стала устойчивой, ей нужно увидеть огромное разнообразие ситуаций: разные формулировки, стили, темы, ошибки людей, редкие случаи.

Вторая: сама модель большая — у неё много весов, которые надо подстроить. Подстройка — это вычисления, которые повторяются снова и снова.

Третья: обучение — это не один проход. Обычно данные «прокручивают» много раз, потому что с первого раза модель не находит хорошие настройки.

А использование модели после обучения — это другой режим, его называют инференс: вы дали вход, модель выдала ответ. В этот момент веса уже не меняются. Модель просто применяет то, что «настроила» раньше. Это похоже на разницу между тренировкой и выступлением: тренировка —

долгие повторения с исправлениями, выступление — один прогон по готовым навыкам. Поэтому обучать дорого (много повторов и пересчётов), а отвечать на запросы — относительно быстро (один расчёт без подстройки).

Как понять, что вы усвоили:

— вы можете объяснить разницу между обучением (веса меняются) и использованием (веса фиксированы);

— вы понимаете, почему «дорого» — это про многократные попытки на огромном количестве примеров.

Представьте задачу: вы хотите, чтобы модель помогала писать короткие вежливые письма клиентам. Берёте набор примеров: слева — исходная ситуация, справа — хороший ответ. Например: «Клиент просит перенести встречу» → «Спасибо, давайте перенесём на…»; «Клиент недоволен задержкой» → «Извините за задержку, вот статус и сроки…». Дальше процесс идёт так.

Шаг 1: подаёте модели вход: «Клиент недоволен задержкой, просит объяснить». Модель генерирует черновик. Он может быть слишком резким, слишком длинным или не по делу.

Шаг 2: сравниваете с эталонным хорошим ответом из примера. Считается ошибка: где модель не попала в тон, структуру, обязательные элементы.

Шаг 3: модель немного подстраивает веса так, чтобы в похожих ситуациях чаще выбирать более вежливые формулировки, нужный порядок фраз, уместную длину.

Шаг 4: повторяете на тысячах похожих и непохожих ситуаций: перенос, отмена, уточнение, возврат, благодарность, напоминание. Постепенно модель начинает выдавать ответы, которые чаще совпадают с тем, что в данных считалось «правильным».

После этого, когда вы в реальной работе пишете: «Составь вежливый ответ клиенту: задержка доставки на 2 дня, предложи компенсацию 5%», модель уже не учится. Она просто быстро применяет настроенные веса и выдаёт вариант письма, похожий по стилю и структуре на те, что «видела» во время обучения.

Что стоит унести и попробовать применить после прочтения:

— думайте об обучении как о многократном цикле «вход → ответ → ошибка → небольшая подстройка», который повторяется на огромном числе примеров;

— помните разницу режимов: обучать долго и дорого, потому что веса постоянно меняют; использовать быстро, потому что веса уже зафиксированы.

Глава 8. Как нейросеть «понимает», что ошиблась

Вы пробуете ИИ-сервис: он распознаёт спам, прогнозирует продажи или пишет текст. Иногда результат «не тот». Возникает естественный вопрос: как модель вообще понимает, что ошиблась, если у неё нет человеческого здравого смысла? И ещё один: почему разработчики не делают так, чтобы ошибок не было совсем — разве это не цель?

Ключевой принцип простой: ошибка модели — это разница между тем, что модель выдала, и тем, какой ответ считается правильным по заранее заданному правилу. У модели нет внутреннего чувства «я неправ». Есть только сравнение с эталоном (правильной меткой, известным числом или ожидаемыми свойствами результата) и численная оценка, насколько она промахнулась.

Как это работает на практике, зависит от типа задачи.

В классификации модель выбирает класс из набора вариантов: «спам/не спам», «кошка/собака», «одобрить/отклонить». Ошибка здесь обычно бинарная: угадала или нет. Но важна не только финальная метка, а уверенность.

Модель может сказать: «спам с вероятностью 0,51» или «спам с вероятностью 0,99». В обоих случаях класс один и тот же, но качество ощущается разным: во втором модель гораздо увереннее. Поэтому разработчики смотрят и на долю правильных ответов, и на то, насколько уверенно модель ошибается.

Как понять, что вы усвоили: в классификации «ошибка» — это несоответствие выбранного класса эталонной метке, а не «плохое объяснение» модели.

В прогнозе (регрессии) модель выдаёт число: цену квартиры, спрос на товар, время доставки. Ошибка здесь — расстояние между предсказанным числом и реальным. Если прогноз продаж был 120, а факт 100, ошибка — 20 (или 20%).

В таких задачах почти никогда не бывает «угадал идеально», и это нормально: данные шумные, мир меняется, измерения неточны. Поэтому качество оценивают средним размером промаха на многих примерах, а не одним удачным попаданием.

