LLM tuning

Обучение LLM

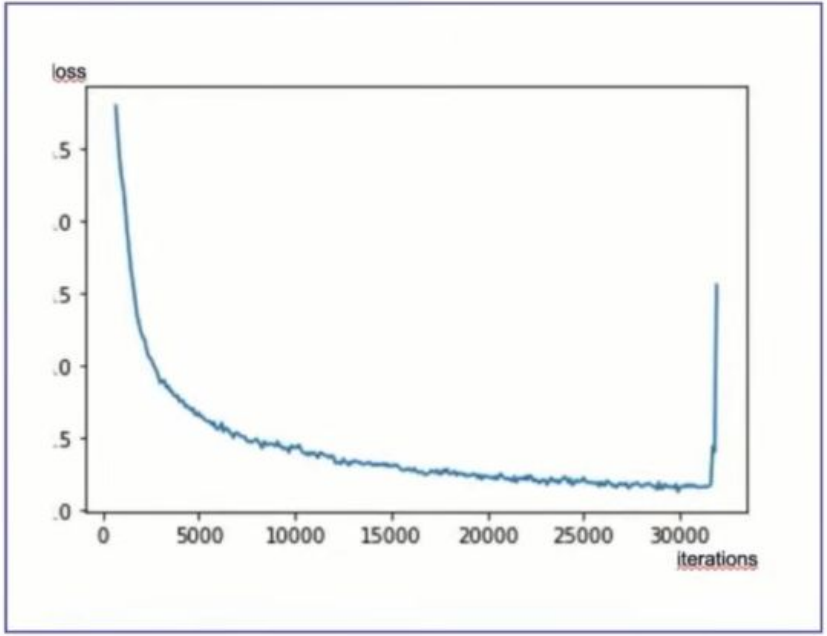
1. Pre training - показываем модели гигабайты неразмеченного
2. Alignment + RL - Учим модель на разметке людей выполнять инструкции. Усиливая возможности модели в zero shot

ИЛИ

1. Обучаем под конкретную продакшн задачу вид X -> Y (суммаризацию)
2. Последующий fine tuning

Проблемы обучения LLM

1. Огромное количество GPU
2. Много данных для обучения модели:
   a. 2 TiB чистых данных для обучения 13B модели
   b. x2 данных за x2 параметров
   c. сбор датасета
   d. очистка
3. Модели могут расходиться



Как использовать языковые модели

- Zero-shot
- Few-shot
- Full finetune
- Parameter efficient finetune (PEFT)

Трюки Zero-shot

- Свод правил - выписываем все требования к нашему запросу.
- Chain of thoughts (CoT)
- Reflection - Можно попросить модель улучшить ответ или найти в нем ошибку

Few-shot

Обучение с ограниченным числом примеров - показываем LLM несколько примеров, вопрос ответ и затем просим сгенерировать новый

Недостатки: Мы ограничены контекстом

Полный finetune

- Если примеров больше 10К то стоит обучать модель целиком:
- Можно обучать с zeroshot подводкой, модель лучше сойдется
- Вход стоит разделять от выхода спец токеном, например [SEP]
- Learning rate должен быть небольшим, сильно меньше, чем в pre-train
- В случае быстрого переобучения, можно попробовать менять dropout

