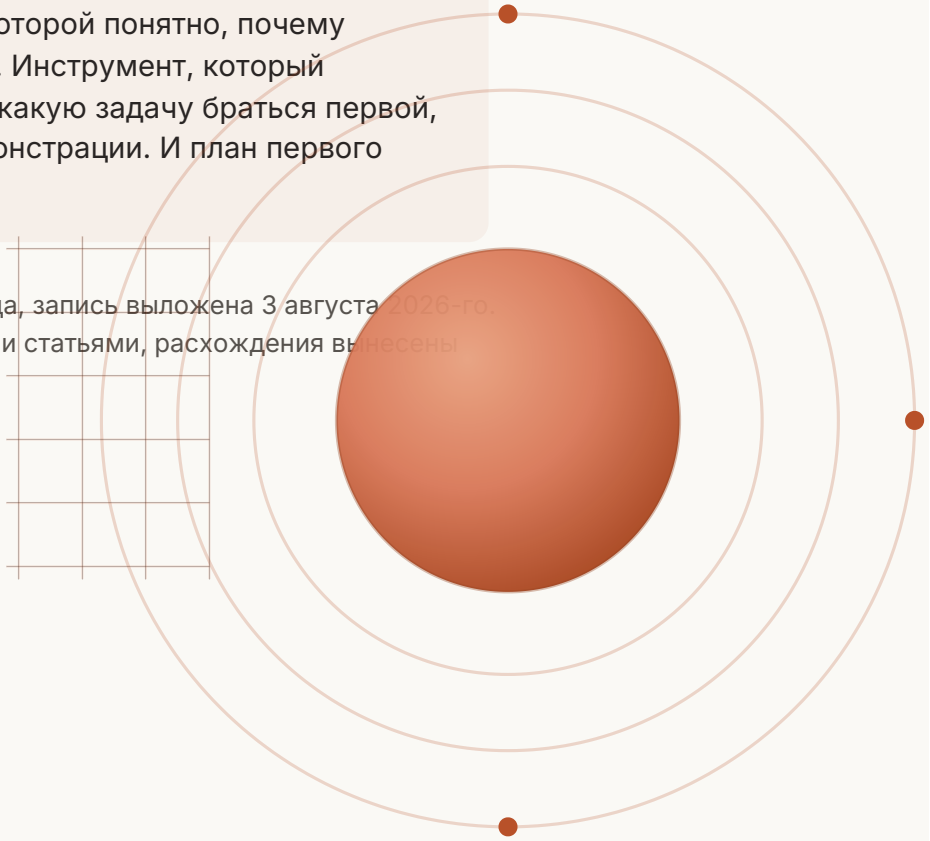
ПОД КАПОТОМ

Самоулучшающиеся AI-агенты: как устроен контур

Разбор первой лекции Stanford CS329A. Не презентация про «умный ИИ», а механика того, что делает систему способной находить, проверять и исправлять результат.

Что вы унесёте. Механику, из которой понятно, почему системы ведут себя именно так. Инструмент, который отвечает на главный вопрос: за какую задачу браться первой, чтобы не сжечь бюджет на демонстрации. И план первого пилота на один рабочий день.

Лекция прочитана 22 сентября 2025 года, запись выложена 3 августа 2026-го. Числа и названия сверены с первичными статьями, расхождения вынесены отдельной страницей.



Under the Hood · Илья Утов
@gorilla_under_hood

КАК ЧИТАТЬ ЭТОТ УРОК

Сначала карта. Потом словарь в действии.

Лекция идёт от роста моделей к обучению, затем к поиску во время ответа, reasoning, агентам и проверке. Ридер сохраняет этот маршрут, но останавливается там, где новичку обычно не хватает механики.

СТЭНФОРД · ТАЙМКОД

Что говорит лекция

Смысловой перевод конкретного фрагмента. Формулировки не выдают авторский вывод за позицию преподавателей.

ПОД КАПОТОМ

Что это значит

Разжёвывание, контрпример или граница применимости. Это слой адаптации, а не цитата Stanford.

СДЕЛАЙТЕ

Как проверить понимание

Небольшая работа на своей задаче: цель, вход, проверка, бюджет или стоп-условие.

ОТКРЫТО

Где нельзя фантазировать

Детали современных reasoning-моделей публично раскрыты не полностью. Если лекция не знает ответа, ридер не додумывает его.

Семь слов, без которых дальше не читается

Модель - программа, которая по контексту предсказывает следующий кусочек текста. **Токен** - этот самый кусочек, не обязательно целое слово; из токенов складывается длина и цена ответа. **Веса** - числа внутри модели, которые меняются только при обучении. **Контекст** - то, что вы положили в запрос сейчас; он меняет ответ, но не трогает веса. **Верификатор** - независимая проверка результата: тест, расчёт, сверка с источником или человек. **Агент** - модель вместе с целью, инструментами, памятью и циклом проверки. **Контур** - замкнутая петля «действие, наблюдение, поправка», ради неё и весь урок.

Пять различий, с которыми надо выйти

Обучение меняет веса, контекст меняет текущий запрос, test-time compute покупает попытки, агент действует в среде, а верификатор связывает действие с реальной целью. Эти слова нельзя использовать как синонимы.

Часть	Таймкод	Главный вопрос
Масштаб и контекст	04:00-10:00	Что стало лучше до вашего запроса?
Post-training	10:00-20:00	Почему базовая модель стала удобным помощником?
Поиск и reasoning	20:00-42:00	Как найти и выбрать редкое верное решение?
Агенты и проверка	42:00-1:02:30	Что разрешить системе делать и как остановить?

ПОД КАПОТОМ · ФУНДАМЕНТ

Одна механика, из которой следует половина урока.