Как понять, что вы усвоили: в прогнозе ошибка измеряется величиной отклонения, и «небольшая ошибка» может быть приемлемой, если она стабильна.

В генерации (тексты, ответы, изображения) всё сложнее, потому что «правильного единственного ответа» нет. Если вы попросили написать письмо клиенту, вариантов хорошего письма — десятки. Поэтому ошибку задают через правила, которые можно проверить: соответствует ли ответ фактам из исходных данных, соблюдён ли формат, нет ли запрещённых утверждений, не пропущены ли обязательные пункты.

Иногда используют сравнение с эталонными примерами (референсами), но чаще качество оценивают набором критериев и проверками. Важно понимать: генеративная модель может написать гладко, но ошибиться по сути — и это будет ошибкой, даже если текст «красивый».

Как понять, что вы усвоили: в генерации ошибка — это нарушение заданных требований (факты, формат, ограничения), а не просто «мне не понравилось».

Теперь — как разработчики измеряют качество на примерах, которых модель не видела. Если проверять модель на тех же данных, на которых её учили, она может «выучить наизусть» и показать отличные результаты, не умея работать в реальной жизни.

Поэтому данные обычно делят на части. Одна часть идёт

на обучение: модель подбирает внутренние настройки так, чтобы на этих примерах ошибаться меньше. Другая часть откладывается для проверки: модель эти примеры не использует при обучении, но на них её тестируют. Смысл простой: проверка должна имитировать будущее использование, когда модель сталкивается с новыми входными данными.

На проверочном наборе считают те же показатели ошибки, что и в реальной задаче: долю правильных классификаций, среднюю величину промаха в прогнозе, выполнение требований в генерации. Если качество на обучении высокое, а на проверке заметно хуже, это признак переобучения: модель слишком подстроилась под конкретные примеры, но плохо обобщает, то есть плохо переносит знание на новые случаи.

Как понять, что вы усвоили: честная оценка качества — это оценка на данных, которые модель не видела при обучении.

Отсюда вытекает третий важный момент: нулевая ошибка — тревожный знак, а не идеальный результат. Для новичка это звучит странно, но у этого есть понятные причины.

Первая причина: утечка данных. Это ситуация, когда в обучение случайно попало то, что должно было остаться

только для проверки, или когда в признаках есть подсказка, которая «выдаёт ответ». Например, вы хотите предсказать, вернёт ли клиент кредит, и среди входных данных случайно оказалась колонка «статус возврата» (пусть даже в завуалированном виде). Модель покажет идеальную точность — но это обман, потому что в реальной жизни такой подсказки не будет.

Вторая причина: запоминание вместо понимания. Если модель слишком сложная для маленького набора данных, она может выучить частные случаи. На обучающих примерах ошибка станет нулевой, но на новых данных появятся промахи. Это похоже на ученика, который заучил ответы к конкретному тесту, но не понял тему: на другом варианте задания он теряется.

Третья причина: слишком простая задача или слишком «чистые» данные в тесте. Если проверочный набор составлен так, что в нём нет сложных случаев, модель легко покажет идеальный результат, но в реальном потоке данных начнёт ошибаться. Поэтому важно, чтобы проверка была похожа на реальность, а не на «удобные» примеры.

Один сценарий, который помогает связать всё вместе. Представьте, что вы делаете простую систему для отдела продаж: классификация «лид горячий/не горячий» и про-

гноз «сколько сделок будет на следующей неделе». Вы собрали таблицу прошлых лидов: источник, регион, сумма, дата обращения, результат (купил/не купил).

Шаг 1: вы решаете, что считать ошибкой. Для классификации — неверная метка «горячий/не горячий». Для прогноза — средний промах в количестве сделок.

Шаг 2: вы делите данные: старые записи — на обучение, часть — на проверку, и следите, чтобы одинаковые клиенты не оказались и там и там (иначе модель узнает их и «схалтурит»).

Шаг 3: вы смотрите результаты. На обучении точность 100%, на проверке тоже 100%. Это выглядит как победа.

Шаг 4: вы проверяете, нет ли утечки: находите колонку «дата выставления счёта» — она появляется только когда лид уже почти точно купил. Модель фактически использовала будущее, которое в момент оценки лида неизвестно. Вы убираете колонку, повторяете оценку — и получаете, например, 78% на проверке. Это не провал, а честная картина.

Теперь можно улучшать: собирать больше данных, уточнять признаки, менять порог уверенности, а главное — понимать, где модель ошибается и какой ценой.

