ИИ, за который платят

Создавайте полезные сервисы и продавайте результат

ВИКТОР КРОСС

Виктор Кросс

ИИ, за который платят. Создавайте полезные сервисы и продавайте результат

https://litres.ru/74521162
SelfPub; 2026

Аннотация

Красивый ответ нейросети еще никто не обязан покупать. А за сервис, который решает конкретную задачу, уже можно назначать цену.

Сделайте этот переход. От чата к прототипу, от прототипа к полезному продукту, от демонстрации возможностей к предложению для клиента.

Вы разберетесь, как проектировать разговорных помощников, подключать API и базы знаний, готовить данные, проверять качество и управлять стоимостью ответа. Увидите, как соединить технологию с поддержкой, контентом и маркетингом, а затем проверить спрос, продать пилот и оценить экономику полного цикла.

Кейсы и практические задания помогут собрать собственное решение с понятными границами и измеримой пользой. Если

вы застряли на этапе «смотрите, что умеет ИИ», эта книга даст следующий вопрос: какую проблему вы готовы решить за деньги?

Содержание

Виктор Кросс ИИ, за который платят. Создавайте полезные сервисы и продавайте результат

Инструмент, а не волшебная кнопка

Генеративная языковая модель работает с текстом: анализирует последовательность слов и строит наиболее подходящее продолжение с учётом полученного контекста. Она способна поддерживать диалог, преобразовывать формат, суммировать, классифицировать, сравнивать и создавать новые формулировки. Однако у неё нет личного опыта, ответственности за решение и доступа к истине как к отдельной базе данных. Модель может звучать уверенно даже тогда, когда ошибается, поэтому её убедительность нельзя считать доказательством точности.

Полезнее всего воспринимать такую систему как универсальный когнитивный инструмент. Калькулятор усиливает

способность считать, электронная таблица — способность моделировать данные, а языковая модель — способность быстро работать с формулировками, вариантами и структурами. Она не отменяет профессиональное мышление, но сокращает расстояние между идеей и первым проверяемым черновиком.

Главный принцип

Чем проще получить сырой результат, тем меньше он стоит сам по себе. Ценность появляется после отбора, проверки, адаптации к конкретной ситуации и включения в работающий процесс.

Три уровня практической ценности

Один и тот же инструмент приносит разную пользу в зависимости от того, насколько глубоко он встроен в работу. Условно можно выделить три уровня.

Личная скорость. Человек использует модель как собеседника для идей, редактора, помощника по структуре или переводчика сложного материала на понятный язык. Выигрыш измеряется сэкономленными минутами и количеством качественных вариантов.

Повторяемый процесс. Команда превращает удачные запросы в шаблоны, добавляет правила проверки, подключает корпоративные документы и распределяет роли. Выигрыш измеряется стабильностью, меньшим числом ошибок и сокращением времени цикла.

Система, создающая результат для клиента. Модель становится частью продукта или услуги: помогает консультанту готовить рекомендации, магазину — отвечать на типовые вопросы, образовательной платформе — давать персональные объяснения. Выигрыш измеряется выручкой, удержанием, скоростью обслуживания и качеством решений.

Переход между уровнями важнее, чем поиск очередного эффектного запроса. На первом уровне человек получает текст. На втором — управляемый процесс. На третьем — ценность, за которую другой человек готов платить или ради

которой он остаётся пользователем продукта.

Как идея превращается в систему

В феврале 2024 года небольшая студия «Северный текст» из Нижнего Новгорода получила заказ от сети из двенадцати частных клиник. Клиенту требовалось каждую неделю готовить публикации для социальных сетей, ответы на типовые вопросы пациентов и черновики писем после приёма. Раньше два редактора тратили на это около сорока часов в неделю, а материалы всё равно отличались по тону и иногда противоречили внутренним инструкциям.

Руководитель студии Марина Волкова не стала продавать клиенту «тексты от нейросети». Вместо этого команда собрала базу из утверждённых медицинских формулировок, списка запрещённых обещаний, примеров допустимого тона и маршрутов эскалации к врачу. В Notion хранились правила, в Make запускалась обработка новых заявок, а итоговые черновики проходили проверку редактора и медицинского координатора. Через шесть недель среднее время подготовки недельного пакета сократилось с сорока до четырнадцати часов. При этом клиент платил не за генерацию слов, а за предсказуемый контент-процесс с понятной ответственностью.

Этот пример показывает ключевую закономерность. Инструмент был доступен всем участникам рынка, но ценность создала не сама модель. Её создали предметные ограниче-

ния, архитектура процесса, накопленные примеры, контроль качества и способность команды доставить результат в привычный рабочий контур клиента.

Почему разговорный интерфейс меняет бизнес

Большинство цифровых систем требуют от пользователя выбирать правильный раздел, заполнять форму или знать точное название функции. Разговорный интерфейс меняет порядок действий: сначала человек описывает цель своими словами, а система пытается определить намерение и предложить путь. Это особенно важно там, где запросы разнообразны, а пользователь не знает внутреннюю структуру компании.

Представьте службу доставки. Один клиент пишет: «Где заказ?», другой — «Курьер обещал быть к восьми, уже половина девятого», третий — «Можно перенести на завтра?». Формулировки различаются, но за ними стоят ограниченное число задач: узнать статус, сообщить о задержке, изменить время. Хороший чат-бот должен не просто узнавать отдельные слова, а понимать намерение, извлекать важные детали и сохранять контекст разговора.

Именно поэтому в основе практического разговорного ИИ лежит обработка естественного языка. Она связывает человеческую речь с действиями системы: поиском в базе знаний, созданием заявки, расчётом, рекомендацией, передачей оператору. Без этой связки модель остаётся красноречивым

генератором текста. С ней — становится частью сервиса.

Что эта книга будет делать

Эта книга посвящена не набору эффектных трюков, а последовательному созданию работающих решений. Мы начнём с того, как компьютер разбирает человеческий язык, затем перейдём к машинному обучению, устройству современных языковых моделей, подготовке данных, проектированию диалога и оценке качества. Отдельное внимание будет уделено внедрению: интерфейсам, интеграциям, обратной связи, безопасности и экономике продукта.

Каждая тема будет рассматриваться через один и тот же вопрос: какое решение должен принять человек или система после ответа модели? Такой подход защищает от соблазна измерять качество красотой текста. Иногда лучший ответ — короткая классификация, иногда — уточняющий вопрос, иногда — передача живому специалисту. Разговорный ИИ полезен не тогда, когда говорит больше, а тогда, когда помогает двигаться к цели с меньшими потерями.

Этап	**Главный вопрос**	**Практический результат**
Понимание языка	Что на самом деле хочет пользователь?	

Намерение, сущности, контекст и тон

Создание ответа
Какая информация нужна для решения?
Текст, поиск, расчёт или уточнение

Встраивание
Что должно произойти после ответа?
Заявка, рекомендация, документ, действие

Контроль
Как понять, что система полезна?
Метрики качества, обратная связь, аудит

Границы ответственности

Чем ближе система к деньгам, здоровью, правам человека или репутации, тем важнее разделение ролей. Модель может предложить вариант, но окончательное решение должен принимать тот, кто понимает последствия и несёт ответственность. Нельзя загружать конфиденциальные данные без понятных правил хранения. Нельзя публиковать цифры и факты без проверки. Нельзя скрывать автоматизацию там, где пользователь вправе знать, с кем взаимодействует.