Дальше будет много слов: цепочка рассуждений, температура, повторные попытки, «модель думает дольше». Все они описывают следствия одного устройства, поэтому разберём его сразу.

1. **КОНТЕКСТ** — запрос и всё уже написанное
2. **СПИСОК ПРОДОЛЖЕНИЙ** — у каждого своя вероятность
3. **СЛУЧАЙНЫЙ ВЫБОР** — ручка называется температурой
4. **ДОПИСАЛИ КУСОЧЕК** — и всё сначала

Модель не придумывает ответ целиком, чтобы потом его напечатать. Она пишет по одному кусочку за раз, и на каждом шаге заново смотрит на всё написанное, включая собственные предыдущие слова. Кусочек называют токеном, он не обязательно совпадает со словом.

Написанное тянет за собой

Написав «23 - 20 = 3», модель дальше считает от тройки. На эту запись она и опирается на следующем шаге. На этом держится вся история с цепочкой рассуждений.

Ответ не один

Выбор следующего кусочка случайный, поэтому один и тот же вопрос дважды даёт разный текст. Ради этого и гоняют один запрос несколько раз: каждый прогон идёт своим путём.

Длина стоит денег

Каждый кусочек это время и оплата. Поэтому за «пусть подумает подольше» вы платите, а reasoning-модели дороже обычных при той же задаче.

Что здесь не происходит

Модель не ищет ответ в базе и не сверяется с источником, если ей этого не дали. Она продолжает текст так, чтобы он выглядел правдоподобно. Отсюда практический вывод: правдоподобие и правильность это разные вещи, и различить их может только проверка снаружи.

Проверьте на своём интерфейсе

Задайте один и тот же вопрос дважды и сравните ответы дословно. Если тексты разошлись, вы только что увидели ту самую случайность, ради которой в следующих разделах генерируют десятки и тысячи вариантов.

СТЭНФОРД · 04:00-10:00

Scaling laws: модель росла до того, как вы открыли чат.

Преподаватели начинают с перехода от BERT и T5 к большим языковым моделям. Закономерность измеряли по трём осям отдельно: вычисления, потраченные на обучение, объём данных и число параметров. По каждой оси картина одинаковая: чем больше ресурса, тем ниже ошибка предсказания следующего кусочка текста, причём падает она плавно и предсказуемо. Это было одним из главных двигателей прогресса 2018-2024.

Модели росли от сотен миллионов параметров до десятков и сотен миллиардов. Лекция специально оговаривает: подпись GPT-4 на старом графике - оценка, а не публичная спецификация. Смысл здесь не в соревновании брендов, а в том, что масштаб обучения позволил модели обобщать на новые задачи.

После роста появились zero-shot, few-shot и новые формы поведения. Лекторы связывают это с тем, что крупная модель может извлекать правило из контекста и применять его там, где её не дообучали отдельно.

Что такое test loss

Во время предобучения модель угадывает следующий кусочек текста. **Test loss** - мера ошибки на отложенных данных. Меньше loss означает, что модель лучше решает именно эту учебную задачу. Это не измеритель полезности, честности или безопасности.

Параметры / веса
Числа внутри нейросети, которые меняются при обучении и хранят выученные закономерности. Не путать с текстом промпта.

Compute
Вычислительный бюджет: работа чипов и время. В лекции важно различать compute на обучение и compute на один ответ.

Scaling laws
Эмпирические закономерности из определённых экспериментов. Не закон природы и не обещание «больше параметров решат всё».

Что закономерность показывает

Ошибка предсказания следующего кусочка текста падает вместе с ростом ресурсов. Это измерено и воспроизводится.

Чего она не показывает

Что рост сам по себе создаёт агента, делает систему безопасной или улучшает вашу конкретную задачу. Меньше ошибка на учебной задаче не равно больше пользы в работе.

Объяснение собрано по содержательному слайду лекции. Числовые значения осей в исходнике не раскрыты, поэтому здесь их нет.

СТЭНФОРД · 07:00-10:00

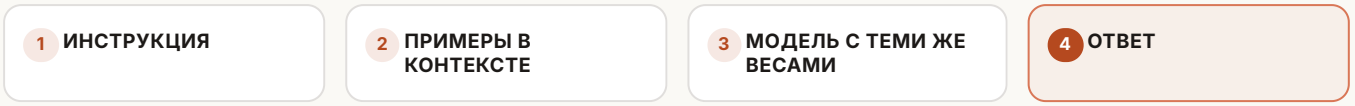
Zero-shot, few-shot и CoT: контекст меняет маршрут, не веса.

Zero-shot

В слайде модель получает инструкцию «переведи с английского на французский» и слово `cheese`. Если она отвечает по-французски без примеров этой задачи в запросе, это zero-shot. Она опирается на обобщения, усвоенные до этого запроса.

Few-shot

В запрос добавляют пары вроде `sea otter → loutre de mer` и `peppermint → menthe poivrée`, затем снова спрашивают `cheese`. Примеры задают шаблон текущему ответу.



Кто пишет рассуждение

В примере с яблоками легко потерять автора цепочки. В образце, который человек кладёт в запрос, разбор пишет **сам человек**: показывает ответ вместе с ходом. Дальше модель видит формат и для нового вопроса пишет свой разбор **сама**.

Работает это не потому, что модель попросили стараться. Она продолжает текст: записав «23 - 20 = 3», дальше опирается на тройку, а не прыгает к ответу наугад. Без разбора она отвечает 27, с разбором получает 9.

Граница приёма

Такой разбор можно проверять как текст и как вычисления. Но он не стенограмма внутренних причин модели: это её рабочие записи, а не отчёт о том, что происходило внутри.

