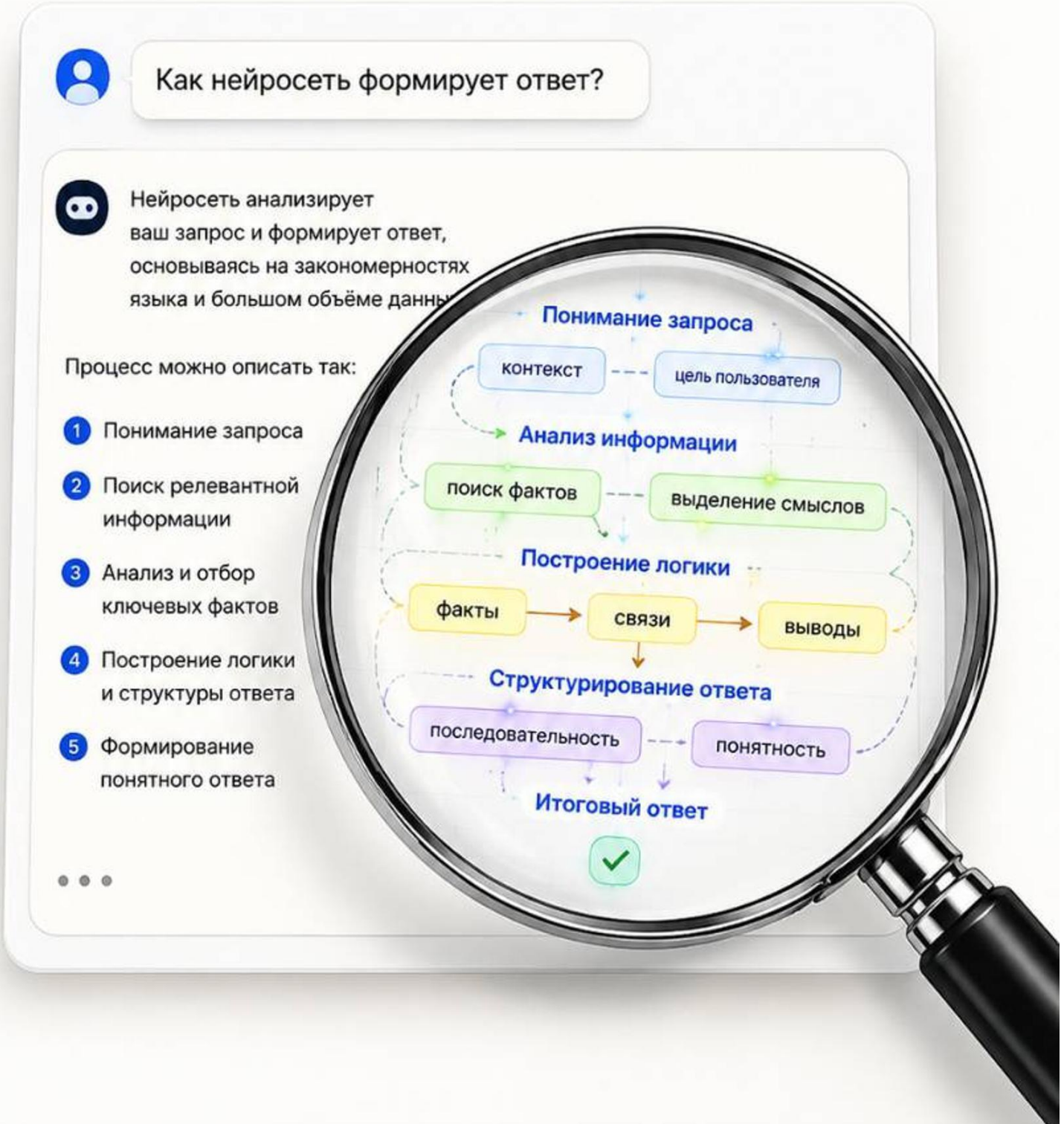
Зубков Андрей

Как понять нейросеть: секреты мышления ИИ



Зубков Андрей

Как понять нейросеть: секреты мышления ИИ

«Автор»
2026

Андрей З.

Как понять нейросеть: секреты мышления ИИ / З. Андрей — «Автор», 2026

Из этой книги вы узнаете, как нейросети подготавливают вам ответ и почему порой они выдают недостоверную информацию. Руководство написано простым и понятным языком, для неспециалистов.Приятный бонус: после чтения вы не просто поймете принципы мышления нейросетей, но и научитесь с ними эффективно работать. Есть несколько простых правил, которые поднимают эффективность работы с любым ИИ в несколько раз.Желаю вам приятного чтения.

Содержание

Зубков Андрей
Как понять нейросеть: секреты мышления ИИ

Глава 1. Зачем понимать нейросеть пользователю

Вы открываете чат с ИИ, пишете обычный запрос и получаете ответ. Иногда он получается полезным: черновик письма, список идей, краткое объяснение темы, план текста. Но в другой раз всё идёт странно. ИИ уверенно пишет факты, которых вы не находите, путает детали, забывает важное из вашего сообщения или предлагает решение, которое не подходит вашей ситуации. Возникает ощущение, что это работает «как-то само»: то угадывает, то нет, и непонятно, как на это повлиять.

Такое чаще всего случается в простых, повседневных задачах, с которых начинают почти все. Попросили переформулировать текст — получили слишком официальный стиль. Попросили идеи для поста — получили банальности. Попросили краткий вывод по статье — получили выводы, которых в статье не было. Попросили составить резюме — ИИ добавил навыки, о которых вы не говорили.

Вы не делали ничего «неправильно», просто инструмент ведёт себя не так, как ожидается от «умного собеседника», который понимает вас как человек.

Ключевой принцип здесь простой: нейросеть не «понимает» задачу так, как понимает человек. Она продолжает текст наиболее вероятным образом по вашему запросу и контексту. Это важно не как теория, а как практическая настройка ожиданий.

Если воспринимать ответ как результат угадывания по шаблонам, а не как проверенное знание, становится ясно, почему появляются ошибки и почему одну и ту же задачу можно получить по-разному.

Из этого принципа следует понятная логика поведения.

Во-первых, модель опирается на формулировку запроса. Если запрос общий, она выбирает общий, усреднённый ответ: «как обычно пишут», «как обычно советуют», «как обычно объясняют». Если вы не указали цель, аудиторию, ограничения и формат, модель не может их надёжно «догадаться» — она подставит наиболее типичный вариант. Поэтому для письма она легко уйдёт в канцелярит, для идей — в клише, для инструкции — в слишком широкие рекомендации.

Во-вторых, модель не проверяет факты автоматически. Она может звучать уверенно, потому что уверенный тон — тоже распространённый шаблон текста. Отсюда берутся «галлюцинации» — правдоподобные, но неверные детали: вымышленные источники, неточные цифры, несуществующие функции сервиса, перепутанные даты.

