# LLM: MoE, LRM, agents

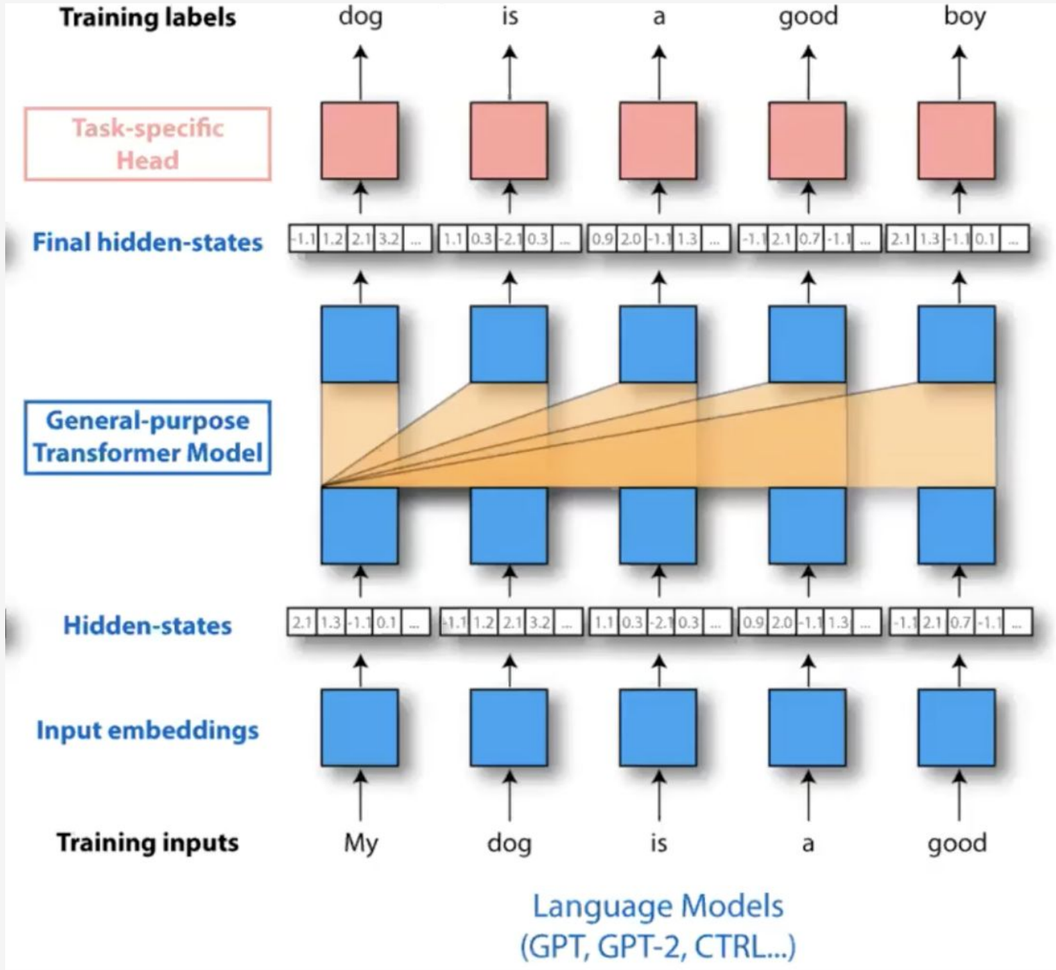
Natural Language Processing

Ника Зыкова, 2025/11/24

# Куда дальше: проблемы

У стандартного GPT-like Трансформера, используемого в text-to-text подходе, есть три проблемы:

- Ограниченная масштабируемость: даже сейчас 300B - это очень много;
- Склонность галлюцинировать и низкая способность отвечать на комплексные вопросы;
- Ограниченность в действиях и источниках информации: только текстовый ввод и его содержание.

