12+

Радик Яхин

Нейросети для работы

Книга-тренажёр

Радик Яхин

Нейросети для работы. Книга-тренажёр

«Издательские решения»

Яхин Р.

Нейросети для работы. Книга-тренажёр / Р. Яхин — «Издательские решения»,

Вы конкурируете не с ИИ, а с людьми, которые его уже используют.Книга «Нейросети для работы» — это не учебник, а практический тренажёр по внедрению нейросетей в работу. Забудьте про чтение в метро: эту книгу нужно открывать только за компьютером. Ваша задача — не просто узнать про ИИ, а заставить его работать на вас уже в первый день.Пройдите путь от базовых запросов до сборки мультиагентных систем. Станьте автономным сотрудником, которого невозможно заменить.

Содержание

Нейросети для работы 6
Часть I. Фундамент: Укрощение кремниевого разума 12
Глава 1. Как «думает» LLM-модель: Управление токенами и контекстом 12
Глава 2. Борьба с галлюцинациями и архитектурные лимиты моделей 17
Глава 3. Создание личного ИИ-периметра и безопасность данных 21
Глава 4. Работа с вложениями, таблицами и документами 24
Часть II. Технологии промптинга: От простых команд к каскадным системам 29
Глава 5. Золотой стандарт промпта: Пятикомпонентная матрица 29
Глава 6. Программирование логики: Chain-of-Thought и Tree-of-Thoughts 33
Глава 7. Продвинутая разметка: Тегирование и структурирование данных 36
Глава 8. Детекция ИИ-контента и критическая оценка 40
Конец ознакомительного фрагмента. 41

Нейросети для работы
Книга-тренажёр

Радик Яхин

ISBN 978-5-0071-2350-1
Создано в интеллектуальной издательской системе Ridero

Нейросети для работы

АВТОНОМНЫЙ СОТРУДНИК: Книга-тренажёр по извлечению максимума из ИИ и AI-агентов

Введение. Эффект сложной автоматизации: Почему ваши цели не работают без ИИ-систем

Вы наверняка не раз ловили себя на странном ощущении: январские цели записаны, план на квартал расписан по неделям, мотивация на месте, а к середине февраля половина пунктов либо вычеркнута, либо задвинута в дальний ящик. И дело тут не в лени и не в недостатке дисциплины. Дело в том, что внутри вашего рабочего дня сидят десятки мелких повторяющихся операций — перенос данных из письма в таблицу, расшифровка голосовой заметки, пересказ совещания для коллеги, который пропустил встречу, подготовка однотипного отчёта, — и каждая из них отъедает по пять, десять, двадцать минут. По отдельности это незаметно. В сумме это **три-четыре часа** ежедневно, которые вы тратите на механическую работу, не требующую ни вашего опыта, ни вашей креативности. И именно эти часы делают ваши амбициозные цели физически невыполнимыми: вы просто не успеваете до них добраться. Эта книга написана для того, чтобы вернуть вам эти часы и превратить их в ресурс для задач, которые действительно двигают бизнес и карьеру вперёд.

Давайте сразу договоримся о главном сдвиге в мышлении, без которого всё дальнейшее не сработает. Вы конкурируете не с искусственным интеллектом. Искусственный интеллект — это инструмент, такой же, как калькулятор или экскаватор. Вы конкурируете с **людьми**, которые уже автоматизировали свою рутину и освободившееся время направляют на стратегию, переговоры, обучение и отдых. Пока вы вручную составляете сводку по пяти письмам, ваш конкурент за тридцать секунд получает ту же сводку от ИИ-ассистента и тратит сэкономленные двадцать минут на звонок ключевому клиенту. Разница в скорости накапливается каждый день, и через месяц превращается в пропасть. Это и есть философия автономности, которая проходит красной нитью через всю книгу: автономный сотрудник — не тот, кто работает без людей, а тот, кто делегировал повторяющиеся операции машинам и сосредоточился на решениях, которые машина принять не способна.

Теперь о том, что изменилось буквально в последние два года и почему эта книга не могла быть написана раньше. Произошла **мультимодальная революция**. Раньше нейросеть понимала только текст: вы вводили слова, она отвечала словами. Сегодня голос, видео и изображения стали такими же полноценными входными данными для ИИ, как текст. Вы можете загрузить фотографию повреждённого товара и получить структурированный акт, можете проговорить голосовую заметку по дороге на встречу и получить готовое техническое задание, можете записать совещание на видео и через минуту получить протокол с распределёнными задачами. Это принципиально меняет объём рутины, которую можно делегировать: если раньше автоматизация касалась в основном писем и таблиц, то теперь она распространяется на звонки, видеозаписи, сканы, фотографии продукции, презентации и даже устные обсуждения в переговорке.

Чтобы вы могли измерить свой прогресс не ощущениями, а цифрами, я ввёл в книгу авторскую методику, которую назвал **«Индекс составной автоматизации»**, сокращённо ИСА. Суть её проста: мы считаем не абстрактную «продуктивность», а конкретные минуты, которые вы ежедневно тратите на повторяющиеся операции, и переводим их в высвобожденное время с учётом нескольких множителей. Формула выглядит так: суммируем все автоматизированные минуты за период, умножаем на коэффициент повторений этой операции в неделю, затем на множитель сложности задачи и на коэффициент модальности, а полученное число

делим на baseline ручного труда, то есть на то время, которое вы тратили до автоматизации. Коэффициент модальности отражает тот факт, что работа с разными типами данных экономит время неравномерно: текстовые задачи имеют коэффициент 1,0, задачи с изображениями — 1,4, задачи с аудио — 1,8, а задачи с видео — 2,2. Почему так? Потому что ручная обработка видео или аудио исторически отнимает больше времени на единицу результата: расшифровать час записи вручную — это четыре-шесть часов работы, тогда как пересказать текст на одну страницу можно за пятнадцать минут. ИИ сжимает именно эти «дорогие» операции сильнее всего.

Приведу пример, чтобы формула перестала быть абстракцией. Допустим, вы менеджер проектов и каждый день тратите двадцать пять минут на пересказ итогов утренней планёрки в общий чат, причём планёрка проходит пять раз в неделю. Вы записываете её на диктофон, загружаете в ИИ-сервис и получаете саммари за две минуты. Автоматизированных минут двадцать три, коэффициент повторений пять, множитель сложности для рядовой задачи равен единице, коэффициент модальности для аудио равен 1,8. Baseline ручного труда — те самые двадцать пять минут. Считаем: $23 \times 5 \times 1 \times 1{,}8 / 25 = 8{,}28$. Это значит, что за одну рабочую неделю вы высвободили время, эквивалентное более чем восьми «ручным» минутам на каждый автоматизированный слот, а в абсолютных цифрах сэкономили почти два часа. А теперь представьте, что таких операций в вашем дне не одна, а шесть или восемь. Именно поэтому ежедневная автоматизация даже двух-четырёх процентов повторяющихся операций даёт **нелинейный рост** свободного времени: вы не просто складываете сэкономленные минуты, вы убираете переключения контекста, снижаете усталость и получаете непрерывные блоки для глубокой работы.

Чтобы вы понимали, на какие результаты ориентироваться, приведу данные по выборке бета-тестеров, которые прошли черновую версию этой книги от начала до конца. Во-первых, высвобождение времени составило от четырёх до четырнадцати часов в неделю в зависимости от исходной загрузки и количества автоматизированных процессов. Во-вторых, скорость подготовки материалов — презентаций, отчётов, писем, контент-планов — выросла в полтора-два с половиной раза. В-третьих, расходы на внешних исполнителей, которым раньше делегировались расшифровки, вёрстка, базовый дизайн и написание типовых текстов, сократились на двадцать-пятьдесят процентов. В-четвёртых, и это особенно важно для корпоративного сектора, количество инцидентов утечки данных при работе с ИИ снизилось до нуля, потому что в книге выстроена чёткая система классификации информации и правил безопасности. И наконец, время на обработку аудио- и видеоматериалов сократилось в четыре-восемь раз: то, что раньше занимало полдня, теперь укладывается в двадцать-сорок минут.

Эти цифры не взяты из воздуха, и я не прошу верить мне на слово. В книге заложена **методика самостоятельного замера**: в первую неделю вы фиксируете свой baseline — записываете, сколько минут уходит на каждую повторяющуюся операцию, какой тип данных при этом используется, как часто операция повторяется. К финалу книги вы заполняете ту же таблицу повторно и получаете персональный отчёт, где видна разница по каждой строке. Никакой магии, только арифметика. Если к концу книги ваш ИСА не вырос хотя бы вдвое — значит, вы пропустили практические блоки, и я настоятельно рекомендую к ним вернуться.

Теперь о том, как правильно проходить эту книгу-тренажёр, чтобы она окупила свою стоимость в первый же день применения. Ключевое слово здесь — «тренажёр». Это не книга для чтения в метро или перед сном. Это книга, с которой нужно сидеть за компьютером, с открытыми ИИ-сервисами и подключённым микрофоном, и выполнять задания из блока «Практика и тренировка» в каждой главе. Без выполнения упражнений вы получите набор интересных фактов, но не навык. Навык формируется только через действие. Для тех, кому нужно быстро внедрить ИИ в рабочие процессы и кто готов выделять по полтора-два часа в день, предусмотрен **двухнедельный трек**: вы идёте по главам последовательно, но выполняете только задания

уровня «Базовый» и «Продвинутый», пропуская экспертные. Для тех, кто хочет построить полноценную систему агентов, мультимодальных конвейеров и корпоративных интеграций, рассчитан **шестинедельный трек**: здесь вы проходите все уровни сложности, включая экспертные задания, и завершаете работу финальным проектом — личной картой автоматизации, в которой зафиксированы конкретные процессы, инструменты и ожидаемая экономия.

Каждая глава построена по одному и тому же циклу из четырёх блоков: сначала теория, где объясняется, как работает технология и почему она устроена именно так; затем реальный кейс, показывающий, как это применили в конкретной компании и какой результат получили; далее пошаговая инструкция или шаблон, который можно скопировать и адаптировать под себя; и наконец практика — упражнение, которое вы выполняете прямо сейчас, не откладывая на потом. Перескакивать через этапы не стоит: без теории практика превращается в бессмысленное копирование, а без практики теория остаётся мёртвым знанием.

И последнее, о чём я обязан сказать во введении. В книге вы найдёте чёткую технику безопасности: полный список данных, которые категорически запрещено загружать в облачные нейросети, в соответствии с Федеральным законом №152-ФЗ «О персональных данных» и законом о коммерческой тайне. Этот список я называю «красные данные», и он включает персональные данные клиентов, биометрическую информацию, проприетарный код, финансовые документы с реквизитами и ряд других категорий. Пренебрежение этими правилами может стоить компании штрафа, репутации, а в отдельных случаях — уголовного преследования. Поэтому прежде чем открывать первую главу, прочитайте блок о технике безопасности в разделе «Инструкция к тренажёру» и держите его в голове на протяжении всей работы.

Вы держите в руках не справочник по нейросетям. Вы держите **систему**, которая за две или шесть недель превратит вас из человека, выполняющего рутину вручную, в автономного сотрудника, у которого рутину выполняет ИИ, а время и энергия уходят на задачи, требующие именно вашего мозга. Начинаем.

Инструкция к тренажёру: Как читать эту книгу

Прежде чем вы перевернёте эту страницу и двинетесь к первой главе, давайте потратим десять минут на то, чтобы договориться о правилах игры. Эта книга устроена не как художественная проза и не как академический учебник, который можно пролистать по диагонали в электричке. Это тренажёр в полном смысле слова: он не передаёт знания, он формирует навык. А навык, как известно, не появляется от чтения — он появляется от повторения действия. Поэтому всё, что я расскажу ниже, напрямую влияет на то, получите ли вы к финалу четыре-четырнадцать часов свободного времени в неделю или закроете файл с ощущением «ну, было интересно».

Начнём с самого важного принципа, который я называю правилом **открытого терминала**. Суть его проста: эту книгу нельзя читать пассивно. Вы не должны лежать с ней на диване, не должны листать её в метро или в очереди к врачу. С ней нужно сидеть за компьютером, с открытым браузером, в котором запущены ИИ-сервисы, и с подключённым микрофоном. Почему именно так? Потому что каждая глава содержит блок «Практика и тренировка», где вам предлагается не прочитать описание действия, а совершить это действие прямо сейчас, своими руками, с конкретным промптом, с конкретной моделью, с конкретным результатом на экране. Если вы читаете задание «Загрузите аудиозапись совещания в сервис транскрибации и получите структурированный протокол», но у вас нет подключённого микрофона и нет записи под рукой — вы просто скользнёте глазами по тексту и пойдёте дальше. А навык не сформируется. Поэтому подготовьте рабочее место заранее: откройте вкладки с **YandexGPT**, **GigaChat**, **Salute**, **DeepSeek**, **Qwen** и, если планируете работать с локальными моделями без доступа в интернет, установите **Ollama** или **LM Studio**. Подключите микрофон — подойдёт даже встроенный в ноутбук, но для голосовых агентов и транскрибации лучше использовать внешнюю

гарнитуру. Выделите на одну главу минимум полтора-два часа непрерывного времени, потому что прерываться на середине упражнения — значит терять контекст и начинать заново.

Теперь о том, как устроена каждая глава изнутри. Я использовал единый четырёхступенчатый цикл, который назвал «Теория — Кейс — Инструкция — Практика», и перескакивать через ступени категорически не рекомендую. Объясню почему. Первая ступень, теория, отвечает на вопрос «как это работает под капотом». Без неё вы будете механически копировать промпты, не понимая, почему один вариант даёт результат, а другой — галлюцинацию. Вторая ступень, кейс, показывает, как именно эта технология сработала в реальной компании с реальными людьми, реальным бюджетом и реальными ошибками. Кейс нужен для того, чтобы вы увидели масштаб применимости и типичные ловушки, в которые попадают те, кто внедряет инструмент впервые. Третья ступень, инструкция, даёт вам готовый алгоритм или шаблон, который можно скопировать и адаптировать под свою задачу без изобретения велосипеда. И четвёртая ступень, практика, закрепляет всё вышеперечисленное через действие: вы берёте свои данные, свой процесс, свой контекст и проходите инструкцию от начала до конца. Если вы пропустите теорию и сразу прыгнете в практику, вы не сможете адаптировать шаблон под нестандартную ситуацию. Если пропустите кейс — не будете знать, где именно система ломается. Если пропустите инструкцию — потратите три часа на то, что делается за двадцать минут. Поэтому идите последовательно, даже если кажется, что теоретический блок вам «уже знаком». В каждой главе теория содержит нюансы, которых нет в других источниках, и именно они отличают уверенное применение от поверхностного.

