

Роман Попов

AI

Для кофейников

Гайд в котором
разберётся даже
твоя бабуля

LLM

MCP

RAG

Attention

MAS

Agents

RMoA

DyLAN

ECON



Роман Попов

**AI для кофейников. От
нейросети до агентов**

«Автор»

2026

Попов Р.

АИ для кофейников. От нейросети до агентов / Р. Попов —
«Автор», 2026

Что на самом деле происходит, когда вы пишете что-то нейросети? Почему она иногда отвечает блестяще, а через минуту уверенно несёт ерунду? Как она учится, откуда берёт знания, зачем ей инструменты и память? И что меняется, когда вместо одного помощника появляется целая команда агентов? «АИ для кофейников» — попытка разобраться в современном ИИ без технического образования и необходимости делать вид, что слова LLM, RAG, Attention или Multi-Agent Systems уже что-то вам объясняют. Здесь всё устроено наоборот: сначала понятная проблема и пример, потом разбираемся, как это работает, и только после этого узнаём, как оно называется. От нейросети до агентов. На пальцах, но без вранья ради простоты.

© Попов Р., 2026

© Автор, 2026

Содержание

Пролог. От автора	7
Глава 1. Как система учится	9
Написать правила или показать примеры	10
Правила ломаются быстрее, чем мы думали	10
Систему готовят на примерах обращений	10
Длинное обращение ставит систему в тупик	11
Как это называется?	11
Что важно не перепутать	11
Попробуй сам	11
Основная мысль	12
Почитать	12
Что изменилось после обучения	13
Ответ приходит за секунды	13
Готовят долго, применяют быстро	13
Прежнее обращение она не открывает	13
Как это называется?	14
Что важно не перепутать	14
Попробуй сам	14
Основная мысль	14
Почитать	14
Проверка на том, чего раньше не было	15
Отличный результат, который ничего не объясняет	15
Часть примеров стоит спрятать заранее	15
Почти одно и то же обращение дважды	15
Как это называется?	16
Что важно не перепутать	16
Попробуй сам	16
Основная мысль	16
Почитать	16
Как появляется новый текст	17
Категория — это ещё не ответ человеку	17
Ответ собирается кусочек за кусочком	17
Вторник, взятый из ниоткуда	17
Как это называется?	17
Что важно не перепутать	18
Попробуй сам	18
Основная мысль	18
Почитать	18
Проверка: собралась ли карта главы 1	19
Задача	19
Если не сходится	19
Что считается пройденным	19
Глава 2. Как работает языковой помощник	20
Кусочки текста и разные продолжения	21
Слово в слово, а ответы разные	21
Текст собирается на ходу, кусок за куском	21

Как это называется?	22
Что важно не перепутать	22
Попробуй сам	22
Основная мысль	22
Почитать	22
Что ему показали прямо сейчас	23
Помощник уверенно называет старое число	23
Три вещи, которые легко перепутать	23
Как это называется?	24
Что важно не перепутать	24
Попробуй сам	24
Основная мысль	24
Почитать	24
Связи между частями текста	25
Кто такой этот «он»	25
Значимость взвешивают заново	25
Как это называется?	26
Конец ознакомительного фрагмента.	27

Роман Попов

AI для кофейников. От нейросети до агентов

AI для кофейников

От нейросети до агентов

Роман Попов

Всё на пальцах, чтобы потом можно было объяснить своей бабушке. Наконец разобраться, как работает весь этот AI. С самого начала, по порядку и без объяснений одной непонятной штуки через три другие

Пролог. От автора

У меня нет технического образования, я не инженер и, честно говоря, довольно плохо знаю матчасть всего, что происходит внутри современного ИИ.

При этом я всю жизнь так или иначе проектирую разные системы: продукты, интерфейсы, команды, операционные структуры, патчи на модульном синтезаторе, в конце концов. Мне всегда интересно понимать, как штука устроена: из каких деталей она состоит, почему работает именно так и что произойдёт, если одну из деталей поменять.

С ИИ у меня долго было странное ощущение. Я много им пользуюсь, что-то с его помощью собираю, читаю про новые подходы, знаю какое-то количество правильных слов. Снаружи всё выглядит вполне прилично.

Но стоит копнуть чуть глубже — довольно быстро обнаруживается место, где понимание заканчивается и начинается «ну, примерно представляю».

И каждый раз, когда я пытаюсь разобраться, происходило одно и то же. Приходишь понять одну незнакомую вещь — её объясняют через две другие незнакомые вещи. Идёшь читать про них — там ещё четыре. Потом появляются сокращения, формулы, греческие буквы, и через двадцать минут у тебя открыто двенадцать вкладок и возникает ощущение, что, возможно, стоило лет двадцать назад всё-таки поступать куда-нибудь ещё.

Меня это бесит ещё со школы.

Потому что я довольно сильно верю в простую вещь: человек способен понять почти что угодно, если ему это нормально объяснить.

Не обязательно сразу знать всю математику под капотом или уметь собрать нейросеть с нуля. Но почти всегда можно сначала понять саму идею: что здесь происходит, какую проблему мы решаем, почему прежнего способа оказалось недостаточно и что придумали вместо него.

Особенно остро я в это упёрся, когда начал собирать open(space).

Я постоянно использую несколько нейросетей параллельно: одна лучше делает одно, другая — другое. Что-то отдаёшь Claude, результат несёшь Codex на ревью, замечания возвращаешь обратно. В какой-то момент понимаешь, что сам работаешь довольно дорогим маршрутизатором для копипаста.

Я подумал: а почему просто не посадить их всех в один чат?

Пусть обсуждают задачу вместе, спорят, проверяют друг друга — и в итоге делают что-нибудь гораздо лучше, чем каждый способен сделать по отдельности.

Звучало прекрасно.

Работало отвратительно.

Вместо суперразума периодически получалось очень плохое совещание. Агенты соглашались друг с другом, повторяли уже сказанное, теряли хорошие мысли, всей компанией уверенно уезжали не туда. Иногда казалось, что несколько хороших моделей, оказавшись в одной комнате, просто коллективно поглупели.

И вот тут стало действительно интересно.