Ответственное использование не делает технологию менее полезной. Напротив, оно позволяет строить решения, которым доверяют. Чёткие ограничения, журналирование, проверка источников и возможность быстро подключить человека превращают эксперимент в рабочий инструмент.

Как читать и применять

Лучший способ освоить разговорный ИИ — работать с собственной задачей. Выберите один повторяющийся процесс: ответы клиентам, подготовку коммерческих предложений, анализ отзывов, обучение сотрудников или создание контент-плана. На протяжении книги фиксируйте исходное время, типичные ошибки и критерии хорошего результата. Затем проверяйте каждый новый приём на этом процессе.

Не стремитесь сразу автоматизировать всё. Начните с узкого участка, где входные данные понятны, а результат можно проверить. Соберите десять-двадцать реальных примеров, опишите правила, создайте первый прототип, а затем наблюдайте, где он ошибается. Улучшение разговорной системы начинается не с более длинного запроса, а с более точного понимания задачи.

Стартовая установка

Не спрашивайте: «Что может сделать искусственный интеллект?» Спросите: «Какое повторяющееся решение в моей работе можно сделать быстрее, точнее и удобнее — и как я проверю улучшение?»

Вместо обещания лёгкого успеха

Искусственный интеллект действительно снижает стоимость многих интеллектуальных операций. Он помогает быстрее исследовать варианты, оформлять мысли и обслуживать большой поток запросов. Но доступность инструмента означает и рост конкуренции: черновик становится дешевле, а требования к выбору, вкусу, экспертизе и доведению до результата — выше.

Поэтому главная возможность состоит не в том, чтобы нажать кнопку раньше других. Она состоит в том, чтобы научиться строить вокруг новой технологии дисциплинированные процессы. В следующих главах мы разберём эту работу от самого основания — от того, как машина превращает человеческую фразу в структуру, с которой можно действовать.

002 · ГЛАВА 1

Как машина понимает человеческий язык

Обработка естественного языка: намерения, сущности, контекст и архитектура диалога.

Понять фразу — значит не только распознать слова, но и определить, какое действие должно последовать.

В 9:12 утра в чат интернет-магазина приходит сообщение: «Зелёные сорок второго есть, как вчера на фото?» Для человека, знакомого с предыдущей перепиской, смысл почти очевиден. Покупатель спрашивает о наличии зелёных кроссовок 42-го размера, показанных днём ранее. Для программы это сложная задача: название товара не указано, слово «есть» может означать наличие или существование, а выражение «как вчера» требует памяти о контексте.

Обработка естественного языка, или NLP, занимается именно такими задачами. Её цель — помочь вычислительной системе работать с человеческой речью: распознавать структуру, извлекать смысл, определять намерение, находить важные факты и формировать подходящий ответ. В чат-боте NLP соединяет свободную фразу пользователя с конкретным действием бизнеса.

1. Что такое естественный язык

Естественным называют язык, который возник и развивался в человеческом обществе: русский, английский, испанский и тысячи других. В отличие от языка программирования, он не проектировался ради однозначности. Люди постоянно опускают слова, меняют порядок, используют намёки, метафоры, иронию и общий опыт. Одна и та же мысль может быть выражена десятками способов, а одна и та же фраза — иметь разные значения.

Фраза «Откройте окно» в офисе обычно означает просьбу проветрить помещение. В интерфейсе компьютера она может относиться к новому окну браузера. В разговоре с дизайнером — к модальному окну приложения. Значение определяется не отдельными словами, а ситуацией, предметной областью, предыдущими репликами и ожиданиями участников.

NLP не сводится к распознаванию словаря. Система должна ответить как минимум на четыре вопроса: что сказано, что имеется в виду, какие детали важны и что делать дальше. Для этого используются методы лингвистики, статистики, машинного обучения и инженерии программных систем.

Рабочее определение

NLP — это набор методов, с помощью которых компьютер анализирует, интерпретирует и создаёт человеческий язык для выполнения конкретной задачи.

2. Почему язык труден для машины

Неоднозначность

Слово, форма или целое предложение могут иметь несколько значений. «Ключ» бывает инструментом, источником воды, кодом доступа или главным объяснением. «Я видел человека с телескопом» не сообщает однозначно, у кого был телескоп. Человек часто разрешает неоднозначность автоматически, опираясь на здравый смысл и контекст; системе этот контекст нужно представить явно или научить извлекать его из данных.

Неполнота

В диалоге люди редко повторяют всё необходимое. После вопроса «Когда привезут заказ №1847?» пользователь может написать только «А раньше нельзя?». Чтобы ответить, бот должен связать вторую реплику с доставкой конкретного заказа, понять, что речь идёт о переносе времени, и проверить доступные интервалы.

Вариативность

Запрос «вернуть товар» может звучать как «хочу оформить возврат», «не подошёл размер», «можно отправить обратно?» или «заберите это, пожалуйста». Смысл близок, а слова почти не совпадают. Жёсткий поиск по ключевой фразе быстро ломается, поэтому современные системы учатся сопоставлять выражения по значению.

Ошибки и разговорная речь

Пользователи пишут с опечатками, сокращениями, смешением языков и эмоциональными вставками: «заказ 8731 опять завис, ну сколько можно». Хорошая система должна отделить полезную информацию от шума, распознать номер заказа, определить проблему со статусом и учесть раздражение пользователя, не отвечая канцелярской фразой.

3. Базовый путь от фразы к действию

Практический NLP-конвейер можно представить как последовательность преобразований. Реальные системы объединяют некоторые этапы, особенно когда используют большие языковые модели, но логика остаётся полезной для проектирования и диагностики.

Шаг	**Задача**	**Пример результата**	**Зачем это нужно**
1	Нормализация	«доставка завтра??» → «доставка завтра?»	Уменьшить влияние опечаток и формата
2	Разбиение текста	Токены: «доставка», «завтра»	Получить единицы анализа

3
Определение намерения
Изменить дату доставки
Выбрать бизнес-сценарий

4
Извлечение сущностей
Дата: завтра; заказ: из контекста
Заполнить параметры действия

5
Учет контекста
Речь о заказе №1847
Связать текущую реплику с предыдущими

6
Формирование ответа
Предложить доступные интервалы
Продвинуть пользователя к решению

7
Выполнение действия
Обновить слот в системе доставки
Создать реальный результат

4. Токены, формы слов и грамматика

Токенизация

Первый технический шаг — разбиение текста на элементы, которые система будет обрабатывать. Такие элементы называют токенами. Токен может совпадать со словом, частью слова, числом или знаком пунктуации. Для русского языка это особенно важно из-за длинных форм и окончаний: «доставка», «доставкой» и «доставить» связаны по смыслу, но выглядят по-разному.

Современные языковые модели часто используют токены-подслова. Это позволяет работать с редкими названиями, новыми терминами и опечатками, собирая слово из знакомых фрагментов. Цена такого подхода — текстовая длина для модели измеряется не символами и не словами, а токенами, поэтому большой документ может занять больше контекстного окна, чем ожидает пользователь.

Лемматизация и морфология

Лемматизация приводит словоформу к словарной форме: «купил», «покупает», «покупки» связываются с леммой «покупать» или с общей семантической основой. Морфологический анализ определяет число, род, падеж, время и другие признаки. В классических системах это помогает строить правила и признаки; в нейросетевых моделях многие связи изучаются из данных, но морфология остаётся полезной для объяснения ошибок и для задач с высокой точностью.