Недостатки: забывание, ограничения по памяти

PEFT

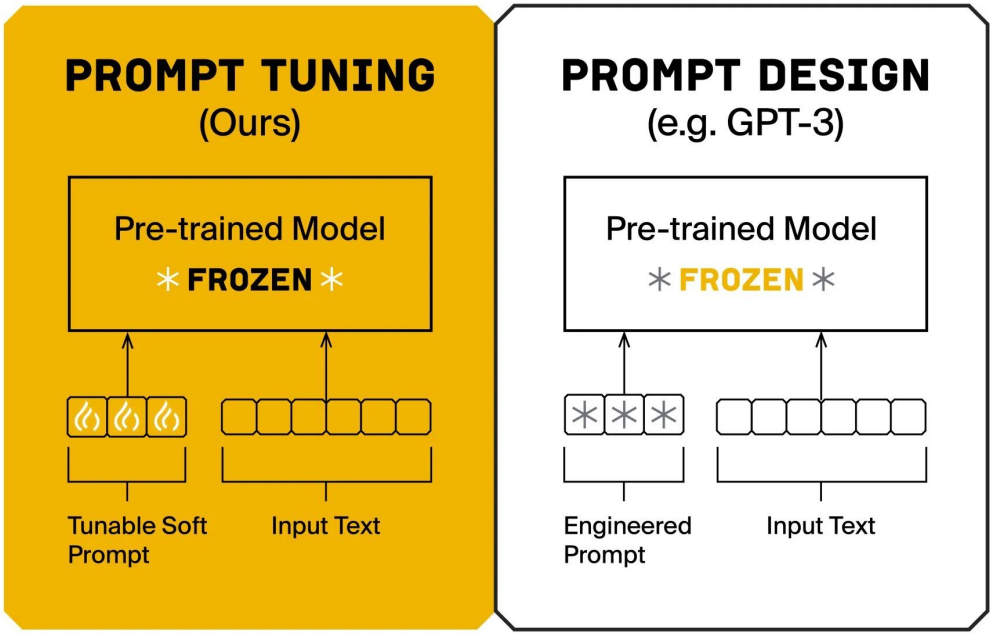
Преимущества PEFT (Parameter-Efficient Fine-Tuning)

1. **Ускорение времени обучения**: PEFT позволяет сократить количество времени, затраченное на обучение, благодаря файнтюнингу небольшого количества параметров, а не всей модели.
2. **Снижение затрат на вычисления и хранение**: PEFT выполняет тонкую настройку только небольшого подмножества параметров, что значительно уменьшает затраты на вычисления и хранение и снижает требования к оборудованию.
3. **Меньший риск переобучения**: Благодаря замораживанию большей части параметров предварительно обученной модели мы можем избежать переобучения на новых данных.
4. **Преодоление катастрофического забывания**: С помощью PEFT модель может адаптироваться к новым задачам, сохраняя ранее полученные знания благодаря заморозке большинства параметров.
5. **Удобство развертывания**: Контрольные точки, созданные при PEFT, компактнее, чем при традиционной тонкой настройке, что делает их развертывание и перенос на другие устройства более простым, поскольку они требуют меньше места для хранения и могут загружаться быстрее.

https://habr.com/ru/articles/791966/

Prompt tuning

Обучаем новые токены на конкретную задачу при этом модель не обучается

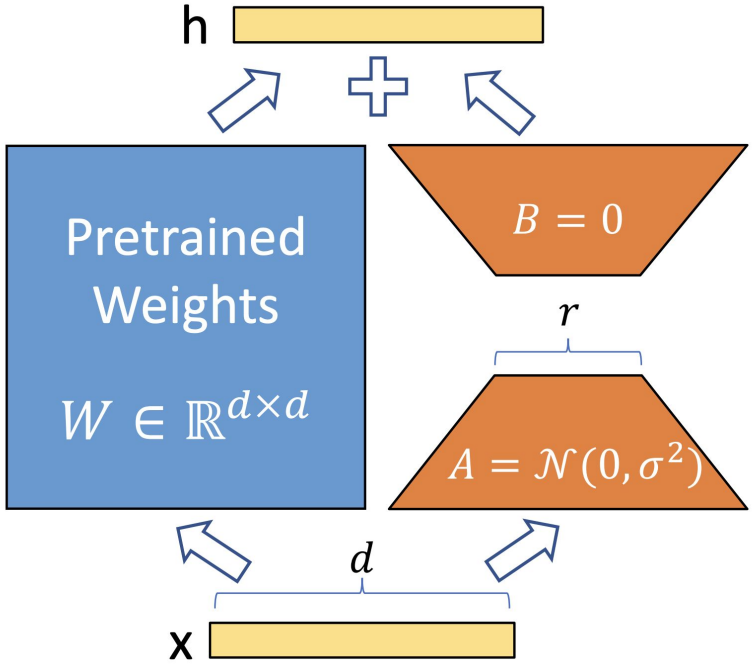


LoRA (Low Rank Adapter)

Обучаем не всю модель, а добавку к ней.

авторы отмечают, что, хотя LLM имеют миллионы параметров, они имеют низкую "внутреннюю размерность" (intrinsic dimension) при адаптации к новой задаче. Проще говоря, большинство параметров являются избыточными. Из чего можно сделать вывод, что матрицы можно представить пространством меньшей размерности, сохраняя при этом большую часть важной информации.