Это не «обман», а побочный эффект генерации. Если в вашем запросе не хватает данных или тема неоднозначна, модель всё равно продолжит текст, заполняя пробелы самым правдоподобным вариантом.

В-третьих, ответы могут быть нестабильными. Даже при похожих запросах результат меняется, потому что генерация текста включает элемент выбора из нескольких подходящих продолжений. Иногда это помогает — появляются новые формулировки и идеи. Иногда мешает — ответ «плывёт», появляются лишние допущения, меняется стиль.

Если воспринимать это как работу вероятностного генератора, становится понятнее, почему «вчера было нормально, а сегодня странно» и почему полезно фиксировать требования в запросе.

И наконец, «магическое» ощущение часто возникает из-за смешения ролей. Пользователь ждёт от ИИ одновременно поиск в интернете, экспертную проверку, понимание контекста жизни и аккуратное следование инструкции. Но модель делает другое: она хорошо собирает связный текст и помогает думать, если вы задаёте рамки. Когда рамок нет, она заполняет их сама — и именно это выглядит как непредсказуемость.

Представьте типичный сценарий. Вам нужно написать письмо в поддержку сервиса: описать проблему, приложить шаги, попросить решение. Вы пишете в чат: «Составь письмо в поддержку, у меня не работает оплата». В ответ приходит длинное письмо с общими фразами, без конкретики, а иногда ещё и с выдуманными деталями: «я пробовал три карты», «ошибка 502», «переустанавливал приложение» — хотя вы этого не говорили. Вы чувствуете, что ИИ «сам придумал», и теперь непонятно, можно ли это отправлять.

Если держать в голове ключевой принцип, ход действий меняется. Вы понимаете: модель продолжает текст и будет заполнять пробелы. Значит, пробелы нужно закрыть вами.

Вы дописываете запрос так, чтобы у модели не было причин фантазировать: указываете, что именно случилось, на каком шаге, какие условия важны, какой тон нужен, и просите не добавлять фактов. Например:

«Напиши короткое письмо в поддержку. Факты: оплата не проходит в веб-версии, на шаге подтверждения появляется сообщение “Платёж отклонён”. Дата: сегодня. Я пробовал одну карту, повторять попытки не хочу. Не добавляй никаких действий, которых я не делал. Стиль: вежливо, по делу. Структура: 1) описание, 2) шаги, 3) что прошу сделать».

Теперь модель делает то, в чём она сильна: собирает аккуратный текст из заданных деталей и формата. А вы уже знаете, что проверять: соответствуют ли все утверждения вашим фактам и нет ли лишних допущений.

После этого вы начинаете пользоваться ИИ спокойнее и точнее. Вы меньше ждёте «чудесного понимания» и больше управляете входными данными: что именно нужно, в каких границах, в каком виде. И вы иначе относитесь к ошибкам: не как к загадке, а как к сигналу, что запрос был слишком общий, контекста не хватило или модель заполнила пробелы правдоподобными догадками.

Запомнить стоит три вещи.

Во-первых, новичок чаще всего использует ИИ для текста, идей и кратких выводов — и именно там непонимание возникает из-за общих запросов и ожидания «как у человека».

Во-вторых, странные ответы и ошибки обычно появляются не потому, что вы «не умеете», а потому что модель генерирует правдоподобный текст и может додумывать недостающее.

В-третьих, базовое понимание принципа даёт практический рычаг: задавайте рамки, уточняйте факты и формат — так ответы становятся заметно полезнее и предсказуемее.

Глава 2. Что такое нейросеть простыми словами

Вы открываете чат с ИИ и задаёте простой вопрос: «Составь письмо клиенту» или «Объясни тему». Иногда ответ получается точным и полезным, а иногда — странным: то слишком официально, то с выдуманными фактами, то с ошибками в деталях. Возникает ощущение, что система то «понимает», то «не понимает», и непонятно, как на это влиять.

Ключевая идея простая: нейросеть — это не программа с жёсткими правилами, а система, которая учится на примерах и поэтому выдаёт вероятный ответ, похожий на то, что она видела в данных.

Обычная программа работает по принципу «если → то». В ней заранее записаны правила: если вы нажали кнопку — сделать строго определённое действие; если число больше 10 — вывести одно сообщение, иначе другое. Такая программа не «догадывается»: она либо следует правилам, либо ломается, если ситуация не предусмотрена.

Нейросеть устроена иначе. Ей не прописывают подробные правила для каждого случая. Вместо этого её обучают на большом количестве примеров, чтобы она научилась находить закономерности.

У языковой модели (так называют нейросеть, которая работает с текстом) задача похожа на продолжение фразы: по началу текста угадать, какие слова обычно идут дальше. На практике это выглядит как ответы на вопросы, пересказ, составление писем и планов. Но внутри это всё равно «подбор следующего фрагмента текста» по вероятности.

Важно понимать, на каких данных она учится. Языковые модели обучаются на текстах: книги, статьи, сайты, инструкции, обсуждения, а также на примерах диалогов, где показано, как люди задают вопросы и как обычно отвечают.

Это не «база знаний» в привычном смысле и не список проверенных фактов. Это огромная коллекция текстовых примеров, из которых модель выучила, как обычно устроены объяснения, письма, аргументы, определения, стили речи и типовые связи между словами.

Отсюда следуют три практических последствия для ожиданий пользователя.

Первое: модель хорошо справляется с задачами, где нужен текст как форма. Черновик письма, план статьи, варианты заголовков, переформулировка, краткое резюме, список идей, объяснение «простыми словами» — всё это похоже на то, что встречается в текстах и диалогах, поэтому модель часто даёт удобную заготовку.

Второе: модель не проверяет реальность того, что пишет. Если в вашем запросе не хватает данных или вопрос требует точных фактов (даты, цифры, ссылки, условия конкретного договора), модель всё равно попытается ответить связно. Она не «останавливается», как программа с ошибкой, а продолжает текст наиболее правдоподобным способом. Поэтому могут появляться выдуманные детали: не из злого умысла, а потому что задача модели — сделать ответ похожим на правильный по форме.

Третье: модель не обязана отвечать одинаково каждый раз, даже на похожие вопросы. Поскольку она выбирает вероятные продолжения, небольшие изменения формулировки, контекста диалога или даже просто «случайность» генерации могут сдвинуть ответ. Это нормально для системы, которая работает по вероятностям, а не по фиксированным правилам.

Представьте бытовой сценарий. Вам нужно написать письмо клиенту: сообщить о переносе срока и предложить варианты. Вы пишете в чат: «Составь письмо клиенту о переносе срока на неделю». Модель выдаёт вежливый текст, но добавляет причину переноса, которую вы не называли, и обещает скидку, о которой вы не говорили.