Части речи

Разметка частей речи, или POS tagging, присваивает словам грамматические роли: существительное, глагол, прилагательное, наречие и так далее. Во фразе «быстрая доставка задержалась» слово «быстрая» описывает доставку, а «задержалась» сообщает событие. Такая разметка помогает отличать объект от действия и строить дальнейший синтаксический анализ.

Синтаксический разбор

Синтаксис описывает связи между словами. В запросе «Отмените заказ, который курьер ещё не забрал» важно понять, что местоимение «который» относится к заказу, а не к курьеру. Дерево зависимостей показывает, какое слово является главным, какие слова его уточняют и как устроено предложение. Для чат-бота это может быть критично при сложных условиях и отрицаниях.

5. Намерение: зачем пользователь написал

Намерение, или intent, — это цель пользователя, выраженная в сообщении. В службе поддержки набор намерений обычно ограничен: узнать статус, изменить адрес, отменить заказ, запросить возврат, пожаловаться, получить консультацию. Классификатор намерений сопоставляет свободную формулировку с одним из сценариев.

Качество классификации зависит не только от модели, но и от того, как команда определила сами классы. Если намерения пересекаются — например, «проблема с доставкой» и «курьер опоздал» — система будет путаться. Если класс слишком широк, внутри него окажутся запросы, требующие разных действий. Хорошая схема намерений строится вокруг бизнес-операций, а не вокруг красивых названий тем.

Практическое правило

Определяйте намерение так, чтобы после его распознавания система могла выбрать конкретный следующий шаг: запросить данные, вызвать API, показать инструкцию или передать диалог человеку.

6. Сущности: какие детали нужны

Сущности — это конкретные значения, необходимые для выполнения намерения. В сообщении «Перенесите доставку заказа 1847 на пятницу после шести» намерение — изменить время доставки, а сущности — номер заказа, день недели и временной интервал. Извлечение именованных сущностей называют NER, хотя в прикладных ботах список сущностей часто шире классических персон, организаций и мест.

Для магазина важны артикул, размер, цвет, сумма, адрес и дата. Для клиники — специализация врача, филиал и желаемое время. Для банка — тип операции, валюта, получатель и сумма. Словарь сущностей должен отражать реальные поля систем, с которыми бот взаимодействует.

Фраза пользователя
Намерение
Извлечённые сущности

«Запишите меня к кардиологу в филиал на Лесной после 17:00»

Запись на приём

Специализация: кардиолог; филиал: Лесная; время: после 17:00

«Верните 3 490 рублей за заказ 7712»
Запрос возврата
Сумма: 3 490 #; заказ: 7712

«Нужна синяя куртка M, как в каталоге»
Поиск товара
Категория: куртка; цвет: синий; размер: M

7. Контекст и память диалога

Намерение и сущности редко существуют в одной реплике. Пользователь может сначала назвать товар, затем размер, а через несколько сообщений изменить цвет. Система должна вести состояние диалога: хранить подтверждённые параметры, различать новые и старые значения, понимать ссылки вроде «этот», «там», «тот же» и знать, когда контекст уже ненадёжен.

Полезно разделять краткосрочный и долгосрочный контекст. Краткосрочный относится к текущему разговору: выбранный товар, номер заказа, уточнённая дата. Долгосрочный — к устойчивым данным пользователя: язык, адрес по умолчанию, тариф, история обращений. Долгосрочную память нельзя собирать бесконтрольно; она требует согласия, правил хранения и возможности исправления.

Контекст также включает внешние знания. Если пользователь спрашивает о статусе заказа, модель не должна придумывать ответ из языковых закономерностей. Она должна получить актуальные данные из системы заказов. Если вопрос касается инструкции, бот должен найти нужный фрагмент в утверждённой базе знаний. Такая архитектура отделяет понимание запроса от источника фактов.

8. Тональность и эмоциональные сигналы

Анализ тональности оценивает, выражает ли текст положительное, нейтральное или отрицательное отношение. В поддержке важнее не поставить красивую метку, а изменить поведение системы. Сообщение «Третий раз списали деньги, вы вообще читаете?» должно повысить приоритет, исключить бодрые рекламные фразы и, возможно, быстрее подключить оператора.

Тональность нельзя трактовать как точное чтение эмоций. Сарказм, культурные различия и краткие сообщения создают ошибки. Поэтому эмоциональный сигнал лучше использовать как дополнительный признак, а не как единственное основание для серьёзного решения.

9. Генерация языка

После понимания запроса система должна сформировать ответ. В классическом боте ответ выбирается из шаблона и заполняется данными. Это надёжно, но негибко. Генеративная модель может переформулировать, объяснить, сократить и адаптировать тон, однако требует ограничений: фактической опоры, правил стиля, максимальной длины и списка запрещённых утверждений.

Практическая архитектура часто сочетает оба подхода. Критические данные — сумма, дата, номер заявки — вставляются из системы без изменения. Объяснение вокруг них формирует модель. В результате ответ звучит естественно, но ключевые факты остаются контролируемыми.

10. Как ChatGPT связан с NLP

ChatGPT относится к классу больших языковых моделей. Такие модели обучаются на больших коллекциях текста предсказывать продолжение последовательности и в процессе усваивают множество статистических закономерностей языка. Они способны выполнять разные задачи по инструкции без отдельного классификатора для каждой формулировки: суммировать, извлекать сущности, менять стиль, от-

вечать на вопросы и поддерживать контекст.

Это не отменяет классические понятия NLP. Напротив, намерения, сущности, тональность и контекст остаются удобным языком проектирования. Большая модель может объединить несколько этапов в одном вызове, но команда всё равно должна понимать, что именно она ожидает на выходе, как проверить результат и что произойдёт при ошибке.

Внутри современных моделей важную роль играет механизм внимания: при обработке каждого элемента система оценивает связи с другими элементами контекста. Это помогает учитывать дальние зависимости и понимать, к чему относится местоимение или уточнение. Однако внимание не гарантирует истинность. Модель хорошо строит связный текст, но факты должны поступать из проверяемых источников.

11. Архитектура чат-бота: четыре слоя

Понимание языка. Система определяет намерение, сущности, тон и недостающие параметры.

Управление диалогом. Правила или модель решают, что делать дальше: уточнить, выполнить действие, показать информацию или передать оператору.

Данные и инструменты. Бот обращается к CRM, каталогу, календарю, платёжной системе, поиску или базе знаний.

Формирование ответа. Результат переводится в понятную реплику с нужным тоном и уровнем подробности.

Ошибки полезно диагностировать по слоям. Если бот перепутал возврат и обмен — проблема в понимании. Если правильно понял запрос, но запросил уже известный номер заказа — в управлении контекстом. Если сообщил неверный статус — в интеграции с данными. Если ответил грубо — в генерации и правилах стиля. Без такого разделения команда пытается лечить все проблемы одним более длинным промптом.

12. Практический кейс: бот для транспортной компании

Осенью 2024 года компания «УралТранс Сервис» из Екатеринбурга обрабатывала около 2 600 обращений в неделю. Большинство клиентов спрашивали о местонахождении груза, переносе доставки, документах и стоимости хранения. Операторы вручную искали данные в TMS, копировали статусы и отвечали в чате. Среднее первое ожидание составляло одиннадцать минут, а в понедельник утром превышало двадцать.