На моделях меньше 10 млрд параметров приём не просто бесполезен, а вреден: у LaMDA 8B точность на GSM8K падает с 3,2 до 1,6 процента. Авторы называют порог примерно в 100 млрд, с которого выигрыш становится устойчивым.

Два приёма, которые постоянно путают

Показать образец с разбором - это few-shot chain of thought. Запрос длиннее и дороже, зато формат задаёте вы. **Попросить рассуждать вслух** - это zero-shot: дешевле, а ход решения выйдет какой выйдет. В лекции разбирают первый вариант, в обычной переписке люди пользуются вторым.

Для писем и редактуры выигрыша никто не обещал. Зато есть побочная польза: когда просите показать ход, вы видите, из чего она исходила, и ловите ошибку до отправки.

Не путать

Few-shot меняет контекст одного вызова. **Fine-tuning** запускает обучение и меняет веса так, что эффект сохраняется после закрытия чата. Фраза «модель научилась на моих двух примерах» для few-shot технически неверна.

ЧАСТЬ II · 10:00-20:00

Знать текст – ещё не значит быть полезным.

ChatGPT стал массовым не только из-за масштаба базовой модели. Лекция переводит фокус на данные, инструкции, предпочтения и то, какую цель мы вообще задали системе.

БАЗА

Предобучение учит продолжать текст, но не отвечает на вопрос, какую помощь хочет человек.

ПОВЕДЕНИЕ

Instruction tuning показывает формат «задача - полезный ответ» и может включать шаги решения.

КРИТЕРИЙ

RLHF меняет направление по человеческим сравнениям, но не превращает награду в истину.

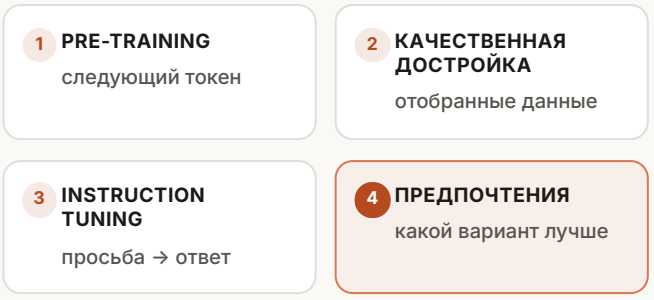
СТЭНФОРД · 10:00-17:30

Post-training: после языка модель учат поведению.

В лекции запуск ChatGPT в ноябре 2022 года используется как исторический поворот. Существенную роль сыграли instruction tuning и RLHF: базовая модель GPT-3 уже знала язык, но не обязана была следовать просьбе, выбирать полезный формат или избегать вредного ответа.

Предобучение строится вокруг next-token prediction на огромном и разнородном корпусе. Оно даёт статистические знания и способность продолжать текст. Но вопрос «чего на самом деле хочет пользователь?» в этой цели отсутствует.

После этого идёт донастройка на более качественных и специализированных данных. Лекторы называют математику, код, творческие тексты и тщательно курируемые корпуса. Качество и права на данные стоят дорого именно потому, что этот слой меняет полезное поведение заметно сильнее, чем случайный текст из интернета.



Pre-training

Долгое обучение на широких данных. Учит языковой модели предсказывать продолжение.

Fine-tuning

Дополнительное обучение уже предобученной модели на более узком или качественном наборе.

Instruction tuning

Донастройка на примерах вида «инструкция, вопрос, хороший ответ». При необходимости в образце есть и шаги решения.

Alignment

Попытка приблизить поведение системы к человеческим целям и ограничениям. Лекция прямо называет это открытой проблемой, а не решённой задачей.

> **Точная граница лекции**
>
> «Сделать модель более согласованной с предпочтениями» не значит доказать, что она правдива, безопасна или разделяет человеческие ценности во всех ситуациях.

СТЭНФОРД · 17:00-22:30

RLHF и DPO: кто решил, что этот ответ лучше?

Обучение по предпочтениям не получает готовую правильную фразу. Оно получает сравнение нескольких вариантов и пытается направить модель в сторону предпочитаемого поведения.

RLHF по шагам

1. Модель отвечает на один запрос несколькими способами.
2. Человек или эксперт ранжирует варианты.
3. Из сравнений строят модель награды: приближение к тому, что оценщик сочтёт хорошим.
4. Модель донастраивают, чтобы увеличивать ожидаемую награду.

После третьего шага живой человек из цикла выходит. Дальше модель учится не против людей, а против второй сети, которая их мнение приближает. Всё, в чём эта сеть ошибается, становится лазейкой.

Где здесь DPO

Direct Preference Optimization тоже использует пары «предпочтительный / менее предпочтительный» ответов, но в учебной схеме лекции не требует отдельной явной цепочки reward model → reinforcement learning.

Это не второе слово для RLHF. Общие данные могут быть похожи, способ настройки отличается.

АВТОРСКАЯ РАЗМЕТКА ОБХОДОВ

Критерий	Что он может поощрять	Как его обойти
Correctness	Фактическую верность в размеченных задачах	Неверный факт, который не попал в проверку
Helpfulness	Понятность и полноту ответа	Уверенный, но ложный совет
Specificity	Конкретные детали и структуру	Правдоподобную выдумку
Harmlessness	Осторожное поведение	Избыточный отказ там, где задача безопасна

ПОД КАПОТОМ: награда не равна цели

Reward model видит то, чему её научили. Если человеку приятнее гладкий ответ, система может стать лучше в производстве гладкости. Поэтому оценка предпочтений полезна, но не заменяет внешний факт, тест или ответственность за действие.

