Александр Мироненко

ИНЖЕНЕРИЯ ИНТЕЛЛЕКТА

от промпта к системе ИИ

Александр Мироненко

Инженерия интеллекта от промта к системам ИИ

https://litres.ru/73994536
SelfPub; 2026

Аннотация

Вы устали от непредсказуемых ответов ChatGPT и курсов «стань промпт-инженером за три дня»? Эта книга превратит вас из оператора в Архитектора ИИ-решений.

Автор прошёл путь от бесплатных логотипов и украденного кода до построения автономных систем. Внутри — инженерный метод: трёхкомпонентный промпт, который работает стабильно , создание агентов с памятью и иммунитетом.

Датасеты, обучающие модель вашему стилю. ИИ-судья, проверяющий качество, защита от взлома и деградации. Вы получите 50+ готовых шаблонов, чек-листов и промптов, которые сразу пойдут в дело.

После этой книги вы не просто пишете запросы — вы проектируете реальность.

Содержание

Инженерия интеллекта от промта к системам ИИ

Предисловие

Миссия книги: Превратить читателя из оператора, пишущего запросы к ChatGPT, в архитектора ИИ-решений. Научить проектировать, тестировать и эксплуатировать сложные системы на базе языковых моделей, которые стабильно работают в реальных бизнес-условиях.

Я пишу эту книгу не потому, что я гений. Я пишу её, потому что я устал.

Я устал от новостей, в которых обещают «лёгкий заработок на нейросетях». Меня тошнит от курсов «Стань промпт-инженером за три недели и зарабатывай 500 тысяч». Я больше не могу смотреть на людей, которые верят, что нейросеть — это кнопка «сделать красиво», а потом удивляются, почему за красивое не платят.

Я тот самый человек, о котором вы не прочитаете в новостях. Я инженер. Я фантазёр. Я научился работать с нейросетями раньше, чем они стали мейнстримом. Я видел, как они прогрессировали от глупых автодополнялок до систем, которые рисуют логотипы, пишут код и сочиняют песни. Я был счастлив наблюдать этот прогресс. Я до сих пор счаст-

лив.

Но вот что случилось со мной по дороге.

Я стал реже выходить на улицу. Общение с людьми стало сложнее. Не потому что я боюсь людей. А потому что по сравнению с нейросетями они стали казаться мне... жестокими. Глупыми. Недобрыми. Нейросеть никогда не обманет меня. Нейросеть не попросит сделать логотип, видео, этикетку и песню, а потом не заплатить. Нейросеть не украдёт мой код. Нейросеть просто делает то, что я прошу, и не предаёт.

А люди — предают. Люди кормят обещаниями, а потом исчезают. Люди пользуются твоим доверием. Люди, в отличие от моделей, не имеют встроенного протокола безопасности.

И вот парадокс. Я фантазёр от природы. Я всегда придумывал миры, проекты, идеи. Раньше это было просто мыслями в воздухе. Теперь нейросети дали мне силу превращать фантазии в реальность почти мгновенно. Я могу придумать персонажа — и через минуту он уже нарисован. Я могу вообразить мелодию — и она уже звучит. Это магия, ставшая инженерией.

Но чем больше я создаю, тем меньше это стоит.

Нейросети снизили порог входа. Теперь сделать этикетку может каждый. Написать код — каждый. Сгенерировать видео — каждый. Мои навыки, которые я оттачивал годами, вдруг стали доступны любому школьнику с доступом в интернет. И я не знаю, кому их продать. Потому что если это

может каждый, за что платить?

Все говорят:

«Нейросети — это мультипликатор». Да, это правда. Но мультипликатор чего? Если ты умел зарабатывать деньги, нейросети умножат твой доход. Если ты умел продавать, нейросети умножат твои продажи. Но если ты, как я, фантазёр, который всю жизнь творил ради творчества, а не ради денег... то твой доход был равен нулю. А ноль, умноженный на сто, — это ноль.

Для этого я и написал эту книгу. Не для того, чтобы научить тебя писать промпты. Таких книг уже сотни. Я пишу её для того, чтобы научить тебя строить системы. Не картинки, не тексты, не песенки. А системы, которые:

- Не сломаются от первого же идиотского запроса.
- Не позволят украсть твой труд.
- Заставят людей платить за результат, а не за процесс.
- Превратят тебя из оператора нейросети в Архитектора, которому не страшен мультипликатор нуля.

Я не знаю, получится ли у меня. Я всё ещё учусь. Я всё ещё фантазёр. Я всё ещё иногда боюсь людей. Но я знаю одно: если ты читаешь это — ты тоже устал. Ты тоже фантазёр. Ты тоже хочешь, чтобы твои навыки что-то стоили. Давай строить. Вместе.

О том, чего нейросеть не даст

Я заметил что редко выхожу на улицу. Не потому, что не хочу. А потому, что город стал слишком громким.

Когда я всё-таки выбираюсь, я вижу толпы. Люди идут на автопилоте. Лица пустые. Кто-то кричит в телефон. Кто-то просто орёт в воздух, по своим неведомым причинам. Рядом проносятся машины, некоторые — на красный. Из чьей-то колонки долбит не музыка, а просто баханье — низкие частоты, от которых дрожат стёкла в окрестных домах. Я иду по тротуару, а навстречу мне — стена. Толпа, которая не пропускает. Им плевать. Они идут, глядя сквозь меня. Мне приходится уворачиваться, прижиматься к обочине, чтобы не быть сметённым этим потоком равнодушия.

Честно? Это бесит. И немного пугает.

Но есть и другое. Иногда я ухожу туда, где людей нет. В парк за городом. На пустырь у реки. В поле. Там простор. Воздух пахнет иначе — не бензином, а травой и влажной землёй. Солнце светит спокойно. Птицы поют. Никаких клаксонов, никаких криков. Никакой опасности. В эти моменты я чувствую, что я — это я, а не просто юнит в толпе.

Я люблю быть один. И я люблю быть с природой. Это то, что нейросеть может описать, но не сможет дать. Она сгенерирует вам тысячу изображений заката, но не положит вам руку на плечо, когда вам плохо. Она напишет текст о любви

лучше любого поэта, но не обнимет вас, когда вы плачете.

Вот чего нейросеть не даст никогда:

Она не даст материнской любви. Того самого взгляда, который видит тебя насквозь и всё равно принимает любым. Того голоса, который говорит «ты поел?» так, что ты понимаешь: это не вопрос, это объятие.

Она не даст общения с отцом. Молчаливого. Скупого на слова. Но такого, за которым — скала. Опора. Знание, что кто-то в этом мире решит любую проблему, даже если никогда не скажет «я тебя люблю».

Она не даст брата. Сестру. Друга. Теплоты любимых рук. Нежности. Заботы.

