LLM training with human feedback

Ivan Reznikov
@qburst @mdxdubai

Shameless Self-Promotion

- PhD in Computational Sciences
- 12+ years of Python and Data Science experience
- Worked for small/medium/large enterprise companies, startups
- Principal Data Scientist at QBurst
- Kaggle Competition Expert
- TEDx Speaker (2017), GITEX/PyCON(2021) CoderHQ(2022, 2023)

https://www.linkedin.com/in/reznikovivan/
https://github.com/IvanReznikov

Roadmap

- Finetuning
- InstructGPT RLHF
- LLAMA2 RLHF
- ~~Pain~~ Fun with code

Why buzz with finetuning?

A person,
that can
search all
doctors
knowledge
(vectorstore)

An person **acting**
as a doctor
(prompt)

A **trained** doctor,
specialized in a
specific domain
(finetuned model)

LLM Training Pipeline

Large language models like can go through a 3-step training process:
1. Pretraining (initial training or self-supervised learning)
2. Supervised finetuning
3. RLHF (human-supervised finetuning)

Initially the models learn from huge amounts of text without specific labels. This way we obtain pretrained models.
In supervised finetuning, they get refined to follow particular instructions better.
During the alignment stage, the models can be finetuned even further to respond in a more helpful and safe way to user's input.

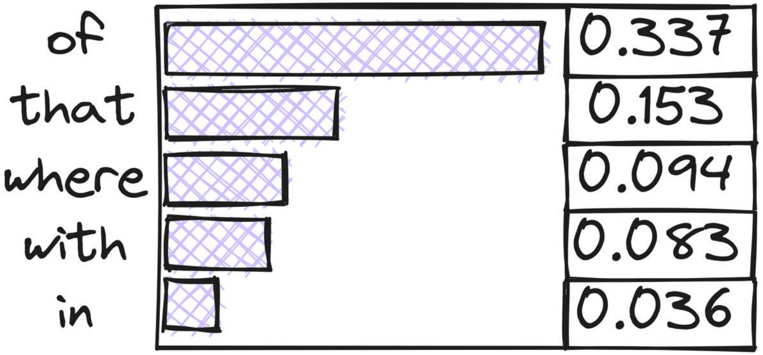
Pretraining

During pretraining, the model learns from a huge collection of text containing billions to trillions of words. For every phase, it's given a sentence and is tasked with predicting the next word or token that should come after it.

Dataset:
100B-10T tokens

Task:
Predict next token on unlabeled texts

Output:
base “pretrained” model



.from_pretrained(<model>)

In many pretrained language models, the tokenizer and the model architecture are designed and trained together. The reason for this coupling is that the tokenizer (“token producer”) and the model architecture need to be compatible with each other to ensure consistent tokenization and text generation.

If the tokenizer and the model architecture were different or not synchronized, it could lead to errors, mismatched embeddings, and incorrect predictions.

```
tokenizer = AutoTokenizer.from_pretrained("some_model")
model = BloomForCausalLM.from_pretrained("some_model")
```