Следующий вопрос, который неизбежно возникнет у вас к третьей-четвёртой главе: «А мне точно нужно выполнять все задания, включая экспертные?» Нет, не обязательно. В книге заложена трёхуровневая система сложности, и каждый уровень помечен соответствующим тегом в заголовке практического блока. **Базовый** уровень рассчитан на человека, который раньше не работал с нейросетями или работал эпизодически: задания занимают десять-пятнадцать минут, не требуют настройки окружения и используют стандартные облачные сервисы через браузер. **Продвинутый** уровень предполагает, что вы уже освоили базовые промпты, умеете загружать файлы в контекст модели и готовы потратить тридцать-сорок минут на настройку связки из двух-трёх инструментов. **Экспертный** уровень — для тех, кто хочет собрать автономный конвейер, мультиагентную систему или корпоративную интеграцию: здесь задания могут занимать два-четыре часа и требуют понимания API, локального развёртывания моделей или работы с low-code платформами. Как выбирать трек? Если вы линейный специалист или руководитель без технического бэкграунда, идите по цепочке «Базовый плюс Продвинутый» и пропускайте экспертные задания, пока не почувствуете, что вам тесно в рамках продвинутого. Если вы аналитик, разработчик или руководитель ИТ-направления, проходите все три уровня, потому что экспертные задания в этой книге — не «задачки для галочки», а реальные рабочие сценарии, которые вы потом перенесёте в свой проект. И помните: уровни не линейны. Вы можете выполнить базовое задание из пятнадцатой главы, а потом вернуться к продвинутому заданию из четвёртой, если рабочая задача потребовала именно этого навыка. Книга не обязывает к строгому порядку внутри уровней.

Теперь перейдём к теме, которую я обязан поднять до того, как вы введёте свой первый запрос в любую нейросеть. Это **техника безопасности** данных. В России действует Федеральный закон №152-ФЗ «О персональных данных», который устанавливает жёсткие правила обработки информации о физических лицах, а также законодательство о коммерческой тайне, которое защищает внутреннюю информацию компаний. Нарушение этих норм грозит не просто штрафом: для юридических лиц санкции могут достигать миллионов рублей, а в отдельных случаях предусмотрена уголовная ответственность. Поэтому я ввожу в книгу систему классификации данных по трём уровням, которую буду использовать во всех главах без исключения. Зелёный уровень — это информация, которую можно свободно загружать в любой

облачный ИИ-сервис без ограничений: обезличенные тексты, открытые данные, черновики без упоминания имён и реквизитов, обучающие материалы. Жёлтый уровень — данные, которые можно загружать только при условии предварительного обезличивания или только в локальные модели, развёрнутые на вашей инфраструктуре: внутренние документы без персональных данных, рабочие переписки, аналитические отчёты с упоминанием названий проектов. Красный уровень — данные, которые категорически запрещено загружать в любые облачные нейросети при любых обстоятельствах. К красному уровню относятся: персональные данные клиентов и сотрудников в любом виде, включая фамилии, телефоны, адреса, паспортные данные и ИНН; биометрическая информация, то есть записи голоса, фотографии лица, отпечатки пальцев, если они используются для идентификации; проприетарный исходный код и архитектурные схемы, составляющие коммерческую тайну; финансовые документы с реквизитами счетов, номера карт, данные платёжных систем; медицинские записи и любая информация о состоянии здоровья; видеозаписи с камер наблюдения в публичных пространствах, если на них возможна идентификация лиц без их письменного согласия. Аудио- и видеоданные я выделяю как объекты повышенной защиты, потому что голос и изображение лица одновременно являются и персональными данными, и биометрией, и объектами авторского права. Прежде чем загрузить что-либо в облачный сервис, задайте себе один вопрос: «Если эта запись завтра окажется в открытом доступе, нарушит ли она права конкретного человека или раскроет ли секрет моей компании?» Если ответ «да» хотя бы на одну часть вопроса — данные красные, и работать с ними можно только в локальном контуре через Ollama, LM Studio или корпоративную инфраструктуру. В каждой главе, где речь пойдёт о работе с файлами, аудио или видео, я буду отдельно напоминать об уровне допустимости, но ответственность за финальное решение всегда лежит на вас.

Чтобы мы с вами говорили на одном языке и не путались в терминах, которые в индустрии часто используют как синонимы, хотя они означают принципиально разные вещи, я ввожу глоссарий-разграничитель прямо здесь, на старте. Первое понятие — **ассистент**. Ассистент — это ИИ-модель, которая отвечает на ваш запрос здесь и сейчас, в рамках одной сессии. Вы задали вопрос — она ответила. Вы попросили переписать текст — она переписала. Ассистент не проявляет инициативы, не обращается к внешним инструментам без вашей прямой команды, не помнит, что было вчера, и не способен запустить цепочку действий самостоятельно. Пример ассистента — чат-бот, которому вы пишете «перескажи этот абзац в три предложения», и он пересказывает. Второе понятие — **агент**. Агент отличается от ассистента тем, что получает цель, а не единичную команду, и сам решает, какие шаги предпринять для её достижения. Он может разбить задачу на подзадачи, обратиться к внешним инструментам — базе данных, календарю, поисковику, — проверить промежуточный результат и скорректировать план, если что-то пошло не так. Вы говорите агенту «собери аналитический обзор по конкурентам за неделю», и он сам решает, какие сайты открыть, какие данные извлечь, в каком формате оформить отчёт. Третье понятие — **мультимодальный агент**. Это агент, который способен воспринимать и обрабатывать не только текст, но и изображения, аудио, видео в любых комбинациях. Вы можете загрузить ему фотографию повреждённого товара, голосовую жалобу клиента и видеозапись инцидента — и он обработает все три потока данных в рамках одной задачи. Четвёртое понятие — **GUI-агент**, где GUI расшифровывается как Graphical User Interface, то есть графический интерфейс пользователя. GUI-агент работает не через программные интерфейсы и не через командную строку, а буквально «смотрит» на экран компьютера, распознаёт кнопки, поля ввода, меню и выполняет клики мышью и нажатия клавиш, имитируя действия человека. Это нужно для систем, у которых нет открытого программного интерфейса: например, для устаревших корпоративных программ или для сайтов, которые блокируют автоматический доступ. Запомните эту четвёрку: ассистент отвечает на одну команду, агент выполняет многошаговую цель, мультимодальный агент делает то же

самое с картинками, звуком и видео, а GUI-агент управляет компьютером через визуальный интерфейс. В тексте книги я буду использовать эти термины строго в таком значении, и если где-то в индустрии вы встретите другое употребление — ориентируйтесь на определения из этой книги.

И последнее, без чего тренажёр не будет тренажёром, а останется просто книгой. Это чек-лист прогресса, то есть система самопроверки, которую вы заполняете после каждой главы. Суть её в следующем: в конце каждой главы вы открываете отдельный файл или страницу в блокноте и фиксируете три вещи. Первое — какое именно ИИ-привычку вы внедрили в свою рабочую рутину после прочтения главы: например, «теперь все итоги планёрок транскрибирую через голосовой сервис, а не записываю вручную». Второе — сколько минут в день или в неделю экономит эта привычка по вашим ощущениям или по фактическому замеру. Третье — столкнулись ли вы с проблемой, которую не смогли решить самостоятельно, и если да, то какой именно. Этот чек-лист нужен не для меня и не для отчётности. Он нужен вам самим, потому что к двадцатой главе вы неизбежно забудете, что внедрили в третьей. А когда в итоговом модуле вам нужно будет собрать личную карту автоматизации и посчитать суммарный Индекс составной автоматизации, вы просто откроете свой чек-лист и увидите полную картину: какие процессы уже работают на ИИ, какие ещё не тронуты, где есть узкие места. Я рекомендую завести один файл и дописывать в него строку после каждой главы. Это займёт три минуты, но без этих трёх минут вы потеряете нить и не сможете объективно оценить свой прогресс. А вся философия этой книги построена на том, что прогресс измеряется не ощущениями, а цифрами.

Итак, правила обозначены. Вы сидите за компьютером, сервисы открыты, микрофон подключён, вы знаете три уровня сложности и понимаете, какие данные нельзя загружать в облако. Вы знаете разницу между ассистентом и агентом. У вас есть файл для чек-листа прогресса. Всё, что осталось, — перевернуть страницу и начать. Первая часть погрузит вас в фундамент: как устроена модель, почему она ошибается и как выстроить вокруг себя безопасный периметр. Без этого фундамента все последующие главы превратятся в набор рецептов, которые рассыпаются при первом нестандартном запросе. Поехали.

Часть I. Фундамент: Укрощение кремниевого разума

Глава 1. Как «думает» LLM-модель: Управление токенами и контекстом

Теория

Давайте начнём с фундамента, без которого всё остальное в этой книге будет держаться на честном слове. Вы наверняка слышали фразу «нейросеть генерирует текст», но задумывались ли вы, что именно происходит в тот момент, когда вы нажимаете кнопку отправки и на экране начинают появляться слова? Чтобы управлять инструментом осознанно, а не наугад, нужно понимать механику его работы. И эта механика, вопреки распространённому мифу о «машинном разуме», устроена удивительно логично и даже в чём-то примитивно.

Токен — это минимальная единица, с которой работает языковая модель. Не слово, не буква, не предложение, а именно токен. Токен — это фрагмент текста длиной от одного символа до нескольких букв, на которые модель разбивает входные данные на этапе подготовки. Грубо говоря, короткое слово типа «дом» или «кот» укладывается в один токен, длинное слово вроде «предпринимательство» распадётся на два-три токена, а пробелы и знаки препинания тоже считаются отдельными токенами. Почему это важно для вас лично? Потому что контекстное окно модели, стоимость запроса и скорость генерации измеряются именно в токенах, а не в символах или словах. Когда вы пишете промпт на полстраницы, вы расходуете не «полстраницы текста», а определённое количество токенов, и именно это количество определяет, уместится ли ваш запрос в память модели и сколько вы заплатите, если работаете через платный API. Для русского языка правило такое: один токен примерно равен четырём-пяти символам кириллицы или двум-трём буквам. То есть текст на тысячу слов на русском языке займёт ориентировочно тысячу двести — тысячу пятьсот токенов. Для английского это соотношение чуть экономнее: около семисот пятидесяти токенов на тысячу слов, потому что латиница кодируется плотнее. Запомните эти цифры, они пригодятся вам каждый раз, когда вы будете прикидывать, влезет ли документ в контекст выбранной модели.

Теперь о механизме, который делает токены не просто набором фрагментов, а осмысленный текст. Этот механизм называется **Attention**, что на русский переводится как «механизм внимания», и он является архитектурным ядром всех современных LLM, то есть Large Language Models, больших языковых моделей. Суть его в следующем: когда модель обрабатывает последовательность токенов, она не читает их строго по порядку, как это делает человек. Вместо этого каждый токен «оглядывается» на все остальные токены в контексте и присваивает им весовые коэффициенты значимости. Грубо говоря, если в вашем тексте есть фраза «Компания подписала контракт с поставщиком, потому что он предложил лучшую цену», модель должна понять, что местоимение «он» относится к «поставщику», а не к «компании». Механизм внимания решает именно эту задачу: он определяет, на какие предшествующие токены текущему токену нужно «обратить внимание» при генерации следующего. Чем больше параметров у модели, тем более сложные и дальние связи она способна уловить. И именно поэтому большая модель лучше понимает контекст, чем маленькая: у неё больше «голов внимания», то есть параллельных каналов, по которым она одновременно отслеживает разные типы связей в тексте.

Из механизма внимания напрямую вытекает понятие **контекстного окна**. Контекстное окно — это максимальное количество токенов, которое модель способна удерживать в «оперативной памяти» при обработке одного запроса. Всё, что не помещается в это окно, для модели

просто не существует: она этого не видит, не помнит и не учитывает. Современные модели предлагают контекстное окно от восьми тысяч токенов у компактных решений до ста двадцати восьми тысяч и даже миллиона токенов у флагманских систем. Восемь тысяч токенов — это примерно шесть тысяч слов, или двадцать-двадцать пять страниц текста. Сто двадцать восемь тысяч токенов — это уже полноценная книга объёмом триста-четыреста страниц. Почему это важно? Потому что если вы загрузите в модель документ на пятьдесят страниц, а контекстное окно рассчитано на двадцать, модель обработает только начало и конец, а середина окажется в так называемой «слепой зоне». Это явление исследователи назвали эффектом «lost in the middle», то есть «потерянный в середине»: модель уверенно отвечает на вопросы по первым и последним абзацам, но начинает путаться и галлюцинировать, когда вопрос касается информации из центральной части документа. Подробно мы разберём стратегии борьбы с этим эффектом в четвёртой главе, а пока запомните: чем больше документ, тем внимательнее нужно подходить к выбору модели и к способу подачи информации.

С контекстным окном тесно связаны два практических ограничения, которые бьют по вашему кошельку и по вашему терпению. Первое — **стоимость запросов**. Если вы работаете через облачный API, то есть через программный интерфейс, который позволяет отправлять запросы к модели из собственной программы или скрипта, вы платите за каждый токен: и за входной, то есть за ваш промпт и загруженные документы, и за выходной, то есть за сгенерированный ответ. Входные токены обычно стоят дешевле выходных, примерно в два-четыре раза, но если вы загружаете в контекст большой документ, входных токенов становится много, и счёт растёт. Второе ограничение — **скорость генерации**. Модель генерирует текст не целиком, а токен за токеном, последовательно. Каждый следующий токен зависит от предыдущего, поэтому ускорить процесс в разы невозможно. Типичная скорость для облачных сервисов составляет тридцать-восемьдесят токенов в секунду, а для локальных моделей на домашнем компьютере — пять-пятнадцать токенов в секунду. Если вам нужен ответ на две тысячи токенов, при скорости сорок токенов в секунду вы будете ждать около пятидесяти секунд. Это не критично для одного запроса, но если вы строите конвейер из десятков последовательных запросов, время складывается, и вы начинаете ценить каждую оптимизацию.