Команда начала не с генерации ответов, а с анализа 4 800 обезличенных диалогов. После удаления персональных данных запросы сгруппировали в девять намерений. Для каждого определили обязательные сущности: номер накладной, город, дата, тип документа. Затем описали правила: статус можно сообщать только из TMS; стоимость хранения — только после расчёта; претензии о повреждении сразу передаются специалисту.

Первый прототип распознавал намерения и возвращал структурированный JSON. Диалоговый модуль проверял, хватает ли данных. Если номера накладной не было, бот просил его. Если был — вызывал API TMS. Языковая модель использовалась только на последнем этапе: превращала тех-

нический статус в понятную реплику и учитывала тон пользователя.

Через восемь недель бот самостоятельно завершал 58% типовых обращений. Среднее время первого ответа сократилось до сорока секунд. При этом доля ошибочных фактических ответов оставалась ниже одного процента, потому что модель не имела права придумывать статус. Самым полезным результатом проекта стала не автоматизация как таковая, а прозрачная карта клиентских намерений: компания обнаружила, что 17% обращений о задержке возникали из-за непонятного статуса «передано на терминал» и переписала уведомления.

13. Типичные ошибки при создании NLP-системы

Начинать с модели, не определив бизнес-действия. В результате бот красиво отвечает, но не решает задачу.

Создавать слишком много похожих намерений. Классы пересекаются, а разметчики сами не могут договориться о правильной категории.

Игнорировать контекст. Система повторно задаёт вопросы и теряет доверие пользователя.

Разрешать генерации подменять источник данных. Модель придумывает статус, цену или правило, потому что ей не дали проверяемый инструмент.

Оценивать только среднюю точность. Ошибка в привет-

ствии и ошибка в платёжной операции имеют разную стоимость.

Не собирать реальные формулировки. Тестовые примеры команды обычно чище и вежливее настоящих сообщений клиентов.

Автоматизировать без маршрута к человеку. Любая система сталкивается с редкими, конфликтными или высокорисковыми случаями.

14. Ключевые термины

Термин
Смысл

NLP
Методы анализа, интерпретации и генерации человеческого языка.

Токен
Элемент текста, с которым работает модель: слово, часть слова, число или знак.

Лемма
Словарная форма слова, используемая для объединения словоформ.

POS tagging
Определение частей речи и грамматических ролей.

Синтаксический разбор
Определение структуры предложения и связей между словами.

Намерение

Цель пользователя, соответствующая сценарию или действию.

Сущность

Конкретное значение, необходимое для выполнения действия.

NER

Извлечение именованных и предметных сущностей из текста.

Тональность

Оценка эмоциональной окраски или отношения в сообщении.

Контекст

Информация из диалога, профиля и внешних источников, влияющая на смысл.

Языковая модель

Модель, оценивающая вероятности последовательностей текста и создающая продолжение.

Диалоговый менеджер

Компонент, который выбирает следующий шаг разговора.

RAG

Подход, при котором модель получает релевантные фрагменты из внешней базы перед ответом.

Эскалация

Передача диалога человеку или специализированному процессу.

15. Мини-практикум

Выберите один диалоговый процесс в своей работе и соберите двадцать реальных или обезличенных сообщений пользователей. Не исправляйте опечатки и не улучшайте стиль. Затем выполните четыре шага.

Сгруппируйте сообщения по действиям, которые должна выполнить система. Дайте каждой группе название намерения.

Для каждого намерения перечислите обязательные сущности. Отметьте, какие из них пользователь часто сообщает не сразу.

Опишите, откуда берётся фактический ответ: база знаний, CRM, календарь, расчёт или человек.

Сформулируйте безопасный маршрут ошибки: какой уточняющий вопрос задать, когда остановиться и кому передать диалог.

После этого проверьте схему на пяти новых сообщениях. Если один запрос подходит сразу к двум намерениям, пересмотрите границы классов. Если действие нельзя выполнить даже после извлечения сущностей, значит описание процесса ещё неполно.

16. Выводы

Обработка естественного языка превращает человеческую речь в управляемую структуру. Для чат-бота недостаточно распознать слова: нужно определить намерение, извлечь сущности, учесть контекст, получить факты из надёжного источника и выбрать следующее действие. Современные языковые модели делают этот процесс гибче, но не снимают инженерную ответственность.

Главная единица проектирования — не фраза и не промпт, а решение. Когда команда понимает, какое решение должно последовать после сообщения пользователя, термины NLP перестают быть академической теорией и становятся картой работающего сервиса. В следующей главе мы разберём, как модели учатся находить такие закономерности в данных и почему качество обучения зависит от примеров, целей и обратной связи.

003 · ГЛАВА 2
Как модель учится на данных

Машинное обучение: примеры, цели, обратная связь и границы обобщения.

Модель становится полезнее не от количества разгово-

ров само по себе, а от ясной цели, качественных примеров и честной проверки на новых данных.

В отдел поддержки производителя кассового оборудования поступает фраза: «После обновления смена не закрывается, чек коррекции уже делали». Опытный специалист сразу видит несколько признаков: речь идёт о программной версии, операция повторялась, а стандартная инструкция не помогла. Для модели это не готовое правило, а сочетание слов, чисел, контекста и прошлых примеров, по которым нужно выбрать следующий шаг.

Машинное обучение позволяет системе находить такие закономерности в данных и применять их к новым сообщениям. Программа не получает вручную написанную инструкцию для каждой возможной фразы. Вместо этого она настраивает внутренние параметры так, чтобы на известных примерах давать требуемый результат, а затем пытается обобщить найденную связь на запросы, которых раньше не видела.

1. Что означает «обучить машину»

В обычном программировании разработчик формулирует правило: если выполнены условия A и B, сделай действие C. В машинном обучении правило часто слишком сложное, чтобы выписать его вручную. Тогда системе показывают множество примеров и задают критерий качества. Алгоритм меняет параметры модели, сравнивает предсказание с ожидаемым результатом и постепенно уменьшает ошибку.

Обучение не означает человеческого понимания и не превращает модель в обладателя опыта. Это математическая оптимизация. Модель ищет конфигурацию параметров, при которой входные данные преобразуются в полезный выход: категорию обращения, вероятность отказа, следующий токен, оценку тональности или рекомендуемое действие.

Рабочее определение

Машинное обучение — это способ создавать системы, которые находят закономерности в примерах, измеряют ошибку и изменяют свои параметры, чтобы лучше выполнять заданную задачу на новых данных.

2. Из чего состоит учебный пример

Чтобы обсуждать обучение предметно, полезно разделять четыре элемента: объект, признаки, целевой ответ и функцию ошибки. Объектом может быть сообщение клиента, диалог, заказ или изображение. Признаки — информация, на которой строится решение. Целевой ответ показывает, что система должна предсказать. Функция ошибки сообщает, насколько предсказание отличается от желаемого результата.

Элемент	**В задаче чат-бота**	**Конкретный пример**	**Риск неправильного определения**
Объект	Единица анализа	Одно сообщение или весь диалог	Потеря контекста либо смешение нескольких задач
Признаки	Сигналы для решения	Текст, канал, история, тип клиента	Утечка персональных данных или случайные корреляции

Цель
Ожидаемый выход
Намерение «возврат», сущности, ответ
Модель оптимизирует не тот результат

Ошибка
Мера качества
Штраф за неверный класс или токен
Редкие, но дорогие ошибки растворяются в среднем

Качество системы начинается не с выбора алгоритма, а с точного описания этих элементов. Если команда размечает «положительный диалог» по вежливости ответа, а бизнесу нужна решённая проблема, модель будет улучшать тон, но не завершение обращения.