Я полез разбираться, как вообще должны работать группы агентов: кто с кем разговаривает, нужно ли им спорить, кто принимает решение, всем ли полезно слышать всех, почему согласие иногда помогает, а иногда делает всех одинаково неправыми.

А дальше произошло самое интересное. Вопрос «как заставить группу агентов хорошо думать вместе?» довольно быстро превратился в вопрос «а как вообще хорошо думает вместе группа?»

Нужен ли ей лидер? Должна ли структура быть плоской? Полезно ли сначала подумать независимо от остальных? Влияет ли на идею то, кто именно её высказал? Когда консенсус

означает, что группа нашла хороший ответ, а когда — что все просто заразились одной ошибкой?

У людей огромная часть этого спрятана в культуре, социальных навыках и опыте, который мы набираем буквально с детства. А когда пытаешься построить общество из голых агентов, внезапно приходится явно описывать то, что между людьми происходит почти незаметно.

И чем дальше я туда лез, тем очевиднее становилась другая проблема: мне самому не хватает фундамента.

Я знаю отдельные карты, но у меня нет колоды.

Где-то слышал одно. Где-то использую другое. Какой-то термин знаю. Какой-то механизм примерно представляю. Но всё это лежит в голове кучей, а не системой.

А мне кажется, в таких вещах самое важное сначала — собрать скелет.

Представьте перемешанную колоду карт. Пока она лежит кучей, перед вами пятьдесят две разные карточки. Но стоит разложить их по мастям и достоинствам — и вместо пятидесяти двух отдельных объектов появляется структура. Теперь у каждой новой карты есть место.

Вот такую структуру я и решил сначала собрать для себя.

Начать буквально с начала. Что значит «модель обучается». Что после обучения остаётся. Как из этого получилась штука, способная разговаривать. Почему ей понадобились знания извне, инструменты и память. В какой момент получается агент. Зачем агентов становится несколько. И уже потом добраться до всех страшных терминов и сокращений.

С одним правилом:

сначала понять штуку — потом узнать, как она называется.

Не объяснять незнакомое через незнакомое. Не доставать формулу там, где идея прекрасно объясняется без неё. На пальцах — настолько, чтобы, разобравшись самому, можно было объяснить своей бабушке.

Изначально я собирал всё это для себя. Потом подумал, что вокруг наверняка полно таких же людей: любопытных, давно пользующихся ИИ — и местами делающих вид, что понимают происходящее чуть лучше, чем понимают на самом деле.

Так появился AI для кофейников.

Параллельно я пишу о том, что делаю, собираю и пытаюсь понять, в телеграм-канале «Романтика Повседневности».

Я здесь такой же кофейник.

Давайте разбираться вместе.

Как устроен курс. Каждый урок собран одинаково: сначала сцена, в которой прежний способ ломается; потом механизм — за счёт чего работает новый и где перестает; потом один разобранный пример, список того, что легко перепутать, и небольшое задание в другом контексте. Название появляется в самом конце урока, когда механизм уже понятен и ему просто нужна метка.

Читать лучше подряд: каждый урок опирается на предыдущий, слова из будущих уроков в текущем не используются. В конце каждой главы — проверка без вариантов ответа: собралась ли карта.

Ядро курса — семь уроков, без которых остальное не соберётся. Седьмая глава «Глубже» — факультативы по свежим исследованиям, их можно пропустить целиком. Все факты в уроках опираются на исследовательскую базу с указанием границ: одна статья здесь нигде не превращается в закон.

Глава 1. Как система учится

правила, примеры, проверка, первый текст

Мы начинаем с самого простого узла: системы, которая раскладывает обращения по двум папкам. На этом узле видно почти всё, что потом будет масштабироваться. Здесь впервые появляется выбор между правилом, которое написал человек, и поведением, которое подобралось по примерам. Здесь же становится ясно, что после обучения в системе остаётся набор внутренних настроек, а не архив обращений, и что успех на знакомых примерах ничего не говорит о новых.

В этой главе обычно ломается одно и то же: слово «учится» слышится как «учится, как человек», и дальше системе либо слепо доверяют, либо так же слепо не доверяют. Оба варианта мешают проектировать. Полезнее держать в голове механизм: примеры, настройка по ним, применение к новому. Тогда вопросы «а что она выучила на самом деле» и «а как это проверить» задаются сами.

Последний урок главы делает шаг, который меняет всё остальное: от выбора папки к написанию текста. Текст появляется по кусочкам, и складность этого текста не имеет отношения к правильности дат и сумм. Эта мысль будет возвращаться в каждой следующей главе, потому что на ней стоит вся проверка результата.

В главе четыре урока: правила или примеры; что остаётся после обучения; проверка на новом; как появляется текст.

От автора

орен(sparse) я собирал так. Идея, название, логика и дизайн-система были мои, а сам чат, лейауты и код делали модели. И первые версии выглядели как любой AI-чат из интернета: серый список сообщений, поле ввода, кнопка. Совсем не та вещь, которую я представлял.

Я пробовал описывать словами. Что должно быть плотнее, где воздух, какая иерархия, как ведёт себя список, когда участников больше трёх. Получал следующую версию, чуть ближе, и снова не то. Потом перестал описывать и стал показывать: скриншот с пометками, экран из Figma, чужой интерфейс со словами «вот так, только у нас три колонки». И дело пошло. Собрали, я пользуюсь, вижу, что не так, показываю, правим, снова пользуюсь.

То есть спецификация продукта не была написана заранее. Она возникала в контакте с работающей штукой, из примеров и разницы между примером и тем, что нужно. Картинка тоже спецификация, только в ней сидят сотни решений, которые словами я бы описывал неделю, и половину всё равно забыл.

Я дизайнер, для меня это старый способ работать, ИИ мне его не подарил. Но когда я начал разбираться, как система вообще чему-то учится, оказалось, что там с самого начала стоит тот же выбор: расписать правила или показать примеры. И у обоих вариантов есть цена.

С неё и начнём.

Написать правила или показать примеры

Можно самим прописать правило, а можно подготовить систему на примерах и посмотреть, что получится на новом обращении.