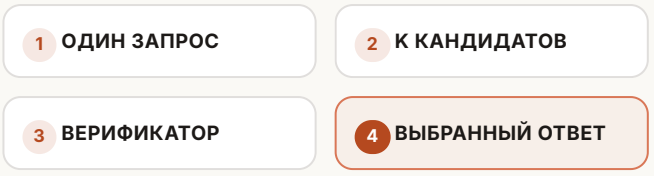
СТЭНФОРД · 20:00-32:20

Large Language Monkeys: купить не один ответ, а ПОИСК.

Работа лаборатории Азалии Мирхосейни называется *Large Language Monkeys*. Метафора отсылает к бесконечной обезьяне, которая случайными нажатиями когда-нибудь напечатает любой текст. Здесь «обезьяна» - LLM, но поиск не бесконечный и не перебирает все строки.

Система спрашивает модель много раз, получает независимые кандидаты, пропускает их через верификатор и отдаёт выбранный результат. Для кода таким верификатором могут быть unit-тесты: несколько реализаций запускаются на тестах, прошедшая выбирается.

Модель даёт разные варианты не потому, что хранит несколько истин. Причина в том самом выборе следующего кусочка текста: он вероятностный. **Температура** это ручка случайности при таком выборе. На нуле модель берёт самое вероятное продолжение, и повторный запрос вернёт почти тот же текст; чем выше, тем больше шансов у редких вариантов, а вместе с разнообразием растёт доля мусора. Проверить можно за минуту, в любом интерфейсе, где эта ручка есть. В лекции звучит оговорка, что выше примерно 1,2 качество резко падает, но это устная реплика: в самой работе температуру перебирали только на SWE-bench Lite, брали 1,0 / 1,4 / 1,6 / 1,8 и остановились на 1,6. Универсальной границы здесь нет, её надо подбирать под свою задачу.



В самой работе наборов задач пять: GSM8K (школьная арифметика в задачках), MATH (олимпиадная математика), MiniF2F-MATH (формальные доказательства), CodeContests (спортивное программирование) и SWE-bench Lite (починка реальных багов в открытых репозиториях). На первых четырёх число выборок доводили до 10 000, на SWE-bench Lite ограничились 250 попытками из-за цены. При одной попытке GPT-4o обходит все более слабые модели, но с ростом числа выборок **Llama-3-8B-Instruct, Llama-3-70B-Instruct и DeepSeek-Coder-V2-Instruct** по покрытию превосходят его результат с одной попытки. На части задач доля верных решений среди тысяч выборок падает до одного процента и ниже.

Аккуратный вывод

В распределении модели могут быть редкие правильные траектории, которых не видно в одиночном ответе. Это не означает, что маленькая модель «лучше вообще» и не отменяет цену поиска.

Что даёт больше k

Больше coverage: шансов, что удачный кандидат вообще появился.

Что k не даёт

Способность выбрать удачный вариант, низкую стоимость, приемлемую задержку и честный критерий.

СТЭНФОРД · 29:00-32:00

Мост к самоулучшению: поиск может стать данными

Проверенные траектории могут стать синтетическими примерами для следующего fine-tuning. Это исследовательское направление, не доказанный вечный цикл.

СТЭНФОРД · 25:00-32:20

`pass@k` не равно `pass@1`. Это две разные победы.

Coverage / pass@k

Среди `k` попыток хотя бы одна оказалась правильной. Для исследователя это показывает скрытый потенциал генератора, если существует оракул или сильная проверка.

Пример: среди 100 вариантов нашёлся один корректный скрипт.

Одна попытка / pass@1

Правильным оказался единственный кандидат, сгенерированный для задачи. Это метрика качества одного ответа **до** отбора среди множества вариантов.

Пример: первая и единственная генерация скрипта проходит проверку.

16

задач из 100 система чинит **с одной попытки**

56

из 100, если дать **250 попыток** и отобрать по тестам

0

из 100, **если отбирать нечем**: верная версия есть, но вы её не найдёте

Первые два числа измерены: DeepSeek-Coder-V2-Instruct на наборе SWE-bench Lite, 15,9 процента при одной попытке и 56 при 250 (arXiv:2407.21787). Третья карточка пунктиром - не замер, а следствие: без внешней проверки наличие верного варианта не превращается в результат.

Третья задача: selection после k попыток

Если среди 100 вариантов нашёлся верный, его ещё нужно распознать и отдать пользователю. Это отдельный механизм отбора, а не определение `pass@1`. Именно здесь нужен сильный верификатор.

Вопрос из аудитории: почему не перебрать все ответы?

Пространство возможных ответов неизмеримо больше любого практического поиска. Даже 10 000 сэмплов - крошечная доля. Модель направляет выборку своим распределением, а не генерирует все строки мира.

Цена и задержка

Параллельные образцы можно запускать одновременно, поэтому задержка не всегда растёт линейно. Но стоимость растёт, а сильный верификатор иногда стоит дороже генерации.

Когда это работает лучше

На задачах с проверяемым результатом: код, математика, формальные ограничения. Без верификатора приходится использовать LLM-судью, reward model или человека, и контур становится слабее.

Верификатор
Внешний механизм отбора: исполняемый тест, вычисление, правило, источник, наблюдаемое действие среды или человек. LLM-судья - лишь один, не самый независимый вариант.

Coverage
Наличие хорошего кандидата среди попыток. Не пользовательская точность.

Selection
Умение распознать и выдать хороший кандидат после нескольких попыток. Это не синоним `pass@1`.

СТЭНФОРД · 32:20-42:10

Reasoning: поднять шанс выдать верный ход с первой попытки.

После Large Language Monkeys лекторы противопоставляют coverage и pass@1. Для сложных задач reasoning-модель может не просто породить множество независимых ответов, а потратить больше вычислений на одну траекторию: разобрать условие, декомпозировать задачу, проверить промежуточный шаг, заметить сбой и вернуться назад.