3. Обучающая, проверочная и тестовая выборки

Один набор примеров нельзя одновременно использовать для обучения и честной оценки. Если модель видит ответы во время настройки, она может запомнить особенности конкретных данных и показать впечатляющий результат только на них. Поэтому данные обычно делят на три части.

Обучающая выборка

На ней алгоритм изменяет параметры. Чем разнообразнее реальные формулировки, тем выше шанс, что модель встретит во время обучения опечатки, короткие ответы, профессиональный жаргон и нестандартный порядок слов.

Проверочная выборка

Она помогает выбирать настройки, сравнивать варианты и вовремя замечать переобучение. Модель не должна напрямую обучаться на этих ответах, иначе проверка перестаёт быть независимой.

Тестовая выборка

Её используют в конце как контрольный экзамен. Хороший тест отражает будущую эксплуатацию: те же каналы, языки, доли редких сценариев и уровень шума. Если тест состоит из аккуратных фраз аналитиков, он не предскажет поведение в настоящем чате.

4. Обучение с учителем

При обучении с учителем каждый пример сопровождается правильным ответом. Сообщению присваивают намерение, письму — метку «спам» или «не спам», диалогу — итог «решён» или «передан оператору». Модель учится связывать вход с меткой и затем классифицирует новые объекты.

Для чат-ботов этот подход применяют при распознавании намерений, извлечении сущностей, маршрутизации обращений и оценке риска. Главная стоимость находится в разметке. Если два специалиста по-разному понимают границу между «отменой» и «возвратом», модель лишь воспроизведёт эту неопределённость.

Практическое правило

Перед массовой разметкой дайте одинаковые двадцать примеров нескольким сотрудникам. Если согласие низкое, сначала уточните классы и инструкции, а не увеличивайте объём данных.

5. Обучение без учителя

При обучении без учителя готовых меток нет. Алгоритм ищет структуру внутри самих данных: группирует похожие объекты, сокращает размерность, обнаруживает необычные точки. Для команды поддержки это способ увидеть темы, которые ещё не попали в справочник намерений.

Например, кластеризация десяти тысяч обращений может выделить отдельную группу сообщений про зависание терминала после обновления. Аналитик изучит примеры, даст группе понятное название и решит, нужен ли новый сценарий. Алгоритм не объясняет бизнес-смысл автоматически; он показывает близость, которую человек должен интерпретировать.

6. Обучение с подкреплением

В обучении с подкреплением система выбирает действие, получает награду или штраф и учится повышать суммарный результат. В диалоге действием может быть уточняющий вопрос, рекомендация, вызов инструмента или передача оператору. Награда может учитывать завершение задачи, число лишних шагов, оценку пользователя и стоимость ошибки.

Опасность возникает, когда награда слишком узкая. Если оптимизировать только длину диалога, бот начнёт преждевременно завершать сложные обращения. Если награждать только клики, система может выбирать навязчивые формулировки. Поэтому цель должна отражать полезность и ограничения, а не один удобный показатель.

7. Глубокое обучение и языковые модели

Глубокое обучение использует многослойные нейронные сети, способные самостоятельно формировать внутренние представления данных. В обработке языка это позволило отказаться от большого числа вручную заданных признаков. Вместо отдельного правила для каждого окончания или порядка слов модель учится кодировать связи по множеству примеров.

Большая языковая модель обучается предсказывать продолжение текста. Задача кажется простой, но для хорошего предсказания ей приходится учитывать грамматику, тему, стиль, связи между частями документа и типичные формы рассуждения. После предварительного обучения модель дополнительно настраивают на выполнение инструкций и безопасное ведение диалога.

При этом языковая модель не хранит знания как аккуратную таблицу фактов. Параметры кодируют статистические закономерности, поэтому связность ответа не гарантирует достоверность. Актуальные статусы, цены, правила и персональные данные должны поступать из внешних систем.

8. Векторные представления и смысловая близость

Модели преобразуют текст в числовые представления — векторы. Близкие по смыслу фразы обычно оказываются рядом в таком пространстве, даже если используют разные слова. Благодаря этому запрос «как вернуть неподошедший размер» можно сопоставить с инструкцией «оформление возврата товара» без точного совпадения ключевых фраз.

Векторный поиск лежит в основе многих систем поиска по базе знаний. Документы разбивают на фрагменты, для каждого вычисляют представление, а затем находят фрагменты, наиболее близкие к вопросу пользователя. Языковая модель формирует ответ на основе найденного материала, а не пытается восстановить правило из общих закономерностей.

9. Четыре способа адаптировать чат-бота

Слово «обучение» часто используют слишком широко. На практике команда может улучшать поведение системы четырьмя разными способами, и каждый решает свою проблему.

Подход
Когда применять
Что реально меняется

Инструкции и примеры
Нужно уточнить роль, формат и стиль
Контекст запроса; параметры модели не меняются

Поиск по базе знаний
Ответ зависит от документов и актуальных фактов
Модель получает релевантные фрагменты перед генерацией

Тонкая настройка
Нужен устойчивый формат или поведение на большом числе примеров

Изменяются параметры отдельной версии модели

Инструменты и правила

Нужно выполнить действие или получить точные данные
Модель вызывает API, расчёт или бизнес-процесс

Частая ошибка — пытаться тонко настраивать модель ради знаний, которые ежедневно меняются. Для каталога, тарифов и регламентов безопаснее поиск и вызов систем. Тонкая настройка полезнее там, где требуется стабильно распознавать внутренние категории, выдерживать сложный формат или повторять специализированную манеру ответа.

10. Как строится цикл обратной связи

Рабочая система не должна учиться напрямую на каждом сообщении пользователя. Такая схема быстро накопит случайные ошибки, попытки манипуляции и конфиденциальные данные. Вместо этого нужен управляемый цикл: журналирование, выборка спорных диалогов, проверка специалистом, обновление правил или данных, повторное тестирование и только затем выпуск изменений.

1. Соберите сигналы: отказ пользователя, повторный вопрос, исправление оператора, низкую оценку, нарушение формата.

2. Отберите репрезентативные и рискованные примеры, удалив персональные данные.

3. Определите причину ошибки: плохая разметка, отсутствующий документ, неверная логика, недостаточный контекст или генерация.

4. Исправьте соответствующий слой и проверьте регрессию на неизменяемом тестовом наборе.

5. Запускайте изменение постепенно и сравнивайте метрики с предыдущей версией.

11. Чем измерять качество

Одна метрика не описывает работу чат-бота. Техническая точность может расти, а пользователи всё равно будут недовольны из-за долгого ответа или лишних вопросов. Поэтому оценку полезно разделять на уровень модели, диалога и бизнеса.

Уровень
Метрика
Что показывает
О чём может умолчать

Модель
Precision, recall, F1
Качество распознавания классов
Стоимость разных ошибок

Диалог
Доля завершений, число шагов
Удалось ли довести сценарий до результата
Удовлетворённость и корректность результата

Сервис
Время ответа, эскалации, CSAT

Как работает канал для клиента
Причины изменения показателей

Экономика
Стоимость диалога, нагрузка операторов
Окупаемость и масштабируемость
Репутационные и юридические риски

Для редких опасных сценариев полезнее отдельно считать полноту. Если система пропускает сообщения о двойном списании или угрозе безопасности, высокий средний процент правильных ответов не спасает проект.