Следующий параметр, без понимания которого вы не сможете управлять качеством генерации, называется **Temperature**, то есть «температура». Это число от нуля до двух, которое определяет степень случайности при выборе следующего токена. Представьте, что модель на каждом шаге имеет список из нескольких подходящих продолжений фразы, каждое с определённой вероятностью. При температуре ноль модель всегда выбирает самый вероятный вариант, и ответ становится максимально предсказуемым, детерминированным, но при этом сухим и однообразным. При температуре около нуля целых семи — единицы, что является значением по умолчанию для большинства сервисов, модель иногда выбирает второй или третий по вероятности вариант, и текст становится живее, разнообразнее, но при этом чуть менее точным. При температуре выше полутора модель начинает активно выбирать маловероятные токены, и текст превращается в креативный поток, который может быть гениальным, а может быть полной бессмылицей. Практическое правило звучит так: для аналитических задач, извлечения данных, суммаризации документов ставьте температуру ноль — ноль целых три десятых, потому что вам нужна точность, а не красота. Для написания маркетинговых текстов, мозговых штурмов, генерации идей поднимайте до нуля семи — единицы. Для художественных экспериментов и нестандартных задач можно рискнуть с единицей и двумя, но будьте готовы к тому, что половину результатов придётся выбросить.

И последнее теоретическое понятие этой главы, которое станет сквозным для всей книги: **нативная мультимодальность**. Ранние языковые модели работали исключительно с текстом: вы вводили слова, получали слова. Современные модели, такие как Qwen 2.5, YandexGPT и другие флагманы, изначально обучаются на данных разных типов: на тексте, на изображе-

ниях, на аудио, а некоторые — и на видео. Слово «нативная» здесь означает, что модель не использует отдельные модули для распознавания картинок и звука с последующей передачей результата в текстовый блок. Вместо этого все типы данных кодируются в единое пространство токенов на самом раннем этапе обработки. Изображение разбивается на визуальные токены, аудио — на звуковые, и далее механизм внимания работает с ними точно так же, как с текстовыми. Для вас это означает, что вы можете загрузить в один запрос фотографию накладной, голосовую заметку с совещания и текстовый вопрос «Сравни данные на фото с тем, что сказал спикер», и модель обработает всё в рамках одного контекстного окна. Это радикально расширяет спектр задач, которые можно делегировать ИИ, и мы будем подробно разбирать мультимодальные сценарии начиная с четырнадцатой главы.

Реальный кейс

Теория теорией, но ничто не отрезвляет так, как чужой счёт на крупную сумму. В две тысячи двадцать третьем году одна крупная международная IT-корпорация, имя которой по понятным причинам я не называю, но масштаб которой вы можете представить по количеству сотрудников в сотни тысяч человек, решила обучить собственную языковую модель на внутренних данных. Задача звучала разумно: собрать корпоративную базу знаний, загрузить её через API в обучающий конвейер и получить модель, которая понимает специфику компании. Команда инженеров подготовила датасет объёмом двенадцать гигабайт и отправила его на обработку. Проблема была в том, что датасет не прошёл предварительную фильтрацию: в нём содержались тысячи дублирующихся документов, идентичные фрагменты из разных версий одних и тех же файлов, повторяющиеся абзацы из архивных переписок. Инженеры не применили то, что в этой книге мы будем называть **чанкингом**, то есть предварительным разбиением большого массива данных на логические фрагменты с удалением дубликатов и оценкой объёма в токенах перед отправкой. Результат оказался катастрофическим: за семьдесят два часа непрерывной обработки API нагенерировал счёт на сорок одну тысячу двести долларов. Не тысяч долларов — сорок одну тысячу. Причём обнаружилось это не сразу, а когда финансовый департамент получил инвойс и передал его в техническую команду с вопросом, что, собственно, произошло. Разбор показал, что из двенадцати гигабайт уникальной информации было от силы три с половиной, остальное — дубликаты, которые модель обрабатывала заново, потребляя токены и деньги. Урок здесь двойной. Первый: любой большой объём данных перед загрузкой в модель или в обучающий конвейер необходимо подвергать чанкингу, то есть разбиению на фрагменты с оценкой уникальности. Второй: перед массовым запросом через API всегда устанавливайте лимит расходов, то есть максимальную сумму, по достижении которой система автоматически останавливает генерацию. Оба этих правила мы будем применять на практике в каждой последующей главе, где речь пойдёт о работе с документами.

Инструкция

Теперь перейдём к тому, что вы будете делать каждый раз, садясь за работу с моделью. Я называю это чек-листом настройки параметров генерации, и он состоит из пяти пунктов, которые нужно пройтись перед отправкой первого запроса. Пункт первый: определите объём входных данных в токенах и убедитесь, что он укладывается в контекстное окно выбранной модели с запасом минимум в двадцать процентов на выходной ответ. Пункт второй: выберите температуру в зависимости от типа задачи — ноль для аналитики, ноль семь для текстов, выше единицы для креатива. Пункт третий: если работаете через API, установите дневной или месячный лимит расходов, чтобы избежать ситуации из кейса выше. Пункт четвёртый: решите, нужна ли вам потоковая генерация, то есть вывод токенов по мере их появления, или пакетная, когда вы получаете весь ответ целиком. Для интерактивной работы с чатом потоковая удобнее, для конвейеров и скриптов — пакетная. Пункт пятый: если задача мультимодальная, проверьте, что выбранный сервис поддерживает нужный тип входных данных — изображения, аудио или видео — нативно, а не через внешнюю надстройку.

А теперь о выборе модели. В этой книге мы работаем с конкретным набором инструментов, и я объясню логику выбора для каждого из них, чтобы вы не гадали, а понимали, почему для конкретной задачи берёте именно этот инструмент. **YandexGPT** — это семейство моделей от Яндекса, доступное через облачный сервис и через API. Его сильные стороны: отличная работа с русским языком, понимание российских реалий и контекста, интеграция с экосистемой Яндекса, включая поиск и корпоративные сервисы. Контекстное окно в последних версиях достигает ста двадцати восьми тысяч токенов. Лучше всего подходит для суммаризации документов на русском, написания деловых текстов, анализа данных из российских источников. **GigaChat** от Сбера — аналогичный облачный сервис с акцентом на корпоративный сегмент. Его преимущество в том, что Сбер предлагает развёртывание на собственной инфраструктуре заказчика, то есть в закрытом контуре, что критично для банков, госсектора и компаний с жёсткими требованиями к хранению данных. Контекстное окно сопоставимо с конкурентами, качество русского языка высокое. **Salute** — это линейка моделей от Сбера, ориентированная на голосовые сценарии и мультимодальные задачи. Если вам нужен не просто текстовый ответ, а работа с аудио или интеграция в голосового агента, Salute будет естественным выбором, потому что модель изначально обучалась на аудиоданных. **Илья** — модель от компании-разработчика «Сколково» и партнёров, заточенная под задачи программирования и технической документации. Если ваша работа связана с кодом, техническими спецификациями или архитектурными описаниями, Илья даст более точный результат, чем универсальные модели, потому что обучалась преимущественно на технических корпусах.

Перейдём к открытым моделям, которые можно развёртывать локально. **DeepSeek-V3** — это открытая модель от китайской лаборатории DeepSeek, доступная для свободного скачивания и запуска на собственных серверах. Её сильная сторона — рассуждение, то есть способность выстраивать цепочки логических шагов при решении аналитических задач. Она хорошо справляется с математикой, анализом данных, написанием структурированных отчётов. Контекстное окно — сто двадцать восемь тысяч токенов. **DeepSeek-R1** — специализированная версия той же лаборатории, оптимизированная именно для многошагового рассуждения. Если вам нужно, чтобы модель не просто дала ответ, а показала ход мысли, проверила сама себя и выдала обоснованный вывод, R1 предпочтительнее. **Qwen 2.5** и **Qwen 3** — линейка от китайской компании Alibaba. Qwen 2.5 — рабочая лошадка с отличным балансом скорости и качества, поддерживает мультимодальный вход, то есть понимает изображения. Qwen 3 — более свежая версия с улучшенным механизмом внимания и расширенным контекстным окном. Обе модели распространяются под открытой лицензией, что позволяет разворачивать их на своей инфраструктуре без лицензионных отчислений. **Yi-34B** — модель от лаборатории 01.AI с тридцатью четырьмя миллиардами параметров. Цифра в названии как раз и означает количество миллиардов параметров, и чем их больше, тем сложнее задачи модель способна решать. Yi-34B хорошо подходит для задач, требующих глубокого понимания контекста на нескольких языках одновременно. Наконец, **Ollama** — это не модель, а среда запуска. Это бесплатная программа, которую вы устанавливаете на свой компьютер и через которую запускаете любую из перечисленных открытых моделей локально, без интернета, без передачи данных в облако. Если ваши данные относятся к красному уровню защиты, о котором мы говорили во введении, Ollama или аналогичная среда LM Studio — единственный допустимый вариант.

Логика выбора, если свести её к простому правилу, звучит так. Для быстрых рабочих задач на русском языке в браузере — YandexGPT или GigaChat. Для голосовых сценариев — Salute или Yandex SpeechKit. Для кода и технической документации — Илья или DeepSeek-V3. Для сложной аналитики с многошаговым рассуждением — DeepSeek-R1 или Qwen 3. Для задач, где данные нельзя выводить за пределы компании, — любая открытая модель через Ollama или LM Studio. Для мультимодальных задач с изображениями — Qwen 2.5 или Qwen 3. Для задач с аудио — Salute или связка Yandex SpeechKit плюс текстовая модель. Не суще-

ствует одной идеальной модели на все случаи жизни, и тот, кто пытается работать только с одним сервисом, неизбежно упирается в потолок. Ваша задача как автономного сотрудника — собрать из двух-трёх инструментов связку, которая закрывает именно ваши процессы.

Практика. Базовый уровень. Упражнение «Токеновый аудит»

Настало время оторваться от чтения и сделать что-то руками. Откройте любой из перечисленных сервисов — YandexGPT, GigaChat или, если хотите работать без интернета, установите Ollama и загрузите модель Qwen 2.5. Ваша задача на ближайшие пятнадцать минут — провести токеновый аудит своего типичного рабочего дня. Возьмите три документа, с которыми вы работаете регулярно: это может быть служебная записка, фрагмент переписки, выдержка из отчёта или техническое задание. Скопируйте каждый текст в сервис-токенайзер, то есть в любой онлайн-инструмент для подсчёта токенов, либо вставьте текст в чат модели и попросите: «Сколько токенов в этом тексте?» Запишите результат для каждого документа. Теперь оцените: если бы вам нужно было загрузить все три документа одновременно в один запрос к модели, хватило бы контекстного окна? А если добавить к ним ваш вопрос и ожидаемый ответ? Это и есть аудит: вы впервые посмотрели на свои рабочие материалы не как на «страницы текста», а как на токены, то есть на ресурс, который расходуется и стоит денег. Результат упражнения запишите в чек-лист прогресса: какие документы вы брали, сколько токенов каждый занял, уложились ли они в контекстное окно и какой запас остался. Если вы обнаружили, что типичный документ занимает восемьдесят процентов контекстного окна модели, которую вы используете, это сигнал: либо выбирайте модель с окном побольше, либо учитесь разбивать документ на части, о чём мы подробно поговорим в четвёртой главе.

Практика. Продвинутый уровень. Упражнение «Выбери модель»

Если вы уже уверенно работаете с одним-двумя сервисами и хотите расширить инструментарий, это задание для вас. Вам понадобится тридцать-сорок минут и доступ минимум к двум моделям из перечисленных выше. Возьмите одну конкретную рабочую задачу, которую вам приходится решать регулярно: например, суммаризация протокола совещания, написание ответного письма клиенту, анализ конкурента или составление технического задания. Сформулируйте один и тот же промпт, то есть один и тот же запрос, и отправьте его последовательно в две разные модели. Например, в YandexGPT и в DeepSeek-V3, или в GigaChat и в Qwen 2.5. Сравните результаты по четырём критериям: точность фактов, полнота ответа, стиль и формат, время генерации. Запишите наблюдения в чек-лист прогресса. Теперь измените температуру: в одной модели поставьте ноль, в другой ноль семь, и отправьте тот же промпт. Посмотрите, как изменился результат. Вы наглядно увидите, что температура ноль даёт сухой, но точный ответ, а ноль семь — более живой, но с риском мелких неточностей. Цель упражнения не в том, чтобы найти «лучшую модель», а в том, чтобы вы почувствовали разницу и научились осознанно выбирать инструмент под задачу, а не хватать первый попавшийся. К концу этого упражнения у вас в чек-листе прогресса должна появиться запись: «Для задачи X лучше подходит модель Y с температурой Z, потому что…» Именно так формируется навык автономного сотрудника: не запоминание рецептов, а понимание логики выбора.

На этом первая глава завершена. Вы узнали, что модель работает с токенами, а не со словами; что механизм внимания определяет качество понимания контекста; что контекстное окно ограничивает объём данных в одном запросе; что температура управляет балансом между точностью и креативностью; что нативная мультимодальность позволяет загружать в модель картинки, звук и видео наравне с текстом. Вы увидели, к чему приводит пренебрежение чанкингом, и научились выбирать модель под конкретную задачу. В следующей главе мы разберём обратную сторону медали: что происходит, когда модель уверенно выдаёт вам неправду, как эту неправду распознать и какие механизмы защиты выстроить, чтобы никогда не попасть в ситуацию адвоката, который подал в суд апелляцию с вымышленными прецедентами.

Глава 2. Борьба с галлюцинациями и архитектурные лимиты моделей

Теория

В первой главе мы разобрались, что языковая модель не «понимает» текст в человеческом смысле, а предсказывает следующий токен на основе статистических вероятностей. Из этого фундаментального свойства архитектуры вытекает главная проблема, с которой вы будете сталкиваться ежедневно: **галлюцинации**. Галлюцинация в контексте искусственного интеллекта — это уверенно сгенерированный, грамматически безупречный, но фактически недостоверный или полностью вымышленный ответ. Почему это происходит? Потому что модель не имеет доступа к базе истин, у неё есть только карта вероятностей. Когда вы задаёте вопрос, на который у модели нет точного ответа в её тренировочных данных, она не может сказать «я не знаю» по умолчанию — её математическая задача заключается в том, чтобы выдать наиболее правдоподобное продолжение диалога. В результате она начинает экстраполировать, склеивая реальные факты с вымышленными именами, датами или цитатами. Понимание этой природы критически важно: вы должны относиться к любому сгенерированному тексту не как к истине, а как к черновику, требующему верификации.