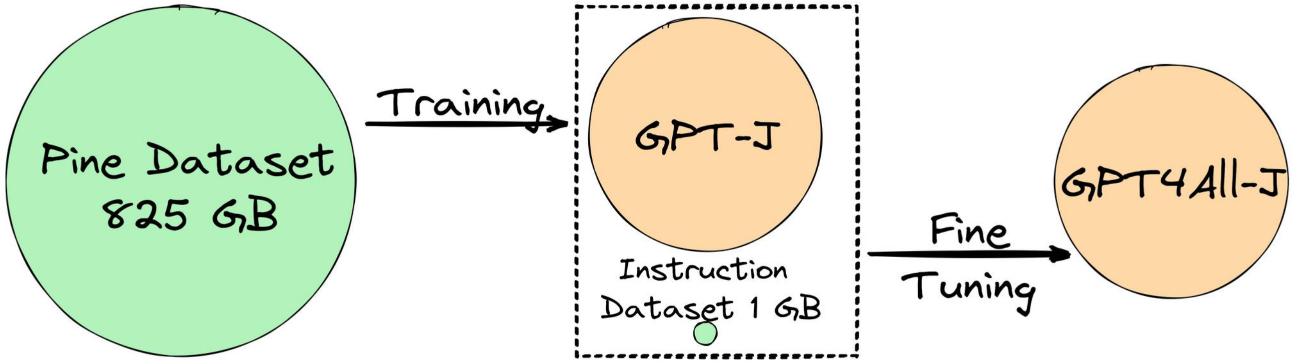
Supervised finetuning

In the supervised finetuning stage, we continue predicting the next token, but now we use pairs of instructions and their corresponding desired outputs.
The instruction is what we give the model as input (sometimes along with optional text), and the output is the response we want the model to generate, similar to our expectations.

Dataset:
10K-100K tokens

Task:
Predict next token on unlabeled texts

Output:
finetuned model



Supervised finetuning

text (string)
"I've learned the nitrogen vacancies used in Memristors are for "switching", between excited…
"Volunteer Services Volunteer Services As Charleston Area Medical Center volunteers, our…
"Q: Python: My return variable is always None So I found a strange thing that happens in python…

prompt (string)	response (string)
"<p>In Angular if I had a few panels, similar to tabs how do you open the panel on click and animat…	"To accomplish this in Angular, you can use ngAnimate and ng-show/ng-hide directives. Here's…
"Given the following scientific paper: This paper presents a novel approach for temporal and semanti…	"This scientific paper describes three different approaches for segmenting and analyzing edited…
" i am using the following code: <pre><code>package Presentacion; import…	"The error is caused by a typo in the code. The correct class name is "PrintWriter" and not…