"r" (ранг) определяет размер матриц низкого ранга.



Our reparametrization. We only train A and B

https://arxiv.org/pdf/2106.09685.pdf

Alignment, что это и для чего он нужен ?

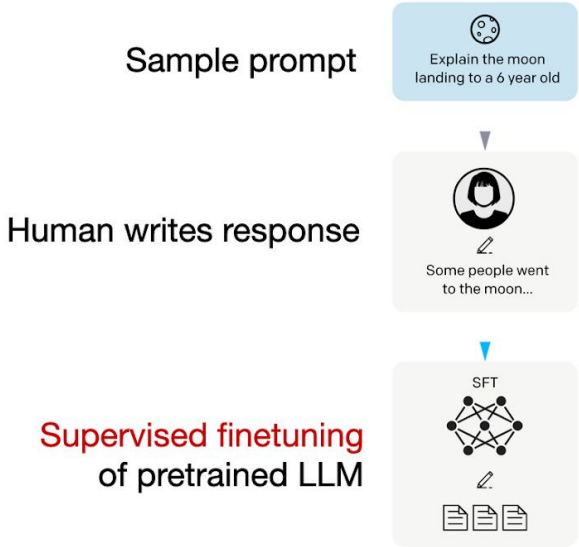
Pre-train - предсказывает следующий токен, нужно придумывать сложный промт, но не факт, что он поможет.

Alignment - обучаем модель выполнять инструкции от пользователя

Supervised Fine-Tuning (SFT)

Самый простой подход собрать датасет.

- Собираем возможные запросы от пользователей
- Пишем правильные ответы на собранные инструкции
- Дообучаем модель на полученных парах



Идеальный ассистент

1. Честный
2. Безопасный
3. Полезный

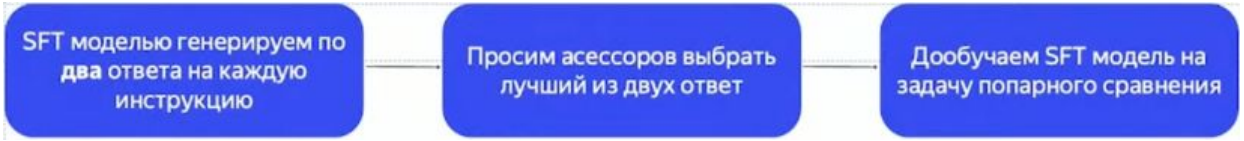
Пользуясь сводом правил разметчики оценивают модели, это дорого и долго

Нюансы асессороской оценки:

- Важен порядок ответов
- Honey pot - периодически нужно давать уже размеченные данные
- Транзитивность, не всегда выполняется

https://arxiv.org/pdf/2112.00861.pdf

Дистилляция асессоров в нейросеть



Обучение reward model

1) Для каждой пары ответов имеем разметку какой хороший, а какой плохо
2) Модель учителя инициализируем предобученой моделью
3) Выбрасываем классификационный слой
4) Обучаем новый линейный слой предсказывать число на сколько ответ хороший

Модель Брэдли-Терри:

$$P(x > y) = \frac{e^{f(x)}}{e^{f(x)} + e^{f(y)}} = sigmoid(f(x) - f(y))$$

Попарная функция ошибки (кросс-энтропия):

$$\mathcal{L}(r_w, r_l) = -\log P(r_w > r_l)$$

Как использовать модель награды ?