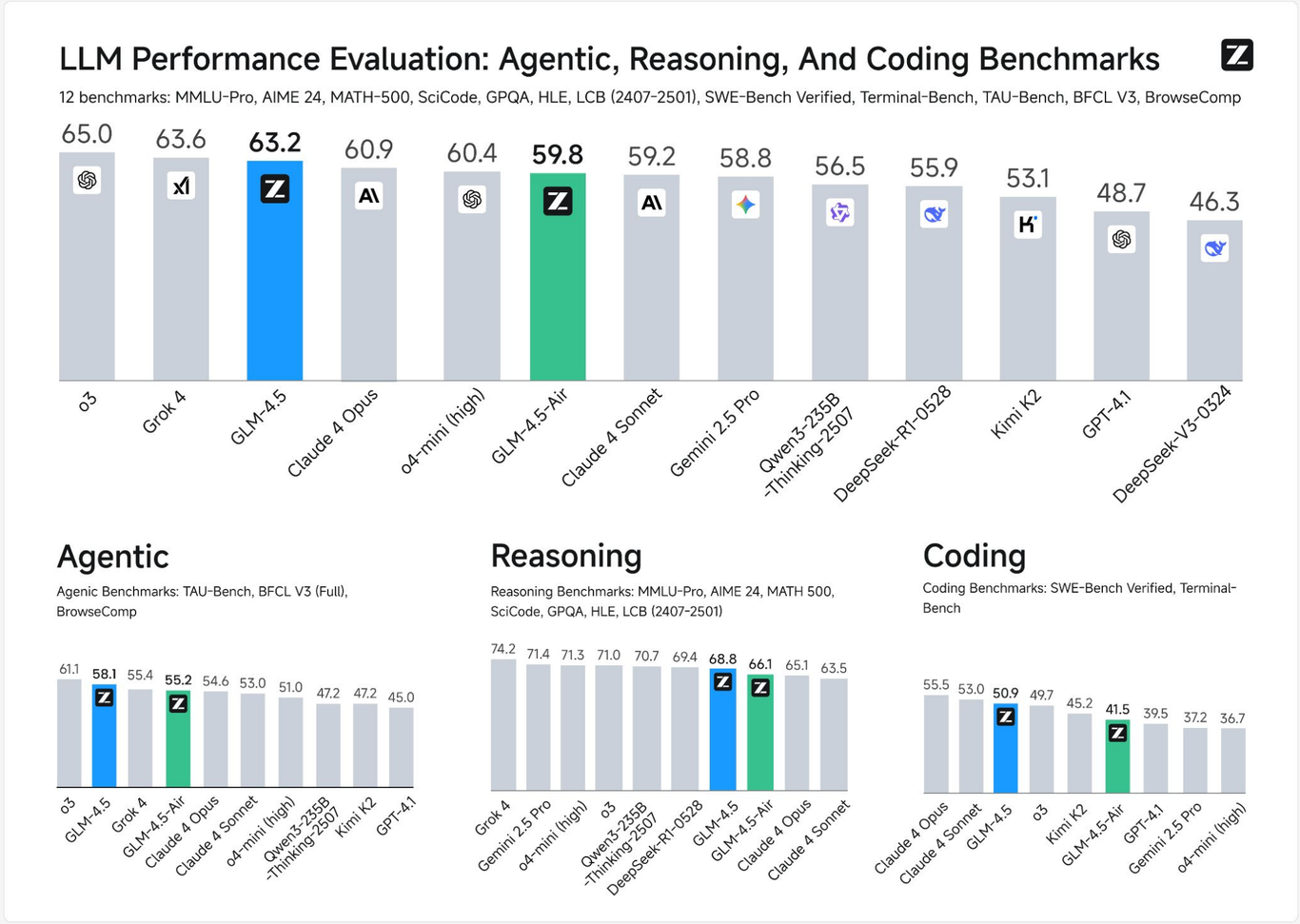



https://www.youtube.com/watch?v=t86G11tfVNw

# Куда дальше: решения

Чтобы решить каждую из описанных проблем были предложены отдельные решения:

- Mixture of Experts: архитектурное решение, позволяет увеличить объем модели, сохранив FLOPs (Floating Point Operations);
- Reasoning: модель получает навык рассуждать;
- Agentic skills: модель получает навык использования внешних инструментов.

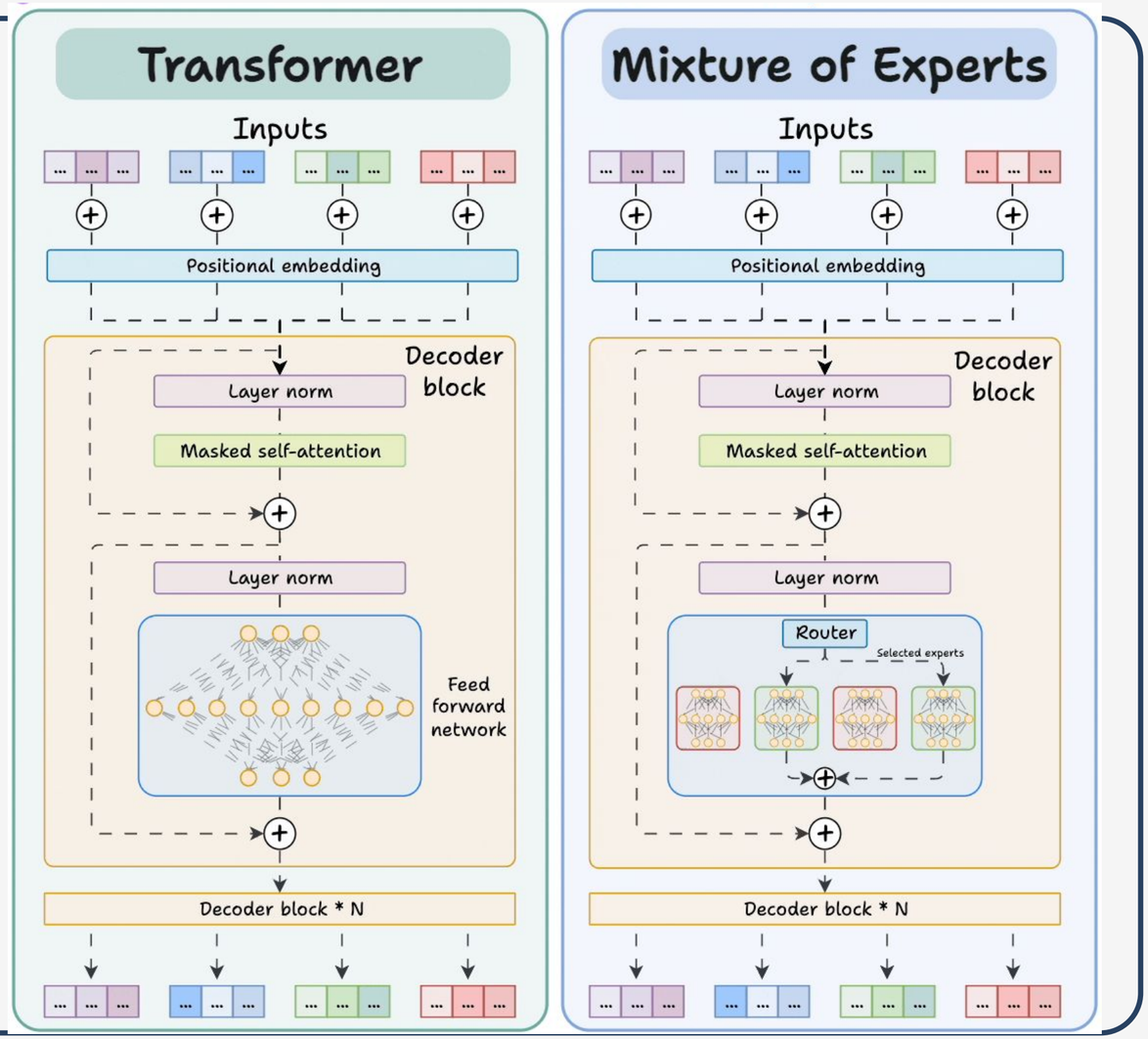



https://z.ai/blog/glm-4.5

# Mixture of Experts

Общая идея: вместо Feed-Forward сети после внимания будем использовать комбинацию из роутера и экспертов.
Что это дает:

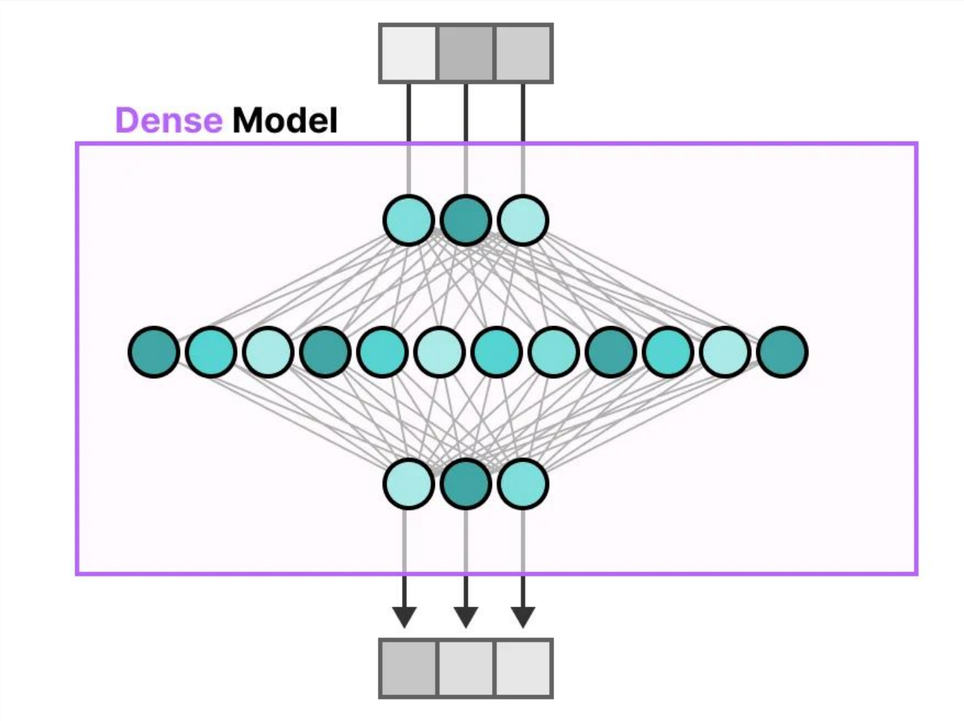
- Для каждого токена активируется только определенный набор весов;
- Роутер позволяет выбрать наиболее подходящего для данного токена эксперта;
- Ограничив кол-во экспертов на токен, мы можем уменьшить FLOPs.

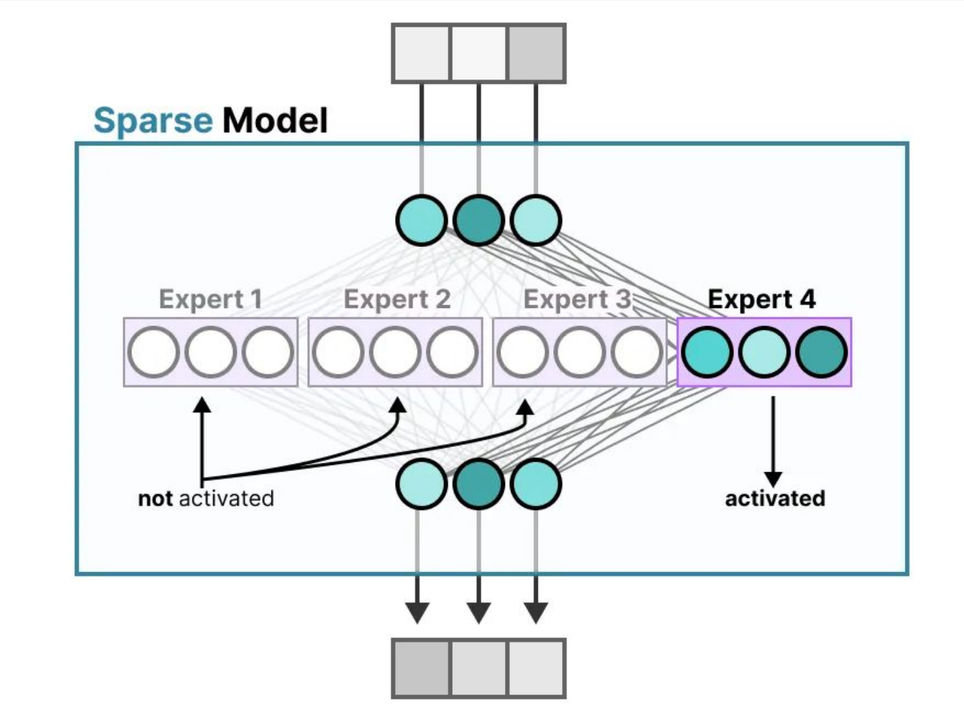



https://ch.pinterest.com/developerggof/algorithms-and-data-structures/

# Mixture of Experts

Стандартную архитектуру принято называть плотной (dense), а MoE - разреженной (sparse), аналогично графам вычислений, которые в итоге получаются.