Если воспринимать ИИ как обычную программу, это выглядит как «ошибка». Если помнить ключевой принцип, становится понятнее: модель видела много писем, где перенос объясняют причиной и смягчают компенсацией, и поэтому «достроила» недостающие элементы.

Как действовать в этом сценарии, если вы хотите управлять результатом? Вы уточняете входные данные, которые нельзя выдумывать, прямо в запросе:

1) кому пишем (роль и тон),
2) что точно произошло (перенос на 7 дней),
3) что нельзя добавлять (не придумывать причины и компенсации),
4) что нужно получить (2 варианта: короткий и развёрнутый).

После этого модель всё равно будет генерировать текст, но уже в более узких рамках. Ей проще «угадать» правильное продолжение, когда вы дали опорные факты и ограничения.

Запомнить стоит следующее:

- Нейросеть отличается от обычной программы тем, что не следует жёстким правилам, а продолжает текст по вероятности на основе выученных примеров.
- Языковая модель обучается на текстах и примерах диалогов, поэтому сильна в типовых текстовых задачах, но не гарантирует фактическую точность.
- В повседневных задачах от неё разумно ожидать хороший черновик и структуру, но факты, детали и обязательства нужно задавать явно и проверять, потому что модель может правдоподобно «достроить» недостающее.

Глава 3. Что делает языковая модель, а чего не делает

Частая ситуация у новичка такая: вы задаёте вопрос, получаете уверенный, связный ответ — и автоматически воспринимаете его как результат «размышления» или «поиска правды». А потом выясняется, что в тексте есть выдуманные факты, странные советы или модель «не понимает», что для вас важно. Из-за этого появляется ощущение непредсказуемости: то помогает, то подводит, и непонятно, как на это влиять.

Ключевой принцип простой: языковая модель не ищет истину и не «думает», она генерирует текст, подбирая наиболее вероятное продолжение по вашему запросу и контексту переписки. «Генерирует» значит: строит последовательность слов так, чтобы она была похожа на те тексты, на которых модель училась, и подходила к текущей фразе. Это не проверка фактов, а очень быстрый подбор продолжения предложения.

Чтобы этот принцип стал практичным, важно понять, как устроена логика ответа. На вход модель получает ваш текст: вопрос, уточнения, примеры, ограничения, а также предыдущие сообщения в диалоге. Всё это становится «контекстом» — набором подсказок, на которые модель опирается, когда продолжает текст.

Дальше модель по шагам выбирает следующее слово (точнее, кусочек слова) и так собирает весь ответ. Она не открывает сайты, не сверяет даты, не смотрит вашу почту и не видит реальную ситуацию вокруг вас, если вы её не описали словами.

Отсюда следует важное ограничение: модель может звучать уверенно, даже когда ошибается. Уверенный тон — это просто стиль текста, который часто встречается в обучающих данных, а не сигнал «проверено».

Если вы спросили «какой закон это регулирует» или «какая точная цифра», модель может сгенерировать правдоподобный вариант, потому что он хорошо выглядит как продолжение вопроса. Но сама по себе она не обязана отличать факт от правдоподобной выдумки, если вы не задали режим проверки (например, попросили источники и критерии) и не перепроверили результат.

Вторая часть принципа — про «личность» модели. У неё нет собственных целей, эмоций и намерений. Она не «хочет помочь», не «обижена», не «пытается вас переубедить» и не «помнит вас как человека» вне того, что написано в текущем диалоге.

Когда кажется, что модель проявляет эмпатию или упрямство, это обычно следствие того, что такой стиль ответа часто подходит к подобным запросам. Модель также не понимает контекст вашей жизни: что для вас допустимо, какие у вас ограничения по времени, бюджету, здоровью, правилам на работе. Если вы этого не указали, она будет заполнять пробелы типичными предположениями и может попасть мимо.

Из этого вытекают ожидания, которые лучше сразу отбросить, чтобы не разочаровываться. Первое: не ждать, что модель «сама догадается», что вы имели в виду, и уточнит важные детали. Иногда она задаёт вопросы, но это тоже сгенерированное поведение, а не гарантия.

Второе: не ждать, что она автоматически проверит факты, документы, ссылки и актуальность. Если вы не попросили проверки и не дали источники, ответ может быть просто правдоподобным текстом.

Третье: не ждать, что она будет «на вашей стороне» как человек и учтёт скрытые цели: например, что вам нужен короткий ответ, что вы не хотите конфликтов в письме, что вы избегаете юридических рисков. Всё это нужно проговаривать.

Представим сценарий. Вы пишете: «Составь письмо в управляющую компанию: у нас в подъезде холодно, пусть решат. Сделай убедительно». Модель выдаёт длинный текст с уверенными формулировками, может даже сослаться на «нормы» и «обязательства», а в конце добавить угрозу жалобой. Вы отправляете — и получаете конфликт, потому что тон слишком жёсткий, а упомянутые нормы не те или вообще выдуманы.

Если помнить ключевой принцип, вы действуете иначе: сначала даёте контекст, который модель сама не знает. Например: «Письмо должно быть вежливым, без угроз. Укажи, что температура в квартире 18°C, измеряли вечером, есть фото термометра. Не упоминай конкретные статьи закона, если не уверен. Сначала задай 3 вопроса, какие данные нужны».

Модель теперь генерирует текст в заданных рамках: тон, структура, ограничения. А вы дополнительно проверяете то, что всё равно не гарантируется: факты и уместность. Если нужны нормы — просите отдельно: «Перечисли возможные нормативы и пометь, где нужна проверка по источнику», а затем сверяете с официальным документом или сайтом вашей УК/города.

После этой главы стоит унести три вещи. Во-первых, языковая модель — это генератор текста по вероятности, а не поисковик истины. Во-вторых, у неё нет собственных целей и «понимания вашей жизни»: она опирается только на написанный контекст. В-третьих, чтобы не разочаровываться, заранее убирайте ожидания «сама проверит» и «сама догадается» и вместо этого явно задавайте рамки: цель, тон, ограничения и то, что нужно уточнить перед ответом.

Глава 4. Как модель видит текст: токены

Иногда кажется, что вы написали короткий запрос, а модель отвечает: «слишком длинно» или внезапно обрывает ответ на полуслове. Или вы видите в сервисе счётчик «символов», «слов» или «токенов», и числа выглядят странно: вроде бы текста немного, а лимит уже близко. Из-за этого сложно планировать: сколько контекста можно дать, поместится ли переписка, почему один и тот же объём текста в разных языках «весит» по-разному.

Ключевая идея простая: модель работает не со словами и не с символами, а с токенами — небольшими кусочками текста, из которых она «собирает» и запрос, и ответ.