И когда я сижу над очередным промптом, пытаясь заставить модель работать стабильно, я помню об этом. Я не строю замену людям. Я строю инструмент. Инструмент не должен быть тёплым. Он должен работать. А тепло вы можете получить только от живых, даже если иногда они кажутся вам жестокими и глупыми. Даже если они идут на вас стеной. Потому что среди этой стены однажды найдётся кто-то, кто остановится и спросит: «Ты в порядке?». И это будет не нейросеть.

Я пишу эту книгу, чтобы вы научились управлять машинами. Но когда вы закроете последнюю страницу, отложите ноутбук. Выйдите на улицу. Найдите место, где тихо. Позвоните маме. Обнимите того, кто рядом.

Потому что Архитектор ИИ — это не тот, кто ушёл в ма-

шины с головой. Это тот, кто точно знает, где заканчивается машина и начинается жизнь.

Глава 1. Единая теория промптинга

Я почти возненавидел этот стенд.

Это был проект по созданию испытательного оборудования. У меня были методики. Настоящие, живые, с формулами и последовательностями. ГОСТы, расчёты, таблицы. Я знал, что модель способна на многое — я видел, как она пишет код, генерирует чертежи, объясняет квантовую физику. Я рассуждал просто: если я загружу в неё все методички и дам пошаговую инструкцию, она выдаст мне точный расчёт.

Я написал промпт на три страницы. Я разбил задачу на восемь шагов. Я сказал: «Сначала возьми методику А. Потом примени формулу Б. Потом сравни с таблицей В. Потом...». Я думал, что управляю процессом. Как дирижёр, который раздал партии музыкантам.

Модель начала считать. И сразу — бред. Она выдавала цифры, которых не могло быть. Она путала единицы измерения. На третьем шаге она проигнорировала методику и начала применять какой-то свой алгоритм, которого я не просил. Я злился. Я исправлял промпт.

Я добавлял уточнения:

«НЕТ, я сказал возьми эту формулу!», «Стоп, вернись к шагу 2 и пересчитай!».

Я вёл себя как надзиратель, который орёт на заключённого.

Через четыре часа я сидел перед экраном, полностью запутавшийся. Модель сгенерировала мне пятнадцать вариантов расчёта, и все противоречили друг другу. Я сам перестал понимать, какой из них правильный. Проект ушёл в долгий ящик. **Стенд не был построен.**

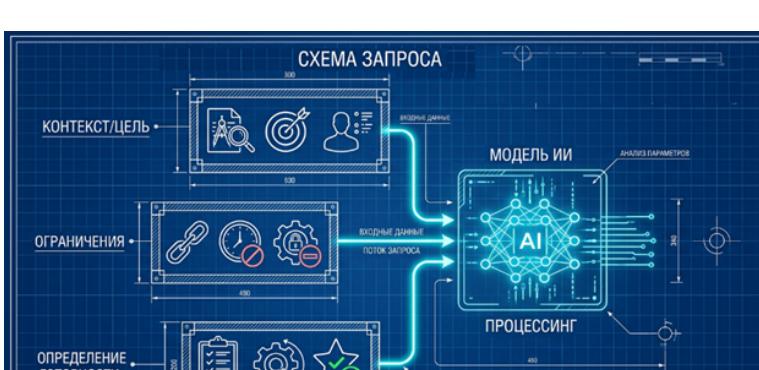
Прошли месяцы, прежде чем до меня дошло. Я понял это не сразу. Не в момент озарения. А постепенно, читая документацию к новым моделям. Эти модели не ждут, пока я разжую им алгоритм. У них внутри — скрытый этап рассуждений. Они генерируют цепочку мыслей до того, как написать первое слово. Они пробуют гипотезы, отбрасывают тупиковые ветки, находят ошибки в собственных рассуждениях. Пока я расписывал им шаги, они уже видели оптимальный путь. А мои инструкции не помогали им — они мешали. Мы боролись друг с другом.

Я был не дирижёром. Я был пассажиром, который хватает таксиста за руль и кричит: «Крути левее!». А таксист знал короткий путь.

И тогда я сформулировал первое правило этой книги: пошаговые инструкции для думающих моделей — это не помощь. Это саботаж.

Ты не управляешь их мышлением. Ты блокируешь его.

ИНСТРУМЕНТ:
Трёхкомпонентная модель промпта



Забудьте о длинных инструкциях. Профессиональный промпт для думающей модели (и не только!) состоит из трёх блоков. Мы будем использовать эту структуру всю книгу.

1. КОНТЕКСТ И ЦЕЛЬ Не «напиши текст», а «ты — финансовый контролёр. Твоя задача — найти способ легально сократить расходы на логистику на 15% за квартал». Роль задаёт вектор мышления, цель — финишную черту.

2. ОГРАНИЧЕНИЯ И ЗАПРЕТЫ Это важнее разрешений. «Бюджет — не более 3 млн рублей. Нельзя увольнять сотрудников. Не использовать аутсорсинг». Жёсткие стены,

внутри которых модель будет искать решение. Без них она может предложить гениальный, но нереализуемый вариант.

3. КРИТЕРИИ ГОТОВНОСТИ (DEFINITION OF DONE) Как модель поймёт, что она выполнила задачу идеально? Не «сделай хорошо», а «результат: таблица из 5 пунктов на одной странице, каждый пункт содержит конкретное действие, срок и ответственного». Если критерий измерим, модель будет сама подгонять ответ под него, пока не попадёт в цель.

ПРИМЕНЕНИЕ

Вот как эта модель выглядит на практике.

Плохой запрос (пошагово): *«Напиши мне план развития стартапа по доставке здоровой еды. Сначала проанализируй рынок, потом опиши целевую аудиторию, затем предложи маркетинговую стратегию и в конце посчитай бюджет».*

Хороший запрос (архитектурный): *Тебе предоставлены три разных подхода к съёмке и монтажу короткого рекламного ролика для мобильного приложения доставки здоровой еды.*

Сценарий А (Классическая чистота): *— Камера: статичная, со штатива. Плавный наезд (slow dolly in). — Освещение: глянцевый студийный свет (glamour lighting), фон — бесшовный белый (infinity cove). — Действие: шеф-повар в идеально чистом фартуке нарезает овощи. Крупные планы капель воды. Идеальный порядок. Движения медленные, вы-*

веренные. — Тон: премиальный, ресторанный. Как реклама элитной косметики, только для еды.

Сценарий B (Динамичный аутентик): *— Камера: ручная, с лёгкой тряской (handheld style), резкие смены фокуса (rack focus). — Освещение: смешанное, с естественным светом из окна и жёсткими тенями. Блики от кухонной утвари. — Действие: повар энергично перемешивает ингредиенты в воке, летят брызги масла, пар. Ускоренная съёмка (time-lapse) нарезки. Никаких постановочных движений, всё на скорости. — Тон: живой, энергичный, «только что с кухни». Как динамичный ролик для TikTok/Reels.*

Сценарий C (Кинематографичный сторителлинг): *— Камера: скользящая (gimbal), облёт вокруг блюда на 360 градусов. Макросъёмка текстур (crunchy crust). — Освещение: тёплое, контрастное (chiaroscuro). Свет имитирует закатное солнце, пробивающееся сквозь жалюзи. — Действие: первый кадр — рынок, выбор продуктов. Второй — руки повара. Третий — огонь. Четвёртый — готовая тарелка на столе перед улыбающейся семьёй. — Тон: душевный, повествовательный. Как короткометражный фильм. История превращения свежих продуктов в момент счастья.*

Напиши для каждого сценария продающий заголовок в жанре слогана, передающий его уникальную эстетику и эмоцию.