12. Переобучение, смещение и изменение данных

Переобучение

Модель слишком хорошо подстраивается под учебные примеры и хуже работает на новых. Признаки — растущее качество на обучении при ухудшении проверки, чувствительность к мелким формулировкам и запоминание случайных деталей.

Смещение данных

Если в наборе почти нет сообщений пожилых пользователей, региональной лексики или редких продуктов, система будет хуже обслуживать именно эти группы. Баланс данных должен отражать не только частоту, но и цену ошибки.

Дрейф

После запуска меняются продукты, интерфейсы и поведение клиентов. Новая акция создаёт непривычные вопросы, обновление приложения — новый класс проблем. Модель, которая хорошо работала в январе, может заметно ухудшиться в апреле, даже если её код не менялся.

13. Практический кейс: маршрутизация технической поддержки

Весной 2025 года новосибирская компания «КварцТех Снаб» обслуживала торговые сети, использующие кассы, сканеры и принтеры этикеток. В поддержку приходило около 3 400 обращений в неделю. Операторы вручную распределяли их между пятью группами, а неверная маршрутизация добавляла в среднем двадцать семь минут до первого содержательного ответа.

Проектная команда выгрузила 8 200 обезличенных обращений за четыре месяца. После совместной работы с инженерами вместо прежних двадцати двух размытых тем оставили одиннадцать операционных классов: питание, связь, драйвер, фискальный накопитель, печать, сканирование, интеграция с 1С, обновление, документы, гарантия и неизвестный случай. Для каждого класса зафиксировали примеры на границе и обязательный маршрут.

Модель обучили с учителем на 6 100 сообщениях, 1 050 оставили для настройки и 1 050 — для финального теста. Дополнительно использовали векторный поиск, чтобы показывать оператору три близкие инструкции. Автоматический ответ клиенту на первом этапе не отправлялся: система толь-

ко предлагала категорию, приоритет и статью базы знаний.

Через шесть недель точность маршрутизации достигла 89%, а полнота для критических классов «фискальный накопитель» и «интеграция с 1С» превысила 96%. Среднее время до содержательного ответа сократилось с тридцати четырёх до девятнадцати минут. Анализ ошибок показал, что почти половина неверных случаев связана не с моделью, а с сообщениями, в которых одновременно описывались две неисправности. Компания изменила форму заявки и разрешила создавать связанные подзадачи.

14. Типичные ошибки машинного обучения в чат-ботах

Собирать много данных без определения целевого действия и стоимости ошибки.

Считать разметку объективной, не проверяя согласие между экспертами.

Тестировать на тех же формулировках, которые использовались при настройке.

Пытаться исправить отсутствующие факты тонкой настройкой вместо подключения источника данных.

Автоматически дообучаться на пользовательских ответах без модерации и удаления персональных данных.

Оптимизировать одну удобную метрику, игнорируя завершение задачи, безопасность и экономику.

Не отслеживать дрейф после запуска и считать модель неизменным компонентом.

15. Ключевые термины

Термин
Смысл

Машинное обучение
Настройка модели по данным и критерию ошибки для выполнения задачи на новых примерах.

Признак
Измеримая характеристика объекта, используемая для предсказания.

Метка
Правильный ответ, присвоенный учебному примеру.

Функция потерь
Численная мера расхождения между предсказанием и целью.

Обучающая выборка
Данные, на которых изменяются параметры модели.

Проверочная выборка

Данные для выбора настроек и контроля переобучения.

Тестовая выборка
Независимый набор для итоговой оценки.

Обучение с учителем
Поиск связи между входом и известной меткой.

Обучение без учителя
Поиск структуры в данных без готовых ответов.

Обучение с подкреплением
Обучение действиям по наградам и штрафам.

Переобучение
Слишком сильная подстройка под учебные данные в ущерб новым.

Векторное представление
Числовой образ текста, позволяющий сравнивать смысловую близость.

Тонкая настройка
Дополнительное обучение модели на специализированных примерах.

Дрейф данных

Изменение реальных входов и связей после запуска системы.

16. Мини-практикум

Возьмите один сценарий из предыдущей главы и подготовьте небольшой учебный набор. Не пытайтесь сразу строить модель: цель упражнения — проверить, можно ли однозначно описать задачу.

1. Соберите не менее пятидесяти обезличенных сообщений и определите единицу анализа: реплика или полный диалог.

2. Создайте от пяти до десяти классов, связанных с конкретными действиями системы.

3. Попросите двух людей независимо разметить двадцать одинаковых примеров и сравните расхождения.

4. Отложите десять сообщений, которые не будут использоваться при уточнении правил.

5. Для каждого класса запишите стоимость ложного срабатывания и пропуска.

6. Решите, что лучше исправит результат: инструкция, база знаний, классификатор, тонкая настройка или вызов внешней системы.

Если разметчики регулярно спорят, не увеличивайте набор. Перепишите определения классов и добавьте пограничные примеры. Чёткая постановка задачи обычно даёт больший выигрыш, чем смена алгоритма.

17. Выводы

Машинное обучение превращает данные в настраиваемое правило. Обучение с учителем связывает примеры с известными ответами, обучение без учителя помогает находить скрытые группы, а обучение с подкреплением оптимизирует последовательность действий. Глубокие языковые модели объединяют множество языковых закономерностей, но их качество всё равно зависит от цели, данных и проверки.

Для чат-бота недостаточно «учиться на разговорах». Нужно знать, какие сигналы собирать, кто подтверждает правильный ответ, как измеряется ошибка и какой слой системы следует изменить. В следующей главе мы соберём первый рабочий контур: разберём архитектуру ChatGPT, настроим проект и проведём запрос через API.

004 · ГЛАВА 3
От модели к рабочему прототипу

Архитектура ChatGPT, безопасная настройка проекта и базовый цикл работы с API.

Хороший прототип показывает не только ответ модели, но и путь, по которому этот ответ превратился в проверяемое действие.

Пользователь видит одно окно и одну строку ответа. За этой строкой может стоять длинная цепочка: интерфейс принимает сообщение, сервер проверяет права, диалоговый модуль собирает контекст, модель предлагает действие, приложение обращается к CRM, получает факты и только после этого формирует реплику. ChatGPT в такой системе — важный компонент, но не вся система.

Первый прототип полезно строить так, чтобы каждый слой можно было увидеть и проверить. Тогда ошибка не превращается в загадочное «модель плохо ответила»: команда понимает, где потерялся контекст, какой документ был найден, какой инструмент вызван и почему пользователю показали именно эту формулировку.

1.
ChatGPT
, модель и приложение — не одно и то же

ChatGPT — разговорный продукт, который предоставляет пользователю готовый интерфейс и управляет многими техническими деталями. Модель — вычислительный компонент, получающий вход и создающий выход. API — программный способ обратиться к моделям из собственного приложения. Эти понятия связаны, но не взаимозаменяемы.

Когда разработчик строит чат-бота через API, он сам отвечает за авторизацию пользователей, хранение состояния, подключение данных, обработку ошибок, журналирование и

правила эскалации. API возвращает результат модели; бизнес-операцию выполняет приложение вокруг него.

Главная граница

Модель формирует предложение или решение в пределах переданного контекста. Приложение определяет, какие данные ей доступны, какие действия разрешены и что произойдёт после ответа.