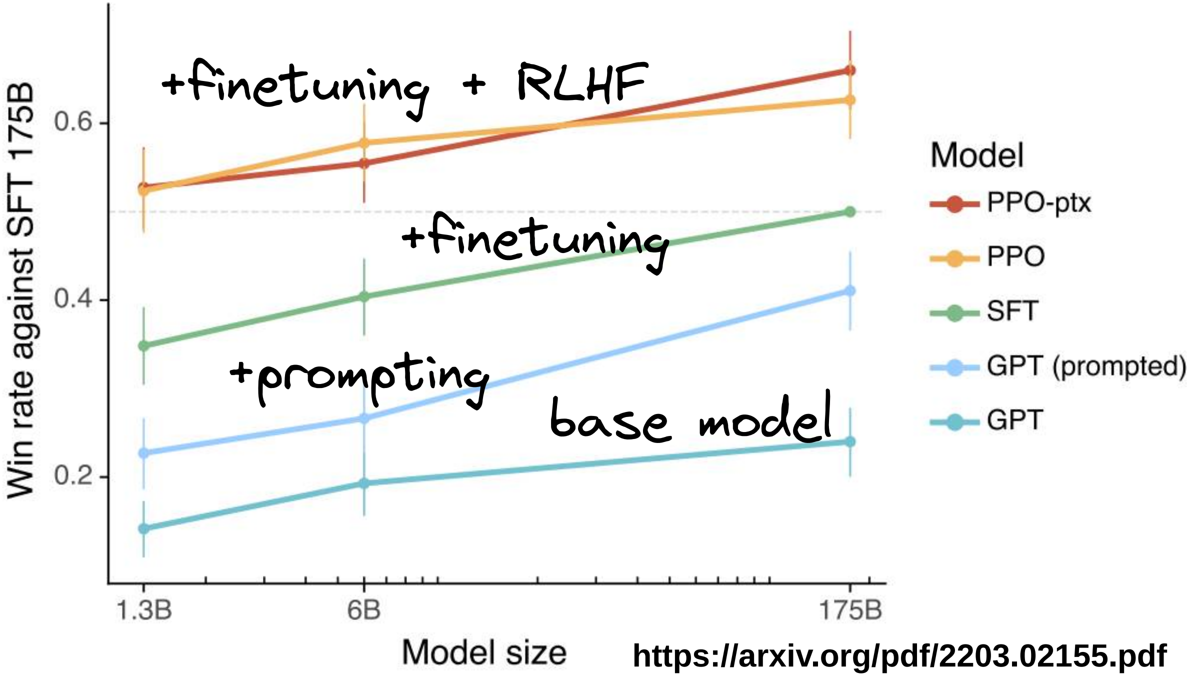
RLHF Alignment

RLHF (Reinforcement Learning from Human Feedback) is an important component of the current method used to train advanced language models. It helps include people's feedback when finetuning models, which ultimately makes the model more useful and secure.

Dataset:
50K-100K tokens

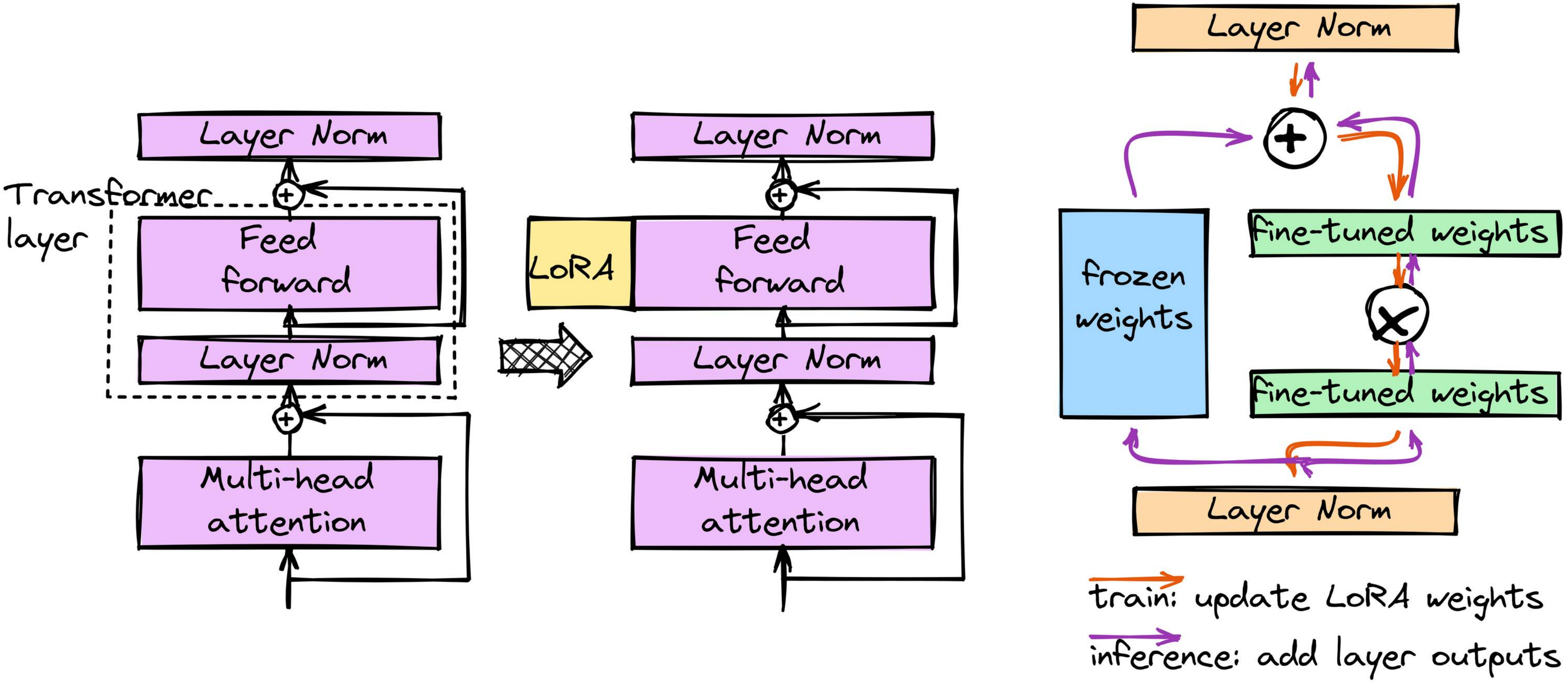
Task:
Align with user (human) preferences