Токен — это фрагмент текста, который модель воспринимает как одну единицу. Это может быть целое короткое слово, часть длинного слова, знак препинания, пробел, кусок числа или даже сочетание букв. Поэтому «одно слово» для человека не гарантирует «один токен» для модели. Например, короткие частые слова часто становятся одним токеном, а редкие, длинные или сложные (особенно с суффиксами, дефисами, смешением языков) могут разбиваться на несколько.

Почему так сделано, вам как пользователю важно не «внутри», а по последствиям. Модель ограничена по количеству токенов, которые она может обработать за раз. Это ограничение включает и ваш запрос, и часть предыдущего диалога, и системные инструкции сервиса, и будущий ответ модели.

То есть вы всегда делите один общий «запас» токенов между входом и выходом: чем длиннее контекст, тем меньше места остаётся на ответ.

Отсюда же вытекает влияние на стоимость и лимиты. Во многих сервисах цена и ограничения считаются именно в токенах: отдельно за входные токены (то, что вы отправили) и за выходные (то, что модель сгенерировала). Если вы вставили большой текст «для справки», вы увеличили вход. Если попросили «напиши подробно на 2000 слов», вы увеличили выход.

Даже если интерфейс показывает «символы» или «слова», внутри почти всегда всё сводится к токенам. Поэтому ощущение «я же не так много написал» может не совпадать с расчётом.

Почему счётчики иногда выглядят нелогично. Во-первых, токены не равны символам: один токен может быть одним символом (например, редкий знак) или несколькими символами. Во-вторых, разные языки и стиль письма дают разную «плотность» токенов. Текст с большим количеством длинных слов, специальных терминов, ссылок, кода, таблиц, эмодзи, необычных кавычек и тире часто превращается в большее количество токенов, чем ожидается. В-третьих, пробелы и пунктуация тоже «весят»: иногда пробел идёт вместе со словом, иногда отдельно — и это меняет счёт.

Третье последствие — неожиданные обрывы ответа. Если модель упирается в максимальное число выходных токенов, она прекращает генерацию, даже если мысль не закончена. Снаружи это выглядит как «оборвалось», «не дописала список», «закончила на половине предложения».

Это не обязательно ошибка логики; чаще это просто лимит по токенам на ответ или общий лимит окна, куда уже не помещается продолжение. Похожий эффект бывает, когда в диалоге накопилось много текста: сервис вынужден «урезать» старые сообщения, чтобы уложиться в предел, и модель продолжает без части контекста. Из-за этого ответ может стать короче или менее связным.

Представьте сценарий. Вы хотите, чтобы модель помогла отредактировать договор на 6 страниц: вставляете весь текст и просите «проверь на риски и предложи правки». Модель начинает отвечать, но через пару абзацев останавливается или пропускает разделы. Вы делаете вывод «она не справилась», хотя проблема может быть в том, что вход уже занял большую часть лимита, а на выход осталось мало токенов.

Что можно сделать в рамках понимания токенов: сократить вход (например, дать только спорные пункты) или разбить задачу на части (по разделам). Можно явно попросить формат, который экономит выход (таблица из 5–7 пунктов вместо подробного комментария на каждую строчку). И отдельно — если ответ оборвался, можно продолжить: «Продолжи с пункта 4, сохраняя тот же формат», потому что модель не «помнит», что хотела написать дальше, она просто продолжает генерацию в новом лимите.

После этой главы стоит унести три вещи. Токен — это кусочек текста, и он не обязан совпадать со словом. Лимиты и стоимость обычно завязаны на токены, поэтому длинный контекст и подробный ответ конкурируют за одно и то же место. Если счёт кажется странным или ответ обрывается, чаще всего причина в токенизации и лимитах: сокращайте вход, упрощайте формат выхода или делите задачу на части.

Глава 5. Принцип «предсказания следующего токена»

Иногда кажется, что нейросеть «знает ответ»: она пишет связно, логично, с уверенными формулировками и без пауз. Но потом вы проверяете один факт — и он неверный. Или просите то же самое другими словами, и ответ меняется. У начинающего возникает вопрос: если текст выглядит таким цельным, почему в нём встречаются ошибки и откуда берётся эта уверенность?

Ключевой принцип простой: языковая модель не ищет истину, а каждый раз выбирает следующий токен по предыдущим. Токен — это кусочек текста, чаще всего слово или часть слова (иногда знак препинания). Модель смотрит на уже написанное и продолжает так, как «чаще всего бывает» в похожих текстах.

Чтобы понять механику, достаточно представить процесс как последовательность маленьких выборов. Вы пишете запрос, и он становится началом контекста. Дальше модель делает шаг: оценивает несколько возможных продолжений и выбирает одно. Это выбранное продолжение добавляется к тексту и становится частью «предыдущих токенов». Затем делается следующий шаг: снова выбор, снова добавление. Так текст растёт по цепочке.

На каждом шаге модель опирается на два источника: на ваш запрос и на уже сгенерированный ею текст. Поэтому ответы обычно получаются последовательными. Если модель начала объяснение в одном стиле и с одной логикой, ей проще продолжать в том же стиле и поддерживать уже сказанное, чем резко менять направление. Это похоже не на проверку фактов, а на аккуратное продолжение фразы: раз уж начато, нужно довести до конца связно.

Но из этого же следует ограничение: последовательность не равна точности. Модель выбирает продолжение, которое хорошо «подходит» к контексту по форме и по типичным связям слов, а не то, которое гарантированно верно в реальном мире. Если в запросе не хватает данных, если тема редкая, если нужны свежие сведения или точные числа, модель всё равно должна продолжать. Она не останавливается, чтобы сходить и проверить источники, если вы отдельно не дали ей такие источники или данные. В результате она может собрать правдоподобную версию ответа — связную, но ошибочную.

Этот принцип также объясняет уверенный тон. Модель обучена на огромном количестве текстов, где ответы часто звучат определённо: «это означает», «причина в том, что», «вот шаги». Такие формулировки сами по себе типичны для объясняющих текстов, поэтому они легко «побеждают» в выборе следующего токена. Уверенность здесь — часть стиля, а не показатель того, что модель действительно проверила информацию. Она может звучать одинаково уверенно и в правильном, и в неправильном ответе, потому что выбирает не «истинно/ложно», а «подходит/не подходит» к контексту и к привычному шаблону речи.

Представьте сценарий: вы просите: «Кратко объясни, что такое контекстное окно в чат-боте, и дай практический совет, как не потерять важные детали». Модель начинает: «Контекстное окно — это объём текста, который модель учитывает…» Это удачное начало: оно типичное и хорошо подходит к запросу. Дальше ей нужно продолжить в том же ключе и дать совет. Она добавляет: «Если диалог длинный, модель может забывать ранние сообщения…» — это тоже типичное продолжение. Затем она предлагает совет: «Периодически делайте резюме и вставляйте его в чат». Всё выглядит гладко и полезно.