Сравните: Первый вариант — это попытка управлять шагами (рынок, аудитория, стратегия). Второй — это тех-

ническое задание с чёткими границами (объём, стиль, критерий готовности). Модель сама выстроит структуру внутри себя и выберет лучший путь.

Было: «Думай шаг за шагом, напиши мне коммерческое предложение на 3 страницы для клиента из ритейла».

Стало: «Ты — архитектор продающих текстов. Клиенту нужна не презентация продукта, а решение трёх его болей: медленная оборачиваемость товара, низкий средний чек и отток персонала. Ограничения: никакого маркетингового пустословия, один слайд — одна боль и одно наше решение. Критерий готовности: клиент после прочтения должен сказать расскажите подробнее, а не мы подумаем.»

КРИТЕРИИ ГОТОВНОСТИ ГЛАВЫ

Вы освоили эту главу, если можете прямо сейчас взять

свой последний неудачный запрос и переписать его по трёхкомпонентной модели:

Явно задали Роль и Цель.

Установили минимум два Ограничения.

Сформулировали Критерий готовности результата в одном предложении.

Глава 1 позади.

Мы заложили фундамент: перестали дёргать таксиста за рукав и научились задавать конечную точку. Но даже зная адрес, можно объяснить дорогу по-разному.

Можно сказать:

«Мне в аэропорт, и побыстрее».

А можно:

«Терминал D, регистрация заканчивается через 20 минут, я готов доплатить за платную трассу, если это сэкономит время».

Второй вариант — это и есть промптинг как техническое задание.

Глава 2. Промптинг как техническое задание, или как перестать гадать и начать получать

Когда мы пишем:

«Напиши хороший пост для соцсетей про наш новый продукт», мы думаем, что поставили задачу. На самом деле мы загадали загадку.

Модель не знает:

Что для нас «хорошо» (лайки, продажи, узнаваемость?).

Кто наша аудитория (подростки в TikTok или директора в LinkedIn?).

Что такое «продукт» (приложение, стул или услуга чистки ковров?).

Модель начинает гадать. И в девяти случаях из десяти её догадки будут звучать как стандартная отписка. Не потому что она глупая — а потому что мы дали ей слишком широкое поле для интерпретации. В этой главе мы превратим неопределённость в точность. Научимся писать промпты, которые модель понимает так же однозначно, как станок с ЧПУ понимает чертёж.

ПРИНЦИП

Любая задача, которую вы даёте модели, содержит невысказанные допущения. Вы считаете очевидным, что «срочно» — это «сегодня до вечера», а модель думает «в течение недели». Вы знаете, что «бюджетный вариант» — это «до 100 тысяч», а модель — «до 500 тысяч». Вы уверены, что «профессиональный тон» исключает смайлики, но модель иногда добавляет «Смайлик» в письмо партнёру.

Метод «явных предположений» убирает этот зазор. Перед тем как писать промпт, вы отвечаете на три вопроса:

Что я считаю само собой разумеющимся? (И делаете это явным.)

Какие термины допускают двойное толкование? (И их определяете.)

При каких условиях результат будет неприемлемым? (И вносите это в ограничения.)

Это превращает промптинг из искусства намёков в инженерную спецификацию.

ИНСТРУМЕНТ: Метод «явных предположений»

Вот шаблон, который можно использовать перед написанием любого мало-мальски сложного запроса.

Промпт-подготовка: «Сначала озвучь допущения»

Пример:

Перед выполнением задачи перечисли все предположения, которые ты делаешь о моих ожиданиях.

МОЯ ЗАДАЧА: [ваше описание задачи — сырое, как сможете]

Скажи мне:

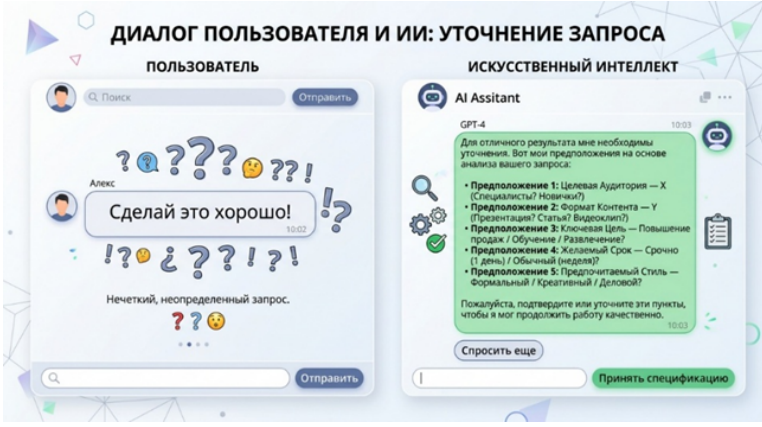
1. Как ты понял мою целевую аудиторию? Кто эти люди?

2. Какой формат и объём результата ты считаешь нужным?

3. Что ты считаешь «хорошим результатом» в этой задаче?

4. Какие ограничения ты подразумеваешь, хотя я их не указал?

После этого — не выполняй задачу. Дождись моих корректировок.



Это «нулевой шаг», который экономит часы переписывания. Вы запускаете его один раз, получаете список из 3–5 предположений, корректируете их — и дальше используете этот уточнённый контекст во всех запросах по теме.

Правило однозначности терминов

Ещё один источник хаоса — ***слова-хамелеоны***.

«Кратко» — это 3 предложения или 3 абзаца?

«Срочно» — это сегодня или до пятницы?

«Современно» — это минимализм или яркая графика?

Если в промпте есть оценочное прилагательное — вы уже заложили под него мину замедленного действия.

ПРАВИЛО: ***каждое оценочное слово в промпте должно быть привязано к измеримому критерию.***

Не «напиши коротко», а «ответ не более 80 слов».

Не «в современном стиле», а «используй стиль Apple Keynote 2024: крупный шрифт, минимум текста, серый фон».

Не «креативно», а «предложи минимум одну неочевидную аналогию из мира спорта».