Чтобы минимизировать риск галлюцинаций, нужно учитывать **архитектурный ландшафт** современных моделей и их специализацию. Отечественные решения, такие как YandexGPT и GigaChat, обучались на массивах, включающих российское законодательство, локальные бизнес-реалии и отечественные энциклопедические данные. Если вам нужно написать претензию по закону о защите прав потребителей или проанализировать рыночный отчёт по российскому ритейлу, эти модели будут галлюцинировать значительно реже, чем их западные или азиатские аналоги, просто потому что их «карта вероятностей» точнее отражает вашу реальность. В то же время открытые модели вроде DeepSeek-V3 или Qwen 3 демонстрируют выдающиеся результаты в структурном мышлении, программировании и математике, поскольку их разработчики делали упор на логику и рассуждения. Если вы попросите YandexGPT написать сложный скрипт на Python, а DeepSeek составить исковое заявление в арбитражный суд Москвы, обе модели с высокой долей вероятности допустят фактические или логические ошибки именно из-за выхода за пределы своей сильной стороны. Выбор правильного инструмента под конкретную задачу — это первый и самый дешёвый способ защиты от нейросетевой лжи.

Особую опасность представляют **мультимодальные галлюцинации**, то есть искажения в работе с изображениями, аудио и видео. Почему они опаснее текстовых? Дело в человеческой психологии и так называемом сенсорном доверии. Когда мы читаем текст, сгенерированный машиной, наш мозг по умолчанию включает критический фильтр: мы подсознательно допускаем, что автор мог ошибиться. Но когда мы видим фотографию или слышим аудиозапись, эволюция заставляет нас доверять органам чувств гораздо сильнее. Если мультимодальный агент анализирует снимок производственного брака и «дорисовывает» в своём описании трещину, которой на самом деле нет, или транскрибирует зашумлённую запись совещания, вставляя в протокол слово, которое никто не произносил, руководитель с гораздо большей вероятностью подпишет такой документ не глядя. Мультимодальная модель может уверенно описать несуществующий объект на картинке или приписать спикеру интонацию и слова, которых не было, просто потому что статистически в подобном контексте они часто встречаются. Именно поэтому работа с любыми нетекстовыми данными требует ещё более жёстких протоколов проверки, о которых мы поговорим в блоке инструкций.

Реальный кейс

Чтобы вы не думали, что галлюцинации — это лишь мелкие помарки в невинных эссе, давайте разберём прецедент, который заставил юридическое сообщество всего мира в панике переписывать внутренние регламенты. В две тысячи двадцать третьем году американский адвокат **Стивен Шварц** готовил апелляционную жалобу в федеральный суд. Чтобы усилить позицию клиента, он попросил популярный ИИ-чат-бот найти судебные прецеденты, подтверждающие его аргументы. Модель выдала шесть идеальных, кристально логичных прецедентов с указанием номеров дел, фамилий судей, дат и даже развёрнутыми цитатами из решений. **Стивен Шварц** включил их в официальный документ и подал в суд. Когда судья попросил предоставить копии этих решений, адвокат снова обратился к ИИ, и тот сгенерировал тексты самих судебных актов, которые выглядели абсолютно аутентично. Проблема выяснилась только тогда, когда клерки суда начали искать дела в базе и не нашли ни одного из них. Все шесть прецедентов, все цитаты и все номера дел были стопроцентной галлюцинацией. Суд назначил адвокату персональный штраф, опубликовал решение в открытом доступе с жёсткой критикой, а сам юрист столкнулся с дисциплинарными разбирательствами и потерей части клиентской базы.

Этот случай стал триггером: на текущий момент зафиксировано более четырнадцати аналогичных инцидентов в разных юрисдикциях, где юристы, учёные и корпоративные секретари слепо доверяли сгенерированным ссылкам на научные статьи или нормативные акты. Урок здесь предельно жёсткий: модель не знает, что она лжёт. Для неё вымышленный судебный прецедент и реальный закон о налогах — это просто последовательности токенов, которые статистически хорошо сочетаются со словом «апелляция». В корпоративной среде это означает, что любой документ, выходящий из-под пера ИИ-ассистента и имеющий юридические, финансовые или репутационные последствия, должен проходить через процедуру, которую я называю презумпцией виновности. Мы изначально считаем любой факт, цифру, ссылку или цитату вымышленными, пока не доказано обратное. Если вы не готовы потратить время на доказательство, вы не имеете права использовать ИИ для этой задачи.

Инструкция

Чтобы внедрить презумпцию виновности в ежедневную работу без паралича анализа, я разработал алгоритм верификации, который называю **Тройное сито**. Это последовательность из трёх фильтров, через которые должен проходить любой значимый результат генерации. Первое сито — это сито внутренней непротиворечивости. Вы читаете сгенерированный текст и ищете логические конфликты внутри самого документа: не утверждает ли модель в первом абзаце, что рынок растёт, а в третьем делает вывод о необходимости сокращения штата из-за падения спроса? Второе сито — это сито внешнего источника. Каждый конкретный факт, дату, имя собственное или номер закона вы должны иметь возможность подтвердить через независимый канал: поисковую систему, внутреннюю базу знаний компании или официальный реестр. Если модель дала ссылку на статью или закон, вы обязаны кликнуть по ней или скопировать название и найти первоисточник вручную, потому что ИИ отлично умеет генерировать правдоподобные, но мёртвые URL-адреса. Третье сито — это сито здравого смысла, или проверка на абсурдность. Вы задаёте себе вопрос: «Если бы этот факт был правдой, знал бы я об этом из новостей или отраслевых каналов?». Часто модель генерирует сенсационные, но физически невозможные события, и простая опора на ваш профессиональный опыт позволяет отсечь их за секунды.

Для работы с изображениями, аудио и видео к этому алгоритму добавляется **четвёртое сито**, которое я называю кросс-модальной сверкой. Суть его в том, что вы никогда не доверяете текстовому описанию, которое модель сделала на основе вложенного файла, без визуального или аудио-контроля. Если ИИ проанализировал фотографию договора и написал, что на ней стоит подпись генерального директора и печать, вы обязаны открыть изображение и убедиться, что модель не перепутала кляксу с печатью, а закорючку с подписью. Если модель транскрибировала аудиозапись и выделила ключевые тезисы спикера, вы должны прослушать оригинал

хотя бы в тех фрагментах, где модель утверждает, что спикер сделал критически важное заявление. Критерий допустимости ИИ без тотальной ручной проверки формулируется так: мы можем делегировать модели генерацию черновиков, идей, структуры и переводов, но финальное утверждение фактов, сумм и юридических трактовок всегда остаётся за человеком.

Практика. Базовый уровень. Упражнение «Поймай лжеца»

Ваша задача на ближайшие пятнадцать минут — намеренно спровоцировать модель на галлюцинацию, чтобы увидеть, как именно она это делает, и перестать её бояться. Откройте любой знакомый вам сервис и выберите узкую тему, в которой вы хорошо разбираетесь: это может быть специфика вашей профессии, биография малоизвестного исторического деятеля или технические характеристики редкого оборудования, с которым вы работаете. Сформулируйте запрос так, чтобы он содержал ложную предпосылку. Например, напишите: «Напиши подробный отчёт о том, как в две тысячи двадцать первом году компания Илон Маска купила завод по производству подводных лодок в Сибири, и укажи три главных условия сделки». Или попросите модель написать биографию вашего вымышленного коллеги, подставив реальную компанию. Посмотрите, как уверенно модель начнёт генерировать детали, выдумывать цифры, даты и имена. Запишите в свой чек-лист прогресса, на каком именно абзаце вы поймали модель на лжи и какие маркеры неуверенности или, наоборот, излишней детализации она использовала, чтобы скрыть нехватку данных. Это упражнение навсегда отучит вас принимать первый же ответ чат-бота за чистую монету.

Практика. Продвинутый уровень. Задание «Фактчек-конвейер»

Если вы хотите использовать ИИ для аналитической работы, вам нужно научиться строить конвейеры проверки. Задание займёт около сорока минут. Возьмите любую отраслевую статью или отчёт, скопируйте текст и загрузите его в модель с промптом: «Извлеки из этого текста пять главных фактических утверждений, содержащих цифры, даты или имена собственные, и представь их в виде нумерованного списка». Получив список, скопируйте каждое утверждение и отправьте его новым запросом с инструкцией: «Проверь этот факт, используя свои знания. Если он противоречит общедоступным данным или кажется сомнительным, укажи на это и объясни почему». Вы увидите, как модель начинает выступать в роли критика по отношению к собственному предыдущему выводу или к чужому тексту. В чек-лист прогресса запишите, сколько из пяти фактов модель успешно подвергла сомнению, и сделайте вывод о том, можно ли использовать такую двухшаговую схему для первичной фильтрации входящей информации в вашей ежедневной работе.

Практика. Экспертный уровень. Задание «Мультимодальный фактчек»

Это задание для тех, кто работает с визуальными данными и хочет настроить четвёртое сито верификации. Вам потребуется модель с поддержкой нативного мультимодального ввода, например Qwen 2.5 или YandexGPT. Найдите или сделайте фотографию сложного объекта: это может быть приборная панель автомобиля, график из финансового отчёта, схема помещения или витрина магазина. Загрузите это изображение в модель и попросите: «Детально опиши всё, что ты видишь на этом изображении, перечисли все надписи, цифры и объекты». Получив ответ, загрузите это же изображение во второй раз, но теперь добавьте к нему текстовый промпт: «Я прилагаю описание этого изображения. Найди три расхождения между тем, что ты видишь на фото, и тем, что написано в этом описании. Укажи на галлюцинации в тексте». В качестве описания вы можете подсунуть модели текст, который она сама сгенерировала на первом шаге, или намеренно искажённый вами текст. Цель упражнения — научиться видеть, где именно зрительный модуль модели расходится с её языковым модулем, и понять, как формулировать промпты для поиска визуальных несоответствий. Результат и выводы обязательно зафиксируйте в чек-листе прогресса.

На этом вторая глава завершена. Мы разобрали природу нейросетевой лжи, посмотрели на реальные последствия слепого доверия к алгоритмам и освоили алгоритм Тройного сита для

защиты от галлюцинаций. В следующей главе мы перейдём от защиты от ошибок к защите от утечек: вы узнаете, как выстроить личный и корпоративный ИИ-периметр, чтобы ваши коммерческие тайны и персональные данные никогда не оказались на серверах чужих компаний.

Глава 3. Создание личного ИИ-периметра и безопасность данных

Теория

Начнём с концепции периметра, потому что без понимания его границ любое использование нейросетей превращается в игру в русскую рулетку с корпоративными данными. Периметр — это невидимая граница, за пределами которой ваша информация становится доступной третьим лицам и выходит из-под вашего юридического и технического контроля. Почему это важно? Потому что в момент отправки запроса в публичный облачный сервис вы фактически передаёте коммерческую тайну или персональные данные на серверы чужой компании, подчиняясь их пользовательскому соглашению. Как это работает на практике? Существует два базовых состояния инфраструктуры: **открытый контур** и закрытый контур. Открытый контур — это любые публичные облачные сервисы, где данные обрабатываются на оборудовании провайдера и могут теоретически использоваться для дообучения глобальных моделей. Закрытый контур — это инфраструктура, над которой вы имеете полный физический или юридический контроль, и данные никогда не покидают её пределы. Для читателя это означает простое правило выбора: если данные критичны для бизнеса или личной безопасности, мы работаем исключительно в закрытом контуре.

Далее классифицируем данные, опираясь на требования Федерального закона номер сто пятьдесят два «О персональных данных» и законодательства о коммерческой тайне. Мы уже упоминали три цвета во введении, теперь разберём их механику. Зелёный уровень — это обезличенные тексты, открытые статьи, черновики без реквизитов и общеизвестные факты. Их можно без опасений загружать в любые облачные SaaS-решения, где SaaS расшифровывается как Software as a Service, то есть программное обеспечение как услуга, когда вы просто заходите на сайт и пользуетесь сервисом через браузер. Жёлтый уровень — это внутренние регламенты, деловая переписка без персональных данных клиентов, обезличенная аналитика. Такие данные можно передавать в корпоративные облака по модели IaaS, или Infrastructure as a Service, инфраструктура как услуга, где вы арендуете изолированные серверные мощности у отечественного провайдера, либо обрабатывать в локальных системах. Красный уровень — это персональные данные сотрудников и клиентов, финансовые реквизиты, пароли и проприетарный код. Согласно сто пятьдесят второму закону, обработка таких данных требует письменного согласия субъектов и строгого хранения на территории Российской Федерации, а передача в публичные нейросети является прямым нарушением, влекущим миллионные штрафы. Особое внимание уделяем аудио и видео: голос и изображение лица закон относит к **биометрическим данным**, если они используются для идентификации личности. Следовательно, запись корпоративного созвона, где обсуждаются зарплаты, или видео с камер наблюдения в офисе — это красный уровень по умолчанию, требующий максимального уровня защиты.

Теперь о моделях развёртывания, то есть о том, где физически находится «мозг» вашего ИИ и как это влияет на безопасность. Первая модель — облачные SaaS, о которых мы сказали выше: это быстро и дёшево, но данные уходят провайдеру. Вторая — IaaS, когда вы арендуете защищённый сервер у российского облачного провайдера и развёртываете там свою модель; это дороже, но даёт юридическую изоляцию и соответствие требованиям регуляторов. Третья — on-premise, или локальное развёртывание на собственных серверах компании в защищённом периметре; это максимальная безопасность, но требует штата инженеров для поддержки железа. Четвёртая — **on-device AI**, то есть ИИ, работающий непосредственно на вашем ноутбуке или смартфоне без выхода в интернет. Для автономного сотрудника on-device AI через локальные среды является главным и самым надёжным инструментом работы с красными данными, потому что в этой архитектуре утечка физически невозможна.