После этой главы стоит унести три вещи. Ошибка — это не «мнение», а заранее определённая мера расхождения с эталоном или требованиями, и она разная для классификации, прогноза и генерации. Качество проверяют на данных, которых модель не видела, иначе оценка будет слишком оптимистичной. И если вы видите нулевую ошибку, это повод искать утечку данных или запоминание, а не повод расслабляться.

Глава 9. Переобучение: когда модель «зазубрила»

Вы пробуете ИИ-сервис: на одних примерах он отвечает идеально, будто «считывает мысли», а на чуть изменённом запросе внезапно начинает путаться. Или вы видите демонстрацию модели, где она безошибочно повторяет ответы из обучающих примеров, но стоит дать новую задачу — и качество резко падает. У начинающих это вызывает простой вопрос: модель «умная» или просто выучила набор заготовок?

Ключевой принцип здесь такой: переобучение — это когда модель слишком хорошо подстроилась под свои учебные примеры и из-за этого хуже работает на новых. «Учебные примеры» — это данные, на которых модель училась. Переобучённая модель не столько понимает общий смысл задачи, сколько запоминает частные случаи и мелкие детали, которые случайно встретились в обучении.

Как это работает на уровне логики. Во время обучения модель ищет закономерности: что в данных повторяется и помогает делать правильный ответ. Но в данных есть два слоя.

Первый — устойчивые сигналы (например, что в письме-жалобе обычно есть проблема, просьба и тон). Второй — случайные детали (например, конкретные фразы, редкие имена, совпадения формата, «любимые» слова в нескольких примерах). Если модель начинает опираться на случайные детали, она выглядит сильной на знакомых примерах, но «ломается» на новых.

Признаки переобучения проще всего понять на бытовой аналогии «знает учебник, но не решает новые задачи». Представьте ученика, который выучил ответы к контрольной из сборника. Если попадаются те же задачи — всё отлично. Если числа поменяли местами или задачу переформулировали — ступор. У модели это проявляется так же: она уверенно отвечает, пока запрос похож на то, что она уже видела, и резко теряет качество, когда меняются условия.

Как понять, что вы усвоили:

— Вы можете объяснить разницу между «запомнила примеры» и «выучила правило» на любом простом примере (школьные задачи, шаблоны писем).

— Вы ожидаете, что качество нужно проверять на новых входных данных, а не только на «похожих как в примере».

Почему переобучение чаще случается, когда модель слишком сложная, а данных мало. «Сложная модель» — это

модель с большим количеством внутренних настроек (параметров), которые можно подогнать под данные. Чем больше таких настроек, тем больше свободы «подстроиться».

Если данных мало, модель может подобрать настройки так, чтобы идеально совпасть именно с этими примерами — включая случайные детали. Это похоже на ситуацию, когда у вас есть огромный набор регуляторов, а вы пытаетесь настроить звук по двум песням. Можно добиться идеала на этих двух, но на третьей всё будет плохо, потому что настройка получилась слишком «под конкретные треки», а не под общий стиль.

На простом уровне это можно представить так:

— Мало данных = мало разных ситуаций, чтобы увидеть, что является общим правилом.

— Слишком много «свободы» в модели = можно выучить не правило, а «память» о примерах.

— Итог = высокая точность на учебных примерах и слабая — на новых.

Как понять, что вы усвоили:

— Вы можете своими словами сказать: «мало данных не “учит обобщать”, а большая модель умеет подгонять ответы слишком точно».

— Вы понимаете, почему идеальная работа на демонстра-

ционных примерах не гарантирует качества в реальности.

Теперь — взгляд со стороны пользователя, без доступа к внутренностям модели. Вам важно заметить, что сервис ведёт себя «слишком узко» и нестабильно. «Слишком узко» — это когда он хорошо справляется только в очень конкретном формате запроса или на одном типе задач, а шаг в сторону резко ухудшает результат. «Нестабильно» — когда небольшие изменения формулировки дают непропорционально разные ответы.

Типичные сигналы, которые можно увидеть в обычном использовании:

— Резкая чувствительность к формулировкам: вы меняете пару слов, а ответ становится то отличным, то слабым.

— Хорошо работает только при «правильном» шаблоне: стоит убрать привычные подсказки (тон, структуру, пример) — и качество падает сильнее, чем ожидается.

— Плохо переносит вариации: просите то же самое, но для другого города, другой аудитории или другого формата — и появляются странные ошибки.

— Уверенный тон при шатких деталях: ответ звучит гладко, но в нём много конкретики, которая не выдерживает проверку, особенно в необычных случаях.

Один цельный сценарий. Допустим, вы используете

ИИ-сервис, чтобы писать ответы клиентам в поддержку. Вы сделали себе «идеальный» запрос и на пяти прошлых письмах получили отличные черновики. Кажется, что задача решена.