Output:
human feedback-tuned model

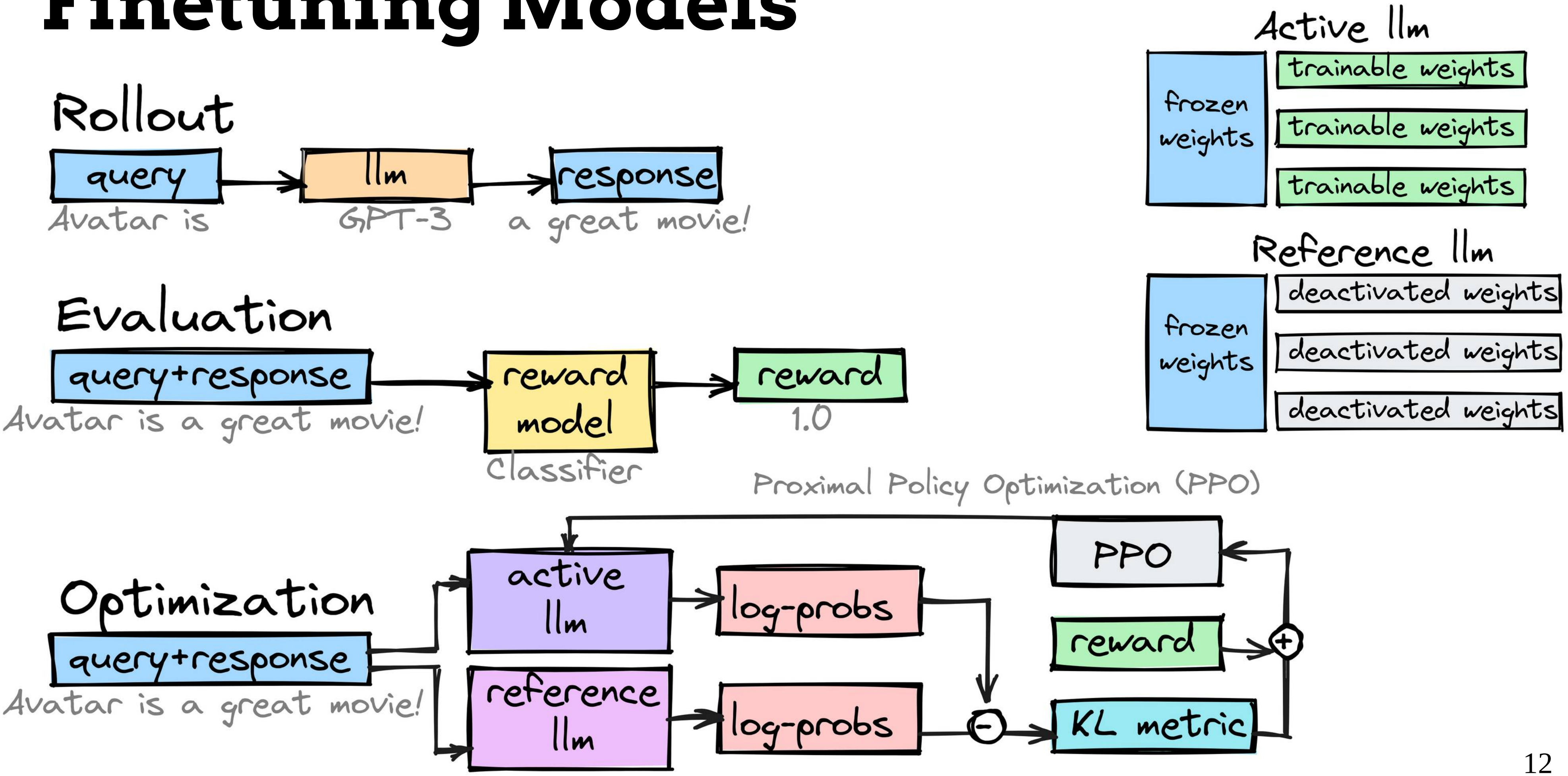


https://arxiv.org/pdf/2203.02155.pdf

Finetuning Models



Finetuning Models



(Instruct)GPT RLHF pipeline

The Instruct(GPT) RLHF pipeline involves taking a pretrained model and refining it through supervised training (similar to step 2 “Supervised finetuning” in the traditional training pipeline). Aftwerwards, the updated model is further refined using proximal policy optimization.