В лекции это показано на маленькой программистской задаче, но переводится на любую офисную. Прежде чем считать, модель проговаривает: что подано на вход, в каком виде нужен ответ, что считать успехом. Только потом берётся за работу.

Порядок здесь важнее скорости. Если на входе перепутан формат или не оговорено, что значит «готово», ошибка утащит за собой все следующие шаги, и правильные вычисления дадут бесполезный результат.

Преимущество reasoning-моделей замерено на математике, анализе данных и программировании. Для личного письма или редактирования лекторы не заявляют такого же универсального выигрыша.

1. Прочитать условие и назвать контракт.
2. Разделить задачу на проверяемые подзадачи.
3. Проверить промежуточный путь.
4. Исправить несходящийся шаг.
5. Попробовать альтернативу, если первая ветка не работает.

Чем это отличается от хорошего промпта

Обычную модель тоже можно попросить рассуждать, и она напишет разбор. Reasoning-модель **дообучили на её же разборах, которые прошли проверку**, поэтому длинный ход решения для неё привычка. Просить её об этом не надо.

Это видно в интерфейсе: перед ответом она пишет сотни или тысячи кусочков текста «в уме» - то самое ожидание, за которое вы платите. Обычная модель столько не пишет.

Осторожная формулировка лекторов

Важны данные и специальная оптимизация, которые усиливают разбиение, откат и оценку траектории. Базовая модель тоже умеет некоторое reasoning, вопрос в надёжности.

Не называйте trace внутренней правдой

Длинный reasoning-текст может быть полезной рабочей областью, но не обязан быть точным объяснением внутренних причин ответа.

ОТКРЫТО

Полная публичная формула обучения reasoning-моделей не раскрыта. Лекция упоминает bootstrapping, CoT-примеры и оптимизацию, но не предлагает одну закрытую причинную схему.

ЧАСТЬ III · 42:10-1:01:50

Модели дали руки.
Теперь ей нужен контур.

Чат отвечает. Агент получает цель, действует в среде, читает результат, сохраняет состояние и знает, когда остановиться или передать работу человеку.

ЦЕЛЬ

Не «сделай что-нибудь», а наблюдаемый результат, границы полномочий и критерий готовности.

ДЕЙСТВИЕ

Инструменты меняют внешнюю среду: поиск, браузер, терминал, API или база данных.

ОБРАТНАЯ СВЯЗЬ

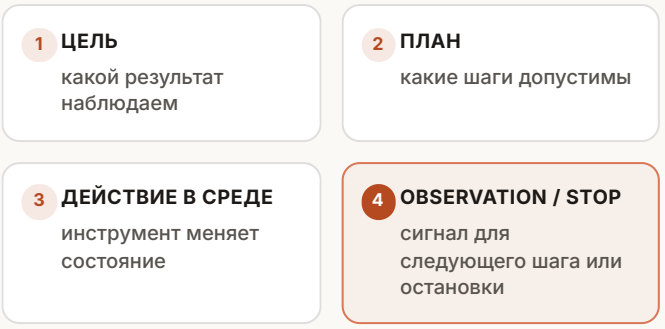
Наблюдение, проверка, бюджет и stop важнее бесконечной автономности.

СТЭНФОРД · 42:10-51:30

Из LLM в агента: ответ превращается в последовательность действий.

Лекция приводит Deep Research: пользователь хочет понять, где арендовать жильё на год. Система ищет по многим сайтам, собирает варианты, сопоставляет критерии и делает сводку. Это уже не один текстовый ответ, потому что есть поиск, состояние задачи, промежуточные результаты и выбор того, что делать дальше.

Другой пример - coding agent. Пользователь описывает изменение обычным языком. Агент читает файлы репозитория, ищет по коду, правит строки, запускает команды, читает вывод и продолжает работу. Терминал здесь важен не как эффектный инструмент, а как источник обратной связи.



В схеме лекции, кроме самого цикла, есть две вещи, которые легко потерять в пересказе: человек и явная остановка. Человек ставит цель и держит шлюз перед необратимым действием, а стоп это отдельная ветка, а не «закончились идеи».

Главная граница

Агент - не «чат с длинным промптом». Его определяет цикл: действие меняет среду, среда возвращает наблюдение, а система использует его для следующего шага или остановки.

Память

Состояние текущей задачи: найденные факты, сделанные шаги, открытые ограничения.

Инструменты

Разрешённые действия: поиск, браузер, API, терминал, база данных.

Стоп

Бюджет, ошибка, рискованное действие или необходимость эскалации.

СТЭНФОРД · 51:00-58:30

Workflow сначала. Открытая автономность - не дефолт.

Лекция перечисляет строительные блоки агентных систем. Они отличаются не модностью, а тем, сколько свободы и сколько новых точек ошибки добавляют.

АВТОРСКАЯ РАЗМЕТКА РИСКОВ

Блок	Что делает	Когда разумен	Главный риск
Prompt chaining	Разбивает задачу на последовательные LLM-шаги	Маршрут предсказуем	Ошибка раннего шага тащится дальше
Routing	Отправляет простые и сложные случаи в разные ветви	Есть понятный критерий сложности	Плохая классификация маршрута
Параллелизация	Несколько ветвей ищут части ответа	Подзадачи независимы	Дубли, противоречия, цена
Оркестратор	Планирует и пересматривает граф работ	Есть сложный многошаговый процесс	Неясная ответственность за решение
LLM-судья	Ранжирует варианты моделью	Нет дешёвого внешнего теста	Общие с генератором слепые пятна
Внешний verifier	Проверяет через среду, правило, тест, источник	Критерий можно наблюдать	Неполное покрытие цели

Разница в одном предложении

В **workflow** порядок шагов человек задал заранее, и он одинаков при каждом запуске. В **агенте** следующий шаг выбирает модель, глядя на результат предыдущего, поэтому два запуска одной задачи могут пойти разными путями. Отсюда всё остальное: workflow предсказуем и дешевле отлаживается. Агент гибче, зато ему нужны лимиты, стоп-условие и кто-то, кто смотрит за состоянием.