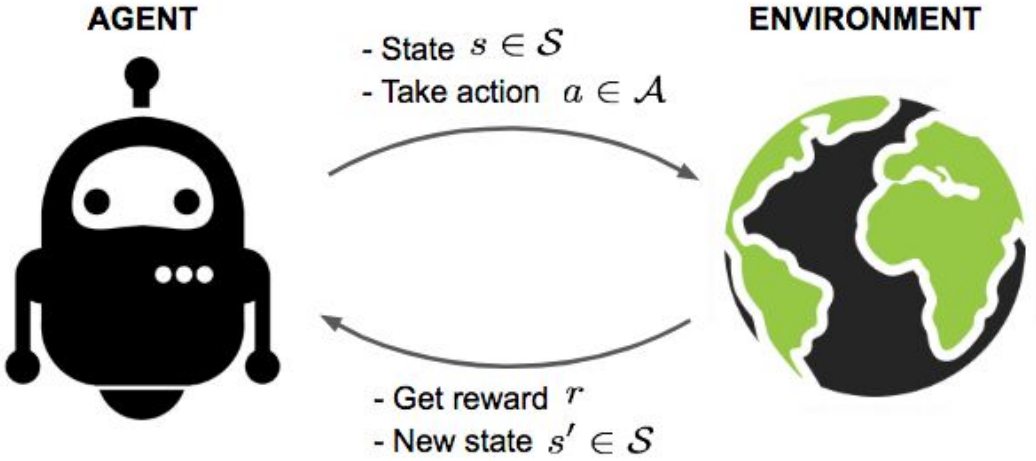
1. Остановка обучения модели с учителем
2. Выбор модели
3. Ранжирование ответов модели
4. Обучение с подкреплением

RL

Политика

$$\pi : \mathcal{A} \times \mathcal{S} \to [0, 1]$$
$$\pi(a, s) = \Pr(A_t = a \mid S_t = s)$$

π (действие|наблюдение)



Задача RL - максимизация награды, среднюю суммарную награду за эпизод взаимодействия со средой

$$\mathbb{E}_{\tau \sim \pi} \sum_{t=0}^{\infty} r_t \to \max_{\pi}$$

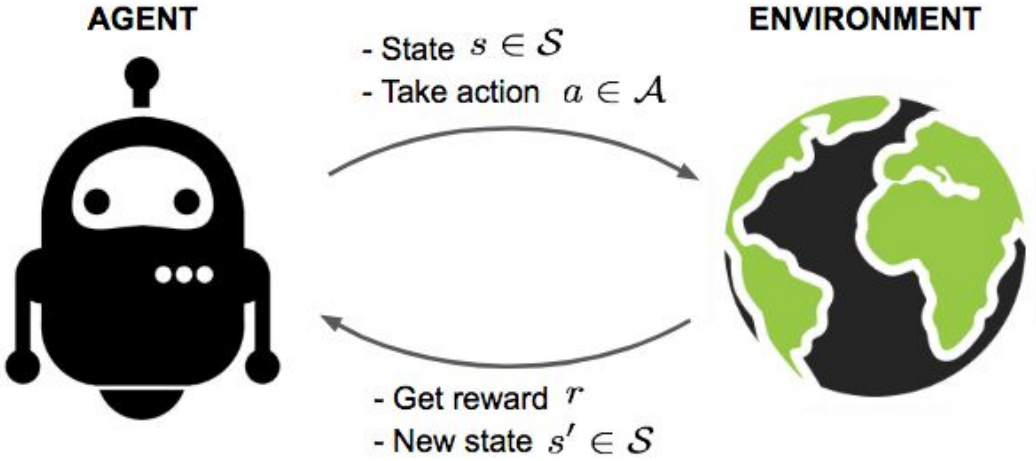
RL

Политика

$$\pi : \mathcal{A} \times \mathcal{S} \to [0, 1]$$
$$\pi(a, s) = \Pr(A_t = a \mid S_t = s)$$