Правила ломаются быстрее, чем мы думали

Ты устроился в молодой модный стартап. Называется «Тесно»: сервис, где люди сдают друг другу свободное место, кладовку, гараж, парковку, половину балкона. Заработает ли на этом кто-нибудь, пока непонятно, но на сайте уже написано «инфраструктура эффективного использования городского пространства». Стартап молодой, поэтому должности у тебя нет, зато есть первая задача: семьсот обращений в поддержку разложи по двум папкам — «спор между людьми» и «вопрос про сервис».

Самый очевидный путь — придумать правила самим: если в обращении есть слово «залог», значит это спор, если «как опубликовать» — вопрос про сервис. Компьютер здесь ничего не придумывает: правила написали мы, он их просто применяет.

Держится это недолго. Приходит «залог вернули, спасибо, а как теперь отменить бронь», и это вопрос про сервис со словом «залог». Приходит «сосед сверху говорит, что гараж его, а не тот, кто мне его сдал», и это спор, в котором нет ни залога, ни слова «спор». Можно дописывать исключения одно за другим, но новые формулировки появляются быстрее, чем мы успеваем их учитывать. И это не значит, что правила всегда плохи: если задача простая и условия устойчивые, нескольких правил вполне достаточно. Ломается именно расчёт на то, что мы заранее предугадаем все формулировки.

Систему готовят на примерах обращений

Есть другой способ. Берём много прежних обращений и для каждого заранее отмечаем, в какую папку оно должно было попасть. Дальше запускаем программу, которую тоже написал человек, только написал он в ней не правила про обращения, а способ подбирать поведение по примерам. Внутри у этой программы большой набор чисел, поначалу случайных. Обращение для неё тоже числа: сколько раз встречается каждое слово, какой длины текст, есть ли в нём фотографии. Программа прогоняет числа обращения через цепочку вычислений, в которой участвует её собственный набор, и получает на выходе одно число: насколько обращение похоже на спор.

Потом она сравнивает свой ответ с нашей пометкой. Если промахнулась, она вычисляет, какие из внутренних чисел и в какую сторону надо чуть сдвинуть, чтобы на этом обращении ошибка стала меньше, и сдвигает их. Затем следующее обращение, и так сотни раз. Сама программа при этом не меняется, строчки кода те же; меняются только значения чисел в наборе. Когда говорят «система нашла закономерность», буквально произошло вот это: числа сдвинулись так, что на большинстве примеров вычисление стало давать верный ответ.

После такой подготовки ей дают новое обращение, которого среди примеров не было, и она отвечает тем же вычислением с уже сдвинутыми числами. Мы не записали отдельное правило под каждую возможную фразу — мы написали программу, которая подобрала числа по примерам, и поведение получилось из этого подбора.

Длинное обращение ставит систему в тупик

Предположим, среди примеров, на которых готовили систему, все споры оказались длинными: с историей, с обидой, с тремя фотографиями кладовки, — а все вопросы про сервис, наоборот, короткими: «Как поменять фото в объявлении?» После такой подготовки приходит новое обращение: Галина Петровна на четырёх абзацах и с пятью фотографиями спрашивает, как правильно сфотографировать балкон, чтобы он «смотрелся просторнее». Система кладёт это в споры просто потому, что обращение длинное и с фотографиями, хотя внутри — обычный вопрос про сервис. Она не читала обращение так, как прочитал бы человек: она подобрала закономерность, которая совпадала с папками на примерах, и длина обращений такой закономерностью и оказалась.

Как это называется?

Машинное обучение (Machine learning, ML). Вот вся история, которую мы только что разобрали: программу не пишут по пунктам про каждое обращение. Ей дают примеры с готовыми ответами, а она сдвигает свои внутренние числа, пока вычисление не начнёт совпадать с ответами. Такой способ получать поведение по примерам называют машинным обучением, по-английски machine learning, сокращённо ML.

Что важно не перепутать

- Правило, которое написал человек напрямую, и поведение, которое система получила через настройку по примерам, — разные вещи, даже если результат снаружи выглядит одинаково.
- Слова «система учится» не означают, что она учится так же, как человек, и не означают, что ей можно доверять безоговорочно.
- Какой признак окажется решающим, определяет сравнение с ответами на примерах, и признак может подвести: в примере с длиной обращений он сработал на примерах и не годится для новых случаев.

Попробуй сам

Возьми два случая из жизни того же «Тесно» и сравни их между собой, а потом перенеси сравнение в совсем другую задачу.

- Случай А. Мы явно указали: любое обращение от Галины Петровны отправлять в отдельную папку.
- Случай Б. Мы показали множество прежних обращений и настроили систему различать похожие случаи.
- Вопрос. В каком случае поведение задано написанным правилом, а в каком получено через обучение? Объясни без специальных слов, а потом попробуй то же самое на фотографии: программа определяет, есть ли на снимке велосипед, и мы не расписывали признаки велосипеда вручную, а показали много фотографий с уже известными ответами. К какому из двух подходов это ближе?

Основная мысль

Правило и обучение по примерам — два разных способа получить нужное поведение системы, и результат снаружи может выглядеть одинаково. Пример с размеченными обращениями — только один из способов обучения, дальше будут другие.

Дальше: Что именно меняется внутри системы во время обучения и что в ней остаётся после того, как обучение закончилось?

Почитать

-

[Introduction to Machine Learning \(Google, документация\)](#)

Что изменилось после обучения

Систему один раз подготовили на примерах, а потом она отвечает на новые обращения за секунды — разберёмся, что происходит между этими двумя моментами.

Ответ приходит за секунды

Систему из прошлого урока готовили один раз, показав ей много примеров обращений с уже расставленными папками. Возникает практический вопрос: значит ли это, что при каждом новом обращении она заново пересматривает все прежние примеры и сравнивает с ними?

Если бы это было так, у «Тесно» быстро накопилась бы тысяча старых обращений, и на каждое новое обращение уходило бы всё больше времени — сверять его пришлось бы со всей историей заново. На практике ответ приходит за секунды, даже когда примеров были тысячи. Значит, происходит что-то другое: обращения из подготовки не хранятся где-то рядом и не перечитываются заново при каждом новом обращении.