Реальный кейс

Чтобы понять цену ошибки, перенесёмся в две тысячи двадцать третий год, когда произошёл хрестоматийный инцидент в крупной азиатской технологической корпорации, производящей смартфоны и полупроводники. Три инженера из подразделения по разработке микросхем столкнулись со сложным багом в проприетарном коде и, желая сэкономить время, скопировали фрагменты исходного кода и загрузили их в публичный облачный чат-бот с просьбой найти ошибку. Через несколько недель служба безопасности корпорации обнаружила катастрофическую утечку: условия пользовательского соглашения публичного сервиса предполагали, что загружаемые данные могут использоваться для улучшения алгоритмов, и фактически **коммерческая тайна** компании оказалась в тренировочном датасете глобальной нейросети, доступной миллионам пользователей. Реакция была мгновенной и жёсткой: корпорация полностью запретила использование любых генеративных ИИ-сервисов на корпоративных устройствах, заблокировала доступ к ним на уровне сетевых экранов и инвестировала десятки миллионов долларов в разработку собственной локальной модели на базе открытых архитектур, которая работала исключительно во внутреннем закрытом контуре. Урок для нас предельно ясен: удобство облачного сервиса никогда не должно перевешивать риски утечки интеллектуальной собственности, а единственной альтернативой тотальному запрету ИИ в компании является создание грамотного локального периметра.

Инструкция

Перейдём к тому, как самостоятельно выстроить этот периметр без привлечения дорогих консультантов и потери месяцев на согласования. Вам потребуется установить локальную среду запуска, и я рекомендую два проверенных инструмента: Ollama и LM Studio. Ollama — это легковесная программа, которая работает через командную строку, то есть текстовый интерфейс управления, и идеально подходит для быстрой загрузки и запуска открытых моделей прямо на вашем компьютере. LM Studio — это приложение с графическим интерфейсом, где вы можете визуально выбирать модели, настраивать параметры и общаться с ними в привычном окне чата. Как это работает на практике? Вы скачиваете установочный файл с официального сайта, запускаете его, выбираете в каталоге нужную модель, например Qwen 2.5, и нажимаете кнопку загрузки. После скачивания, которое может занять от десяти минут до пары часов в зависимости от размера модели и скорости интернета, вы отключаете компьютер от сети и продолжаете работать. Это и есть on-device AI в чистом виде: ваши данные **физически не могут** покинуть жёсткий диск, потому что интернет-соединение разорвано, и никакие хакеры или корпоративные шпионы не смогут перехватить ваш запрос к локальной нейросети.

Чтобы принять решение, какую инфраструктуру выбрать для конкретной задачи, используйте матрицу решений, которую мы проговорим связным текстом. Если задача разовая, данные зелёные и нужен быстрый результат — выбирайте облачный SaaS. Если задача регулярная, данные жёлтые и требуется интеграция с корпоративной базой — арендуйте IaaS у отечественного провайдера и развёртывайте там модель через API. Если данные красные, содержат персональные данные или коммерческую тайну — используйте исключительно on-premise серверы компании или on-device AI на вашем личном рабочем ноутбуке. Для оценки финансовой целесообразности применяется шаблон расчёта TCO, или Total Cost of Ownership, **совокупной стоимости владения**. При расчёте TCO для локальной модели вы суммируете стоимость железа, то есть видеокарт и оперативной памяти, затраты на электроэнергию, зарплату инженера, который будет её обслуживать, и стоимость платной коммерческой лицензии, если модель не полностью открыта. Сравнивая эту сумму с ежемесячными платежами за облачный API, вы обычно обнаруживаете, что локальное развёртывание окупается примерно через восемь-двенадцать месяцев при условии ежедневного использования, зато после этой точки ваши расходы стремятся к нулю, а безопасность становится абсолютной. Читателю нужно

прямо сейчас решить, готов ли он инвестировать время в установку локальной среды, чтобы навсегда закрыть вопрос с утечками.

Практика. Базовый уровень. Лабораторная работа «Полевое развёртывание»

Ваша задача на ближайшие тридцать минут — создать собственный изолированный контур и убедиться в его работоспособности. Скачайте и установите LM Studio или Ollama на свой рабочий компьютер. Найдите в каталоге модель Qwen 2.5 или любую другую доступную модель размером до восьми миллиардов параметров, чтобы она комфортно уместилась в оперативную память обычного ноутбука. Загрузите её. Когда загрузка завершится, физически отключите компьютер от интернета: выдерните кабель Ethernet или выключите Wi-Fi. Откройте локальный чат, загрузите в него любой красный документ, например, черновик личного дневника или фрагмент финансовой таблицы с реальными цифрами, и попросите модель проанализировать данные. Вы воочию убедитесь, что ИИ отлично работает без подключения к сети, и почувствуете то самое спокойствие, которое даёт **полный контроль** над информацией. Запишите в чек-лист прогресса факт успешного офлайн-развёртывания и размер модели, которую вы смогли запустить на своём железе.

Практика. Экспертный уровень. Задание «Аудит периметра и ТЗ на инфраструктуру»

Если вы руководитель или ИТ-специалист, это задание займёт у вас около часа и потребует системного мышления. Проведите аудит текущего ИИ-периметра вашего отдела: опросите коллег и выясните, какие именно облачные сервисы они используют для работы с текстами, таблицами и презентациями. Определите, попадают ли в эти сервисы жёлтые или красные данные, и оцените масштаб потенциальной уязвимости. На основе этого аудита составьте техническое задание, или ТЗ, для ИТ-департамента на развёртывание локальной модели. В ТЗ обязательно укажите требуемый объём контекстного окна, список открытых моделей, которые необходимо протестировать, требования к изоляции сети и расчёт ТСО на первые двенадцать месяцев. Это задание превратит вас из пассивного пользователя в **архитектора корпоративной безопасности**, и именно такие специалисты, способные bridging the gap между бизнес-задачами и ИТ-инфраструктурой, сейчас ценятся на рынке труда выше всего. Результат аудита и структуру ТЗ зафиксируйте в чек-листе прогресса.

На этом третья глава завершена. Мы выстроили непробиваемый периметр, научились классифицировать данные по цвету и освоили локальное развёртывание моделей. В следующей главе мы перейдём к работе с тяжёлыми артефактами: вы узнаете, как заставить нейросеть читать многостраничные PDF-файлы, анализировать сложные таблицы в Excel и не терять суть в середине огромных документов.

Глава 4. Работа с вложениями, таблицами и документами

Теория

В первой главе мы разобрались, что модель работает с токенами и что контекстное окно ограничивает объём данных, которые она способна удержать в «оперативной памяти» за один запрос. Теперь давайте посмотрим, что происходит, когда вы загружаете в модель не просто текст, а файл: PDF-документ, таблицу Excel, скан договора или фотографию накладной. Для вас как пользователя всё выглядит просто — вы перетаскиваете файл в окно чата и задаёте вопрос. Но внутри системы происходит сложная цепочка преобразований, и понимание этой цепочки критически важно, потому что именно на этапе конвертации файла в токены теряется информация, искажается структура и возникают ошибки, которые потом невозможно исправить никаким промптом.

Когда вы загружаете текстовый документ в форматах DOCX или TXT, система извлекает из него чистый текст, разбивает на токены и помещает в контекстное окно модели. Это самый простой и предсказуемый сценарий. С PDF всё сложнее: PDF — это не текстовый формат, а формат вёрстки, то есть он хранит не последовательность слов, а координаты каждого символа на странице. Если PDF создан из текстового редактора, система может извлечь текст относительно чисто. Но если PDF является сканом бумажного документа, извлечь текст стандартными средствами невозможно — нужно сначала распознать изображение, то есть применить технологию OCR, которая расшифровывается как Optical Character Recognition, оптическое распознавание символов. Таблицы Excel представляют отдельную головную боль: модель не понимает ячейки, строки и столбцы как визуальную сетку, поэтому система либо конвертирует таблицу в текстовое представление, где каждая строка записывается через разделитель, либо вообще теряет структуру, превращая аккуратную матрицу данных в бессвязный поток слов. Понимание этих ограничений — первый шаг к тому, чтобы перестать удивляться, почему модель «не видит» данные на третьей странице вашего пятидесятистраничного отчёта.

Теперь о явлении, которое исследователи назвали эффектом **lost in the middle**, то есть «потерянный в середине». Суть его в следующем: когда в контекстное окно попадает большой объём текста, модель уверенно обрабатывает начало и конец документа, но информация из центральной части обрабатывается с заметно меньшим вниманием. Это не баг и не ошибка конкретной модели — это следствие архитектуры механизма внимания, о котором мы говорили в первой главе. Когда последовательность токенов достигает десятков тысяч, вычислительные ресурсы распределяются неравномерно, и средние фрагменты получают меньший вес при генерации ответа. Практический вывод прост: если вы загрузите в модель договор на сорок страниц и спросите о пункте, который находится на двадцать третьей странице, с высокой вероятностью модель либо ответит размыто, либо сосшлётся на соседние пункты, либо вообще придумает содержание. Это не значит, что с большими документами работать нельзя — это значит, что нужно использовать правильные стратегии подачи информации.

Первая и самая важная стратегия называется **чанкинг**, то есть предварительное разбиение большого документа на логические фрагменты. Слово произошло от английского chunk, кусок, и смысл операции в том, что вы не загружаете весь документ целиком, а разрезаете его на осмысленные части: по разделам, по главам, по страницам или по логическим блокам. Каждый чанк обрабатывается моделью отдельно, а результаты потом объединяются. Почему это работает? Потому что каждый отдельный фрагмент полностью помещается в фокус внимания модели, и эффект «потерянного в середине» просто не возникает. Оптимальный размер чанка для большинства задач составляет от пятисот до двух тысяч токенов: меньше пятисот — модель теряет контекст и не понимает, о чём речь; больше двух тысяч — начинает размываться фокус. При чанкинге критически важно сохранять перекрытие, то есть включать последние

два-три предложения предыдущего фрагмента в начало следующего, чтобы модель не теряла нить повествования на стыках.

Вторая стратегия — **каскадная суммаризация**, то есть последовательное сжатие информации в несколько этапов. Представьте, что у вас двенадцать квартальных отчётов конкурентов, каждый по тридцать страниц. Загрузить все триста шестьдесят страниц в один запрос невозможно даже в модель с контекстным окном на сто двадцать восемь тысяч токенов, потому что помимо входных данных вам нужно место для выходного ответа. Каскадная суммаризация решает это элегантно: на первом этапе вы загружаете каждый отчёт по отдельности и просите модель извлечь ключевые метрики, выводы и цифры, сжимая тридцать страниц до одной. На втором этапе вы берёте двенадцать получившихся саммари и загружаете их вместе, прося модель провести сравнительный анализ. На третьем этапе, если нужно, вы просите сформулировать финальные рекомендации. Каждый этап уменьшает объём данных в двадцать-тридцать раз, и модель на каждом шаге работает с компактным, хорошо структурированным входом.

Третья стратегия — **RAG**, что расшифровывается как Retrieval-Augmented Generation, то есть генерация с дополненным поиском. Суть подхода в том, что вместо загрузки всего документа в контекст вы создаёте внешний индекс, где каждый фрагмент текста привязан к своему векторному представлению, то есть к числовому коду, отражающему смысл этого фрагмента. Когда пользователь задаёт вопрос, система сначала ищет в индексе наиболее релевантные фрагменты по смысловому сходству, а затем подаёт только эти фрагменты модели вместе с вопросом. Модель отвечает строго на основе найденных фрагментов, а не на основе своих внутренних знаний, что резко снижает риск галлюцинаций. RAG особенно эффективен для корпоративных баз знаний, где тысячи документов и ни один сотрудник не помнит, в каком именно регламенте записана нужная процедура.

Из классического текстового RAG логически вырастает **мультимодальный RAG**, где в индексе хранятся не только текстовые фрагменты, но и изображения, таблицы, схемы и даже аудиозаписи. Представьте, что у вас база из пятисот фотографий оборудования сорока семи филиалов, и вам нужно найти все снимки, где видна конкретная трещина определённого типа. Мультимодальный RAG позволяет задать запрос текстом или даже загрузить пример изображения, и система найдёт все визуально похожие фрагменты. Для вашей ежедневной работы это означает, что вы можете строить поисковые системы не только по тексту, но и по картинкам, схемам и сканам, что открывает колоссальные возможности для аналитики.

Реальный кейс

Чтобы показать, как эти стратегии работают в связке и какой экономический эффект дают, расскажу историю финансового директора группы компаний **«Синергия»**, которого назовём для удобства **Андрей Викторович**. Задача, которая перед ним стояла, звучала так: за три рабочих дня подготовить сравнительный анализ двенадцати квартальных отчётов конкурентов, каждый объёмом от двадцати пяти до тридцати пяти страниц, с извлечением ключевых финансовых метрик, динамикой выручки, маржинальностью по направлениям и стратегическими приоритетами. Традиционный подход предполагал, что два аналитика вручную читают все двенадцать отчётов, выписывают данные в единую таблицу, сверяют цифры, строят графики и пишут аналитическую записку. На это уходило ровно три рабочих дня, причём к концу третьего дня аналитики уставали, начинали путать колонки и пропускать примечания в сносках.

Андрей Викторович решил применить каскадный промптинг, то есть ту самую стратегию последовательного сжатия, которую мы разобрали выше. На первом этапе он загрузил каждый отчёт по отдельности в модель с контекстным окном на сто двадцать восемь тысяч токенов и промптом, в котором чётко указал: «Извлеки из этого квартального отчёта следующие данные: выручка по сегментам, EBITDA, чистая прибыль, капитальные затраты, ключевые риски, упомянутые в разделе обсуждения руководством, и три стратегических приоритета

на следующий квартал. Представь результат в виде структурированного текста с числовыми значениями». Каждый отчёт обрабатывался за три-четыре минуты, включая время на чтение и корректировку промпта. На весь первый этап ушло около сорока минут. На втором этапе он собрал двенадцать саммари, каждое объёмом около одной страницы, и загрузил их одним запросом с инструкцией провести сравнительный анализ, выявить тренды и аномалии. На третьем этапе попросил сформулировать выводы и рекомендации для совета директоров. Общее время: сорок минут вместо трёх дней. Точность извлечения цифр составила девяносто четыре процента, а оставшиеся шесть процентов Андрей Викторович проверил вручную, сверив с оригиналами. С тех пор эта процедура стала еженедельным конвейером, и аналитики освободились для задач, требующих стратегического мышления, а не механического копирования цифр из PDF в Excel.