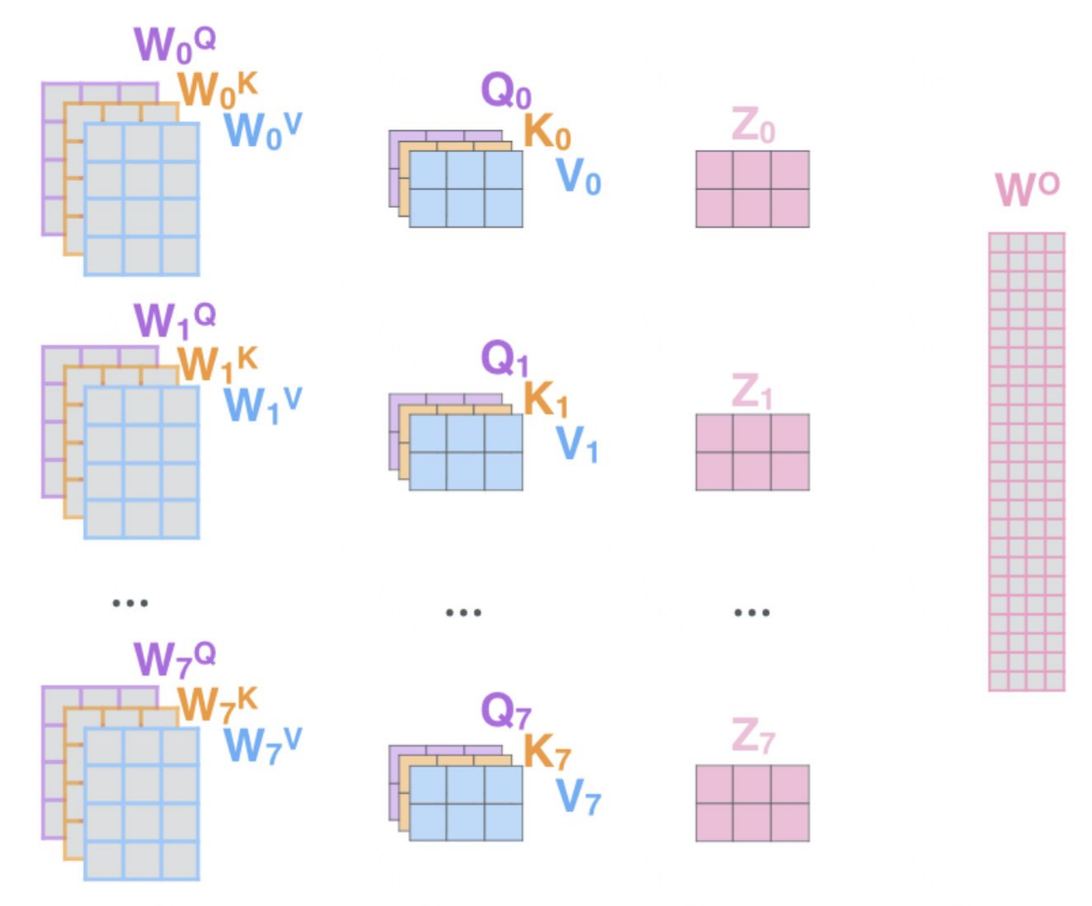

https://habr.com/ru/articles/882948/

# MoE: почему FF-слои?

Веса в модели распределены неравномерно по ее архитектурным частям. В итоге получается, что большая модель состоит из:

- 70–85% весов из MLP/FFN слоев;
- 10–20% весов из Self-Attention;
- 5–10% весов из эмбеддингов;
- <1% сохраненных значений из LayerNorm и прочего.

Поэтому логичнее всего сокращать кол-во вычислений именно в FFN части. Полносвязные слои почти в любой сети самые дорогие.



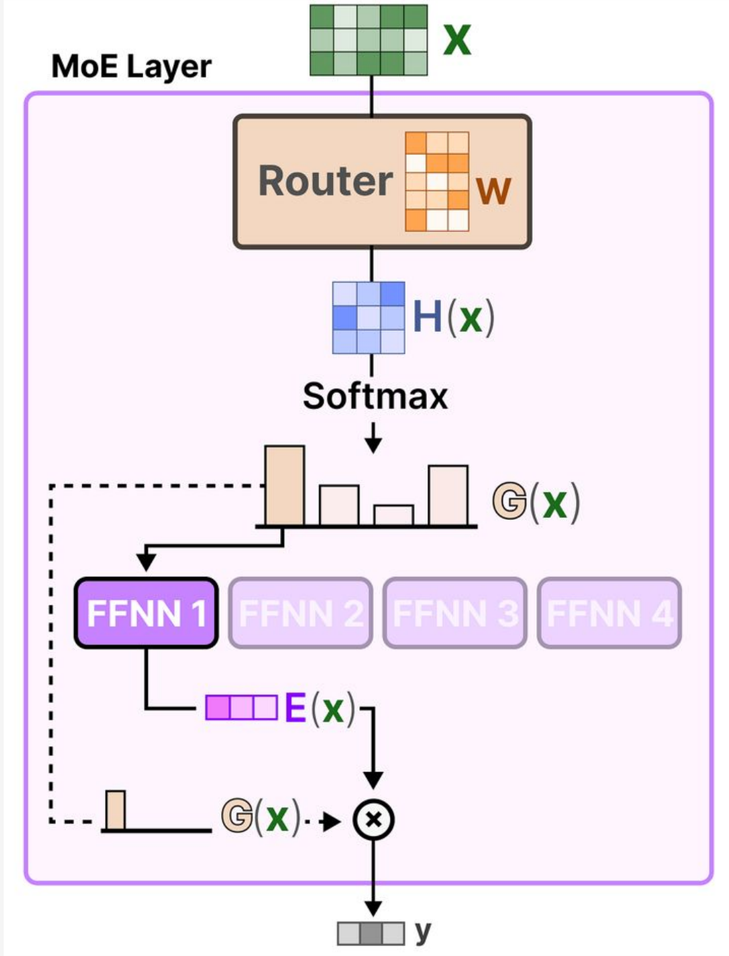

https://jalammar.github.io/illustrated-transformer/

# MoE: подробнее

Разреженная сеть состоит из двух компонент:

- Роутера: классификатора экспертов (линейный слой плюс софтмакс, на выходе столько значений, сколько у нас экспертов);
- Экспертов: полносвязных FF сетей одинакового размера.

Для выбранных экспертов (обычно топ-1/2 по мнению роутера) считается средневзвешенная сумма выходов (веса - вероятности от роутера).




https://llmstudio.ru/blog/mixture-of-experts-moe

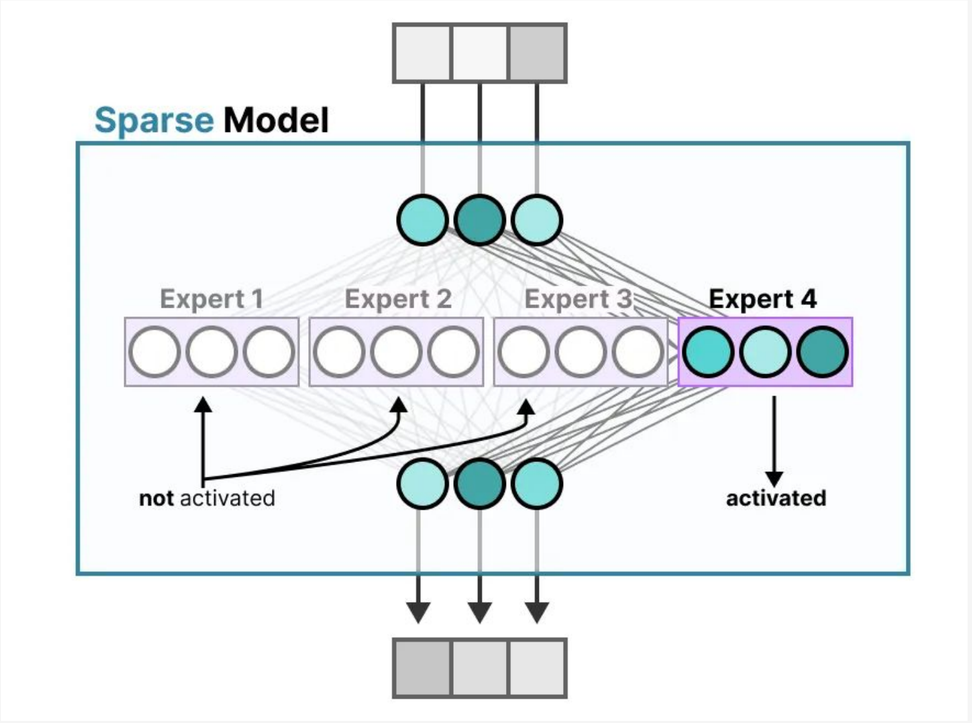
# MoE: где выгода

Пусть у нас есть модель с плотным FF-слоем:
d = 4096, h = 4d = 16384
Всего весов в таком слое:
2 * 4096 * 16384 = 134 217 728 (134M)

И MoE модель с 16 экспертами, где каждый эксперт размером с FF-слой плотной модели.