Готовят долго, применяют быстро

Подготовку и применение стоит разделить на два разных по природе процесса. Подготовка идёт один раз, до начала работы: программа проходит по тысячам примеров, на каждом сравнивает своё вычисление с пометкой и чуть сдвигает внутренние числа. Это медленно: примеров много, и по каждому нужно вычислить, что и куда двигать.

Применение происходит потом, при каждом новом обращении: обращение превращают в числа, прогоняют через ту же цепочку вычислений с уже подобранными внутренними числами и получают ответ. При применении модели эти примеры не открываются заново: их влияние осталось в сдвигах внутренних чисел. Из этого не следует, что от примеров в системе не осталось ничего конкретного: отдельные куски данных она иногда способна воспроизвести, и «не ищет по архиву» и «ничего из обучения не запомнила» — разные утверждения.

Сколько таких чисел? В системе, которая делит обращения на две папки, их могут быть сотни. В системах, которые пишут текст, счёт идёт на миллиарды, но устроено всё так же: набор чисел плюс цепочка вычислений над ними. Удобно представлять прибор с миллиардами ручек, каждую из которых подготовка чуть подкрутила. Только помни, что ручки — картинка. В компьютере это файл с числами и программа, которая умеет их применять.

Прежнее обращение она не открывает

Представь два обращения от одного и того же арендатора с разницей в неделю: «Хозяин гаража не отвечает третий день, а мне нужно забрать велосипед» и «Он так и не ответил, что мне делать?» Система не открывает первое обращение, когда обрабатывает второе: в момент применения она не обращается ни к прежним обращениям, ни к примерам из подготовки. Оба обращения проходят через одни и те же подстроенные настройки, и у системы, которая только выбирает папку, итог совпадает — «спор между людьми», — потому что оба обращения похожи на закономерности, которые в настройках уже закрепились. Оговорка «только выбирает папку» важна: у системы, которая пишет текст, одинаковый вход не обязан давать одинаковый ответ, и почему так — увидим через два урока.

Как это называется?

Модель. Вот то, что остаётся после подготовки: набор подобранных чисел вместе с цепочкой вычислений, а не сами примеры. Этот набор и называют моделью, а числа внутри — её параметрами или весами. Файл с весами можно скопировать на другой компьютер и использовать там ту же модель.

Что важно не перепутать

- Долгая подготовка на множестве примеров и быстрое применение к одному новому обращению — разные по масштабу и по скорости процессы, и их легко перепутать местами.
- Веса — не список готовых ответов на конкретные обращения. В модели не «лежат» прежние примеры, из которых она подбирает подходящий; результат вычисляется заново, когда новый текст пропускают через подобранные числа.
- Устройство вычислений может быть разным, но принцип один: сначала долгая подготовка, потом быстрое применение. Внутри не стоит представлять таблицу ответов: там числа, сдвинутые примерами.

Попробуй сам

Перенеси то же различие в другую задачу.

- Помощник путешественника подсказывает, какой город подходит по описанному бюджету и интересам. Настройку на множестве прошлых поездок он проходил заранее, один раз. Где здесь подготовка, а где применение?
- Он получает совсем новый запрос, которого раньше не видел. Значит ли это, что он заново пересматривает все прошлые поездки, на которых его готовили, — или он просто пропускает новый запрос через то, что уже подобралось?
- Вопрос. Если бы помощник хранил и пересматривал каждую прошлую поездку при каждом новом запросе, что бы это значило для скорости его ответа при тысячах прошлых примеров?

Основная мысль

Один из способов устроить модель — нейросеть: длинная цепочка простых вычислений с миллионами или миллиардами весов, подобранных по примерам. При применении модели ей не нужно заново открывать эти примеры.

Дальше: Как проверить, что модель научилась чему-то полезному?

Почитать

- [Introduction to Machine Learning \(Google, документация\)](#)
- [Курс Hugging Face о работе современных моделей обработки текста \(глава 1\)](#)
- [Vaswani et al. \(2017\) — статья, заложившая основу современных моделей обработки текста](#)

Проверка на том, чего раньше не было

Система почти не ошибается на тех обращениях, на которых её готовили, но настоящие новые обращения вдруг даются ей хуже — стоит разобраться, откуда взялось это расхождение.

Отличный результат, который ничего не объясняет

Систему из прошлых уроков подготовили на всех примерах обращений, которые у нас были, и на них она отвечает почти без ошибок. Кажется, что дело сделано. Но когда ей начинают показывать настоящие новые обращения пользователей, ошибок вдруг становится заметно больше, чем на подготовленных примерах.

Возникает вопрос: если система так хорошо справилась с тем, на чём её готовили, откуда взялись новые ошибки? И можно ли было заметить эту разницу заранее, ещё до того, как система начала работать с настоящими обращениями?

Часть примеров стоит спрятать заранее

Проверять систему на тех же обращениях, на которых её готовили, — нечестный экзамен: это как проверять ученика по тем самым задачам, которые он уже решал с ответами перед глазами. Разница в том, что ученик хотя бы мог случайно наткнуться на настоящее понимание помимо зазубренных ответов, а у системы за высоким результатом вообще ничего, кроме подстроенных под эти же примеры настроек, может не стоять. Такой результат ничего не говорит о том, как система поведёт себя на обращении, которого раньше не видела.

Честная проверка устроена иначе. Из всех собранных примеров заранее откладывают часть в сторону и не показывают системе во время подготовки вообще. Это отдельная проверочная подборка. После подготовки систему проверяют именно на ней — на обращениях, которые она не видела, хотя они были собраны и размечены точно так же, как обучающие.

Если результат на такой подборке заметно хуже, чем на обучающих примерах, значит система не столько нашла общую закономерность, сколько подстроилась под конкретные подготовленные обращения и их особенности — вплоть до случайных деталей, которые к сути задачи не имеют отношения.

Почти одно и то же обращение дважды

Допустим, при разделении примеров в проверочную подборку случайно попали обращения одного и того же арендатора, который писал про один и тот же гараж пять раз за день почти слово в слово: «хозяин не отвечает», «хозяин так и не ответил», «он всё ещё не ответил». Система показывает на этой подборке отличный результат — но только потому, что почти такие же обращения уже встречались среди обучающих примеров: перемешали дни, а не содержимое. Когда систему проверяют на обращениях от новых пользователей, которых не было ни в обучении, ни в этой подборке, результат оказывается заметно хуже. Дело было не в системе, а в том, как разделили примеры: подборка выглядела отдельной, но фактически повторяла обучение.