Шаг 1. Вы проверяете перенос на новые ситуации. Берёте два новых письма: одно короткое и эмоциональное, другое — длинное, с несколькими вопросами. Просите сервис ответить тем же стилем.

Шаг 2. Вы делаете небольшие изменения и смотрите, «плывёт» ли качество. Меняете в запросе только одно: вместо «ответь вежливо и по делу» пишете «ответь коротко и по делу». Потом меняете формат: «сначала извинение, потом решение» → «сначала решение, потом извинение».

Шаг 3. Вы фиксируете признаки «узости». Если сервис внезапно начинает:

— забывать часть вопросов из длинного письма,

— давать слишком общий ответ без конкретных шагов,

— путать условия возврата/сроки, которые раньше писал правильно,

— или сильно менять тон без причины,

это похоже на поведение, когда он «держится» за конкретный шаблон и хуже справляется с вариациями.

Шаг 4. Вы делаете практичный вывод как пользователь. Вы решаете использовать сервис не как «автоматический ответчик», а как генератор черновика, который обязательно проходит быструю проверку по чек-листу: все вопросы учтены, сроки и условия не выдуманы, тон соответствует ситуации. То есть вы учитываете, что при узком и нестабильном поведении нельзя доверять результату без контроля.

Что унести из этой главы:

— Переобучение — это не «модель плохая», а ситуация, когда она слишком подстроилась под учебные примеры и хуже переносит знания на новые случаи.

— Риск выше, когда данных мало, а модель слишком «мощная»: ей проще запомнить детали, чем выучить общее правило.

— Как пользователь вы замечаете это по «узости» и нестабильности: небольшая смена формулировки или условий даёт непропорционально разные ответы, поэтому качество нужно проверять на вариациях задачи, а не только на одном удачном шаблоне.

Глава 10. Что происходит, когда вы делаете запрос?

Вы открываете чат с ИИ, пишете вопрос и ждёте ответ. Иногда он приходит мгновенно, иногда — через заметную паузу. Иногда модель «помнит», о чём вы говорили выше, а иногда будто игнорирует важную деталь. Бывает и так: в одном сервисе ответ длинный и подробный, в другом — короче и осторожнее, хотя запрос почти тот же. На старте это выглядит как каприз или «настроение» системы, но на деле там есть понятная логика.

Ключевая идея простая: в момент запроса модель не «думает как человек», а быстро считает наиболее вероятный ответ на основе вашего текста и контекста. Этот момент называется инференс — применение уже обученной модели для получения результата. Обучение было раньше, а инференс — это «использование» модели здесь и сейчас.

Когда вы отправляете запрос, сначала происходит перевод текста в числа. Компьютер не умеет работать с буквами напрямую, ему нужны числовые представления. В языковых моделях текст режется на кусочки — токены (это могут быть слова, части слов или знаки). Каждый токен превращается в

набор чисел, который отражает, «что это за кусочек» с точки зрения модели. Это не словарь значений, а удобный для вычислений код.

Дальше модель прогоняет эти числа через свои внутренние слои. Если очень упрощать, внутри есть много «переключателей» — веса. Вес — это число, которое показывает, насколько сильно одна часть модели влияет на другую. В процессе обучения эти веса настроили так, чтобы модель чаще угадывала правильное продолжение текста. Во время инференса веса уже не меняются: модель просто применяет их, как настроенный фильтр.

Результат вычислений — не готовое предложение, а вероятности: какой следующий токен сейчас наиболее вероятен. Модель выбирает следующий токен, добавляет его к уже написанному и повторяет процесс много раз, пока не получится ответ нужной длины или пока сервис не остановит генерацию. Поэтому полезно помнить: модель буквально «дописывает текст» шаг за шагом.

Как понять, что вы усвоили:

— Вы можете объяснить фразой: «Модель превращает мой текст в числа и по шагам выбирает следующие токены по вероятности».

— Вы понимаете, что при инференсе модель не учится

заново, а использует уже настроенные веса.

Почему же ответ так сильно зависит от контекста и истории диалога? Потому что для модели «вход» — это не только последняя ваша фраза. Входом становится весь текст, который сервис отправил модели: ваш вопрос, предыдущие реплики, иногда системные инструкции (например, «отвечай вежливо», «не выдавай опасные советы»), а также скрытые подсказки вроде выбранного тона или роли. Всё это вместе называется контекстом — набор текста, на который модель опирается, когда предсказывает следующий токен.

Отсюда два практических следствия. Первое: если в истории диалога есть неточность, модель может продолжать опираться на неё, потому что «видит» её как часть входа. Второе: если важная деталь была сказана давно, она может не попасть в текущий контекст — не потому что модель «забыла», а потому что сервис не передал ей всю историю из-за ограничений по длине.

Конец ознакомительного фрагмента.