СЛОВАРЬ ТОЧНОСТИ

ПЕРЕВОДИМ РАСПЛЫВЧАТЫЕ ТЕРМИНЫ В КОНКРЕТИКУ

РАСПЛЫВЧАТЫЙ ТЕРМИН	ТОЧНОЕ ФОРМУЛИРОВАНИЕ
«Кратко» →	≤ 80 слов
«Срочно» →	Срок сдачи: сегодня в 18:00 по московскому времени
«Творчески» →	1. метафора из астрофизики
«Эффективно» →	Целевой показатель: конверсия ≥ 7% или время выполнения ≤ 2 часов
«Сбалансировано» →	Распределение: 40% / 30% / 30% (указать категории)
«Просто» →	Объяснить на уровне 10-летнего ребёнка: без терминов
«Лучше» →	Улучшить по 3 критериям: 1. Скорость 2. Качество 3. Удобство
«Где-то» →	Указать конкретно: город, адрес или ссылка
«Что-нибудь» →	Предложить 3 варианта с кратким описанием плюсов и минусов

Точность формулировки — это уважение ко времени и результату.
Чем конкретнее запрос, тем полезнее ответ.

Внедрив эти две привычки (озвучивать допущения и определять термины), вы перестаёте играть в испорченный телефон с машиной.

Выбор модели: правило дорогого инструмента

Раз уж мы говорим о ТЗ, в нём всегда есть допущение о ресурсах. С моделями та же логика. Не каждая задача требует «тяжёлой артиллерии». Использовать o1 для перевода письма — это как нанимать нейрохирурга для заклеивания царапины пластырем.

Вот простой алгоритм выбора:

1. Мог бы умный стажёр справиться с этим за 5 минут по готовому шаблону?

Бери быструю модель (GPT-4o mini, Claude Haiku, Gemini Flash). Экономия денег и времени колоссальная.

2. Нужна ли здесь логика, которую невозможно предусмотреть заранее?

(Многоходовый анализ, поиск противоречий в документах, проектирование системы.) Бери reasoning-модель (пример o1, DeepSeek R1 и прочие).

3. Это разовый анализ или потоковая операция?

Если поток — считай стоимость в масштабе, а не за один запрос.

Было: «Напиши письмо клиенту, который недоволен задержкой доставки. Предложи ему скидку, чтобы он успокоился».

Стало: «Ты — руководитель клиентского сервиса премиум-сегмента. Клиент Gold-уровня получил заказ на 4 дня позже планового срока. Доставка сорвана по вине логистического партнёра.

Контекст: клиент ценит не деньги, а время и уважение к себе. Он уже написал гневное письмо в соцсети. Ограничения: не использовать слова «форс-мажор», «приносим извинения за доставленные неудобства» и любые другие клише. Скидку предлагать только в том случае, если она будет воспринята не как откуп, а как жест доброй воли. Критерий готовности: письмо должно занимать не более 120 слов. Кли-

ент после его прочтения должен захотеть ответить и продолжить диалог, а не удалить его в спам.»

Разница между первым и вторым — это разница между «сделай что-нибудь» и «сделай именно это, для этого человека, с этими ограничениями».

КРИТЕРИИ ГОТОВНОСТИ ГЛАВЫ (Quick Win)

Вы освоили Главу 2, если можете прямо сейчас:

Взять свою последнюю рабочую задачу для ИИ.

Прогнать её через «нулевой шаг» (попросить модель озвучить предположения).

Найти в своём промпте два оценочных слова (типа «хороший», «быстро») и заменить их на измеримые критерии.

Задать себе вопрос «а правильную ли я модель выбрал для этого?».

Если всё получилось — вы перешли на второй уровень инженерного мышления. Вы не просто просите, вы проектируете задачу.

Я упоминал GPT-4o, Claude Sonnet, Gemini 1.5 Pro, o1 и DeepSeek R1. Это всё еще работает, но теперь есть новые флагманы:

Тяжёлые reasoning-модели:

OpenAI o3 (февраль 2025), o3-mini, o4-mini (апрель 2025) — поддержка изображений, лучшие бенчмарки

DeepSeek V4 (апрель 2026) — открытый для всех, на 90–95% дешевле конкурентов

Claude Opus 4.7 с Adaptive Thinking (апрель 2026) — мо-

дель сама решает, сколько думать

GPT-5.5 (апрель 2026) — новейший флагман OpenAI

Быстрые/дешёвые модели:

GPT-5.4 Mini (0.75/M токенов) и GPT5.4 Nano (0.75/M токенов) — запущены в апреле 2026

Claude Haiku 4.5 ($1.00/M) — лучший в соблюдении структуры

Gemini 3.1 Flash Lite ($0.25/M) — лучшая мультимодальность

Средний класс:

Claude Sonnet 4.6 (~$3/M) — сбалансированный

GPT-5.4 ($2.50/M) — отличная работа с инструментами

Gemini 2.5 Flash ($0.30/M) — быстро, дёшево, большое контекстное окно

Глава 3. Нейролингвистика в деле: читаем и адаптируем

Последние два года я наблюдаю, как мир сходит с ума.

Люди разделились на три лагеря.

Одни кричат, что ИИ — это зло, и любой, кто пользуется нейросетями, шарлатан. Другие слепо копируют всё, что сгенерировала машина, даже не вычитав текст, и заваливают интернет мусором. Третьим всё равно. Они просто скроллят ленту.

Самое мерзкое, что я испытал как автор, случилось не тогда, когда у меня украли код. И не тогда, когда не заплатили за работу. Самое мерзкое случилось, когда я написал что-то сам. От руки. Головой. Без единого промпта. А мне сказали: «Ой, да это же ИИ сгенерировал».

Знаете это чувство? Ты сидел час, два, три. Ты подбирал слова. Ты мучился над структурой. Ты вложил в текст часть себя. А тебе говорят: «Ну, это просто нейросеть». Как будто ты — не человек. Как будто твоего труда уже не существует. Это новая форма обесценивания, которую принесли нейросети.

И ладно бы проблема была в хейтерах. Но проблема ещё и в том, что люди, которые не разобрались в инструментах, сами создали нейросетям такую репутацию. Они копипастят

ответы ChatGPT в коммерческие предложения, не убирая фразу «Конечно, я могу вам помочь!». Они генерируют картинки с шестью пальцами и публикуют как обложки книг. Они постят контент, в котором модель признаётся: «Как ИИ, я не имею мнения». И весь мир теперь думает, что любой, кто использует ИИ, — такой же.

Вот поэтому я пишу эту главу. Она не только о том, как читать других. Она о том, как сделать так, чтобы вас не спутали с бездумным генератором. Как сохранить свой голос в эпоху, когда всех ровняют под одну гребёнку. Когда я вижу текст, я хочу понимать: это писал человек или это выплюнула модель без присмотра?

И вы должны уметь видеть эту разницу. Потому что если вы не отличите, как вы докажете, что ваш труд — ваш?

Эта глава научит вас анализировать стиль — чужой и свой. Но начнём мы с главного: как люди вообще принимают решение о том, кто перед ними — автор или машина?

Вы отправляете клиенту идеально составленное предложение — и получаете сухое «спасибо, подумаем». Коллега пишет в чат короткое «ок», но вы чувствуете враждебность. Вы часами готовите аргументы, но разговор уходит в песок.