Но теперь вы уточняете: «Сколько именно символов у контекстного окна в этой модели?» Если у модели нет точных данных, она всё равно будет продолжать. Она может написать конкретное число, потому что конкретные числа часто встречаются в технических объяснениях и делают текст убедительным. При этом число может оказаться выдуманным или устаревшим. С точки зрения механики это не «обман», а следствие процесса: модель выбирает следующий токен, который делает ответ похожим на нормальную справку, даже если у неё нет надёжной опоры для точной цифры.

Из этой главы стоит унести три вещи. Во-первых, модель отвечает шаг за шагом, выбирая следующий токен по предыдущим, поэтому контекст и формулировка запроса сильно влияют на продолжение. Во-вторых, связность текста — ожидаемый эффект такого механизма, но она не гарантирует точность фактов. В-третьих, уверенный тон и плавные формулировки — это часто просто «типичный стиль» продолжения, поэтому важные утверждения лучше отдельно просить обосновать данными, ограничениями или проверяемыми источниками.

Глава 6. Откуда берутся знания модели

Частая ситуация: вы спрашиваете у нейросети факт — например, «какие сейчас правила подачи на визу» или «кто победил на последней конференции», — и получаете уверенный ответ. Потом проверяете и видите, что часть деталей устарела или вообще не совпадает с реальностью. Возникает вопрос: откуда она это взяла, если «всё знает», и почему иногда попадает, а иногда нет.

Ключевой принцип такой: знания языковой модели — это не библиотека документов, а «статистическая память» о том, какие слова и мысли обычно встречаются вместе в текстах. «Статистическая память» здесь означает привычки, выученные на множестве примеров: модель не хранит страницы как в папке, а запоминает общие закономерности языка и содержания.

Эта память формируется во время обучения на больших массивах текстов. Модели показывают огромное количество фраз и учат продолжать их: какое слово, выражение или мысль чаще всего подходит дальше. Постепенно она подхватывает устойчивые связи: что обычно пишут после «причины инфляции», как выглядит структура резюме, какие аргументы часто приводят в теме «плюсы и минусы удалённой работы».

Важно, что это обучение не похоже на чтение с пометками «вот истинный факт». Модель просто настраивается так, чтобы хорошо угадывать продолжение текста. Из этого вырастает способность объяснять, пересказывать и обобщать.

Из-за этого модель не «знает» конкретный документ. Даже если в обучающих данных когда-то был нужный вам отчёт или статья, модель не открывает их по запросу и не цитирует как источник. Она воспроизводит общий рисунок: типичные формулировки, распространённые версии, стандартные шаги и выводы, которые часто встречались в похожих текстах.

Поэтому она может дать правдоподобный ответ, даже когда точных данных у неё нет. Она собирает фразу так, как это обычно пишут люди, а не так, как написано в конкретном файле.

Это объясняет два эффекта, которые новички часто путают.

Первый: модель хорошо справляется с «типовыми» задачами, где важны общие закономерности — написать письмо, предложить план, объяснить понятие, перечислить распространённые причины и последствия. Там как раз и помогают обобщения.

Второй: модель может ошибаться в деталях, которые завязаны на конкретный источник, дату, версию документа или редкий случай. Там требуется не «как обычно», а «как именно сейчас» или «как написано в этом документе».

Отсюда напрямую следует вопрос актуальности. У модели есть граница по времени: она обучена на текстах, которые были доступны до определённого момента, и сама по себе не «подтягивает» новости. Поэтому про новые события, свежие правила, недавние релизы и текущие цены она может говорить только в режиме предположений: по старым шаблонам и по тому, как обычно развиваются такие истории.

Иногда это выглядит убедительно, потому что формулировки гладкие. Но гладкость — не признак свежести. Чем ближе вопрос к «что изменилось на этой неделе» или «какая версия актуальна сегодня», тем выше риск устаревания или выдуманных деталей.

Представим сценарий. Вам нужно быстро подготовить сообщение команде: «Что известно про новый закон, который вчера обсуждали в новостях, и как он может повлиять на наш процесс?» Вы задаёте вопрос нейросети и получаете структурированный ответ: основные пункты закона, даты, штрафы, комментарии ведомств.

Текст выглядит аккуратно, но вы замечаете, что он слишком «универсальный»: много общих фраз и точные числа без ссылок. Если помнить про статистическую память, ход действий меняется. Вы переформулируете задачу так, чтобы модель работала в своей сильной зоне: просите не «сообщить факты», а помочь организовать проверку и подготовить черновик.

Например: «Составь список вопросов, которые нужно уточнить по закону (даты вступления, кого касается, исключения). Предложи шаблон сообщения команде: что мы знаем, что проверяем, какие действия возможны. Отдельно отметь, какие пункты требуют источника».

В ответ вы получаете структуру, которую можно быстро заполнить после проверки в официальной публикации или в надёжной новости. Модель не подменяет источник, но помогает не забыть важные части и быстро собрать понятный текст.

Запомнить стоит следующее:

- Модель обучается на больших массивах текстов и формирует «статистическую память»: умение продолжать и обобщать по закономерностям, а не доставать документы из архива.
- Она не «знает» конкретный документ по запросу, поэтому уверенный тон не гарантирует точность деталей, особенно дат, чисел и редких исключений.
- Для новых событий и актуальных правил используйте модель как помощника по структуре и вопросам, а факты подтверждайте внешним источником или данными, которые вы сами ей дали.

Глава 7. Ограничения памяти и времени модели

Иногда кажется, что модель «всё поняла»: вы подробно объяснили задачу, она ответила разумно, а через несколько дней в новом чате ведёт себя так, будто вас видит впервые. Или в длинном диалоге она внезапно путает детали: меняет даты, забывает ограничения, возвращается к старой версии требований. Ещё одна частая ситуация — вы просите: «подумай подольше и найди идеальное решение», а ответ всё равно получается поверхностным или обрывается на полпути, как будто у модели кончилось терпение.

Ключевой принцип простой: модель отвечает только на основе текста, который видит прямо сейчас, и делает это в ограниченное время и объёме. Она не хранит вашу личную историю «в голове» и не может бесконечно долго размышлять. У неё есть границы по тому, сколько текста она учитывает и сколько шагов может сделать, пока формирует ответ.

Отсюда следуют три важных следствия для работы.

Во-первых, почему модель не помнит прошлые сессии и не хранит личную историю. Обычный чат для модели — это отдельный эпизод: вы открыли диалог, написали сообщения, модель увидела их и ответила. Когда вы начинаете новый чат, у модели нет доступа к вашим прошлым перепискам, если только конкретный сервис специально не подставляет их в текущий разговор.