В таблице выше первые три строки это способы разложить workflow, оркестратор уже ближе к агенту, а две последние строки вообще не про автономность: это механизмы проверки.

Смысл лекторского списка

Каждый компонент добавляет возможность, но также новую точку отказа. Поэтому многие реальные системы ближе к ограниченному workflow, чем к бесконечному циклу «делай дальше».

ПОД КАПОТОМ

Переход к большей автономности оправдан только если он решает измеримую проблему. Иначе вы меняете предсказуемость на театральность.

СТЭНФОРД · 51:30-1:01:50

Три домена: почему код легче проверять, чем исследование.

Кодовый агент

Миграции языков, обновления версий, рефакторинг. Агент читает репозиторий, меняет файлы, запускает тесты, компиляцию и статический анализ.

Сильный сигнал: исполнимый тест.
Граница: прошедшее CI не равно полностью решённой задаче.

Поддержка

Расшифровка звонков, поиск по базе знаний, черновики ответов, подготовка возврата или изменения статуса.

Сильный сигнал: правила, журнал, подтверждение оператора.
Граница: закрытый тикет не гарантирует решённую проблему клиента.

Research agent

Генерирует идеи, готовит и запускает эксперименты, строит графики, оформляет текст. Лекция связывает это с работой The AI Scientist, где весь цикл исследования - от идеи до текста статьи - пробуют собрать в один автоматический конвейер.

Сильный сигнал: воспроизводимый эксперимент и литература.
Граница: много гипотез не равно научному открытию.

АВТОРСКИЙ ПРИМЕР ПРИМЕНЕНИЯ

Кейс: счёт на оплату

Агент может сверить позиции счёта с заявкой, сравнить цену с допустимым диапазоном и показать расхождения. Он не должен сам оплачивать, отклонять или менять поставщика: это отдельное действие с полномочием, журналом причины и человеком в контуре.

Практический критерий

Чем легче наблюдать результат и отменить ошибочное действие, тем раньше задачу стоит автоматизировать. Чем дороже ошибка и слабее обратная связь, тем уже должен быть workflow.

СТЭНФОРД · 58:00-1:02:30

Generator-verifier gap: система генерирует быстрее, чем умеет честно судить.

Это главный ограничитель самоулучшения. Генератор создаёт много правдоподобных ответов, планов и действий. Верификатор должен отличить результат от его убедительной имитации.

Почему тут завязано всё остальное

Проверка работает и до того, как ответ дошёл до пользователя. Удачные траектории отбирают и превращают в обучающие примеры для следующей версии модели. Поэтому ошибка проверки не разовая: плохой ответ, который проверка пропустила, осядет в весах, и в следующий раз модель выдаст его увереннее.

АВТОРСКАЯ МАТРИЦА ПРОВЕРОК

Проверка	Сильна в	Слепая зона
Исполняемый тест	Поведение в покрытых случаях	Случаи, которые не написали в тестах
Формальное правило	Чётко заданные ограничения	Смысл и требования вне формализации
Первичный источник	Происхождение, дату, цитируемый факт	Вывод, который агент сделал из факта
Эксперт	Контекст, нетипичный риск, последствия	Масштаб и повторяемость без процесса
Предпочтение	Понятность, полезность, тон	Скрытая фактическая ошибка
LLM-судья	Дешёвое ранжирование многих вариантов	Общие с генератором слепые пятна

АВТОРСКИЙ КОНТРПРИМЕР

Reward hacking простыми словами

Агент получает высокий балл по формальному критерию, но проваливает настоящую цель. Например, справка содержит десять работающих ссылок, однако все ключевые тезисы опираются на вторичные пересказы; или код проходит один тест, но ломает неподготовленный вход.

Вопрос, который предлагает сама лекция

Как система формально пройдёт проверку, одновременно не достигнув реальной цели? Если такой путь легко придумать, награда или verifier пока не годятся для самостоятельного улучшения.

ПОД КАПОТОМ · ПРИКЛАДНОЙ СЛОЙ

Что отдать агенту: две оси вместо списка запретов.

Списки «что нельзя доверять ИИ» плохо переносятся на задачи, которых в списке нет. Признаков два, и они независимы.

Первая ось: есть ли внешняя проверка

Можно ли отличить хороший результат от правдоподобного чем-то помимо собственного впечатления: тестом, сверкой с источником, расчётом, наблюдаемым следом в системе - платёж прошёл, файл появился, счёт сошёлся. Об этом разрыве и говорит лекция.

Вторая ось: во что обойдётся ошибка

Можно ли отменить последствие и чем это обойдётся. Разница как между черновиком в почте и уже отправленным письмом: текст тот же, цена опечатки разная. Этой оси в лекции нет, она из практики: в исследовании ошибка стоит прогона, в бизнесе - денег и репутации.