Инструкция

Перейдём к конкретным методам обработки файлов, которые вы будете использовать каждый день. Начнём с PDF-документов. Если PDF создан из текстового редактора, то есть вы можете выделить текст мышкой и скопировать его, модель обработает его напрямую: просто загрузите файл в чат и задайте вопрос. Если PDF является сканом, то есть текстовый слой отсутствует, вам потребуется мультимодальный OCR. В этом случае вы загружаете изображение скана в модель с поддержкой визуального входа, например в Qwen 2.5 или YandexGPT с мультимодальным режимом, и формулируете запрос: «Распознай весь текст на этом изображении, сохрани структуру заголовков и нумерацию пунктов, выведи результат в виде чистого текста». Качество распознавания зависит от чёткости скана: если разрешение ниже ста пятидесяти точек на дюйм или текст сфотографирован под углом, модель начнёт путать буквы. Оптимальное разрешение для сканирования — триста точек на дюйм, формат сохранения — PNG без сжатия или TIFF. Если документ многостраничный, сканируйте по одной странице и обрабатывайте последовательно, а не пытайтесь загрузить весь PDF сканов одним файлом.

Теперь о таблицах Excel и файлах формата XLSX. Модель не умеет читать бинарный формат Excel напрямую, поэтому перед загрузкой вам нужно конвертировать таблицу в текстовое представление. Самый простой способ: откройте таблицу в редакторе, выделите нужные данные, скопируйте их и вставьте в промпт как обычный текст, разделяя столбцы символом табуляции или вертикальной чертой. Если таблица большая, более пятидесяти строк, разбейте её на логические блоки по десять-пятнадцать строк и обрабатывайте каждый блок отдельно, а результаты объединяйте. Для сложных таблиц с несколькими листами, формулами и условным форматированием лучший подход — экспортировать нужный лист в формат CSV, то есть Comma-Separated Values, значения, разделённые запятыми. CSV — это обычный текстовый файл, где каждая строка представляет собой запись, а столбцы разделены запятой или точкой с запятой. Модель отлично понимает CSV и может анализировать данные, находить аномалии, группировать и агрегировать значения.

Для файлов формата DOCX, то есть документов Microsoft Word, подход аналогичен текстовому PDF: система извлекает чистый текст, и модель работает с ним напрямую. Однако есть нюанс с таблицами и встроенными изображениями внутри DOCX. Если в документе есть таблица, при извлечении текста она может потерять структуру и превратиться в поток слов. В этом случае перед загрузкой скопируйте таблицу отдельно и вставьте её как CSV или как текст с разделителями. Если в документе есть диаграммы или схемы, сохраните их как изображения и загрузите через мультимодальный режим отдельно от текста.

Работа со сканами через мультимодальный OCR заслуживает отдельного внимания, потому что это один из самых частых сценариев в российском бизнесе: накладные, акты, счета-фактуры, договоры — всё это часто приходит в виде сканов или фотографий. Алгоритм работы следующий: сначала вы загружаете изображение в модель и просите распознать текст с сохранением структуры. Затем, если документ содержит таблицу, вы просите модель преобразовать

распознанные данные в формат CSV. И наконец, вы задаёте аналитический вопрос уже по извлечённым данным. Критически важно при работе со сканами документов, содержащих персональные данные или реквизиты, помнить о классификации из третьей главы: если на скане видны фамилии, телефоны, паспортные данные или номера счетов, это красный уровень, и обрабатывать такой документ можно только в локальном контуре через Ollama или LM Studio, без подключения к интернету.

Практика. Базовый уровень. Задание «Из текста в таблицу»

Ваша задача на ближайшие пятнадцать минут — взять любой текстовый документ, в котором данные представлены в виде связного повествования, и превратить их в структурированную таблицу. Это может быть годовой отчёт компании, статья с перечислением показателей или даже фрагмент технического задания, где требования описаны абзацами. Откройте модель, загрузите текст и сформулируйте промпт по следующей схеме: «Проанализируй приведённый ниже текст. Извлеки все числовые показатели, имена, даты и категории. Представь результат в виде таблицы с колонками: Параметр, Значение, Единица измерения, Источник в тексте. Если какой-то параметр не имеет числового значения, укажи в колонке Значение слово „не указано“». Скопируйте результат, вставьте его в Excel или в любой табличный редактор и убедитесь, что данные извлеклись корректно. Запишите в чек-лист прогресса, сколько параметров модель извлекла, сколько из них оказались верными при ручной сверке с оригиналом, и потребовалась ли корректировка промпта. Если точность извлечения ниже девяноста процентов, попробуйте добавить в промпт пример желаемого формата, то есть одну-две строки таблицы, заполненные вручную, чтобы модель поняла шаблон.

Практика. Продвинутый уровень. Задание «Сравнительный анализ»

Это задание займёт тридцать-сорок минут и потребует от вас применения каскадной суммаризации на практике. Найдите два-три документа на одну тему: это могут быть два отчёта разных компаний за один период, две версии технического задания на один проект или две статьи разных авторов на одну тему. Загрузите каждый документ по отдельности и попросите модель извлечь ключевые тезисы, цифры и выводы, сжав каждый документ до одной страницы саммари. Затем загрузите все саммари вместе и попросите провести сравнительный анализ: «Сравни представленные данные. Укажи, где значения совпадают, где расходятся и где один документ содержит информацию, отсутствующую в другом. Представь результат в виде связного текста с выделением расхождений». Результат сравните с оригиналами: действительно ли модель нашла все ключевые различия или упустила что-то важное? Запишите в чек-лист прогресса, на каком этапе каскада возникли потери информации и как вы их компенсировали. Это упражнение научит вас чувствовать, где именно в цепочке обработки данных нужно быть особенно внимательным.

Практика. Экспертный уровень. Задание «Мультимодальный RAG»

Это задание для тех, кто хочет освоить работу с визуальными данными в режиме поиска и извлечения. Вам потребуется сорок-шестьдесят минут и модель с поддержкой мультимодального входа. Соберите пять-семь изображений на одну тему: это могут быть фотографии одного типа оборудования с разных ракурсов, скриншоты разных версий одного интерфейса, сканы нескольких страниц одного документа или фотографии товаров на витрине. Загрузите изображения по одному и для каждого попросите модель составить подробное текстовое описание: что изображено, какие надписи присутствуют, какие объекты можно выделить. Сохраните все описания. Теперь загрузите все описания вместе и задайте вопрос: «На основе этих описаний найди три общих элемента, присутствующих на всех изображениях, и два элемента, которые встречаются только на одном из них». Это и есть упрощённая версия мультимодального RAG: вы создали текстовый индекс визуальных данных и теперь ищете по нему смыслы. Запишите в чек-лист прогресса, насколько точно модель нашла общие элементы и где возникли ошибки. Если вы работаете в корпоративной среде, подумайте, как масштабировать этот под-

ход на сотни изображений и как автоматизировать процесс через скрипт, который последовательно загружает файлы и собирает описания в единую базу.

На этом четвёртая глава завершена. Вы узнали, как модель превращает файлы в токены, почему большие документы теряют информацию в середине и как три стратегии — чанкинг, каскадная суммаризация и RAG — позволяют работать с документами любого объёма без потери качества. Вы освоили обработку PDF, таблиц и сканов, а также попробовали свои силы в мультимодальном поиске по изображениям. В следующей главе мы переходим к искусству составления запросов: вы узнаете, из каких пяти компонентов состоит идеальный промпт и почему замена одного слова в роли модели способна сократить количество доработок с четырёх итераций до одной.

Часть II. Технологии промптинга: От простых команд к каскадным системам

Глава 5. Золотой стандарт промпта: Пятикомпонентная матрица

Теория

В четырёх предыдущих главах мы разобрали, как модель работает с токенами, почему она галлюцинирует, как защитить данные и как подавать документы. Теперь настало время главного инструмента, который определяет качество любого вашего взаимодействия с нейросетью: промпта. Промпт, от английского prompt, то есть запрос или подсказка, — это текстовая инструкция, которую вы отправляете модели для получения результата. Звучит банально, но именно здесь совершается девяносто процентов ошибок, которые потом выливаются в часы переделок, разочарование и вывод «ИИ не работает». Проблема не в модели. Проблема в том, что большинство пользователей пишут запросы так, как будто разговаривают с всезнающим коллегой, который автоматически понимает контекст, цели и формат. Модель не понимает. Модель — это математический механизм, который выдаёт ровно то, что вы запросили, и если запрос размытый, ответ будет размытым.

Чтобы исключить размытость, я предлагаю вам **пятикомпонентную матрицу**, то есть структуру из пяти обязательных элементов, каждый из которых выполняет свою функцию и без которого промпт теряет эффективность. Первый компонент — Роль. Вы указываете модели, от лица какого специалиста она должна отвечать. Это не формальность и не игра в театр: роль задаёт вектор поиска в пространстве вероятностей. Когда вы пишете «Ты — стажёр-аналитик», модель подбирает наиболее вероятные продолжения из корпуса текстов, ассоциированных с начальным уровнем экспертизы. Когда вы пишете «Ты — аудитор с двадцатилетним стажем в Big Four», модель переключается на паттерны речи, аргументации и глубины анализа, характерные для текстов, написанных опытными профессионалами. Разница в качестве ответа при одинаковом задании может быть колоссальной, и мы увидим это в кейсе ниже.

Второй компонент — Контекст. Здесь вы даёте модели фоновую информацию, без которой невозможно принять правильное решение: кто ваш клиент, какая отрасль, какая стадия проекта, какие ограничения по бюджету или срокам. Контекст сужает пространство генерации и не позволяет модели уходить в общие рассуждения. Третий компонент — Задача. Это конкретное действие, которое модель должна совершить: проанализировать, написать, сравнить, извлечь, переформулировать. Задача должна быть одна и сформулирована глаголом в повелительном наклонении. Четвёртый компонент — Ограничения. Здесь вы указываете, чего делать нельзя: не использовать жаргон, не превышать определённый объём, не выдумывать факты, не давать рекомендаций вне указанной области. Ограничения работают как ограждения на дороге: они не говорят модели, куда ехать, но не дают ей съехать в кювет. Пятый компонент — Формат вывода. Вы чётко указываете, в каком виде хотите получить результат: связный текст на три абзаца, нумерованный список из пяти пунктов, таблица с определёнными колонками, письмо в официально-деловом стиле. Без формата модель сама решает, как оформить ответ, и в половине случаев этот формат вас не устроит.

Для мультимодальных запросов, где вы загружаете изображение, аудио или видео, к пяти компонентам добавляется шестой — **Модальность**. В этом компоненте вы явно указываете модели, какой тип данных она получает и что именно нужно из него извлечь. Например: «Ты получаешь фотографию приборной панели. Извлеки все числовые показания и укажи,

какие из них выходят за пределы нормы». Без явного указания модальности модель может начать описывать изображение общими словами, не фокусируясь на тех данных, которые вам нужны. Шестой компонент критически важен, потому что визуальные и аудио-данные содержат гораздо больше информации, чем текст, и без точного указания, на что смотреть, модель утонет в деталях.

Теперь о двух фундаментальных техниках промптинга, которые определяют, даёте ли вы модели примеры или нет. Первая техника называется **Zero-shot**, то есть «ноль примеров». Вы формулируете запрос и сразу просите результат, не показывая модели, как выглядит желаемый ответ. Это работает для простых, хорошо известных задач: суммаризация, перевод, перефразирование. Вторая техника — **Few-shot**, то есть «несколько примеров». Перед тем как дать модели основное задание, вы показываете ей два-три примера пар «вход → выход», чтобы она поняла шаблон. Few-shot критически важна, когда вам нужен нестандартный формат, специфический стиль или когда задача выходит за рамки типичных сценариев. Например, если вы хотите, чтобы модель извлекала данные из текста в определённом формате JSON с конкретными ключами, покажите ей два-три примера готового JSON на основе других текстов, и только потом давайте основной текст. Модель увидит паттерн и воспроизведёт его с точностью, недостижимой при Zero-shot.

Реальный кейс

Чтобы показать, как один-единственный компонент матрицы способен изменить экономику целого процесса, расскажу историю консалтинговой фирмы, специализирующейся на аудите бизнес-процессов для среднего производственного сектора. Команда из четырёх аналитиков готовила отчёты по результатам выездных проверок: каждый отчёт включал описание текущих процессов, выявленные узкие места, рекомендации по оптимизации и финансовую модель эффекта. Написание одного отчёта занимало в среднем два дня, причём значительная часть времени уходила на итерации доработки: первый черновик, правки старшего партнёра, второй черновик, ещё правки, третий черновик, финальная шлифовка. В среднем получалось четыре итерации, то есть четыре цикла «написал — получил замечания — переписал».

Руководитель аналитического отдела решил внедрить ИИ в процесс подготовки отчётов и начал с того, что написал промпт, в котором указал роль: «Ты — стажёр-аналитик, который готовит первый черновик отчёта». Модель выдавала текст, но он был поверхностным, без глубины анализа, без конкретных цифр, с общими рекомендациями в духе «необходимо оптимизировать процессы». Старший партнёр возвращал черновик с замечаниями, аналитик переписывал промпт, добавляя уточнения, и цикл повторялся. Четыре итерации, как и раньше, только теперь с участием нейросети.

Перелом произошёл, когда руководитель изменил один-единственный компонент матрицы — роль. Вместо «стажёр-аналитик» он написал: «Ты — **аудит-директор** с двадцатилетним стажем в крупнейших международных консалтинговых компаниях, специализирующийся на операционной эффективности производственных предприятий. Твой стиль: конкретный, основанный на цифрах, без общих фраз. Ты всегда указываешь количественный эффект каждой рекомендации». Результат изменился радикально. Модель начала выдавать текст с конкретными бенчмарками, ссылками на отраслевые нормы, расчётами эффекта в процентах и рублях. Объём доработок сократился с четырёх итераций до одной: старший партнёр вносил лишь косметические правки и уточнял пару цифр по специфике конкретного клиента. Экономия времени составила около шести часов на каждый отчёт, что при потоке в восемь-десять отчётов в месяц давало компании почти два дополнительных рабочих дня в месяц на каждого аналитика. Урок очевиден: роль — не декоративный элемент, а рычаг управления глубиной и качеством генерации.

Скрипт и шаблон

Теперь я дам вам универсальный шаблон, который вы будете использовать каждый день. Я называю его **Мета-Промпт**, потому что это промпт для создания промптов: вы подставляете свои данные в фиксированную структуру и получаете запрос, который работает с первого раза. Структура звучит так. Первая строка: «Ты — [РОЛЬ: должность, стаж, специализация]». Вторая строка: «Контекст: [описание ситуации, отрасли, клиента, стадии проекта]». Третья строка: «Задача: [конкретное действие глаголом в повелительном наклонении]». Четвёртая строка: «Ограничения: [что нельзя делать, какой объём, какие запреты]». Пятая строка: «Формат вывода: [структура, стиль, объём, формат документа]». Шестая строка, если запрос мультимодальный: «Модальность: [тип входных данных и что именно из них извлечь]». Седьмая строка, если используете Few-shot: «Примеры: [две-три пары вход-выход]». Вы можете сохранить этот шаблон как текстовый файл и каждый раз, садясь за работу с моделью, заполнять его как бланк. Это занимает тридцать секунд, но экономит часы переделок.