Даже если вам кажется, что «это один и тот же аккаунт», для модели это не означает автоматическую память о вас. Поэтому фразы вроде «ты же знаешь, чем я занимаюсь» или «как мы обсуждали вчера» не дают опоры: в новом чате этих данных просто нет. Если вам важно, чтобы модель учитывала контекст о вас или проекте, его нужно снова дать в тексте — коротко и по делу.

Во-вторых, как устроено «забывание» внутри одного диалога. У модели есть ограничение на объём текста, который она может удерживать в работе одновременно. Это часто называют контекстным окном: условная «рамка», внутри которой находятся последние сообщения и часть предыдущих.

Когда диалог разрастается, старые куски текста начинают не помещаться в эту рамку. Тогда сервис либо обрезает начало переписки, либо модель фактически перестаёт опираться на ранние детали, потому что они не попали в текущий вход. Снаружи это выглядит как забывание: вы когда-то указали «не упоминай цены» или «пиши на “вы”», модель соблюдала, а потом перестала. Причина обычно не в упрямстве и не в «плохом настроении», а в том, что важное условие оказалось слишком далеко в начале диалога и больше не учитывается.

Практический вывод: чем длиннее разговор, тем важнее держать ключевые требования ближе к текущему месту. Если условия критичны, их стоит повторять в сжатом виде, а не надеяться, что модель «помнит всё с самого начала».

В-третьих, как планировать работу, учитывая, что модель не может «думать долго» как человек. Модель генерирует ответ по шагам, и у неё есть ограничения: сколько текста она может выдать и сколько вычислений сервис готов потратить на один ответ. Даже если попросить «размышляй час», это не превратится в человеческое долгое обдумывание.

Модель либо даст стандартный по глубине результат, либо начнёт растекаться по общим словам, либо упрётся в лимит и прервётся. Поэтому «долго думать» лучше заменять на «работать по этапам». Вы не увеличиваете волшебным образом время мысли, вы структурируете задачу так, чтобы модель последовательно делала небольшие понятные куски работы, а вы проверяли и направляли.

Один сценарий, который показывает, как это применять. Допустим, вы готовите письмо клиенту с итогами встречи и договорённостями. Сначала вы в одном сообщении даёте вводные: кто клиент, цель письма, тон (деловой, без фамильярности), три ключевых договорённости, что нельзя обещать и желаемый объём. Модель пишет черновик — всё выглядит хорошо.

Потом вы начинаете уточнять: «добавь абзац про сроки», «перепиши вступление мягче», «вставь список рисков», «сделай две версии — короткую и длинную», и диалог разрастается. На каком-то шаге модель вдруг добавляет обещание по срокам, которое вы просили не давать, и меняет тон на слишком дружеский.

Вместо того чтобы спорить с моделью, вы делаете следующее:

1) Останавливаете диалог и пишете короткий «якорь» — сжатый список неизменных правил: тон, запреты, факты, формат. Это 5–8 строк, без лишних объяснений.

2) Просите модель сначала повторить эти правила своими словами в 3–5 пунктах, чтобы убедиться, что они попали в текущий контекст.

3) Дальше даёте одну задачу за раз: «перепиши письмо, соблюдая якорь; не добавляй новых фактов; сделай 120–150 слов».

4) Если нужно «глубже», вы не просите «думай дольше», а делите работу: отдельно — список рисков, отдельно — формулировки про сроки в безопасном виде, отдельно — финальная сборка письма. Так вы укладываете рассуждение в управляемые шаги и снижаете шанс, что модель начнёт выдумывать или потеряет ограничения.

Что стоит унести и попробовать после прочтения:

- Если контекст важен, не рассчитывайте на память между сессиями: нужные факты и правила нужно заново дать в тексте.
- В длинных диалогах держите ключевые условия «рядом»: периодически делайте короткий якорь с неизменными требованиями.
- Вместо просьбы «думай долго» планируйте работу этапами: маленькие задачи, проверка, затем сборка результата.

Глава 8. Роль случайности в ответах

Иногда это выглядит странно: вы задаёте один и тот же вопрос два раза — и получаете два разных ответа. В первый раз нейросеть пишет коротко и по делу, во второй — добавляет лишние детали, меняет порядок шагов или предлагает другие идеи. Возникает ощущение, что она «передумала» или «капризничает». На практике причина обычно проще: в генерации ответа есть доля случайности.

Ключевой принцип такой: языковая модель не выбирает единственно правильную фразу. Она каждый раз собирает ответ как последовательность наиболее вероятных вариантов. В этот выбор часто добавляется случайность, чтобы текст не был одинаковым и «застывшим». Случайность здесь — это не хаос, а регулируемая примесь, которая позволяет модели иногда выбирать не самый очевидный следующий кусочек текста.

Чтобы понять механику, полезно представить процесс по шагам. Модель пишет не «сразу весь ответ», а по частям: слово за словом (точнее, кусочками слов). На каждом шаге она оценивает несколько возможных продолжений и назначает им вероятности: какие варианты сейчас подходят лучше, а какие хуже. Дальше нужно выбрать один вариант и перейти к следующему шагу — и так до конца.

Если бы модель всегда брала самый вероятный вариант, ответы были бы очень стабильными, но часто однообразными. Формулировки повторялись бы, идеи были бы предсказуемыми, а иногда текст становился бы «заезженным»: одинаковые вступления, одинаковые выводы, одинаковые списки. Поэтому во многих режимах работы модель выбирает не строго «самое вероятное», а из нескольких вероятных — и тут появляется случайность.

Эта случайность напрямую влияет на разнообразие. Когда она выше, модель чаще «разрешает себе» взять вариант, который чуть менее вероятен, но всё ещё подходит по смыслу. В результате меняются формулировки: другие слова, другой тон, другой порядок. А иногда и сами идеи: другие примеры, другие варианты решения, другие пункты списка. Это особенно заметно в задачах, где нет одного правильного ответа: придумать темы для поста, варианты заголовков, способы сформулировать просьбу, идеи подарков, черновик письма.

Отсюда и ответ на вопрос, почему один и тот же запрос может давать разные ответы. Во-первых, при каждом запуске модель может сделать другие выборы на нескольких шагах подряд, и маленькое отличие в начале «разветвляет» весь текст дальше. Во-вторых, в запросе часто есть пространство для интерпретации: вы не указали длину, формат, аудиторию, критерии качества — и модель каждый раз может «додумать» это по-разному. В-третьих, влияет контекст диалога: даже если вам кажется, что запрос тот же, модель опирается на предыдущие сообщения, а значит, стартовые условия уже не идентичны.