Как это называется?

Обобщение и переобучение (Generalization and overfitting). Способность переносить найденную закономерность на новые случаи — это обобщение, а её срыв, когда система слишком точно подстроилась под обучающие примеры, — переобучение. Часть примеров заранее откладывают для честной проверки на новых данных.

Что важно не перепутать

- Хороший результат на тех же примерах, на которых систему готовили, — не доказательство, что она справится с новыми обращениями. Это проверка на тех же данных, под которые система уже подстраивалась, а не проверка на новом.
- Проверочная подборка должна быть отделена от обучающих примеров по-настоящему, а не только по названию папки: если в неё случайно попали почти такие же обращения, что и в обучение, утечка ответов останется незамеченной.
- Случайное перемешивание примеров перед разделением помогает не всегда: если несколько примеров сильно похожи между собой или связаны между собой, обычное перемешивание может развести похожие копии по разным сторонам и создать ту же утечку.

Попробуй сам

Перенеси эту же проверку на другую задачу и найди, где в ней может спрятаться утечка.

- Редактор документов подготовили на большом наборе текстов и предложенных исправлений, а проверили на отложенной части того же набора. Позже выяснилось, что несколько документов из проверочной подборки были более ранними черновиками текстов, вошедших в обучение. Что здесь пошло не так?
- Службу поддержки банка обучили на обращениях клиентов за целый год и проверяли на случайно отобранной десятой части всех обращений, без учёта того, что часть обращений были повторными сообщениями одного и того же человека по одному и тому же вопросу. Могла ли здесь возникнуть утечка и какая?
- Вопрос. Что нужно было бы сделать иначе в обоих случаях, чтобы проверочная подборка была честной?

Основная мысль

Хороший результат на обучающих примерах ещё не значит, что система справится с новым обращением — это разные вопросы.

Далее: Как перейти от выбора готовой категории к тому, чтобы система сама написала связный ответ человеку?

Почитать

- [Generalization and overfitting \(Google, документация\)](#)

Как появляется новый текст

Разложить обращение по папке — не то же самое, что написать арендатору связный ответ; посмотрим, откуда вообще берётся этот текст.

Категория — это ещё не ответ человеку

Прежняя система из наших уроков умеет разложить обращение по одной из готовых папок: «спор между людьми» или «вопрос про сервис». Для внутренней сортировки этого достаточно, но человеку нужно не имя папки, а настоящий ответ: вернут ли залог и когда.

Выбрать одну из заранее известных категорий и написать связный ответ своими словами — задачи разной сложности. Категорий у нас было всего две, а возможных связных ответов практически бесконечно много. Откуда система вообще берёт этот текст, если мы не писали ей заранее заготовленные фразы под каждый случай?

Ответ собирается кусочек за кусочком

Ответ система собирает не целиком за один раз, а маленькими кусочками текста один за другим. Сначала появляется первый кусочек, затем следующий, и так далее, пока ответ не будет закончен.

Важная деталь: каждый следующий кусочек выбирается с учётом всего, что уже появилось до него, — то есть с учётом и исходного обращения, и уже написанной части ответа. Из-за этого текст получается связным: система не собирает фразы наугад, а достраивает то, что уже начала.

На каждом шаге подходящих продолжений обычно несколько, а не одно-единственное, и заранее известного правильного продолжения среди них нет. Система выбирает одно из этих вариантов, поэтому одинаковые по смыслу обращения получают разные, но одинаково складные ответы. Как именно устроен этот выбор среди нескольких вариантов, разберём в следующем уроке; сейчас достаточно знать, что он не сводится к одному заранее предопределённому продолжению.

Вторник, взятый из ниоткуда

Арендатор написал: «Когда вернут залог за гараж?» Система начинает собирать ответ кусочек за кусочком: «Ваш», «залог», «будет», «возвращён», «во», «вторник». И получается складное, вежливое предложение: «Ваш залог будет возвращён во вторник». Читается оно гладко и звучит уверенно. Но настоящая дата возврата хранится в платёжной системе «Тесно», а не в том, как система подбирала кусочки текста. Складный и правдоподобный на вид ответ мог собраться и без сверки с этой настоящей датой — тогда во вторник залог может и не вернуться.

Как это называется?

Генерация (Generation). Вот то, что мы только что наблюдали: ответ появляется не целиком и не из готового списка фраз, а кусочек за кусочком, каждый — с учётом всего, что написано раньше. Систему, устроенную для этого, называют языковой моделью.

Что важно не перепутать

- Складность предложения и его фактическая правильность — разные вещи. Ровный, уверенно звучащий текст не гарантирует, что названная дата, номер или условие верны.
- Система не подбирает ответ из заранее заготовленного списка фраз под каждый случай — она строит его кусочек за кусочком заново, с учётом уже написанного.
- Выбор следующего кусочка не сводится к одному заранее известному правильному продолжению. Подходящих вариантов обычно несколько, и то, как именно система выбирает среди них, — отдельный вопрос, не решённый в этом уроке.

Попробуй сам

Перенеси ту же идею на два других примера.

- Помощник-редактор продолжает начатое письмо, каждый раз опираясь на то, что уже написано выше. Где здесь работает та же идея, что и с ответом арендатору «Тесно»?
- Служба поддержки банка складно и уверенно сообщает клиенту дату разблокировки карты. Что стоит проверить отдельно, прежде чем считать эту дату верной?
- Вопрос. Почему складный и связный текст сам по себе не может служить доказательством того, что названный в нём факт правильный?

Основная мысль

Складность получившегося текста — заслуга самого способа собирать его кусочек за кусочком; она ничего не говорит о том, сверялся ли названный в нём факт с настоящими данными.

Дальше: Какими кусочками модель на самом деле обрабатывает текст, и почему один и тот же запрос может дать чуть разные формулировки ответа?