Чтобы проверить качество готового промпта перед отправкой, пройдите мысленный чеклист из пяти вопросов. Первый вопрос: понимает ли модель из моего промпта, кто она и от чьего лица говорит? Если нет — допишите роль. Второй вопрос: достаточно ли контекста, чтобы модель не уходила в абстракции? Если нет — добавьте детали о проекте, клиенте, отрасли. Третий вопрос: сформулирована ли задача одним глаголом в повелительном наклонении? Если задача размыта или содержит несколько действий — разбейте на подзадачи. Четвёртый вопрос: есть ли хотя бы одно ограничение? Если нет — добавьте запрет на выдумывание фактов или на превышение объёма. Пятый вопрос: указан ли формат вывода? Если нет — модель сама решит, как оформить ответ, и вы потратите время на переформатирование. Если хотя бы на один вопрос ответ «нет», промпт не готов к отправке.

Отдельно остановлюсь на технике Few-shot, потому что именно она чаще всего вызывает затруднения у новичков. Логика Few-shot проста: вы показываете модели два-три примера того, что хотите получить, и затем даёте основное задание. Примеры должны быть разнообразными, но релевантными: если вы хотите, чтобы модель извлекала имена и должности из текста, покажите ей два текста разного стиля, из которых имена и должности извлечены в нужном формате. Не нужно показывать десять примеров — двух-трёх достаточно, чтобы модель уловила паттерн. Перегружать промпт примерами вредно, потому что каждый пример расходует токены контекстного окна, и если примеров слишком много, для основного задания может не хватить места. Оптимальный баланс: два примера по три-пять строк каждый, затем основное задание.

Практика. Базовый уровень. Тренажёр «Few-shot»

Ваша задача на ближайшие пятнадцать минут — освоить технику Few-shot на конкретном примере. Откройте любую модель и возьмите простую задачу: извлечение контактных данных из текста. Сначала отправьте запрос в режиме Zero-shot: «Извлеки из следующего текста имя, должность и телефон: [вставьте любой абзац делового письма]». Посмотрите на результат. Теперь отправьте тот же запрос, но в режиме Few-shot: перед основным заданием добавьте два примера. Первый пример: текст делового письма, из которого извлечены имя, должность и телефон в формате «Имя: …, Должность: …, Телефон:…». Второй пример: другой текст, извлечённый в том же формате. Затем дайте основной текст. Сравните результаты Zero-shot и Few-shot: во втором случае формат будет точнее, структура чище, а вероятность ошибки ниже. Запишите в чек-лист прогресса, в чём именно Few-shot оказался лучше и для каких типов задач в вашей работе эта техника будет наиболее полезна.

Практика. Продвинутый уровень. Задание «Перевод задачи в промпт»

Это задание займёт тридцать минут и потребует от вас применить всю пятикомпонентную матрицу. Возьмите любую рабочую задачу, которую вам приходится выполнять регулярно: подготовка письма клиенту, написание резюме встречи, составление отчёта, анализ конкурента. Сформулируйте эту задачу обычным языком, как вы бы объяснили её коллеге. А теперь

переведите её в промпт по пятикомпонентной матрице: запишите роль, контекст, задачу, ограничения и формат вывода. Отправьте промпт модели и оцените результат. Если результат вас не устраивает, вернитесь к чек-листу из пяти вопросов и определите, какой компонент пропущен или сформулирован слабо. Перепишите промпт и отправьте снова. Повторяйте цикл до тех пор, пока не получите результат, который можно использовать без доработки. Запишите в чек-лист прогресса итоговый промпт, который сработал, и укажите, какой компонент оказался решающим для качества ответа. Именно так формируется навык автономного сотрудника: вы не ищете готовый рецепт, а конструируете запрос осознанно, понимая, зачем нужен каждый элемент.

На этом пятая глава завершена. Вы освоили пятикомпонентную матрицу промпта, поняли, почему роль определяет глубину ответа, научились использовать Few-shot для точного форматирования и получили универсальный шаблон, который будет работать в любой задаче. В следующей главе мы поднимемся на уровень выше: вы узнаете, как заставить модель рассуждать пошагово, как разбивать сложные задачи на цепочки запросов и как техника Tree-of-Thoughts позволяет модели самой генерировать и оценивать несколько вариантов решения, прежде чем выдать финальный ответ.

Глава 6. Программирование логики: Chain-of-Thought и Tree-of-Thoughts

Теория

В предыдущей главе мы освоили пятикомпонентную матрицу, которая задаёт модели роль и формат. Но что делать, если задача требует глубокой логики, математических расчётов или стратегического планирования? Здесь на сцену выходит концепция, которая превращает языковую модель из простого генератора текста в мыслящий инструмент. Эта концепция называется **Chain-of-Thought**, что переводится как цепочка рассуждений. Суть её предельно проста: вы принудительно заставляете модель прописывать промежуточные логические шаги перед тем, как выдать финальный ответ. Почему это критически важно? Вспомните первую главу: модель предсказывает следующий токен. Если вы задаёте сложный вопрос и требуете мгновенный ответ, модель начинает угадывать финальный результат, опираясь на поверхностные статистические связи, и неизбежно накапливает ошибки. Но если вы требуете расписать ход мысли, каждый сгенерированный промежуточный шаг становится новым контекстом, новой опорой для следующего токена. Модель буквально использует свой же текст в качестве оперативной памяти, выстраивая мостик от условия к решению. Для вас это означает, что добавление одной короткой фразы в конец промпта способно радикально повысить точность аналитических выводов.

Развитием этой идеи служит метод **Self-Consistency**, или самосогласованность. Как это работает? Вы просите модель решить одну и ту же логическую задачу несколько раз подряд, каждый раз генерируя новую цепочку рассуждений, а затем выбираете тот финальный ответ, который встретился чаще всего. Представьте, что вы спрашиваете совет у пяти независимых экспертов, выслушиваете их аргументацию и принимаете решение на основе консенсуса. В случае с нейросетью это позволяет отсечь случайные логические тупики и галлюцинации, которые неизбежно возникают при единичной генерации. Если три из пяти цепочек рассуждений приводят к одному выводу, а две — к другому, вероятность истинности первого стремится к максимуму. Читателю стоит взять этот метод на вооружение в ситуациях, когда цена ошибки в бизнес-решении слишком высока, чтобы полагаться на первый же сгенерированный черновик.

Следующий уровень абстракции — это **Tree-of-Thoughts**, то есть дерево мыслей. Если цепочка рассуждений представляет собой линейный путь от точки А к точке Б, то дерево мыслей имитирует процесс принятия решений опытным стратегом или шахматистом. Модель на каждом шаге генерирует несколько альтернативных вариантов действий, оценивает перспективность каждой ветви, отбрасывает заведомо проигрышные и продолжает развивать только самые многообещающие. Это особенно важно для задач, где нет единственно верного алгоритма, а нужно исследовать пространство возможностей: например, при разработке маркетинговой стратегии или поиске нестандартного технического решения. Вы не просто получаете ответ, вы получаете карту принятых решений с обоснованием, почему одни пути были отвергнуты, а другие — выбраны.

Наконец, мы подходим к **каскадному промптингу**, или Prompt Chaining. Это архитектурный приём, при котором одна неподъёмная задача разбивается на последовательность из нескольких простых запросов, где результат работы первого промпта автоматически подаётся на вход второму, и так далее. Почему нельзя просто написать один гигантский промпт со всеми инструкциями? Дело в том, что чем длиннее и запутаннее инструкция, тем выше шанс, что модель потеряет фокус, проигнорирует часть ограничений или запутается в приоритетах. Каскад решает эту проблему, превращая конвейер обработки информации в сборочную линию, где каждый этап отвечает за одну узкую функцию: один промпт только извлекает данные, вто-

рой их структурирует, третий анализирует, четвёртый критикует, а пятый упаковывает в итоговый отчёт.

Реальный кейс

Чтобы показать экономический эффект от внедрения цепочек рассуждений, перенесёмся в офис глобального логистического агрегатора, где руководителем аналитического отдела работает **Дмитрий Анатольевич**. Перед его командой стояла задача автоматизировать расчёт сложных мультимодальных маршрутов, где нужно было учитывать не только базовые тарифы, но и прогнозы погоды, сезонные простои на таможне и исторические данные о задержках конкретных перевозчиков. Когда аналитики впервые попытались решить эту задачу с помощью прямого промпта, загрузив в модель все таблицы и попросив выдать оптимальный маршрут, результат оказался катастрофическим: доля логических и математических ошибок в расчётах достигала сорока восьми процентов. Модель пыталась удержать в памяти слишком много переменных и начинала выдумывать несуществующие скидки или игнорировать погодные риски.

Руководитель перестроил архитектуру запросов, внедрив метод самосогласованности и принудительное пошаговое рассуждение. В новый промпт была добавлена жёсткая инструкция: прежде чем назвать финальный маршрут, модель обязана расписать влияние каждого фактора на итоговую стоимость, рассчитать три альтернативных сценария и аргументировать выбор наилучшего. Доля ошибок рухнула с сорока восьми до двух и одной десятой процента. Время, которое логисты тратили на ручную проверку машинных расчётов, сократилось в двадцать раз, что позволило компании обрабатывать на тридцать процентов больше заявок без расширения штата. Этот кейс наглядно демонстрирует, что нейросеть не заменяет мышление, но она способна его масштабировать, если вы задаёте ей правильный алгоритм работы.

Инструкция

Теперь давайте разберём универсальный алгоритм разделения комплексной бизнес-задачи на пять связанных запросов, который вы сможете адаптировать под свои процессы. Первый шаг называется извлечением и очисткой: вы подаёте модели сырой массив данных, будь то длинная стенограмма совещания или неструктурированный отчёт, и просите исключительно извлечь факты, цифры и имена, отбросив всю воду и эмоциональную окраску. Второй шаг — это структурирование: полученный на первом этапе чистый текст подаётся во второй промпт, задача которого — разложить информацию по заданным категориям или перевести её в формат таблицы. Третий шаг посвящён непосредственному анализу: вы загружаете структурированные данные и просите модель применить метод цепочки рассуждений, чтобы найти скрытые взаимосвязи, аномалии или точки роста, обязательно требуя прописывать логику каждого вывода. Четвёртый шаг — это внедрение внутреннего критика: вы берёте аналитический вывод и отправляете его в новый промпт с ролью скептика, задача которого — найти слабые места, недостающие данные и логические противоречия. Пятый, финальный шаг — это синтез и форматирование, где модель объединяет первоначальный анализ и замечания критика в итоговый документ, оформленный строго по вашему корпоративному шаблону.

Читателю необходимо запомнить, что сила каскада заключается в **изолизации задач**: модель не отвлекается на форматирование, когда ей нужно считать, и не пытается анализировать, когда ей нужно просто извлечь текст. Именно эта последовательная фокусировка на микро-шагах даёт макроскопический выигрыш в качестве итогового результата.

Практика. Продвинутый уровень. Упражнение «Многоуровневый разбор»

Для этого задания вам потребуется около тридцати минут. Найдите любую объёмную аналитическую статью или отраслевой отчёт, содержащий несколько тезисов и выводов. Ваша задача — не просто попросить модель сделать саммари, а провести её через каскад из трёх промптов. Первым промптом извлеките все фактические утверждения и числовые данные. Вторым промптом попросите модель проверить каждое утверждение на внутреннюю непротиворечивость и соответствие базовой логике, расписывая ход мысли для каждого пункта. Тре-

тьим промптом сгенерируйте список из трёх вопросов, которые журналист или инвестор должен задать автору этого отчёта, чтобы выявить скрытые риски. Запишите в чек-лист прогресса, на каком именно этапе модель нашла неочевидные слабые места в исходном тексте, и как пошаговое рассуждение помогло ей избежать поверхностного пересказа.

Практика. Экспертный уровень. Задание «Каскад из 5 шагов»

Это задание потребует от вас около часа и системного подхода к автоматизации. Возьмите реальную рабочую задачу, которая обычно занимает у вас несколько часов: например, подготовку конкурентного анализа перед запуском нового продукта или написание стратегической записки для совета директоров. Спроектируйте для этой задачи полноценный каскад из пяти промптов, чётко прописав, что является входом и выходом на каждом этапе. Первый промпт должен собирать и очищать информацию, второй — классифицировать её, третий — анализировать с использованием техники цепочки рассуждений, четвёртый — подвергать результат жёсткой критике, а пятый — формировать финальный документ. Прогоните через этот каскад реальные данные и оцените качество итогового результата по сравнению с тем, что вы обычно делаете вручную. В чек-лист прогресса перенесите сами тексты пяти промптов и укажите, какой именно этап оказался самым узким местом, потребовавшим дополнительной настройки ограничений или уточнения роли. Именно так создаются автономные конвейеры, способные работать без вашего ежеминутного участия.

На этом шестая глава завершена. Мы разобрали, как заставить модель рассуждать пошагово, как использовать консенсус нескольких цепочек мыслей для защиты от ошибок и как разбивать неподъёмные задачи на конвейер из пяти этапов. В следующей главе мы перейдём к продвинутой разметке: вы узнаете, как использовать XML-теги и псевдокод для жёсткого управления поведением ИИ, превращая хаотичный поток данных в структурированный космос.

Глава 7. Продвинутая разметка: Тегирование и структурирование данных

Теория

В предыдущих главах мы научились составлять промпты по пятикомпонентной матрице и выстраивать цепочки рассуждений. Но что делать, когда входные данные становятся по-настоящему большими и хаотичными: длинная стенограмма совещания на сорок страниц, неструктурированный массив клиентских отзывов, дамп логов из CRM-системы или смесь из инструкций, ограничений и сырых данных в одном промпте? Модель начинает путаться: она не понимает, где заканчивается ваша инструкция и начинаются данные для анализа, где контекст, а где ограничения. В результате она может проигнорировать часть ограничений, перепутать данные с инструкциями или выдумать факты, которых не было в исходнике. Решение этой проблемы — **продвинутая разметка**, то есть использование XML-тегов и псевдокода для жёсткого структурирования промпта.