Эта же логика объясняет, почему иногда ответы отличаются не только стилем, но и содержанием. Если вы спрашиваете: «Составь план статьи про здоровое питание», то возможны десятки нормальных планов. Модель не «вспоминает один правильный», она каждый раз строит один из допустимых вариантов. А когда вы спрашиваете факты (например, «какая дата…»), случайность может привести к более рискованным догадкам, если модель не уверена или если в запросе не хватает уточнений. Снаружи это выглядит как нестабильность, но внутри это та же механика выбора из нескольких продолжений.

Дальше важно понять, когда вам нужна стабильность, а когда — разнообразие. Стабильность полезна, когда вы хотите повторяемый результат и минимальный риск «уехать» в сторону: итоговое письмо клиенту, краткая инструкция для коллег, сводка по встрече, формулировка требований, ответы, где важна точность и одинаковый формат. В таких задачах вы обычно не выигрываете от неожиданностей: вам нужно, чтобы модель держалась рамок, не добавляла лишнего и не меняла структуру.

Разнообразие полезно, когда вы ищете варианты и сравниваете: идеи, черновики, альтернативные формулировки, разные углы зрения, несколько сценариев. Здесь как раз ценны небольшие отклонения: они помогают увидеть то, что вы сами не сформулировали с первого раза. В таких задачах «разные ответы» — это не проблема, а инструмент.

Представим сценарий. Вы хотите написать короткое описание услуги для сайта и не знаете, как лучше сформулировать. Вы пишете запрос: «Сделай описание услуги “настройка рекламы” на 3–4 предложения». В первый раз модель выдаёт сухой текст, во второй — добавляет обещания, в третий — уходит в технические детали. Вы решаете управлять тем, что вам важнее: стабильность формата или разнообразие вариантов.

Действия могут быть такими.

1) Сначала вы выбираете режим «разнообразия», потому что вам нужны варианты. Вы уточняете запрос: «Дай 5 разных вариантов описания услуги, каждый по 3–4 предложения. Варианты должны отличаться тоном: нейтральный, дружелюбный, деловой, короткий, более подробный. Без обещаний “гарантируем рост”, без профессионального жаргона». Теперь случайность работает на вас: модель может предложить разные формулировки, но вы заранее ограничили опасные зоны.

2) Затем вы переходите в режим «стабильности», потому что вам нужно выбрать один вариант и довести его до финала. Вы берёте понравившийся текст и просите: «Перепиши этот вариант, сохрани смысл и структуру, сделай на 10% короче, оставь нейтральный тон, не добавляй новых фактов». Здесь вы уменьшаете пространство для интерпретации, и даже если формулировки чуть изменятся, результат будет ближе к повторяемому.

В этом сценарии вы не боретесь со случайностью «в лоб», а используете её по назначению: сначала как генератор вариантов, потом как инструмент аккуратной правки в заданных рамках.

Запомнить стоит три вещи. Во-первых, разные ответы на один запрос чаще всего нормальны: модель выбирает из нескольких подходящих продолжений, и небольшая случайность разветвляет текст. Во-вторых, случайность увеличивает разнообразие формулировок и идей — это полезно в творческих и поисковых задачах. В-третьих, когда нужна стабильность, уменьшайте неопределённость в запросе: фиксируйте формат, ограничения и критерии, а разнообразие включайте осознанно — когда вы действительно ищете варианты.

Глава 9. Настройки генерации: температура и топ-p

Иногда вы задаёте один и тот же вопрос нейросети два раза и получаете разные ответы. В одном случае текст получается аккуратным и «по делу», в другом — более смелым, с необычными формулировками и лишними допущениями. Возникает ощущение, что модель «в настроении» или что-то «решила по-своему». На практике чаще всего дело не в теме вопроса, а в настройках генерации — в том, насколько модели разрешают выбирать редкие варианты продолжения.

Ключевой принцип простой: температура и топ-p управляют степенью случайности в выборе следующего фрагмента текста. Поэтому они меняют предсказуемость и «креативность» ответа. Под «креативностью» здесь не имеется в виду талант — это всего лишь готовность модели уходить от самых очевидных формулировок и выбирать менее вероятные продолжения.

Чтобы понять механику, полезно помнить базовую вещь: языковая модель пишет текст по шагам, каждый раз выбирая следующий маленький кусочек (часть слова или слово). У неё есть несколько подходящих вариантов, и у каждого — своя вероятность. Настройки генерации влияют на то, какие варианты вообще допускаются и насколько охотно модель берёт не самые вероятные.

Температура — это «ручка», которая делает выбор более строгим или более свободным. При низкой температуре модель чаще берёт самый вероятный вариант и старается идти по проторенной дорожке. Ответы обычно получаются ровнее, ближе к шаблонным, с меньшим количеством неожиданных поворотов. При высокой температуре модель чаще выбирает не самый вероятный вариант. Тогда текст становится разнообразнее по словам и идеям, но растёт риск странных деталей, лишних допущений и ошибок, потому что модель чаще «сворачивает» туда, где у неё меньше опоры на типичные продолжения.

Топ-p работает иначе: он не «усиливает» или «ослабляет» все варианты сразу, а ограничивает круг выбора. Представьте, что модель на каждом шаге составляет список самых вероятных продолжений и берёт из него только верхнюю часть — до тех пор, пока суммарная вероятность этих вариантов не наберёт заданную долю p. Это и есть top-p (его ещё называют nucleus sampling).

Если top-p низкий, в списке остаются только самые вероятные продолжения — стиль будет более осторожным и однообразным. Если top-p высокий, список шире, туда попадают более редкие варианты — стиль становится разнообразнее, но может стать менее стабильным.

Важно, что температура и топ-p решают похожую задачу разными способами и могут усиливать друг друга. Низкая температура плюс низкий top-p обычно дают максимально предсказуемый текст: меньше сюрпризов, но и меньше свежих формулировок. Высокая температура плюс высокий top-p дают больше разнообразия, но и больше «разброса» по качеству. Если вы не можете управлять обеими настройками, достаточно понимать общий смысл: вы либо сужаете выбор модели, либо расширяете.

Представим сценарий. Вам нужно подготовить короткое письмо клиенту: вежливо напомнить про оплату и предложить два варианта даты созвона. Вы делаете первый запрос и получаете нормальный текст. Потом просите «переформулировать иначе» — и вдруг нейро-

сеть добавляет лишние обещания («мы гарантируем скидку»), меняет тон на слишком фамильярный или придумывает детали, которых вы не говорили.

Если в этот момент снизить креативность (уменьшить температуру и/или top-p), модель будет реже добавлять неожиданные элементы и чаще держаться исходных фактов и нейтрального стиля. А вот если задача другая — например, придумать 15 тем для постов или 10 вариантов заголовка, — вы, наоборот, выигрываете от повышенной креативности: пусть варианты будут менее «вылизанными», зато более разнообразными, и вы сможете выбрать лучшее.