Почитать

- [Курс Hugging Face о работе современных моделей обработки текста \(глава 1\)](#)
- [Vaswani et al. \(2017\) — статья, заложившая основу современных моделей обработки текста](#)

Проверка: собралась ли карта главы 1

Четыре урока позади. Прежде чем идти дальше, проверь без подсказок, что различения работают на чужом примере.

Задача

На складе стоит камера, и программа должна отличать повреждённые коробки от целых. Опиши своими словами, не заглядывая в уроки, как эту задачу можно решить двумя разными способами, и что именно человек делает в каждом из них. Потом ответь на вопросы ниже. Ответы лучше записать: устное «понял» проверку не проходит.

- Что происходит в системе в момент подготовки и что остаётся в ней после? Почему в нашей обученной системе ответ на новой коробке не требует каждый раз открывать архив прежних снимков?

- Все повреждённые коробки на подготовленных снимках стояли в тёмном углу. Что может пойти не так и как ты это обнаружишь до того, как программу включают на складе?

- Теперь программа должна не только пометить коробку, но и написать претензию поставщику. Почему складной текст претензии ничего не говорит о том, верна ли в ней дата поставки?

Если не сходится

Не можешь объяснить, чем правило отличается от обучения по примерам — вернись к уроку «Написать правила или показать примеры». Путаешь подготовку и применение или думаешь, что система хранит снимки — «Что изменилось после обучения». Не видишь ловушку с тёмным углом — «Проверка на том, чего раньше не было». Кажется, что связный текст сам по себе надёжен — «Как появляется новый текст».

Что считается пройденным

Ты пересказал оба способа и три ответа на новом примере, ни разу не открыв уроки. Если пришлось подглядывать хотя бы в один — это ещё не владение, это понимание с подсказкой. Разница станет заметна через неделю.

Глава 2. Как работает языковой помощник

кусочки текста, контекст, связи, настройка поведения

Теперь у нас есть система, которая пишет ответы. Вторая глава разбирает, из чего состоит один её ответ: на какие кусочки режется текст и почему одинаковые запросы дают разные формулировки; что модель получила при подготовке, а что ей показали прямо сейчас; как она связывает части текста между собой; и как уже обученную модель доводят до полезного поведения.

Главная путаница этой главы — между тремя уровнями, которые снаружи выглядят одинаково: настройками модели, приложенным к вопросу документом и инструкцией о поведении. Все три меняют ответ, но живут в разных местах и стоят по-разному. Приложить документ — не переобучить. Попросить быть аккуратнее — не настроить веса. Когда эти уровни различены, половина обсуждений «почему она так ответила» становится короче.

Второй риск — приписать механизму человеческие свойства. У модели есть устройство, которое при обработке каждого кусочка текста взвешивает, какие другие кусочки сейчас важнее. Это техническое решение задачи про местоимения и связи, а не внимание в психологическом смысле. Имя у него, впрочем, именно такое, и на него часто ловятся.

В главе четыре урока: кусочки текста и варианты; что ей показали сейчас; связи между частями текста; как меняют поведение обученной модели.

От автора

Есть у меня штука под названием open(space). Идея простая: несколько агентов сидят в одном пространстве и думают вместе. Я собрал, запустил, написал им первый вопрос и стал ждать умного разговора.

Разговор был. Только не со мной. Они отвечали друг другу, повторялись, соглашались, куда-то уезжали и очень активно генерировали деятельность. Я писал «стоп, вернитесь к теме», и реакция была как у людей на планёрке, когда в дверь заглядывает кто-то не из отдела. Кивнули и продолжили.

Я, конечно, решил, что они недостаточно умные, и пошёл делать их умнее. То есть дописывать инструкции. Как себя вести, что учитывать, чего не делать, как оформлять. Через пару вечеров у каждого агента был свод правил размером с должностную инструкцию. Стало хуже.

Сам я причину не нашёл бы, признаюсь. Я попросил Claude разобрать, почему они меня не слышат, и он показал: мой вопрос приходил к агенту вместе с сорока правилами и был среди них одним пунктом из сорока одного. Слушать пользователя стало одной из задач. Не главной. Мои сообщения буквально терялись в том, что я сам же и написал. Убрали лишнее, агенты снова начали отвечать мне.

Чтобы понять, почему так вообще получается, пришлось разобраться, что помощник получает в момент ответа, где это лежит и что из этого весит больше. Про это вторая глава.

Кусочки текста и разные продолжения

Помощник «Тесно» уже отвечает людям, и один и тот же запрос иногда получает два непохожих ответа. Разберёмся, что эта разница означает, а что — нет.

Слово в слово, а ответы разные

Раз помощник такой умный, ему разрешили отвечать людям. В первый же день пишет Галина Петровна: сдала третий балкон, спасибо. Помощник отвечает тепло, поздравляет и добавляет, что она вошла в топ-3 арендодателей Москвы и что её персональный менеджер Артём свяжется с ней завтра. Топа нет, Артёма нет, в «Тесно» четыре человека. На завтра Галина Петровна звонит и просит Артёма; с тех пор Артём — это основатель. Помощник ничего не выдумывал специально: он продолжил письмо так, как такие письма обычно продолжают, — тот же вторник, только с именем. Спроси его ещё раз, и Артёма, может, не будет. Выдуманным он от этого быть не перестанет.

После Артёма ты два раза подряд отправляешь помощнику один и тот же запрос: «Напиши короткий ответ человеку, который написал через форму на сайте, что владелец гаража третий день не выходит на связь». Формулировка совпадает до последней запятой. Результаты не совпадают. В первый раз помощник обещает напомнить владельцу сегодня же. Во второй раз сразу спрашивает номер брони, чтобы найти владельца. Оба годятся, но слова, порядок мыслей и даже следующий шаг различаются.

Как это трактовать: новая идея — спросить номер брони — или просто другие слова про то же самое? От ответа зависит, брать ли второй вариант на вооружение.