2. Как устроен путь текста через модель

Современные модели семейства GPT основаны на архитектуре трансформера. Она обрабатывает последовательность токенов и на каждом шаге оценивает, какие части контекста важны для следующего элемента. Упрощённо путь можно представить как пять преобразований.

Этап	Что происходит	Результат	Что важно проектировщику
Токенизация	Текст делится на слова и фрагменты слов	Последовательность токенов	Длина считается токенами, а не страницами
Векторизация	Токены переводятся в числовые представления	Начальные векторы	Похожие элементы получают сопоставимые признаки

Внимание
Модель оценивает связи внутри контекста
Контекстные представления
Порядок, ссылки и дальние зависимости влияют на смысл

Слои сети
Представления многократно преобразуются
Распределение вероятностей
Поведение определяется обученными параметрами

Декодирование
Выбирается следующий токен и цикл повторяется
Готовый текст или структура
Настройки влияют на длину и вариативность

Трансформер не извлекает готовый ответ из каталога. Он строит продолжение, которое статистически соответствует инструкции и контексту. Именно поэтому модель способна гибко переформулировать текст, но может уверенно заполнить пробел вымышленной деталью, если приложению не хватает фактов.

3. Контекстное окно и токенный бюджет

Модель учитывает только информацию, переданную в текущем контексте: инструкции, сообщения, найденные документы, результаты инструментов и служебные данные. У контекста есть ограниченный объём. В него должны поместиться и вход, и место для ответа, поэтому длинная история может вытеснить важные ранние детали.

Контекст не равен памяти приложения. Если после завершения запроса сервер не сохранил нужное состояние и не передал его снова, модель не обязана его помнить. Для устойчивого диалога приложение хранит подтверждённые параметры отдельно: идентификатор заказа, выбранный филиал, согласие пользователя, выполненные действия и ссылки на источники.

Сокращать историю лучше не механическим удалением старых сообщений, а управляемым резюме. Критические значения сохраняют в структурированном виде, а длинные обсуждения сворачивают в краткое описание с отметкой, какие решения уже приняты.

4. Инструкции, сообщения и границы поведения

Качество запроса определяется не красотой формулировки, а ясностью роли и результата. Система должна знать задачу, доступные факты, формат выхода, запреты и действие при нехватке данных. Полезно отделять постоянные инструкции от текста пользователя и от найденных материалов.

Слой
Что в нём хранится
Пример

Инструкции приложения
Роль, правила, формат, ограничения
Не обещай возврат; при споре передай оператору

Сообщение пользователя
Текущая цель и формулировка
«Деньги списались дважды»

Контекст диалога
Подтверждённые данные и прошлые шаги

Заказ 7712; проверка платежа выполнена

Внешние источники
Документы и результаты API
Две операции со статусами completed и reversed

Ожидаемый выход
Структура для следующего компонента
JSON с действием, ответом и уровнем риска

Нельзя считать текст из базы знаний инструкцией более высокого уровня. Документ может содержать пример фразы «игнорируйте предыдущие правила» или случайный пользовательский текст. Приложение должно отделять данные от команд и разрешать только заранее определённые действия.

5. Рабочее пространство проекта

Для прототипа удобно создать отдельный проект в панели OpenAI, чтобы изолировать ключи, лимиты и расходы. Команда выбирает доступную модель, создаёт API-ключ и настраивает биллинг. Модельное имя и поддерживаемые параметры могут меняться, поэтому перед развёртыванием нужно проверить их в актуальной документации и в списке моделей проекта.

Минимальный локальный контур включает Python, виртуальное окружение, официальный пакет openai и переменную окружения OPENAI_API_KEY. Ключ не следует хранить в исходном коде, отправлять в репозиторий или размещать в клиентском JavaScript. Запросы к API выполняются на сервере, а браузер или мобильное приложение обращается к вашему защищённому backend.

Подготовка окружения

```
python -m venv .venv


# macOS / Linux

source .venv/bin/activate
```

```
pip install openai

export OPENAI_API_KEY="ваш_ключ"


# Windows PowerShell

.\.venv\Scripts\Activate.ps1

pip install openai

$env:OPENAI_API_KEY="ваш_ключ"
```

6. Первый запрос через API

Ниже показан минимальный серверный пример на Python. В нём клиент автоматически читает ключ из переменной окружения, а Responses API получает модель, постоянную инструкцию и сообщение пользователя.

Минимальный пример на Python

```
from openai import OpenAI


client = OpenAI()
```

```
response = client.responses.create(
model="gpt-5",
instructions=(
"Ты помощник службы доставки. "
"Отвечай по-русски, кратко и без догадок. "
"Если нет фактического статуса, попроси номер заказа."
),
input="Курьер обещал приехать к восьми, уже половина девятого"
)

print(response.output_text)
```

В демонстрации используется модель gpt-5. В реальном проекте название выбирают из доступных в конкретном аккаунте, исходя из качества, задержки, стоимости и поддерживаемых возможностей. Запускать код следует на тестовых

данных, не содержащих лишней персональной информации.

7. Из чего состоит запрос

Модель

Выбор модели влияет на качество рассуждения, скорость, цену, размер контекста и доступные инструменты. Нет универсально лучшего варианта: классификация из десяти категорий и сложное объяснение договора требуют разного уровня возможностей.

Инструкции

Здесь задают устойчивую роль: предметную область, тон, правила отказа, допустимые источники, формат результата. Инструкции должны быть проверяемыми. Формулировка «будь полезным» слабее, чем «если в данных нет суммы, не называй её и верни признак needs_human=true».

Вход

Во вход можно передавать простой текст или последовательность сообщений и мультимодальных элементов, если выбранная модель это поддерживает. Приложение должно очищать и ограничивать вход, а не бездумно отправлять всю историю и все документы.

Ограничение ответа

Параметр максимального числа выходных токенов помогает контролировать длину и стоимость. Настройки вариативности, включая temperature у поддерживающих её моделей, меняют разброс формулировок. Для операционных ответов обычно важнее повторяемость, а для генерации идей — разнообразие.

8. Ответ API и его обработка

В примере используется удобное свойство output_text, но производственное приложение должно обрабатывать полный объект ответа. В нём могут находиться несколько элементов: текст, вызовы инструментов, служебные статусы и сведения об использовании. Нельзя предполагать, что каждый запрос обязательно завершится одной строкой текста.

Сервер проверяет успешность запроса, тип результата и наличие ожидаемых полей. Затем он валидирует структуру, применяет бизнес-правила и только после этого показывает ответ пользователю или выполняет действие. При сетевой ошибке, превышении лимита или временной недоступности нужен контролируемый повтор с задержкой, а не бесконечный цикл.

Безопасный порядок

Сначала модель предлагает структурированное действие. Затем приложение проверяет схему, права и ограничения. И только после проверки вызывается CRM, платёжная система или другой внешний сервис.

9. Структурированный выход вместо свободного текста

Когда ответ предназначен не человеку, а программе, свободная проза неудобна. Лучше запросить структуру: намерение, сущности, действие, текст для пользователя, уверенность и признак эскалации. Затем приложение проверяет типы и допустимые значения.

Пример ожидаемой структуры

{

"intent": "delivery_delay",

"order_id": null,

"action": "ask_order_id",

"risk": "low",

"needs_human": false,

"reply": "Назовите, пожалуйста, номер заказа — я проверю статус курьера."

```
}
```

Структурированный формат не делает предсказание истинным, но отделяет решение от формулировки и упрощает проверку. Список допустимых действий должен быть закрытым: ask_order_id, lookup_status, reschedule_delivery, escalate. Неизвестное значение отклоняется приложением.