Запомнить стоит следующее:

- Температура отвечает за то, насколько модель склонна отходить от самых вероятных продолжений: ниже — предсказуемее, выше — разнообразнее.
- Топ-p ограничивает набор вариантов, из которых вообще выбирается продолжение: ниже — уже и осторожнее, выше — шире и смелее.
- Снижайте креативность для задач, где важны точность, единый стиль и отсутствие выдумок (письма, резюме фактов, инструкции). Повышайте — для задач на варианты и идеи (заголовки, концепции, черновые наброски), где вы всё равно будете отбирать и редактировать.

Глава 10. Что такое контекстное окно

Почти у каждого, кто долго переписывается с ИИ, случается одинаковая ситуация. В начале вы подробно описали задачу, договорились о формате и ограничениях, а через 20–30 сообщений модель вдруг «забывает» важную деталь. Например, начинает писать не тем тоном, путает исходные данные или снова задаёт вопрос, на который уже отвечали. Со стороны это выглядит как невнимательность, хотя причина обычно техническая и довольно простая.

Ключевой принцип такой: у модели есть ограниченный «буфер текста», который она учитывает прямо сейчас. Всё, что не помещается в этот буфер, перестаёт влиять на ответ. Этот буфер называют контекстным окном. Контекстное окно — это максимальный объём текста, который модель может держать «перед глазами» при генерации следующего ответа.

Важно правильно понимать, что именно считается объёмом контекста. В контекст входит не только ваш последний запрос. Туда попадает сразу три части: ваша текущая реплика, предыдущая история диалога (включая ваши сообщения и ответы модели) и тот ответ, который модель прямо сейчас генерирует. То есть место в контекстном окне «занимает» всё: и постановка задачи, и уточнения, и примеры, и промежуточные версии текста, и длинные ответы самой модели.

Из-за этого длинные диалоги со временем начинают «съедать» контекст. Модель не хранит бесконечную переписку как человек, который может мысленно вернуться к началу разговора. Когда общий объём текста становится больше, чем позволяет контекстное окно, системе приходится обрезать часть истории. Обычно «выпадает» самое старое — начало диалога. В результате модель продолжает отвечать, опираясь на то, что осталось внутри окна. А то, что оказалось за его пределами, для неё как будто не было сказано.

Это и создаёт эффект: вы уверены, что правило было задано («пиши вежливо и коротко», «не используй англицизмы», «опирайся только на эти цифры»), но модель его нарушает. Не потому что «решила иначе», а потому что в момент ответа она буквально не видит этот кусок текста. Иногда остаются обрывки: модель помнит часть требований, но теряет детали. Тогда ответы становятся нестабильными: в одном сообщении она соблюдает условие, в следующем — нет, потому что условие уже «уехало» за пределы окна.

Практический вопрос для пользователя — как прикинуть, поместится ли задача в контекстное окно выбранной модели. Точного подсчёта «на глаз» обычно не получится, потому что модели считают объём не в словах и не в символах, а в более мелких единицах текста. Но оценивать всё равно можно, если мыслить не числами, а объёмами и рисками.

Полезная логика такая:

1) Сначала мысленно соберите «пакет текста», который модели нужно учитывать одновременно: исходные требования, данные, примеры, ограничения, плюс текущий запрос. Если вы хотите, чтобы модель строго следовала правилам, эти правила должны быть внутри окна именно в тот момент, когда она пишет ответ.

2) Добавьте к этому предполагаемый размер ответа. Если вы просите «подробно», «с примерами», «на 2 страницы», вы заранее занимаете значительную часть окна будущим ответом — и тем самым уменьшаете место для истории и исходных условий.

3) Оцените длину истории диалога. Если вы уже много раз уточняли одно и то же, вставляли черновики, просили переписать текст, то вы, возможно, уже близко к пределу. В такой ситуации даже правильный запрос может не сработать, потому что важные куски оказались в начале и уже не помещаются.

Один рабочий сценарий. Допустим, вы ведёте диалог, чтобы подготовить резюме. В начале вы дали вводные: должность, опыт, список проектов, ограничения («без личных данных», «до одной страницы», «деловой тон»). Дальше вы 15 раз просили переписать пункты, добавляли новые версии, обсуждали формулировки и вставляли целые абзацы текста.

В какой-то момент вы пишете: «Собери финальную версию и обязательно оставь только одну страницу». Модель выдаёт текст на две страницы и добавляет детали, которые вы просили не упоминать.

Если рассуждать через контекстное окно, картина такая: длинная история с черновиками и правками заняла почти всё доступное место. Начальные правила («до одной страницы», «без личных данных») оказались в самом старом фрагменте и «выпали». Внутри окна остались последние обсуждения формулировок и ваш финальный запрос, но без исходных ограничений. Поэтому модель честно делает то, что видит: собирает «финальную версию», но не может опереться на правила, которые больше не присутствуют в её текущем контексте.

Что можно унести из этой главы:

- Контекстное окно — это лимит текста, который модель учитывает сейчас; в него входит запрос, история диалога и генерируемый ответ.
- В длинных переписках начало часто перестаёт влиять на ответы, потому что выходит за пределы окна, и это выглядит как «забывание».
- Оценивайте вместимость задачи так: что должно быть перед глазами модели одновременно + какой длины вы просите ответ + насколько разрослась история; если важное было в начале, есть риск, что оно уже не учитывается.

Глава 11. Почему модель «забывает» сказанное ранее

Частая ситуация: вы долго обсуждаете задачу с нейросетью, постепенно уточняете детали, договариваетесь о стиле и ограничениях. А потом в какой-то момент модель внезапно предлагает решение, будто этих договорённостей не было: меняет формат, забывает важное условие, снова задаёт вопрос, на который уже отвечали. Возникает ощущение, что она «не слушает» или «не помнит», хотя раньше всё шло нормально.

Ключевой принцип простой: модель видит не весь диалог, а только его последнюю часть и отвечает, опираясь на то, что поместилось в её «контекстное окно». Контекстное окно — это ограниченный объём текста, который модель учитывает прямо сейчас. Когда диалог становится длинным, самые ранние сообщения перестают помещаться и фактически перестают влиять на ответ.

Это «забывание» происходит не потому, что модель решила игнорировать вас. Оно связано с обрезкой старых сообщений: системе нужно уместить входной текст в ограничение по длине, и при переполнении из рассмотрения выпадают первые куски переписки. В результате качество ответа меняется предсказуемо: модель начинает опираться на более свежие реплики, а не на исходные требования.

Конец ознакомительного фрагмента.