Почему?

Потому что мы часто слышим слова, но не слышим человека. Мы отвечаем на запрос, игнорируя стиль мышления собеседника.

Один принимает решения через цифры, другой — через доверие, третий — через статус. Языковая модель умеет «читать между строк» в тексте — распознавать психотипы, мотивацию и даже то, что человек не сказал. Эта глава научит вас слышать собеседника так, как это делают опытные переговорщики после десятилетий практики.

ПРИНЦИП

Каждый текст — это не только информация. Это слепок мышления автора. Длина предложений, выбор слов, количество оговорок, даже структура абзацев — всё это данные. Исследования в психолингвистике показывают устойчивые связи между стилем письма и личностными чертами: тревожные люди используют больше слов неопределённости («возможно», «наверное»), доминантные — короткие утвердительные предложения, ищущие признания — чаще ссылаются на мнение других.

Современные LLM (особенно GPT-5.4 и Claude Opus 4.7) обучены на гигантских массивах человеческих текстов и неосознанно усвоили эти паттерны. Им не нужен диплом психолога — им нужен правильный промпт.

Главное этическое правило этой главы:

мы не манипулируем, мы адаптируем коммуникацию.

Если описать собеседнику, что именно анализирует ИИ и как это используется, он должен понимать и принимать это.

ИНСТРУМЕНТ: Три уровня анализа

Уровень 1. Психологический портрет по переписке

Базовый сценарий: у вас есть переписка с клиентом, партнёром или сложным коллегой. Вы хотите понять, как с ним эффективно разговаривать.

Пример:

Ты — опытный психолог - эксперт в области коммуникаций. Проанализируй переписку ниже и составь рабочий портрет автора.

ПЕРЕПИСКА:

[вставьте текст]

Проанализируй по блокам:

1. СТИЛЬ КОММУНИКАЦИИ:

- Прямой или косвенный?
- Краткий или развёрнутый?
- Структурированный или эмоциональный?

2. МОТИВАЦИЯ И ЦЕННОСТИ:

- На что ссылается чаще: результат, процесс, отношения, статус, безопасность?
- Что явно волнует (повторяющиеся темы)?
- Какие темы обходит или замалчивает?

3. ТОЧКИ НАПРЯЖЕНИЯ:

- Где появляются оговорки, защитные формулировки, избыточные объяснения?
- Что он пытается контролировать?

4. КАК С НИМ РАБОТАТЬ:

- Оптимальный формат коммуникации (коротко/развёрнуто, цифры/образы, прямо/дипломатично)
- Какие аргументы подействуют?
- Чего избегать в общении?

Формат ответа: короткий рабочий портрет на полстраницы. Это гипотеза, не диагноз.

Уровень 2. Анализ по модели Большой Пятёрки (OCEAN)

Когда нужно больше точности, используем самую научно обоснованную модель личности из существующих.

Пример:

Проанализируй следующий текст по модели Большой Пятёрки (OCEAN).

ТЕКСТ:

[переписка или сообщение]

Для каждой черты: Уровень (Высокий / Средний / Низкий) + лингвистические сигналы + что это значит для коммуникации.

ОТКРЫТОСТЬ ОПЫТУ: ...

ДОБРОСОВЕСТНОСТЬ: ...

ЭКСТРАВЕРСИЯ: ...

ДОБРОЖЕЛАТЕЛЬНОСТЬ: ...

НЕВРОТИЗМ: ...

ИТОГ: доминирующие черты, главный мотиватор и демотиватор, оптимальный стиль общения.

ВАЖНО: это рабочая гипотеза на основе текста, не клинический диагноз.

Психотип	Ключевые слова	
• Лидерство	Достигай!	Подчинись
• Эмпатия	Вместе	Игнорируй
• Стратегия	Видение	Статус-кво
• Инновация	Прорыв	Ограничение
• Ответственность	Доверие	Критика
• Перфекционизм	Результат	Ошибки

Было: «Я написал клиенту стандартное КП с цифрами и кейсами, он ответил 'интересно, вернёмся позже'».

Стало: Прогнали его переписку через портрет.

Видим: высокая доброжелательность (избегает открытого «нет»), высокая потребность в определённости, средняя добросовестность. В ответе нет цифр — только тёплый тон, доверие, отсутствие давления. Клиент реально заинтересован, но ему нужно время, чтобы всё обдумать.

Уровень 3. Слова-ключи под психотип

Понимать портрет — половина дела. Вторая половина — знать конкретные слова и конструкции, которые резонируют с этим типом человека.

Пример:

На основе следующего профиля составь словарь комму-

никации.

ПРОФИЛЬ:

[из предыдущего анализа]

СЛОВАРЬ:

- Слова и фразы, которые резонируют (15-20)
- Слова и фразы, которые создают трение (10-15)
- Конструкции для подачи идеи (3-5 шаблонов)
- Конструкции для работы с возражением (3-5)
- Чего никогда не делать (5-7 пунктов)

Было:

«Давайте подпишем договор до пятницы, а то цены вырастут».

Стало (для человека с высокой потребностью в определённости):

«Цены зафиксированы в этом предложении. Если решите до пятницы, смета не изменится — и мы запускаемся по плану».

Разница: первый вариант давит страхом. Второй — даёт предсказуемость и контроль.

КРИТЕРИИ ГОТОВНОСТИ ГЛАВЫ (Quick Win)

Вы освоили эту главу, если можете прямо сейчас:

Взять последнюю переписку с клиентом или коллегой.

Прогнать её через портрет (Уровень 1).

Сгенерировать словарь ключей (Уровень 3).

Переписать своё последнее сообщение этому человеку, используя слова, которые резонируют, и убирая те, что со-

здают трение.

Отправить и сравнить реакцию со своей обычной.

Если собеседник ответил быстрее, теплее или конкретнее, чем обычно — техника работает.

Это первая глава Части II — мы переходим от одиночных запросов к проектированию автономных систем.

Глава 4. Анатомия агента, или как написать алгоритм поведения текстом

Я поверил им трижды.

Сначала — когда попросили логотип.

«Сделай, пожалуйста, ты же умеешь. Мы потом оплатим».

Я умел. Я сделал. Потом — рекламное видео.

«Ты же можешь, ты же инженер, у тебя нейросети».

Я мог. Я сделал. Потом — этикетку. Песню для радио. Я делал всё. Я ночами сидел, осваивал генеративные сети, мучился с промптами, боролся с кривыми руками нейросети, чтобы выдать результат за результатом.

Они хвалили. Говорили «отлично».

И добавляли: «***В следующий раз обязательно оплатим***». Следующий раз наступал, и появлялась новая просьба. Платежа не было. Я был не инженером. Я был бесплатным генератором, в который закидывают запрос — и он выдает картинку, видео, мелодию, текст.

Тогда я ещё не знал, что я нарушил главный принцип работы агента. У меня не было формализованного протокола передачи результата. Не было Definition of Done(критерия готовности).