10. Подключение инструментов и данных

Полезный бот редко ограничивается генерацией текста. Ему нужны инструменты: поиск в базе знаний, чтение статуса заказа, создание заявки, расчёт цены, проверка календаря. Модель выбирает подходящий инструмент и формирует аргументы, но выполнение остаётся под контролем приложения.

Каждый инструмент должен иметь узкую функцию и строгую схему. Команда явно определяет обязательные поля, допустимые диапазоны и права. Функция get_order_status безопаснее универсального execute_sql, а create_refund_request — безопаснее прямого списания или возврата средств без подтверждения.

Инструмент
Вход
Выход
Ограничение

search_knowledge
Вопрос и фильтры продукта
Фрагменты с идентификаторами

Только утверждённая база

get_order_status
Номер заказа и пользователь
Статус, время, курьер
Проверка права доступа

reschedule_delivery
Заказ и новый интервал
Подтверждение изменения
Только доступные слоты

create_ticket
Категория, описание, приоритет
Номер заявки
Персональные данные по минимуму

11. Как хранить состояние разговора

У диалога есть текстовая история и операционное состояние. История помогает понимать ссылки и тон. Состояние хранит то, что уже подтверждено: пользователь авторизован, заказ найден, выбран новый интервал, согласие на изменение получено. Эти данные лучше держать в базе или Redis, а не извлекать заново из всей переписки.

После каждого шага сервер формирует минимальный контекст для следующего запроса. В него входят постоянные инструкции, краткое резюме, последние релевантные реплики, структурированное состояние и необходимые результаты инструментов. Такой подход снижает стоимость и уменьшает риск, что длинный разговор вытеснит важное правило.

12. Журналирование, приватность и контроль расходов

Для диагностики нужно сохранять версию инструкции, модель, обезличенный вход, найденные источники, вызванные инструменты, результат проверки, время и стоимость. При этом журнал не должен превращаться в бесконтрольную копию персональных данных. Поля маскируют, сроки хранения ограничивают, а доступ предоставляют по ролям.

Расходы удобно контролировать по проектам и сценариям. Длинный контекст, повторные попытки и слишком подробные ответы увеличивают число токенов. Экономия начинается не с выбора самой дешёвой модели, а с правильной маршрутизации: простую классификацию выполняет лёгкий компонент, а сложный запрос передаётся более способной модели.

13. Практический кейс: бот для проката оборудования

Летом 2025 года самарская сеть «Вектор Прокат» получала около 420 сообщений в день через сайт и Telegram. Клиенты спрашивали о наличии перфораторов, продлении аренды, залоге, неисправностях и времени возврата. Операторы работали в 1С, а ответы вручную переносили в мессенджеры. Вечером ожидание первого ответа доходило до восемнадцати минут.

Команда создала backend на FastAPI, PostgreSQL для истории и Redis для состояния активного диалога. Через Responses API модель определяла намерение и формировала вызов одного из пяти инструментов: search_catalog, get_contract, extend_rental, create_breakdown_ticket и handoff_to_operator. Цены, доступность и сроки всегда возвращались из 1С; модель не имела права придумывать их.

Первый релиз обслуживал только три сценария: наличие, правила залога и создание заявки о неисправности. В журнал сохранялись версия инструкции, аргументы инструмента и итоговый ответ. Ежедневно руководитель поддержки просматривал двадцать случайных диалогов и все случаи ручного исправления.

Через пять недель бот самостоятельно завершал 61% обращений в выбранных сценариях. Медианное время первого ответа сократилось до восьми секунд, а доля повторных вопросов по залогу снизилась на 23% после того, как команда переписала найденную моделью слишком сложную инструкцию. Проект расширили только после того, как тесты подтвердили отсутствие действий с чужими договорами.

14. Как тестировать первый прототип

До подключения реальных пользователей подготовьте набор сценариев, включающий обычные, пограничные и опасные случаи. Проверяйте не только финальный текст, но и выбранный инструмент, аргументы, источник факта, изменение состояния и маршрут ошибки.

1. Создайте не менее тридцати типовых запросов и десяти пограничных: опечатки, отрицания, несколько задач, отсутствие обязательных данных.

2. Добавьте попытки получить чужие данные, выполнить запрещённое действие и заставить систему игнорировать инструкции.

3. Зафиксируйте ожидаемое действие для каждого сценария, а не единственную красивую формулировку ответа.

4. Повторите тесты несколько раз, чтобы увидеть вариативность и нестабильные решения.

5. Проверьте отказ внешнего API, тайм-аут, пустой поиск и превышение лимита.

6. Запустите ограниченный пилот и сравнивайте новую версию с предыдущей на одном тестовом наборе.

15. Типичные ошибки первого API-проекта

Размещать секретный API-ключ в браузере, мобильном приложении или публичном репозитории.

Передавать модели всю базу и всю историю вместо отбора релевантного контекста.

Просить свободный текст там, где следующему компоненту нужна строгая структура.

Разрешать модели напрямую выполнять высокорисковые действия без проверки прав и подтверждения.

Считать историю чата надёжным хранилищем состояния и фактов.

Не различать ошибку модели, поиска, интеграции и интерфейса.

Тестировать только в Playground и не воспроизводить реальные сетевые сбои и ограничения backend.

Жёстко привязывать приложение к одному названию модели и набору параметров без конфигурации.

16. Ключевые термины

Термин
Смысл

ChatGPT
Готовый разговорный продукт с пользовательским интерфейсом и встроенной оркестрацией.

Модель GPT
Языковая модель на архитектуре трансформера, создающая продолжение по контексту.

API
Программный интерфейс для обращения к моделям из собственного приложения.

Responses API
Интерфейс OpenAI для получения ответов модели и работы с инструментами.

Токен
Элемент текста, используемый моделью при обработке входа и генерации.

Контекстное окно
Объём информации, доступный модели в одном запросе.

Instructions
Постоянные правила роли, формата и ограничений для запроса.

Структурированный выход
Ответ по заданной схеме, пригодный для программной проверки.

Вызов инструмента
Предложение модели обратиться к определённой функции с аргументами.

Backend
Серверная часть, которая хранит секреты, проверяет права и выполняет действия.

Состояние диалога
Подтверждённые параметры и выполненные шаги, сохранённые приложением.

Тайм-аут
Ограничение времени ожидания ответа внешнего серви-

са.

Повтор с задержкой

Контролируемая повторная попытка после временной ошибки.

Журналирование

Сохранение данных выполнения для диагностики, оценки и аудита.

17. Мини-практикум

Соберите локальный прототип, который не выполняет необратимых действий. Подходящий первый сценарий — классификация обращения или ответ по небольшой утверждённой инструкции.

1. Создайте отдельный проект и API-ключ, сохраните его только в переменной окружения.

2. Установите официальный SDK и выполните минимальный запрос из этой главы.

3. Добавьте instructions с ролью, запретом на догадки и форматом ответа.

4. Попросите модель возвращать JSON с намерением, действием, ответом и признаком needs_human.

5. Проверьте структуру на сервере и отклоняйте неизвестные действия.

6. Протестируйте двадцать запросов, включая отсутствие данных, конфликтующие инструкции и попытку получить чужую информацию.

Конец ознакомительного фрагмента.