Текст собирается на ходу, кусок за куском

Когда помощник создаёт текст, он не имеет дела с целыми словами как с неделимыми единицами. Текст заранее режется на более мелкие кусочки, и они не обязаны совпадать с привычными словами. Для примера: слово «кладовка» условно можно разложить на кусочки «кла» и «довка», а короткое слово «да» вообще не делить. Это придуманное разбиение только для иллюстрации идеи, а не описание конкретной системы — оно зависит от языка и настройки и не совпадает одинаково даже у разных помощников. У каждого такого кусочка есть свой номер в заранее составленном списке, и этот номер — обычное число — компьютер получает на входе вместо буквенного текста, а дальше работает с числами, не с буквами.

Вторая часть устройства касается того, как из уже написанного куска текста получается следующий. Помощник не выбирает единственно возможное продолжение: на каждом шаге программа рассчитывает для возможных следующих кусочков числа: какие продолжения здесь подходят сильнее, а какие слабее. Из этих чисел получают шансы и разыгрывают следующий кусочек, как жребий: более подходящий выпадает чаще, но не гарантированно. Из-за этого при полностью одинаковом запросе цепочка выбранных кусочков может пойти другим путём уже с первых шагов, а дальше расхождение только нарастает, потому что каждый следующий кусочек создаётся с оглядкой на все предыдущие.

Как выбирать по этим шансам — отдельное решение, а не свойство помощника. Можно всякий раз брать кусочек с самым большим числом, и тогда на одинаковый запрос ответ будет повторяться слово в слово. При генерации розыгрыш можно использовать, а можно и нет; насколько сильно ответы будут различаться, зависит от настроек.

Вот оба ответа целиком. Первый: «Здравствуйтесь! Спасибо, что написали. Мы напомним владельцу гаража ответить вам сегодня же». Второй: «Здравствуйтесь! Подскажите,

пожалуйста, номер вашей брони — так я быстрее найду владельца и передам ему, что вы ждёте ответа». Но только второй меняет ход дела: через форму на сайте пишут без входа в аккаунт, и без номера брони проверка не может начаться. Получи ты первый вариант оба раза, ты бы не заметил, что для проверки не хватает детали.

Как это называется?

Токен (Token). Вот вся штука, которую мы только что видели: текст режется на кусочки меньше слова, для возможных следующих кусочков помощник рассчитывает числа, из которых получают шансы, и по выбранному правилу берёт один. Кусочки называют токенами, а розыгрыш по этим шансам — сэмплированием.

Что важно не перепутать

- Разная формулировка одного и того же решения — не то же самое, что разное решение по сути. Прежде чем радоваться новой идее, проверь, изменилось ли предложение, а не только слова вокруг.
- Несколько ответов, полученных подряд от одного и того же помощника, не превращаются в мнения нескольких независимых специалистов. Это варианты одного и того же процесса, и они вполне могут ошибаться в одну и ту же сторону.
- Кусочки, на которые режется текст внутри помощника, не обязаны совпадать со словами — это условность, которая зависит от конкретной настройки и языка, а не удобные слоги для чтения.

Попробуй сам

Перенеси это различие в другую ситуацию. Представь помощника путешественника, который дважды подряд отвечает на один и тот же вопрос про визу в выбранную страну.

- Случай А. В первый раз помощник пишет: «Виза не нужна на срок до девяноста дней». Во второй раз: «На срок до трёх месяцев виза не требуется». Числа и смысл совпадают, слова разные.
- Случай Б. В первый раз помощник пишет, что виза не нужна. Во второй раз — что визу нужно оформить заранее через консульство.
- Вопрос: в каком случае перед тобой просто новая формулировка, а в каком — другая, причём противоречащая первой, гипотеза? Что стоило бы сделать дальше, прежде чем довериться любому из двух ответов?

Основная мысль

Разные слова ещё не значат разное решение — иногда это просто другой вариант, выпавший при розыгрыше тех же шансов. Само разнообразие формулировок ничего не говорит о том, правильно ли решена задача человека.

Дальше: Откуда вообще помощник знает, какую именно задачу мы ему только что поставили?

Почитать

- [Tokenizers \(Hugging Face, 2026\)](#)

Что ему показали прямо сейчас

«Тесно» обновило правило страховки уже после того, как языковой помощник закончил свою подготовку. Разберёмся, что в этой ситуации меняется, а что остаётся прежним.

Помощник уверенно называет старое число

«Тесно» меняет правило страховки: раньше вещи в кладовке были застрахованы на двадцать тысяч рублей, теперь — на пятьдесят. Правило поменялось уже после того, как языковой помощник закончил свою подготовку. Ты на всякий случай спрашиваешь у помощника, на какую сумму застрахованы вещи, и тот уверенно называет двадцать тысяч — старое число.

Тогда ты вставляешь в тот же самый разговор абзац с новым текстом правил и повторяешь вопрос. На этот раз помощник называет пятьдесят тысяч верно. Возникает вопрос: помощник только что заново обучился новому правилу, или произошло что-то совсем другое, что исчезнет, стоит только закрыть этот разговор?

Три вещи, которые легко перепутать

Стоит различать три разные вещи, которые легко смешать в одну. Первая — то, что закладывалось во время подготовки помощника: тогда на множестве примеров подбирались его внутренние настройки, и именно они определяли уверенный, но устаревший ответ про двадцать тысяч. Эти настройки не меняются от одного разговора к другому и не меняются от того, что кто-то вставил свежий абзац с правилами.

Вторая вещь — то, что оказалось приложено к конкретному обращению именно сейчас: свежий текст правил страховки, история переписки с этим арендатором, номер брони. Это не переучивание помощника — это разовое дополнение к тому, что он видит в момент ответа. Сами по себе эти пятьдесят тысяч не попадут в настройки помощника: в новом разговоре он узнает их, только если система снова даст ему этот абзац, — иначе снова назовёт старое число, потому что внутри для него ничего не поменялось.

Третья вещь — отдельная инструкция о том, как вообще себя вести: например, указание сверяться с приложенным документом раньше, чем с общими знаниями, или писать коротко и вежливо. Это тоже не переучивание, а отдельная договорённость, которую можно менять от разговора к разговору, не трогая ничего внутри самого помощника.

Важно и другое: большой объём приложенных сведений не гарантирует, что каждая их часть будет использована одинаково хорошо. Если в один разговор вставить сразу весь договор со страховой на полсотни страниц, помощник может упустить именно нужный абзац про пятьдесят тысяч среди остального, особенно если он спрятан в середине длинного текста.