Тогда в MoE:

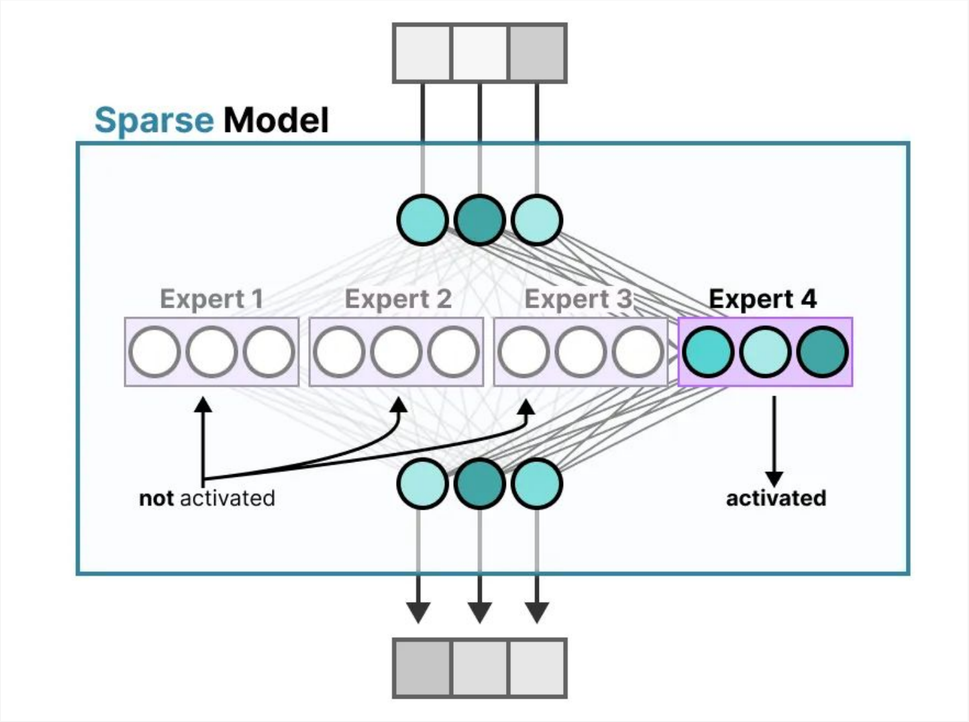
- 134M * 16 = **2 147 483 648 (2.1B) весов всего**;
- **134M весов активируется при топ-1 стратегии**.




https://habr.com/ru/articles/882948/

# MoE: преимущества

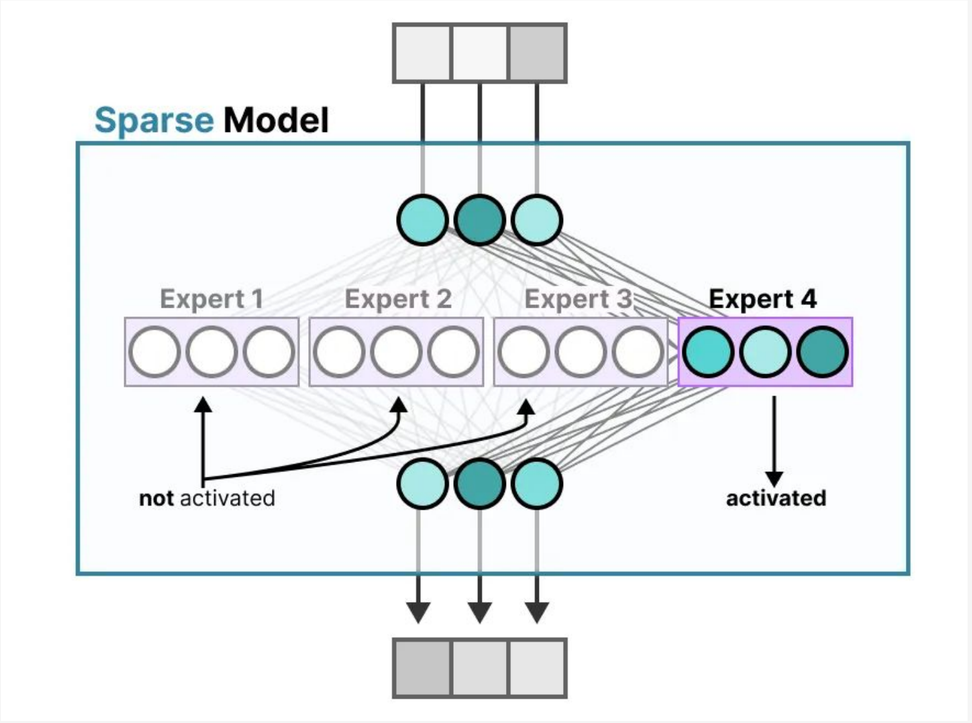
- Масштабируемость: увеличиваем общее кол-во параметров, но жестко ограничиваем кол-во активированных;
- Специальизация: модель использует разных экспертов под разные типы задач (код, перевод, математика и т.д);
- Мультиязычность и домены: комбинации экспертов позволяют модели более гибко подстраиваться под разные языки и домены без потери в качестве.




https://habr.com/ru/articles/882948/

# MoE: проблемы

- Load balancing: роутер может предпочитать небольшой набор экспертов, из-за чего работа модели становится нестабильной;
- Collapse: последствие load balancing, когда несколько экспертов становятся универсальными, из-за чего остальные не обучаются и не работают, но “занимают место”;
- Routing instability: роутер резко меняет выбранных экспертов в разных батчах, но в схожих контекстах;
- Сложный inference.