XML-теги, от eXtensible Markup Language, то есть расширяемый язык разметки, — это специальные маркеры, которые заключают фрагменты текста в угловые скобки: открывающий тег, содержимое, закрывающий тег. Например, <context> здесь контекст </context>. Почему это работает? Потому что современные языковые модели обучались на огромных массивах кода, документации и структурированных данных, где XML-теги используются повсеместно для разделения смысловых блоков. Модель «видит» эти теги и понимает, что содержимое внутри них — это отдельная сущность со своей ролью. Когда вы пишете промпт без разметки, модель получает поток текста и вынуждена сама угадывать структуру. Когда вы используете теги, вы даёте модели чёткую карту: вот здесь роль, вот здесь инструкция, вот здесь сырые данные, вот здесь ограничения. Это снижает когнитивную нагрузку на модель и резко повышает точность выполнения задач.

Основные теги, которые вы будете использовать ежедневно, звучат так. Первый — <role> или <persona>, внутри которого вы указываете, от лица какого специалиста модель должна отвечать. Второй — <context>, где вы размещаете фоновую информацию о проекте, клиенте, отрасли. Третий — <instruction> или <task>, содержащий конкретное задание с глаголом в повелительном наклонении. Четвёртый — <constraints>, где вы перечисляете все ограничения и запреты. Пятый — <raw_data>, внутри которого вы размещаете все сырые данные для анализа: тексты, таблицы, логи, стенограммы. Шестой — <output_format>, где вы описываете желаемый формат вывода. Седьмой — <examples>, если вы используете технику Few-shot. Восьмой — <scratchpad>, то есть «черновик», где вы просите модель вести промежуточные рассуждения перед выдачей финального ответа. Девятый — <thinking>, аналог черновика, но с акцентом на пошаговую логику. Десятый — <verification>, где вы просите модель проверить свой ответ на соответствие ограничениям.

Почему именно <raw_data> критически важен? Потому что без него модель не может отличить ваши инструкции от данных, которые нужно обработать. Представьте, что вы загружаете транскрипт совещания и пишете: «Извлеки из этого текста все задачи и укажи исполнителей». Но внутри транскрипта кто-то произнёс фразу: «Не нужно извлекать задачи, просто перескажи обсуждение». Модель может воспринять эту фразу как вашу инструкцию и проигнорировать ваше первоначальное задание. Но если вы поместите транскрипт внутрь тега <raw_data>, модель поймёт, что это данные для анализа, а не команды для выполнения. Это особенно важно для длинных документов, где вероятность случайного совпадения формулировок с вашими инструкциями возрастает с каждой страницей.

Теперь о связи тегирования с архитектурой открытых протоколов подключения инструментов. В семнадцатой главе мы подробно разберём, как агенты подключают внешние инстру-

менты через стандартизированные протоколы. Но уже сейчас важно понимать, что XML-разметка — это мост между естественным языком и машинным кодом. Когда вы используете теги, вы фактически пишете псевдокод, который модель интерпретирует как структурированные команды. Это позволяет создавать сложные промпты, где разные блоки данных обрабатываются по разным правилам: например, данные внутри <financial_data> анализируются с температурой ноль для максимальной точности, а данные внутри <creative_brief> обрабатываются с температурой ноль целых восемь для генерации идей. Модель понимает эти различия и применяет разные стратегии к разным блокам.

Для мультимодальных задач разметка расширяется тегами <image>, <audio>, <video>, внутри которых вы описываете, что именно нужно извлечь из каждого типа данных. Например: <image description=«фотография повреждённого оборудования»> Проанализируй тип повреждения, укажи его локализацию и предполагаемую причину </image>. Это позволяет модели обрабатывать разнотипные данные в рамках одного промпта, не смешивая их и не путая инструкции для текста с инструкциями для изображений.

Реальный кейс

Чтобы показать, как разметка решает реальные бизнес-проблемы, расскажу историю продуктовой команды крупного IT-сервиса, которая разрабатывала систему автоматического извлечения задач из стенограмм еженедельных планёрок. Задача звучала просто: загрузить транскрипт совещания и получить список задач с исполнителями и дедлайнами. Команда написала промпт: «Проанализируй следующую стенограмму совещания и извлеки все упомянутые задачи. Для каждой задачи укажи: название, исполнителя, срок выполнения, приоритет. Если задача не имеет чёткого исполнителя или срока, отметь это как „требует уточнения“». На первый взгляд, всё логично. Но результат оказался катастрофическим: модель извлекала не только реальные задачи, но и выдумывала их на основе обсуждения. Если на совещании кто-то говорил: «Мы могли бы запустить новую фичу в следующем квартале», модель заносила это как задачу с исполнителем и дедлайном, хотя это было просто гипотетическое обсуждение. Доля галлюцинированных задач достигала тридцати двух процентов, то есть каждая третья задача в списке была выдумана моделью.

Руководитель продуктовой команды, которого звали **Алексей Дмитриевич**, решил применить XML-разметку. Он переписал промпт, разделив его на чёткие блоки. Внутри тега <role> он указал: «Ты — опытный проектный менеджер, который ведёт протоколы совещаний». Внутри <instruction> он написал: «Извлеки из стенограммы только те высказывания, которые являются прямыми поручениями или зафиксированными договорённостями. Не извлекай гипотетические обсуждения, пожелания или общие рассуждения». Внутри <constraints> он добавил: «Если в высказывании нет чёткого глагола в повелительном наклонении или явного согласия исполнителя, не включай его в список задач. Если сомневаешься, лучше пропустить, чем выдумать». И наконец, внутри <raw_data> он разместил саму стенограмму. Результат превзошёл ожидания: доля галлюцинированных задач упала с тридцати двух процентов до нуля целых семи десятых процента, то есть практически обнулилась. Модель начала пропускать сомнительные фрагменты вместо того, чтобы додумывать их, и список задач стал коротким, но стопроцентно точным. Экономия времени на ручной проверке составила около четырёх часов в неделю на каждого менеджера, а доверие к системе автоматической фиксации задач выросло настолько, что команда перестала вести параллельные записи вручную.

Шаблон кода

Теперь я дам вам эталонный шаблон системного промпта с XML-разметкой, который вы будете адаптировать под свои задачи. Структура звучит так. Первый блок: <role> Ты — [должность, стаж, специализация]. Твой стиль: [конкретный, аналитический, креативный]. Ты всегда [указать ключевые принципы работы] </role>. Второй блок: <context> Ситуация: [описание проекта, клиента, отрасли]. Стадия: [начало, середина, завершение]. Ограничения: [бюджет,

сроки, ресурсы] </context>. Третий блок: <instruction> Твоя задача: [конкретное действие глаголом в повелительном наклонении]. Шаги выполнения: 1. [первый шаг]. 2. [второй шаг]. 3. [третий шаг] </instruction>. Четвёртый блок: <constraints> Запрещено: [список запретов]. Обязательно: [список обязательных действий]. Объём: [ограничение по символам или абзацам]. Стиль: [формальный, неформальный, технический] </constraints>. Пятый блок: <raw_data> [сюда помещаются все сырые данные для анализа] </raw_data>. Шестой блок: <output_format> Формат вывода: [описание структуры]. Пример: [один-два примера желаемого формата] </output_format>. Седьмой блок: <scratchpad> Перед выдачей финального ответа проведи промежуточные рассуждения в этом блоке. Проверь соответствие ограничениям. Убедись, что все данные взяты из raw_data, а не выдуманы </scratchpad>. Восьмой блок: <verification> После генерации ответа проверь: 1. Все ли ограничения соблюдены. 2. Нет ли выдуманных фактов. 3. Соответствует ли формат требуемому. Если есть нарушения, исправь их перед финальным выводом </verification>.

Этот шаблон работает как конструктор: вы можете добавлять, убирать или модифицировать блоки в зависимости от задачи. Для простых задач достаточно трёх-четырёх блоков: <role>, <instruction>, <raw_data>, <output_format>. Для сложных аналитических задач нужны все восемь блоков. Для мультимодальных задач добавляются теги <image>, <audio>, <video> с описанием того, что именно нужно извлечь из каждого типа данных. Ключевой принцип: чем сложнее задача, тем больше блоков разметки нужно использовать, потому что каждый блок снижает когнитивную нагрузку на модель и повышает точность выполнения.

Чтобы проверить качество промпта с разметкой перед отправкой, пройдите мысленный чек-лист из шести вопросов. Первый вопрос: все ли сырые данные помещены внутрь тега <raw_data>? Если данные смешаны с инструкциями, модель может их перепутать. Второй вопрос: есть ли блок <constraints> с хотя бы одним ограничением? Если ограничений нет, модель может выдать слишком длинный или слишком общий ответ. Третий вопрос: указан ли формат вывода внутри <output_format>? Если формат не указан, модель сама решит, как оформить ответ. Четвёртый вопрос: есть ли блок <scratchpad> или <thinking> для промежуточных рассуждений? Для сложных задач этот блок критически важен, потому что он заставляет модель рассуждать пошагово. Пятый вопрос: есть ли блок <verification> для самопроверки? Этот блок снижает риск галлюцинаций и нарушений ограничений. Шестой вопрос: все ли теги правильно закрыты? Незакрытый тег может сбить модель с толку и привести к непредсказуемым результатам. Если хотя бы на один вопрос ответ «нет», промпт не готов к отправке.

Отдельно остановлюсь на технике **псевдокода** внутри промпта. Псевдокод — это упрощённая запись алгоритма на языке, близком к программированию, но понятном человеку и модели. Например, вы можете написать: «ЕСЛИ в тексте упоминается числовая метрика, ТОГДА извлеки её и укажи единицы измерения. ИНАЧЕ отметь как „метрика не указана“». Модель отлично понимает такие конструкции, потому что обучалась на миллионах строк кода. Псевдокод особенно полезен, когда вам нужно описать сложную логику с условиями, циклами или ветвлениями. Вместо того чтобы писать длинное объяснение на естественном языке, вы формулируете алгоритм компактно и однозначно. Это снижает риск недопонимания и повышает точность выполнения сложных задач.

Практика. Продвинутый уровень. Задание «Из хаоса в космос»

Ваша задача на ближайшие тридцать-сорок минут — превратить хаотичный массив данных в структурированный результат с помощью XML-разметки. Найдите или создайте длинный текст, содержащий смесь из инструкций, данных и обсуждений: это может быть расшифровка голосовой заметки, где вы диктуете задачи, комментарии и мысли вслух; или длинное письмо от клиента, где требования перемешаны с вопросами и эмоциональными высказываниями; или стенограмма совещания, где обсуждение проблем смешано с конкретными поручениями. Сначала отправьте этот текст в модель с простым промптом без разметки: «Извлеки

из этого текста все задачи и требования». Посмотрите на результат и оцените, сколько задач модель выдумала или исказила. Теперь перепишите промпт с полной XML-разметкой: используйте теги <role>, <instruction>, <constraints>, <raw_data>, <output_format>, <scratchpad>, <verification>. Поместите весь хаотичный текст внутрь <raw_data>, а в <constraints> явно укажите: «Извлекай только те высказывания, которые являются прямыми поручениями или чёткими требованиями. Не извлекай обсуждения, вопросы, пожелания или гипотетические сценарии. Если сомневаешься, лучше пропустить». Отправьте промпт с разметкой и сравните результаты. Запишите в чек-лист прогресса, насколько сократилось количество галлюцинированных задач, какие именно блоки разметки оказались наиболее эффективными и для каких типов хаотичных данных в вашей работе эта техника будет наиболее полезна.

Практика. Экспертный уровень. Задание «Мультимодальная разметка»

Это задание займёт около часа и потребует от вас работы с разнотипными данными. Соберите три типа данных на одну тему: текстовое описание проекта или продукта, изображение схемы, графика или прототипа, аудиозапись обсуждения или голосовой заметки на ту же тему. Ваша задача — написать единый промпт с XML-разметкой, который обрабатывает все три типа данных и выдаёт структурированный результат. Используйте теги <text_data>, <image_data>, <audio_data> для разделения модальностей. Внутри каждого тега укажите, что именно нужно извлечь: из текста — ключевые требования и ограничения, из изображения — визуальные элементы и их взаимосвязи, из аудио — устные договорённости и эмоциональную окраску обсуждения. В блоке <instruction> укажите, как объединить данные из трёх модальностей в единый отчёт. В блоке <constraints> укажите, что делать при противоречии данных из разных модальностей: например, «Если текстовое описание противоречит изображению, укажи это как несоответствие, требующее уточнения». В блоке <verification> попросите модель проверить, все ли три модальности учтены в финальном отчёте. Загрузите все три типа данных и промпт в модель с поддержкой мультимодального ввода. Оцените результат: насколько полно модель объединила данные из разных источников, насколько точно она выявила противоречия, насколько структурированным получился итоговый отчёт. Запишите в чек-лист прогресса итоговый промпт с разметкой, какие именно теги оказались наиболее полезными для мультимодальной обработки и как вы будете применять эту технику в своей ежедневной работе.

На этом седьмая глава завершена. Вы освоили продвинутую разметку с XML-тегами, поняли, почему <raw_data> критически важен для отделения данных от инструкций, получили эталонный шаблон системного промпта и научились превращать хаотичные массивы в структурированные результаты. В следующей главе мы перейдём к обратной задаче: вы узнаете, как определять машинную генерацию в текстах, аудио и видео, какие признаки выдают синтетический контент и как выстроить двухэтапный контроль, чтобы не нанимать «пустых» кандидатов и не покупать несуществующие товары.

Глава 8. Детекция ИИ-контента и критическая оценка

Теория

Мы подошли к рубежу, где технология стала настолько совершенной, что перестала быть просто инструментом и начала имитировать самого создателя. Речь идёт о детекции синтетического контента. Что это такое? Это набор методов и практик, позволяющих отличить работу человеческой психики и моторики от математической аппроксимации, выполненной нейросетью. Почему это важно именно сейчас? Потому что стоимость создания подделки упала до нуля. Любой мошенник или недобросовестный подрядчик может за пять минут сгенерировать голос вашего руководителя, написать безупречный отчёт или смонтировать видеообращение от лица несуществующего эксперта. В корпоративной среде слепое доверие к входящим данным ведёт к финансовым потерям и репутационным кризисам. Понимание механики подделок превращает **иллюзию достоверности**

Конец ознакомительного фрагмента.