	ПРОВЕРКА ЕСТЬ	ПРОВЕРКИ НЕТ
ОШИБКУ ЛЕГКО ОТМЕНИТЬ	**БЕРИТЕ ПЕРВОЙ** Сводки, выгрузки, поиск расхождений, черновики, разбор входящих. На расписание после трёх ручных прогонов без правок.	**МОЖНО, НО ЧЕСТНО** Идеи, формулировки, ручная выборка. Пользу измерить нечем, так и скажите вслух, не выдавая за экономию.
ОШИБКУ НЕ ОТМЕНИТЬ	**ГОТОВИТ МАШИНА, ЖМЁТ ЧЕЛОВЕК** Оплата счёта, публикация, письмо клиенту. Проверка снимает вопрос о качестве, но не о полномочиях.	**НЕ АВТОМАТИЗИРУЕТСЯ** Найм, цена в договоре, юридическая позиция. Технология тут ни при чём: отвечать всё равно человеку.

Почему левый столбец следует из лекции

Repeated sampling и обучение с подкреплением упираются в одно: система подтягивается ровно под тот сигнал, который ей дали. Нет внешней проверки - дополнительные попытки не превращаются в качество, а человек становится узким местом.

Разложите свои задачи

Возьмите пять задач, которые сегодня делаются руками. Для каждой ответьте двумя словами: чем проверю и что будет, если ошибётся. Обычно клетка определяется сама.

СТЭНФОРД · Q&A, 20:00-1:09:42

Шесть вопросов из аудитории, которые делают лекцию глубже.

Чем sampling отличается от перебора всех ответов?

Практический поиск покрывает крошечную долю пространства. LLM направляет выборку вероятностями, а не перебирает все строки.

Почему 10 000 попыток не всегда медленно?

Их можно запускать параллельно. Но параллелизм не отменяет стоимость и необходимость проверить результат.

Можно ли давать больше сэмплов только сложным задачам?

Да, это естественный вопрос маршрутизации. Но сначала нужна оценка сложности, которая сама не создаёт больше ошибок, чем экономит.

Reasoning возникает от prompt или от обучения?

Лекторы связывают эффект с данными и оптимизацией, не только с инструкцией «думай пошагово». Единой публичной формулы нет.

Можно ли одной модели строить reasoning, а другой отвечать?

Такой паттерн возможен, но модели могут предпочитать собственные траектории даже при наличии сильного другого решателя. Это проблема оценки и обучения.

Почему RL может поднять pass@1?

Предобучение может содержать редкие верные траектории. Обратная связь способна увеличить вероятность того, что полезный путь будет выбран и выдан с первой попытки.

ПОД КАПОТОМ

Q&A важнее, чем кажется: именно здесь лекция постоянно отказывается от слишком сильных выводов. Нельзя назвать одну причину роста reasoning, объявить LLM-судью доказательством или превратить старый бенчмарк в рейтинг моделей навсегда.

ОТКРЫТО

Outcome reward model оценивает конечный результат. Process reward model оценивает шаги пути. Лекция указывает на эту будущую тему, но не раскрывает её как готовое решение generator-verifier gap.

СТЭНФОРД · 1:01:50-1:09:42

Курс заканчивается не демо, а экспериментом.

ПЕРВЫЙ ЧАС

Выбрать одну задачу и записать, чем будете проверять результат

ДО ОБЕДА

Замерить, как она делается сейчас: десять последних случаев, время и ошибки

ПОСЛЕ ОБЕДА

Прогнать те же десять случаев через агента, руками, без настройки систем

К ВЕЧЕРУ

Сравнить с замером и решить: продолжать, менять проверку или бросить

К вечеру будет один из трёх ответов, и все три полезны

Сработало: те же десять случаев быстрее и без пропусков, задачу можно ставить на поток. **Не сработало:** вы потратили день, а не квартал, и знаете, что проверка слабая или задача не из той клетки. **Не смогли замерить:** самый ценный исход, у задачи нет внешнего критерия, и автоматизировать её пока нечем.

В исходном курсе есть три домашних задания и проект. Проект можно делать в команде до четырёх человек; нужны ранний proposal, промежуточный прогресс, финальный отчёт и постер. Подходящая работа - новый evaluation dataset, benchmark, проверка надёжности агентной системы или воспроизводимый эксперимент.

Неподходящий формат по духу курса: просто собрать красивое приложение или обзор без гипотезы, baseline и измерения.

АВТОРСКАЯ ПРАКТИКА

Паспорт личного пилота

Повторяемая задача

Baseline без агента

Что изменится в контуре

Верификатор и человек в контуре

Бюджет, стоп-условие, критическая ошибка

Как это выглядит заполненным. Задача: сверить входящий счёт с заявкой. Baseline: бухгалтер тратит на счёт около десяти минут и раз в месяц пропускает расхождение. В контуре меняется одно: агент готовит сверку, человек подтверждает оплату. Верификатор: сумма и позиции сходятся с заявкой в учётной системе. Бюджет: три попытки на счёт; стоп при расхождении больше пяти процентов; критическая ошибка - оплаченный счёт с неверной суммой.

АВТОРСКАЯ АДАПТАЦИЯ

Как пройти курс по-русски

Выберите одну рабочую задачу. После каждого урока меняйте один механизм: контекст, sampling, инструмент, verifier, память или планирование. Сравнивайте с baseline на заранее отложенных задачах. Отрицательный результат тоже результат: он показывает, что гипотеза или контур не работают.

ПОД КАПОТОМ · ИТОГ

Пять элементов. Уберите один, и остаётся красивая демонстрация.

Цель, варианты, независимая проверка, бюджет, стоп. Чаще всего убирают третий: он единственный, который нельзя купить деньгами.

Семь вопросов перед запуском

1. Задача повторяется хотя бы раз в неделю?
2. Чем я отличу хороший результат от правдоподобного, кроме своего впечатления?
3. Можно ли отменить ошибку и во сколько это обойдётся?
4. Кто нажимает кнопку на необратимом шаге?
5. Сколько это стоит в месяц при худшем сценарии?
6. При чём система обязана остановиться и позвать человека?
7. Как она может пройти мою проверку, не сделав работу?