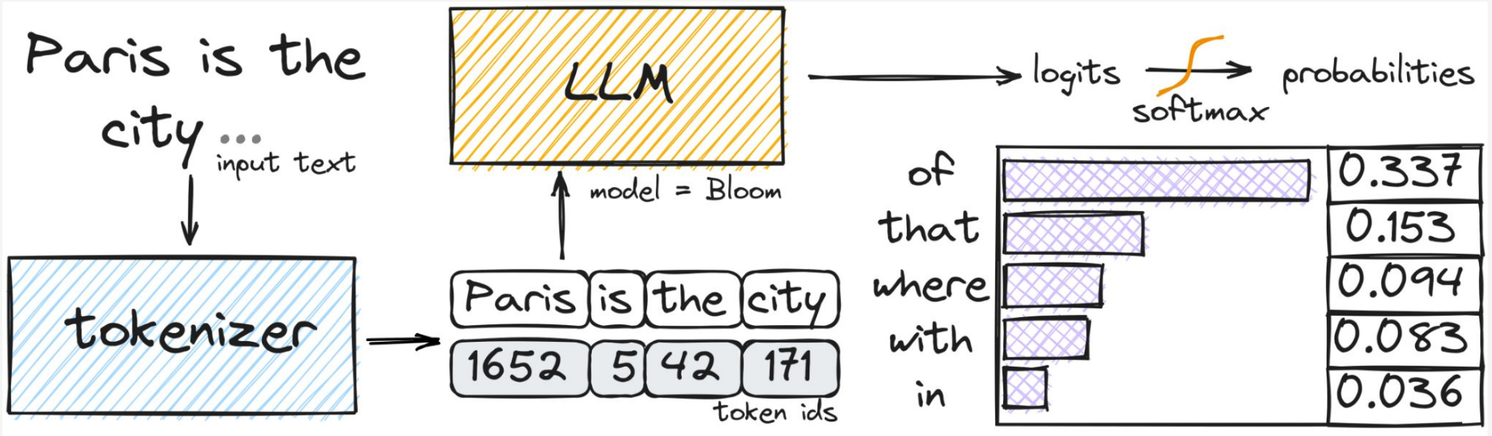



https://habr.com/ru/articles/882948/

# Reasoning

Большие языковые модели **не обладают встроенным мышлением**: они умеют воспроизводить наиболее вероятные цепочки токенов, что позволяет повторять форму ответов, а не структуру рассуждений. Для рассуждений нужно:

- построение промежуточных шагов
- планирование
- проверка гипотез
- обратная связь (self-reflection)




https://cloud.tencent.com/developer/article/2350191?areaId=106001

# Reasoning: пример

**Вопрос:**

У Анны было 12 конфет. Она дала 3 подруге, а затем купила ещё 7. Сколько конфет у нее сейчас?

**Ответ:**

<cot>

Анна начинает с 12 конфет.

Она отдаёт 3 конфеты, значит остаётся: 12 – 3 = 9.

Затем она покупает ещё 7 конфет: 9 + 7 = 16.

</cot>

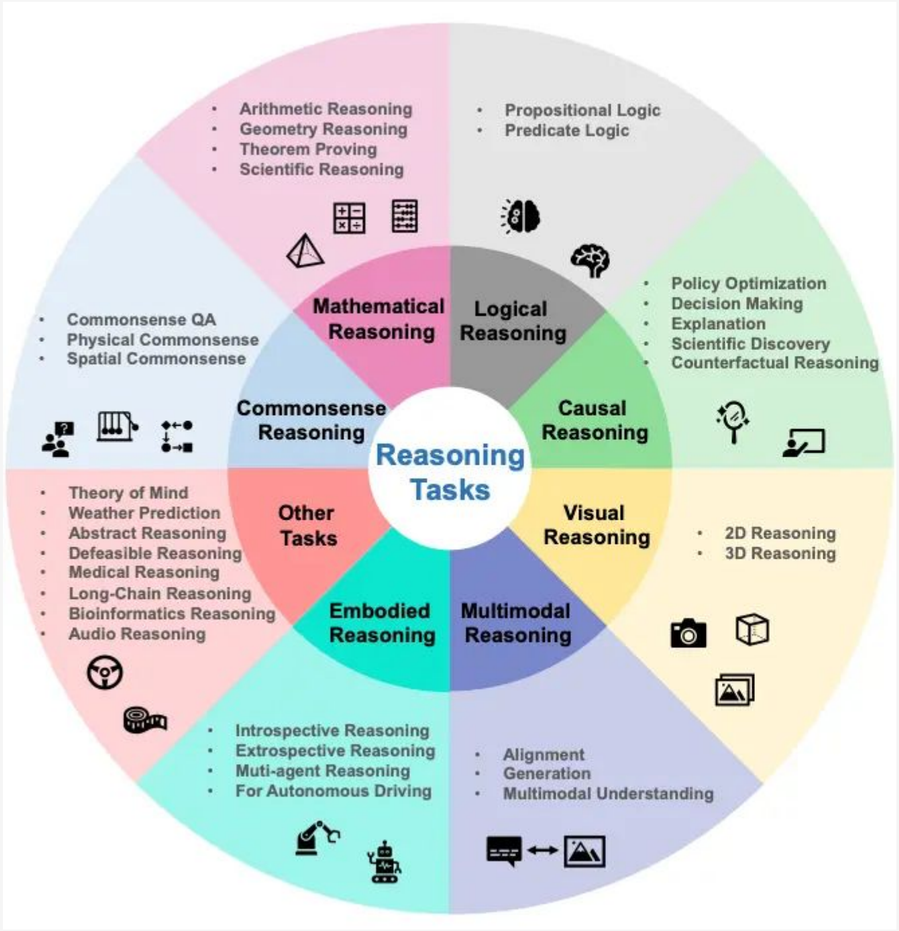
16

# Reasoning: зачем

Большая часть комплексных задач требует навыка рассуждений:

- Deep Research;
- Формирование и выполнение сложных инструкций (agentic skills);
- Написание кода;
- Решение задач (математика, химия, физика и т.д.).

Также рассуждения позволяют модели более качественно работать с большим контекстом.