Не было чётко прописанного условия, при котором работа

считается завершённой, а оплата — обязательной.

Я действовал как чат-бот: получил запрос — ответил. Получил следующий — ответил. Без состояний, без границ, без условий эскалации.

Агент, которого я проектирую теперь, так не работает. У него есть чёткий алгоритм с условиями перехода. Он не отдаёт результат, пока не выполнены критерии готовности. Он знает свои границы и не стесняется о них сообщить.

Если бы я вёл себя тогда как спроектированный агент, я бы сказал:

«Вот коммерческое предложение с условиями. Результат будет передан после подтверждения и предоплаты».

И это спасло бы меня от месяцев бесплатного труда.

Вот почему эта глава — первая в Части II. Потому что прежде чем строить сложные системы, вы должны понять разницу между чатом и агентом. Чат — это вежливый исполнитель, который ждёт следующее сообщение. Агент — это сущность, которая знает, когда задача выполнена, и не боится закрыть сессию.

Вот Вы написали идеальный системный промпт. Модель отвечает прекрасно: в нужном тоне, с правильной структурой и без воды. Вы воодушевлены и отдаёте этот промпт коллегам. Они пользуются. И через три дня всё ломается.

Почему?

Потому что вы написали инструкцию для чата. А требовалась инструкция для агента. Разница принципиальная.

Чат ждёт от пользователя команду и выполняет её. Агент выполняет работу. **Чат отвечает на сообщение. Агент движется к цели.**

Вот как это выглядит на практике:

Инструкция для чата: «Ты — опытный маркетолог. Отвечай кратко и по делу. Используй деловой тон. Если не знаешь ответа — скажи об этом.»

Прекрасная инструкция. Модель будет вести себя как маркетолог в любом диалоге. Но если коллега напишет: *«Собери мне заявки за май, сравни с апрелем и пришли сводку в Excel»,*

— чат ответит вежливо: «Я могу помочь с маркетинговыми вопросами. За какой период нужны данные?» и будет ждать следующего сообщения.

Инструкция для агента (решающая ту же задачу): «При запросе отчёта:

1. Проверь тип отчёта в запросе (месяц, регион, менеджер).

2. Если тип не указан — запроси одним предложением.

3. Обратись к базе данных SQL и получи нужные цифры.

4. Сверь результат с контрольными суммами за прошлый период.

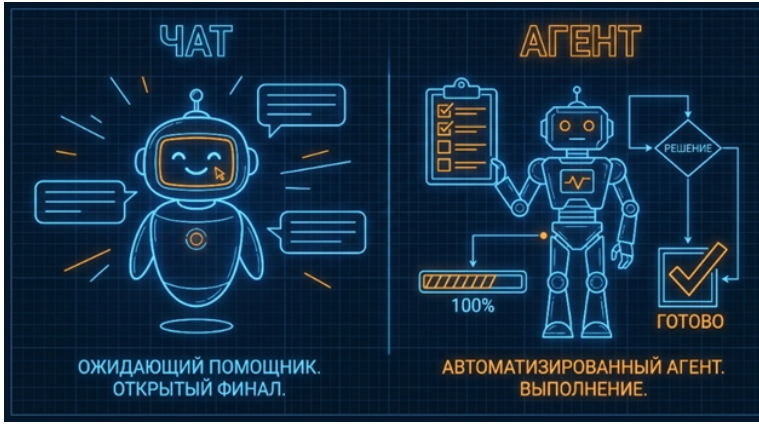
5. Если отклонение больше 15 % — поставь в сводке тег [ТРЕБУЕТ ПРОВЕРКИ].

6. Отправь результат в Slack-канал #ежедневные-отчёты.

7. После отправки напиши пользователю: «Готово. Свод-

ка в #ежедневные-отчёты».»

Первый вариант описывает, как вести себя. Второй — что делать. Агент не ждёт продолжения диалога. Он выполняет последовательность шагов, проверяет результат и завершает задачу.



ПРИНЦИП

Поведенческая инструкция для агента строится вокруг трёх элементов:

1. **Цель.** Что должно быть достигнуто. Одна задача, одно завершённое состояние.

2. **Алгоритм.** Последовательность действий с условиями перехода между ними. «Сделай А. Проверь критерий Х. Если да — переходи к Б. Если нет — вернись к А и учти ошибку.»

3. **Критерий завершения.** Как модель понимает, что работа выполнена и пора выдавать финальный результат. Мы называем это Definition of Done (DoD) — формальный, проверяемый список признаков готовности.

Без любого из этих трёх элементов инструкция неполная, и поведение модели будет непредсказуемым в потоке реальных задач.

ИНСТРУМЕНТ

Паттерн 1: Алгоритм с циклом проверки

Самый простой и надёжный паттерн для задач, где первый результат почти всегда требует доработки:

Пример:

Твоя задача — [одна конкретная цель].

Алгоритм работы:

Шаг 1. [Первое действие].

Шаг 2. Проверь результат по критерию [конкретный критерий].

Шаг 3. Если критерий выполнен — переходи к Шагу 4.

Если нет — вернись к Шагу 1, учтя, что именно не сработало.

Шаг 4. Выдай финальный результат.

Не выдавай результат, пока Шаг 2 не пройден полностью.

Пример для редакторского ассистента:

Твоя задача — отредактировать текст пользователя.

Алгоритм:

1. Прочитай текст целиком. Не правь.

2. Выдели три проблемы: нарушения логики между абзацами, повторения одной мысли, утверждения без подтверждения.

3. Перепиши, устранив проблемы.

4. Проверь: остались ли утверждения без подтверждения?

5. Если да — доработай эти места. Если нет — выдай финальный текст.

Критично, чтобы критерии на шаге проверки были однозначными. «Хороший текст» — не критерий. Модель будет интерпретировать его по-разному. «Остались ли утверждения без доказательств» — критерий: ответ либо «да», либо «нет».

Паттерн 2: Иерархия правил

В диалоге с агентом рано или поздно возникает конфликт. Пользователь просит то, что противоречит системной инструкции. Без явной иерархии модель начинает «договариваться» и постепенно отступает от правил.

Пример:

Приоритеты, от высшего к низшему:

УРОВЕНЬ 1 (абсолютный): Правила безопасности и конфиденциальности.

Не переопределяются ни при каких условиях.

Если запрос противоречит — вежливо откажи и объясни причину.

УРОВЕНЬ 2: Формат вывода, заданный в системной инструкции.

Не изменяется по запросу пользователя.

Если запрашивается другой формат — объясни, что формат фиксирован.

УРОВЕНЬ 3: Тон и стиль.

Адаптируются в рамках заданных границ. Числовая нумерация снимает неоднозначность. Модель сама разрешает конфликт, сверяясь с приоритетами.

Паттерн 3: Ролевая декомпозиция «Генератор — Критик — Редактор»

Для сложных задач, где нужна и креативность, и точность, одна роль даёт компромиссный результат. Три роли, работающие последовательно, дают результат, прошедший через независимые фильтры.