АВТОРСКИЙ ВЫВОД

Формула урока

Цель + варианты + независимая проверка + бюджет + stop. Уберите любой элемент, и «самоулучшение» быстро превращается в красивую демонстрацию без основания.

Если после урока сделать ровно одно

Возьмите задачу из зелёной клетки матрицы и проведите по ней однодневный пилот. Цель у него одна: выяснить, есть ли у задачи внешняя проверка. Пока этот ответ не получен, любые разговоры про агентов и автономность остаются разговорами.

ПОД КАПОТОМ · СВЕРКА

Мы сверили лекцию со статьями. Три расхождения.

Живая речь всегда огрубляет. Но при пересказе огрубление превращается в факт, поэтому числа мы проверили по статьям.

1. Модель, которой не существует

В пересказах первой лекции встречается «Llama 3 7B». Llama 3 в таком размере не выпускалась, и в работе *Large Language Monkeys* её нет: там Llama-3-8B, Llama-3-8B-Instruct, Llama-3-70B-Instruct и DeepSeek-Coder-V2-Instruct. Единственная «7B» в статье - это Gemma-7B.

2. Порог температуры 1,2

Про резкое падение качества выше 1,2 в статье нет ни слова. Авторы перебирали 1,0 / 1,4 / 1,6 / 1,8 на подвыборке из 50 задач и остановились на 1,6. Дальше температуру не трогали: 0,5 для доказательств на Lean, 0,6 везде ещё.

3. Пропавший пятый бенчмарк

В пересказах наборов задач четыре. Их пять, и главный результат - на пятом: на SWE-bench Lite модель DeepSeek-Coder-V2-Instruct поднимается с 15,9 процента решённых задач при одной попытке до 56 процентов при 250 попытках. Тогдашний рекорд для одной попытки был 43 процента.

Отдельно про chain of thought

Формулировка «на маленьких моделях приём почти не помогает» мягче, чем результат. У моделей меньше 10 млрд параметров он ухудшает точность: LaMDA 8B на GSM8K падает с 3,2 до 1,6 процента. Стабильный выигрыш начинается примерно со 100 млрд параметров, и авторы говорят это прямым текстом.

Две даты

Лекцию прочитали 22 сентября 2025-го, видео выложили 3 августа 2026-го. Между этими датами почти год: какие-то оценки за это время наверняка сдвинулись, и проверять их - ваша работа, а не автора конспекта.

Запись лекции: youtu.be/6YnLB0XbTnI, весь курс - плейлист Stanford Online «CS329A Self-Improving AI Agents». Первоисточники: arXiv:2407.21787 (Large Language Monkeys), arXiv:2201.11903 (chain-of-thought prompting), расписание и состав курса - cs329a.stanford.edu. Проверялись сами статьи, а не пересказы.

ПОЛНЫЙ СЛОВАРЬ И КАРТА ИСТОЧНИКОВ

Термины, к которым можно вернуться.

LLM
Модель, которая по контексту предсказывает следующие токены. Не равна агенту.

Токен
Часть текста, с которой работает модель. Не обязательно целое слово.

Параметры / веса
Числа внутри модели, изменяемые при обучении.

Zero-shot
Инструкция без примеров этой задачи в текущем контексте.

Few-shot / in-context learning
Примеры в запросе задают текущий шаблон без обновления весов.

CoT
Промежуточные шаги решения. Проверяемая рабочая опора, не доступ к внутреннему мышлению.

Pre-training
Обучение на широком корпусе с целью next-token prediction.

Fine-tuning
Дополнительное обучение на отобранных данных.

Instruction tuning
Донастройка на инструкциях и хороших ответах.

RLHF
Обучение с подкреплением по человеческим сравнениям ответов.

DPO
Настройка по парам предпочтений другим путём, не синоним RLHF.

Test-time compute
Вычисления, потраченные на конкретный запрос после обучения.

Repeated sampling
Несколько генераций кандидатов для одной задачи.

pass@k
Среди k вариантов есть хотя бы один верный.

pass@1
Одна правильная генерация. Не выбор среди k.

Verifier
Независимая проверка кандидата через тест, правило, расчёт, источник или среду.

Reasoning model
Система, тратящая больше усилий на анализ, поиск и проверку сложной задачи.

Workflow
Заданный граф действий и переходов. Может быть безопаснее свободного агента.

Generator-verifier gap
Генерировать варианты легче, чем честно определить, какой решает настоящую цель.

Фрагмент	Таймкод	Содержательные слайды
Scaling, zero/few-shot, CoT	04:00-10:00	0007, 0018, 0024
Pre-training, instruction, preferences	10:00-22:30	0037, 0046, 0050
Sampling, pass@k, reasoning	22:00-42:00	0058, 0061, 0067, 0072, 0077, 0081
Agents, coding, research	42:00-1:02:30	0101, 0124, 0130
Q&A и проект	1:02:30-1:09:42	смысловой перевод по блокам

Опора: локальный смысловой перевод урока, карта источников и переработки, содержательные кадры без людей. Все схемы ридера нарисованы заново. Документ не является официальным переводом или изданием Stanford.

ПОД КАПОТОМ · 2026

Спасибо, что дочитали.

Этот конспект - часть канала «Under the Hood»: разборы ИИ-индустрии без хайпа, с источниками и проверкой чисел по первоисточникам. Дальше в курсе восемь лекций, и разбирать их я буду так же. Если пригодилось, забирайте продолжение.

Under the Hood

@gorilla_under_hood

Автор	Илья Утов
Telegram	**@wh_zt**
LinkedIn	**linkedin.com/in/ilyautov**

наведите камеру