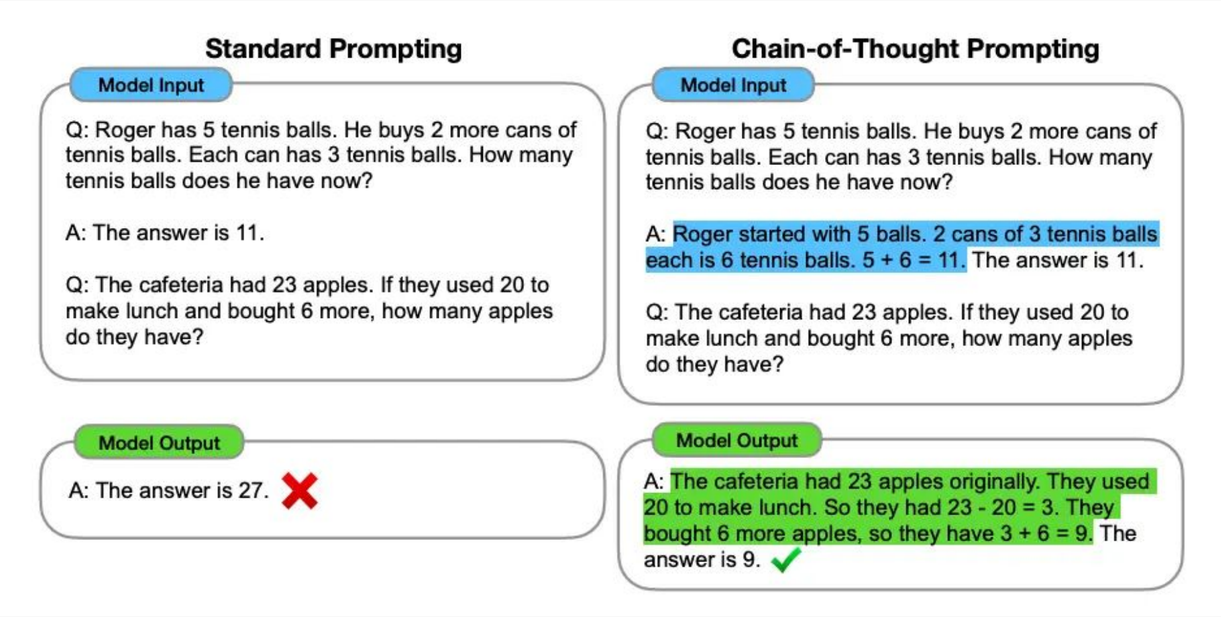

https://www.promptingguide.ai/research/llm-reasoning

# Reasoning: методы

Большую часть форматов рассуждений сначала попробовали в виде few-shot подхода в промптинге:

- Chain-of-Thought (CoT)
- Self-Consistency: генерируем несколько рассуждений, смотрим согласованность;
- Tree-of-Thought (ToT)
- Reflexion / Self-Refine: модель анализирует и исправляет ошибки после вывода;

Но в итоге оказалось, что для сложных задач этому надо учить.




https://www.promptingguide.ai/ru/techniques/cot

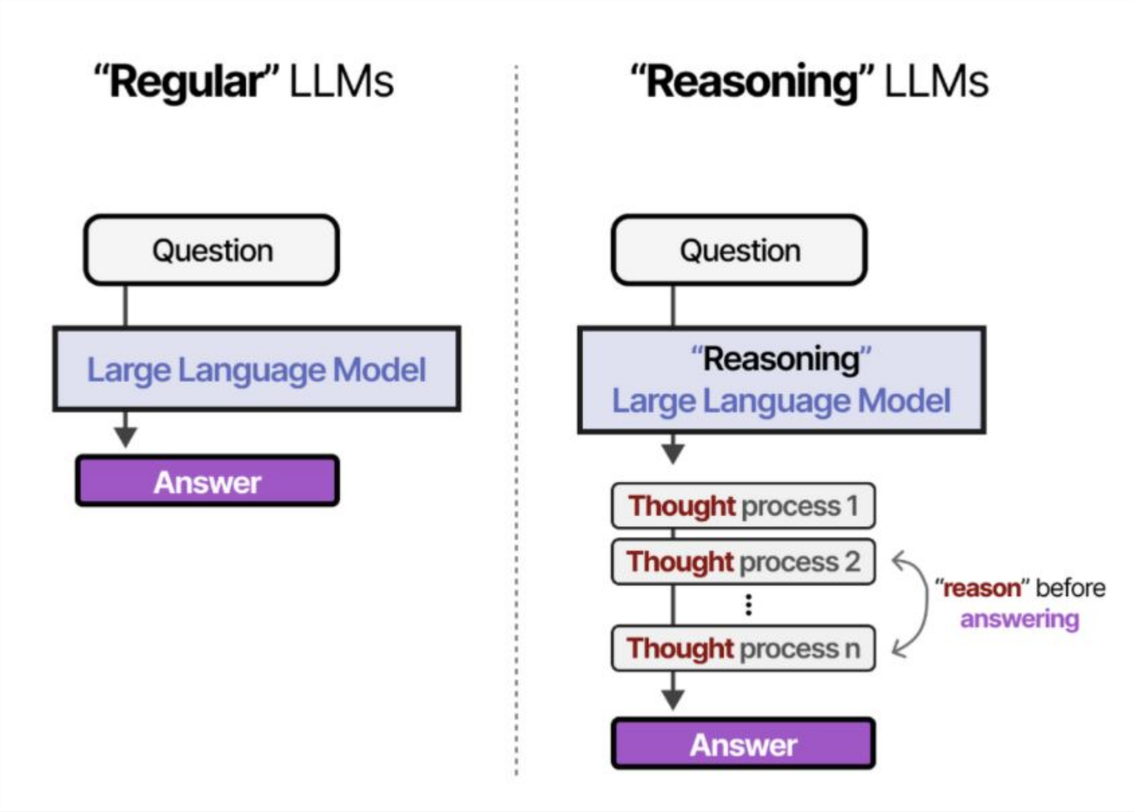
# Reasoning: обучение

Адаптация модели к рассуждениям может быть с двух точек зрений:

- Архитектурной: улучшение механизмов памяти и глубокого контекста;
- Обучающей: либо данные, либо подход к обучению показывает модели, что теперь она умеет рассуждать о задаче.