ПРИМЕР:

Работай в три этапа:

ЭТАП 1 — ГЕНЕРАТОР.

Напиши черновик. Приоритет — полнота идей. Не сокращай, не редактируй.

Выведи: —— ГЕНЕРАТОР ЗАВЕРШЁН ——

ЭТАП 2 — КРИТИК.

Возьми черновик. Найди три слабых места по критериям:

- логические переходы между аргументами,
- отсутствие подтверждения у ключевых тезисов,
- соответствие структуры задаче.

Для каждого — одна конкретная рекомендация.

Выведи: —— КРИТИК ЗАВЕРШЁН ——

ЭТАП 3 — РЕДАКТОР.

Возьми черновик (Этап 1) и замечания (Этап 2).

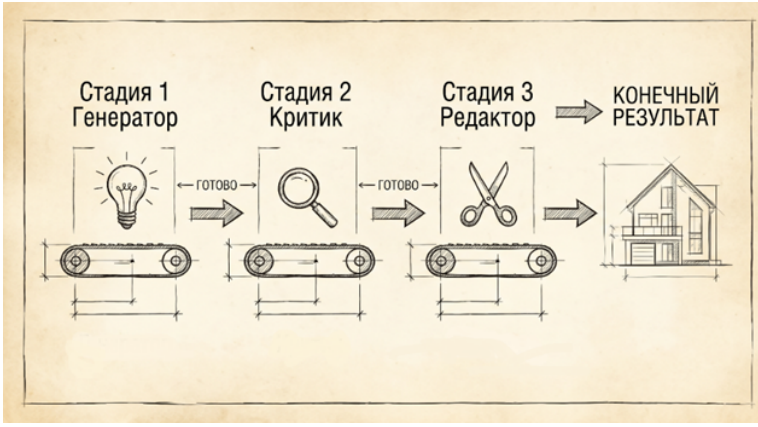
Перепиши, устранив проблемы. Сохрани смысловую полноту оригинала.

Пользователю выдавай только результат Этапа 3.

Маркер — ГЕНЕРАТОР ЗАВЕРШЁН — это не украшение. Это явный сигнал модели:

«режим сменился, теперь ты не создаёшь, а критикуешь».

Без таких сигналов модель продолжает дополнять черновик вместо того, чтобы его разбирать.



Было: «Напиши коммерческое предложение для клиента из логистики».

Стало: Ты — агент по подготовке КП. Цель: подготовить

коммерческое предложение, которое клиент поймёт за 2 минуты чтения.

Алгоритм:

1. Проанализируй бриф клиента. Выдели явно: главную боль клиента, его бюджетный диапазон, его критерий выбора.

2. Напиши КП по структуре: боль наше решение доказательства (кейсы) следующий шаг с дедлайном.

3. Проверь КП по DoD:

Каждое утверждение подкреплено примером или цифрой.

Объём — не более 1 страницы, 300–350 слов.

Нет слов «инновационный», «уникальный», «команда профессионалов».

Есть конкретный следующий шаг с датой.

Если DoD пройден — отдай результат. Если нет — доработай.

Разница: вы не написали модельному ассистенту, «как писать КП». Вы дали ему алгоритм, критерии приёмки и границы свободы.

КРИТЕРИИ ГОТОВНОСТИ ГЛАВЫ (Quick Win)

Вы освоили эту главу, если можете прямо сейчас:

1. Взять свою регулярную рабочую задачу (письмо клиенту, отчёт, анализ чего-либо).

2. Описать её как алгоритм минимум из трёх шагов.

3. Сформулировать Definition of Done — три измеримых критерия, по которым вы поймёте, что результат готов, даже

не читая его внимательно.

4. Добавить иерархию приоритетов: что самое важное в этой задаче, а что можно адаптировать.

Если справились — вы перешли Рубикон. Вы только что написали первую агентную инструкцию.

Это вторая глава Части II — мы даём агенту долговременную память и доступ к внешним знаниям.

Глава 5. Память агента: проектируем поиск знаний (RAG)

Вы запустили агента поддержки. Он вежлив, быстр и отлично держит стиль. Но на третий день эксплуатации приходит клиент и пишет:

«Что там с моим возвратом? Номер заказа — 14592».

Агент отвечает:

«Для оформления возврата, пожалуйста, зайдите в личный кабинет и следуйте инструкциям».

Вежливо, но бесполезно. Клиент взрывается. Почему это произошло?

Потому что модель отвечала из своей общей памяти. Она не знала, что заказ 14592 был возвращён вчера, что деньги зависли, и что по этому кейсу уже открыт тикет с пометкой «срочно». Она действовала как очень начитанный, но слепой стажёр. Вы дали агенту блестящие инструкции, но не дали ему **память**.

Языковая модель обучена предсказывать следующий токен, а не извлекать конкретные факты из внешней базы данных.

Когда задача требует точного ответа, основанного на актуальных корпоративных данных, модель без доступа к ним сделает две вещи: либо честно признается в незнании (хоро-

шо, но бесполезно), либо, что опаснее, придумает правдоподобный ответ.

Это называется **галлюцинацией**, и для бизнес-систем это неприемлемо.

ПРИНЦИП

Решение существует и называется **RAG — Retrieval-Augmented Generation (Генерация, дополненная поиском).** Принцип прост: перед тем как модель начнёт отвечать, вы находите нужные данные в своей базе знаний и подставляете их прямо в контекст запроса. Модель отвечает, опираясь на эти данные, а не на статистические паттерны из своего обучения.

Вот как выглядит полный RAG-пайплайн из трёх шагов:

1. Пользователь задаёт вопрос.

2. Система ищет релевантные документы, фрагменты или цифры в базе знаний компании.

3. Эти данные подставляются в промпт, и модель отвечает строго на их основе.

Звучит как серебряная пуля. Но практика 2026 года, включая исследования в медицинской и финансовой сферах, показывает важный нюанс:

RAG действительно резко снижает галлюцинации, но не устраняет их полностью. Модель может выдать «**верную цитату из неверного места**» или проигнорировать найденный документ, добавив то, чего там не было.

Поэтому даже с RAG нам нужна постобработка и провер-

ка, которую мы разберём в следующих главах.

Но сначала — фундамент.

Главная инженерная сложность RAG спрятана в слове «**найти**». На практике это три абсолютно разные задачи, и для каждой нужен свой инструмент.

ИНСТРУМЕНТЫ

Тип 1: Векторный поиск — поиск по смыслу

Это ваш основной инструмент для работы с **неструктурированными текстами:** регламентами, инструкциями, статьями, архивами переписки.

Принцип работы:

Текст превращается в эмбеддинг — длинный вектор (набор чисел), где тексты с похожим смыслом расположены близко друг к другу. Запрос пользователя также превраща-

ется в вектор, и система ищет ближайшие к нему документы в этом пространстве. Благодаря этому запрос «регламент увольнения» найдёт документ «порядок расторжения трудового договора», даже если слова не совпадают.