Ты присылаешь помощнику такое сообщение: «Новое правило: вещи в кладовке застрахованы на пятьдесят тысяч рублей, старое правило про двадцать тысяч больше не действует. Арендатор хранит велосипед за тридцать пять тысяч и спрашивает, покрывает ли страховка — покрывает?» Помощник отвечает верно: раз велосипед стоит тридцать пять, а новый предел — пятьдесят, покрывает. Стоит заметить: правило не осело в его настройках навсегда — оно просто оказалось перед ним именно в этом сообщении, и ответ получился из него. С этого дня абзац про страховку прикладывают к каждому ответу.

Как это называется?

Контекст (Context). Вот то, что мы только что различили: всё, что помощник получает для текущего ответа — просьбу, приложенные сведения, предыдущую переписку, — называют контекстом. Саму просьбу или инструкцию, которую мы ему даём, часто называют промптом. Ни то ни другое не меняет настройки помощника.

Что важно не перепутать

- Приложить к разговору свежий документ — не то же самое, что заново обучить помощника. Один раз добавленные пятьдесят тысяч не остаются с ним для следующего, отдельного разговора.
- Инструкция о том, как себя вести, и сведения о конкретном деле — разные вещи, хотя обе появляются в одном и том же обращении к помощнику.
- Больше приложенного текста не значит, что каждая его часть будет одинаково замечена и учтена. Почему часть текста иногда теряется среди остального — работа отдельного механизма, до которого мы доберёмся в следующем уроке.

Попробуй сам

Перенеси это на редактор документов, который помогает готовить договоры для клиентов. Ниже — две ситуации и вопрос к ним.

- Случай А. Редактор с самого начала умел определять типовые пункты договора — этому его научили заранее на множестве примеров.
- Случай Б. Сегодня в разговор с редактором вставили черновик конкретного договора этого клиента вместе с короткой просьбой: сначала перечислить риски, а предложения по правке — только после этого списка.
- Вопрос: какая часть ситуации про заранее выученные настройки редактора, а какая — про то, что оказалось приложено к разговору именно сейчас, включая саму просьбу о порядке ответа? Что из этого исчезнет, если начать совершенно новый разговор?

Основная мысль

Держи в голове разницу между тремя вещами: что впечатано в настройки помощника надолго, что система даёт ему прямо сейчас в виде контекста, и отдельную договорённость о том, как себя вести. Последние два не меняют саму модель.

Дальше: Как слова внутри одного такого текста вообще помогают понять связи между ними?

Почитать

- [Effective context engineering for AI agents \(Anthropic, 2025\)](#)

Связи между частями текста

В переписке «Тесно» слово «он» иногда означает велосипед, а иногда — арендатора, и без соседних слов не разобрать, что имеется в виду. Разберёмся, как помощник вообще с этим справляется.

Кто такой ЭТОТ «он»

Владелец гаража пишет в «Тесно»: «Арендатор написал, что велосипед стоял в гараже, а теперь его нет. Где он сейчас?» Само по себе слово «он» ничего не говорит: это может быть вопрос про велосипед, а может — про арендатора. Только по соседним предложениям понятно, что речь, скорее всего, про велосипед: именно про его пропажу говорилось прямо перед этим. Прочитай помощник это иначе — и в споре о пропаже он ответит не на тот вопрос.

Похожая путаница легко случается и в более длинных обращениях. Арендатор упоминает сразу две брони, а затем пишет «отмените её» — и неясно, какая из двух броней имеется в виду, пока не сопоставить эту фразу с предыдущими предложениями.

Значимость взвешивают заново

Чтобы дать связный ответ, помощнику приходится решать похожую задачу при обработке каждого маленького кусочка текста. Обработывая очередной кусочек, система заново рассчитывает, какие из предыдущих частей текста сейчас могут быть полезнее: одни получают больший вес, другие меньший, и их сведения смешиваются с учётом этих весов. Поэтому при обработке кусочка «он» сведения о велосипеде могут повлиять сильнее, чем сведения об арендаторе. В настоящей системе таких сравнений много: они идут параллельно и повторяются слой за слоем.

Это происходит заново на каждом шаге, а не один раз для всего текста целиком: слово «он» в одном месте получит одни веса, а то же самое слово чуть дальше — совсем другие. Это полезная первая картина, и она намеренно узкая: такие веса связывают не только местоимения с именами, но и любые части текста, которые оказались полезны друг другу при обучении, и ни один отдельный вес сам по себе не означает решения «это слово про велосипед». Разрешение местоимений — самый наглядный случай, а не единственная работа механизма.

Это устройство иногда хочется представить как нечто похожее на человеческую сосредоточенность или переживание процесса чтения. Граница здесь важна: пересчёт значимости происходит по одному и тому же принципу при обработке любого текста — интересного или скучного, важного или нет — и не предполагает ни осознанного выбора, ни усталости, ни вообще какого-либо переживания происходящего. Это техническое устройство, а не аналог человеческой психики.

Возьмём фразу: «Владелец обещал перезвонить, но так и не позвонил, и арендатор теперь спрашивает, где он». Слово «он» стоит рядом сразу с двумя кандидатами — «владелец» и «арендатор». Для простоты представим, что в одном из таких сравнений больший вес получает место про владельца: это помогает следующему вычислению удержат связь с тем, кто должен был позвонить, и в ответ перевешивает «владельца». Если бы фраза была другой — «Владелец передал ключ арендатору, и тот теперь пишет, что доволен», — больший вес, скорее всего, получило бы место про арендатора, хотя предложения устроены очень похоже. Разница не в самих словах, а в том, что оказалось рядом по смыслу в конкретном предложении.

Как это называется?

Механизм внимания

Конец ознакомительного фрагмента.

Текст предоставлен ООО «Литрес».

Прочитайте эту книгу целиком, [купив полную легальную версию](#) на Литрес.

Безопасно оплатить книгу можно банковской картой Visa, MasterCard, Maestro, со счета мобильного телефона, с платежного терминала, в салоне МТС или Связной, через PayPal, WebMoney, Яндекс.Деньги, QIWI Кошелек, бонусными картами или другим удобным Вам способом.