Самые частые подходы к дообучению:

- Учимся на примерах с рассуждениями от людей (учебники, разметка);
- Учимся на синтетике от более умной модели;
- Reinforcement Learning



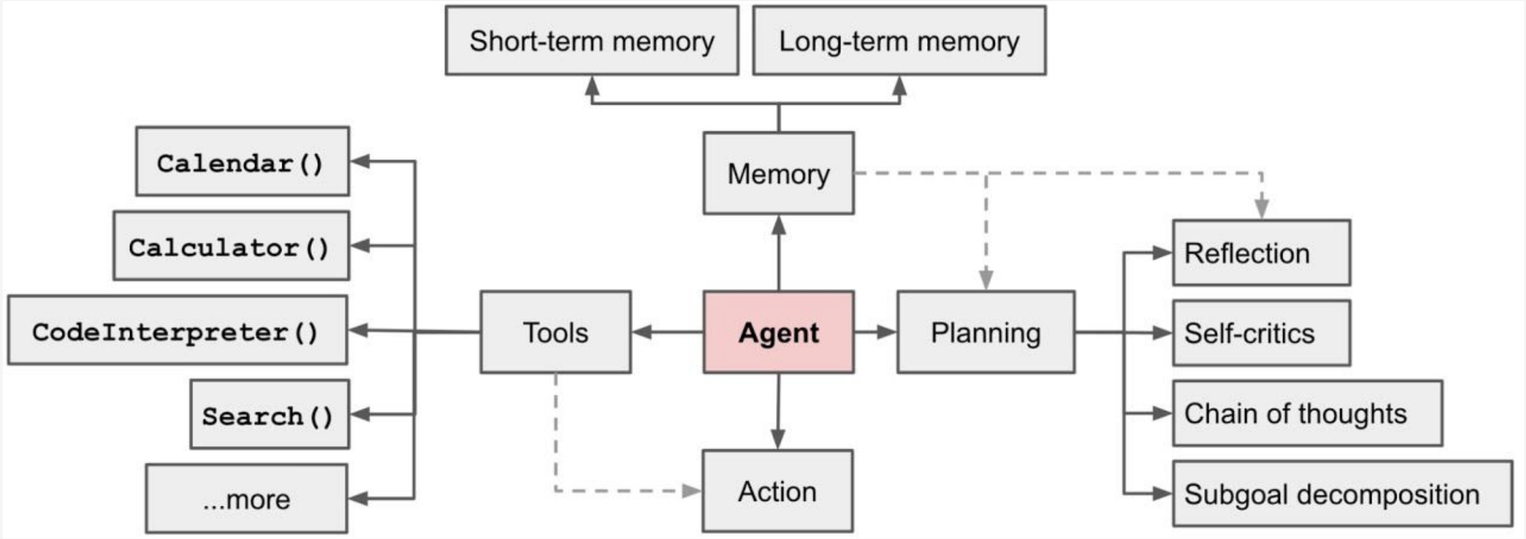



https://community.cloudera.com/t5/Community-Articles/Understanding-Reasoning-Models-with-GRPO-A-Conceptual/ta-p/411569

# Agents

Агент - это модель, способная использовать внешние инструменты для выполнения запроса пользователя. Чаще всего этот процесс состоит из трех зацикленных шагов:

- Рассцждение (reasoning): модель формирует последовательность необходимых вызовов функций;
- Действие (action): модель вызывает функции (по факту, просто пишет об этом);
- Наблюдение (observation): модель обрабатывает результат.




https://community.cloudera.com/t5/Community-Articles/Understanding-Reasoning-Models-with-GRPO-A-Conceptual/ta-p/411569

# Agents: пример

**Вопрос:**

Сколько рублей будет стоить купить 100 евро?

Не видно пользователю в чате

**Ответ:**

<cot>Нужно получить актуальный курс</cot>
<tool_call>call_api("currency", "euro")</tool_call>
<tool_answer>92.3<tool_answer>
<cot>Теперь могу посчитать итог, для этого необходимо умножить 100 на 92.3</cot>
<tool_call>call_api("calculator", "92.3 * 100")</tool_call>
<tool_answer>9230<tool_answer>
<cot>Получил ответ 9230, спрашивали про рубли, значит</cot>

100 евро будут стоить 9230 рублей.



https://community.cloudera.com/t5/Community-Articles/Understanding-Reasoning-Models-with-GRPO-A-Conceptual/ta-p/411569

# Agents: зачем

Необходимость прививать модели  агентские навыки возникла по нескольким причинам:

- Модели части ошибаются в вещах, которые легко делаются через код (арифметика, рисование графиков, операции со строками), при этом хорошо пишут этот код;
- Невозможно написать руками процедуры на все случаи жизни, но можно научить модель писать их самостоятельно (включить мультик, который смотрели вчера; открыть вклад и перевести на него N рублей со счета; посмотреть погоду на день, на который запланирована поездка);
- Модель превращается в самостоятельного участника процесса, который умеет планировать и принимать решения.



https://community.cloudera.com/t5/Community-Articles/Understanding-Reasoning-Models-with-GRPO-A-Conceptual/ta-p/411569

# Agents: подробнее

Создание агента возможно даже через промптинг, однако чаще всего все-таки делается через дообучение и дополнительные архитектурные решения:

**Добавление памяти**

Модель получает некое хранилище (чаще всего векторная база данных) и функции записи и чтения для него. Это позволяет не хранить важную информацию в ограниченном контексте.

**Информация о тулах**

Это любые функции, реализованные в формате API. Модель должна получить информацию о их сути и формате вызова и входа/выхода.

**Бизнес-ограничения**

Любая модель работает в рамках какие-то ограничений: не всем можно выдавать кредит, нельзя хамить клиенту, есть чувствительная информация, которую нельзя разглашать, но можно использовать.



https://community.cloudera.com/t5/Community-Articles/Understanding-Reasoning-Models-with-GRPO-A-Conceptual/ta-p/411569

# Agents: обучение

В контексте обучения агентов у нас появляется понятие **траектории** - последовательности рассуждений и вызовов тулов. Обучаются агенты именно на траекториях, длина которых в шагах (turns) зачастую играет большую роль, чем кол-во токенов. Методы применяются разные:

- Дообучение на правильных траекториях (с оценкой всей работы или каждого шага);
- Reinforcement Learning: учимся в среде, оцениваем, пришли ли к правильному ответу;
- Self-play / Multi-agent training: в среде несколько агентов и они конкурируют/кооперируются в рамках решения задачи (чем-то идейно похоже на бустинг);
- Tool-use datasets: учим модель самим правилам вызова тулов, остальное решаем промптингом.



https://community.cloudera.com/t5/Community-Articles/Understanding-Reasoning-Models-with-GRPO-A-Conceptual/ta-p/411569

# Примеры

| Модель | MoE | Reasoning | Agentic Skills | Open / Closed |
|---|---|---|---|---|
| **OpenAI o1** | – | + | – | Closed |
| **GPT-4.1 / GPT-4.2** | – | + | + | Closed |
| **Claude 3.5** | – | + | + | Closed |
| **DeepSeek-R1** | – | + | ± | Open |
| **DeepSeek-V3** | + | + | ± | Open |
| **DeepSeek-R1-MoE** | + | + | + | Open |
| **Qwen2.5-MoE** | + | ± | ± | Open |
| **Qwen2.5-Math** | – | + | – | Open |