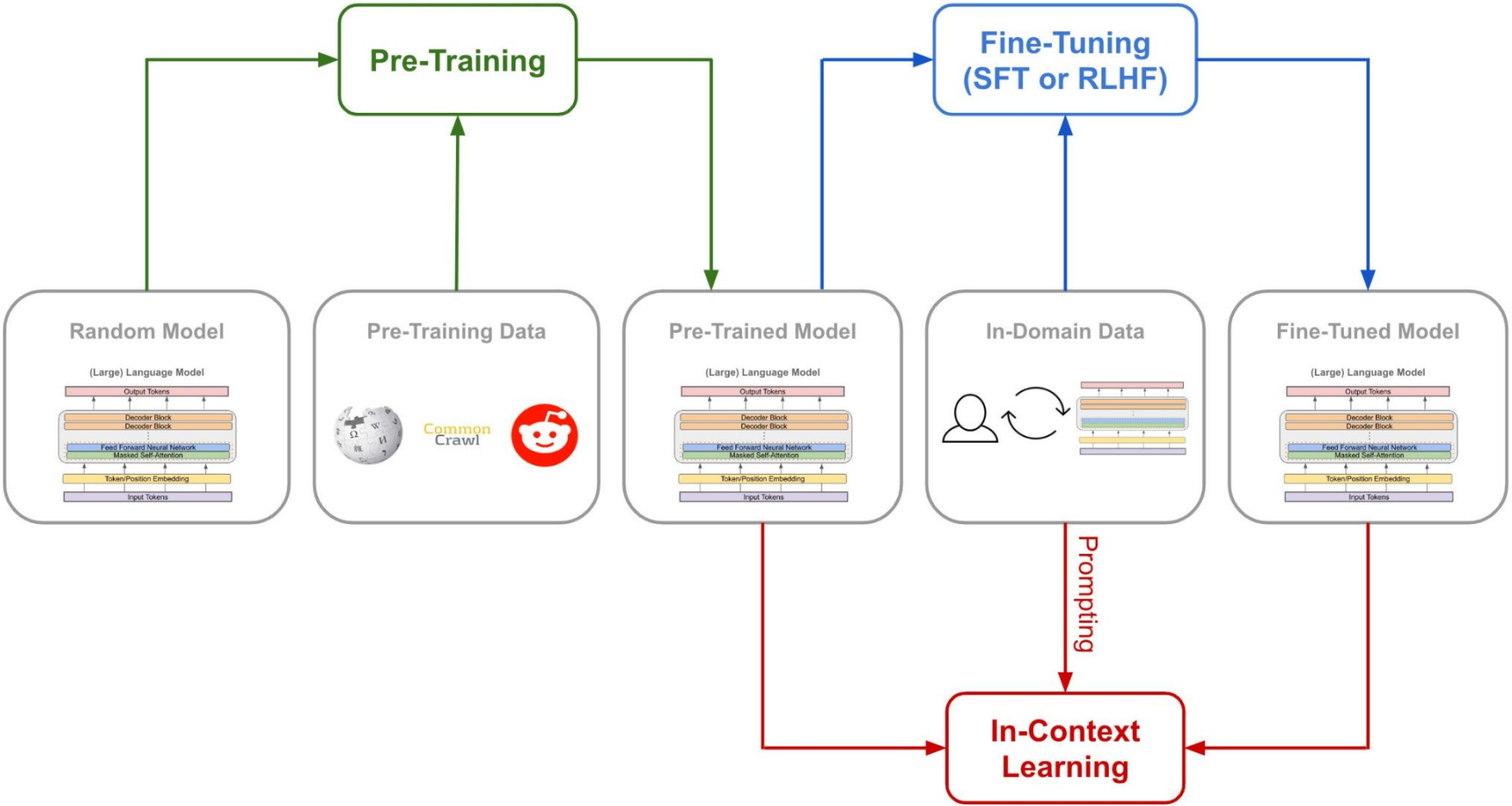
π (инструкция |ответ модели)

x - инструкция
y - ответы модели



Задача RL - максимизация награды, меняя поведение агента мы хотим выбирать политики с максимальной наградой

$$\mathbb{E}_{x \sim \mathcal{D}} \mathbb{E}_{y \sim \pi(\cdot|x)} r(x, y) \to \max_{\pi}$$


Pre-Training
Fine-Tuning
(SFT or RLHF)
Random Model
(Large) Language Model
Output Tokens
Decoder Block
Decoder Block
Feed Forward Neural Network
Masked Self-Attention
Token/Position Embedding
Input Tokens
Pre-Training Data
Common
Crawl
Pre-Trained Model
(Large) Language Model
Output Tokens
Decoder Block
Decoder Block
Feed Forward Neural Network
Masked Self-Attention
Token/Position Embedding
Input Tokens
In-Domain Data
Fine-Tuned Model
(Large) Language Model
Output Tokens
Decoder Block
Decoder Block
Feed Forward Neural Network
Masked Self-Attention
Token/Position Embedding
Input Tokens
Prompting
In-Context
Learning

https://www.youtube.com/watch?v=c7VUzWKFECg&list=PL6Wui14DvQPzMqtOOnfL00ZQ61JcN8V87&pp=iAQB