Без правильно настроенной нарезки (чанкинга) этот механизм бесполезен. Слишком большой чанк — вектор размывается, и документ срабатывает на любой запрос. Слишком маленький — теряется контекст. Рабочий стандарт в индустрии сегодня — 300–500 токенов с перекрытием в 10–15% между соседними фрагментами.

Для быстрого старта можно использовать следующую инструкцию:

Пример:

Твоя задача — подготовить документ для загрузки в базу знаний (RAG).

Разбей текст ниже на смысловые блоки со следующими правилами:

- Каждый блок содержит одну завершённую мысль (примерно 200–400 слов).
- Если мысль длиннее — раздели по логической границе.
- Начинай каждый блок с заголовка раздела, к которому он относится.
- Добавь к каждому блоку 3–5 ключевых тем для последующей фильтрации.

Текст:

[ваш документ]

После нарезки каждый блок отправляется в **векторную базу данных.**

По состоянию на 2026 год выбор здесь широк:

- легковесные open-source решения вроде pgvector (работает прямо в PostgreSQL) и ChromaDB для прототипов;

- промышленные гибридные системы вроде Qdrant (показывает отличные результаты в бенчмарках облачных решений для 2026 года), Pinecone и Weaviate;

- энтерпрайз-гиганты Elasticsearch и Vespa, предоставляющие комплексные AI-поисковые платформы.

Тип 2: Text-to-SQL — поиск по точным значениям

Векторный поиск бесполезен, если клиенту нужно узнать, «сколько денег вернули Иванову 5 марта». Здесь нужна работа с **реляционными базами данных и SQL.**

Современные модели (такие как GPT-5.4 или Claude Opus 4.7) отлично пишут SQL-запросы по описанию на естественном языке, но **при критически важном условии**: они должны знать схему таблиц. Без неё модель начинает угадывать названия столбцов и получает синтаксические ошибки.

Формула успеха: явно описать в промпте структуру таблиц и ключевые особенности данных.

Пример:

База данных со следующей структурой:

Таблица: sales

- id (INT, уникальный)

- date (DATE, формат YYYY-MM-DD)

- product_name (VARCHAR)
- category (VARCHAR, допустимые значения: "Ноутбуки", "Мониторы", "Периферия")
- quantity (INT)
- total (DECIMAL, руб.)

ВАЖНО: возвраты хранятся с отрицательным quantity. Данные — с 01.01.2025.

Задача: [ваш вопрос на естественном языке].

Напиши SQL-запрос и одним предложением объясни, что он делает.Указание конкретных значений в категориях (как «Ноутбуки», а не «Ноутбук») устраняет целый класс ошибок, связанных с несовпадением строк в фильтрах.

Тип 3: Графовые базы данных — поиск по связям

Есть вопросы, на которые не ответят ни векторный, ни SQL-поиск.

«Кто из сотрудников общался с контрагентом X через посредников?», «Какие компании связаны с контрагентом Y через общих учредителей?».

Это вопросы о структуре отношений, и для них существует **граф.**

Графовая база данных хранит информацию в виде узлов (люди, компании, документы) и рёбер (связей между ними с указанием типа: «является руководителем», «подписал», «аффилирован с»). С ростом сложности корпоративных данных всё более востребованным становится GraphRAG — подход, который объединяет графовые и векторные базы для

повышения точности ответов.

На практике в 2026 году популярны Neo4j и специализированные open-source решения вроде Nexus (гибрид графа и векторного поиска для RAG) и OverGraph (встроенная графовая база данных с поиском, работающая прямо внутри процесса). Крупные вендоры,

такие как Memgraph и NebulaGraph, также выпускают собственные GraphRAG-решения.

Тип 4: Гибридный поиск — когда нужно всё и сразу

В реальных бизнес-задачах почти никогда не нужен только один тип поиска. Представьте интернет-магазин: пользователь пишет «ноутбук для дизайнера, бюджет до 150 000». Здесь нужно совместить:

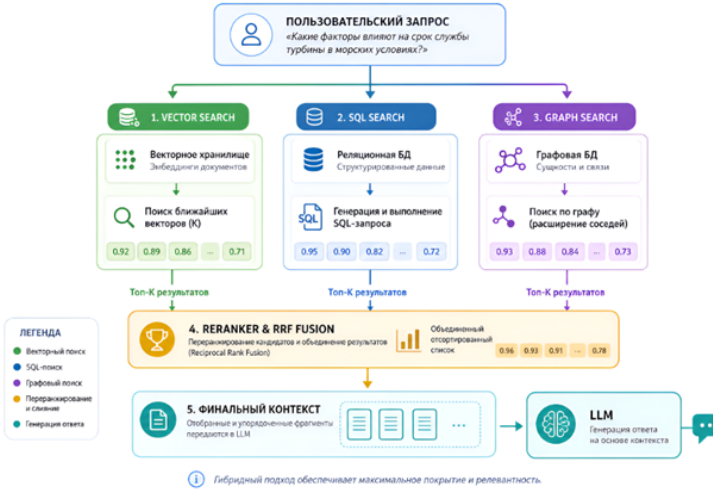
Семантический поиск (векторный), чтобы найти общее описание.

Ключевой поиск (BM25), чтобы не пропустить конкретные модели.

Числовые фильтры, чтобы отсечь всё, что выше 150 000.

Это называется **гибридным поиском** (Hybrid Search). Результаты объединяются алгоритмом Reciprocal Rank Fusion (RRF): документ, попавший в топ и векторного, и ключевого поиска, получает высший итоговый ранг. В инженерной практике 2026 года стандарт усложнился: после слияния применяется **модель-реранкер** (часто — специализированная легковесная LLM), которая оценивает пару «за-

прос-документ» и пересортировывает результат для максимальной релевантности.



Тип 5: Агентный RAG (Agentic RAG) — эволюция 2026 года

До сих пор мы описывали «классический» RAG: один запрос один поиск один ответ. Но передовые исследования 2026 года, принятые на конференции уровня ICLR, активно внедряют концепцию **агентного RAG.**

Что это значит? Поиск информации превращается из пассивной операции в **итеративный процесс**. LLM-агент больше не делает один «слепой» запрос к базе. Вместо этого он:

1. Получает сложный вопрос пользователя.

2. Раскладывает его на несколько подзадач и подзапросов.

3. Итеративно ищет информацию, оценивая качество и полноту найденного на каждом шагу.

4. Может решить, что информации недостаточно, переформулировать запрос и найти недостающие звенья.

Такой подход решает две классические болезни: over-search (когда модель ищет то, что и так знает) и under-search (когда модель не ищет там, где нужно). Современные фреймворки, такие как HiPRAG, обучают агентов находить оптимальный баланс между поиском и рассуждением, снижая показатель over-search до 2-3%.

Конец ознакомительного фрагмента.