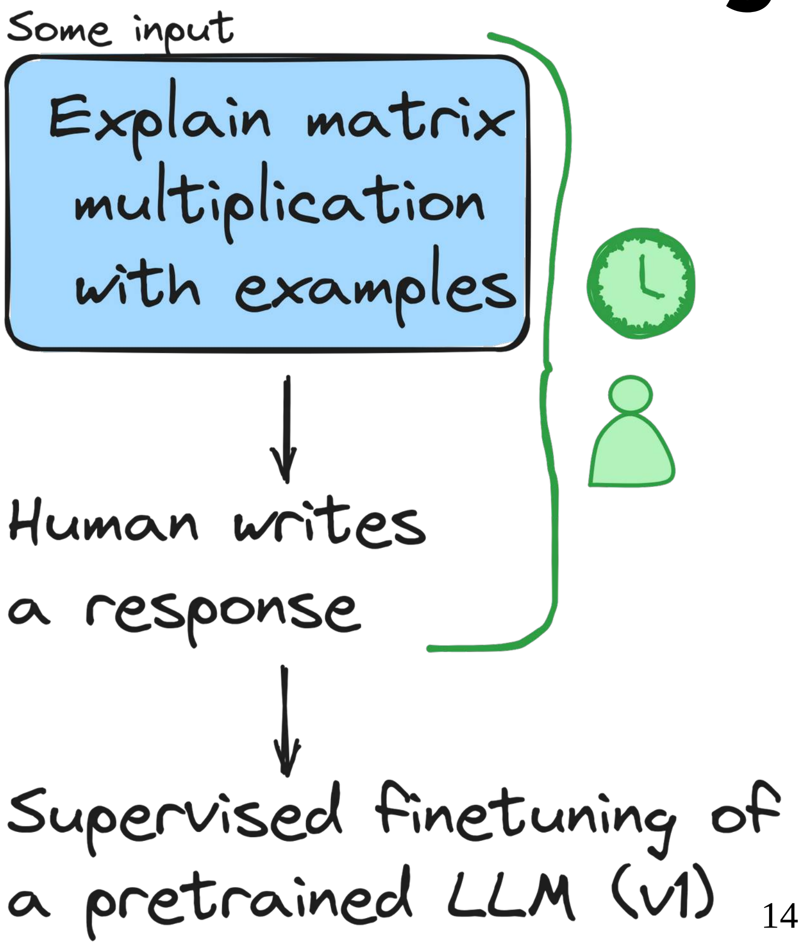
The RLHF pipeline is a 3-step training process:

1. Refined training of the pretrained model through supervision
2. Development of a model for providing rewards
3. Additional refinement using proximal policy optimization (PPO)

RLHF pipeline: Supervised finetuning

In the first step of RLHF pipeline, we either generate or select prompts (potentially from a dataset or database) and request humans to produce high-quality responses.

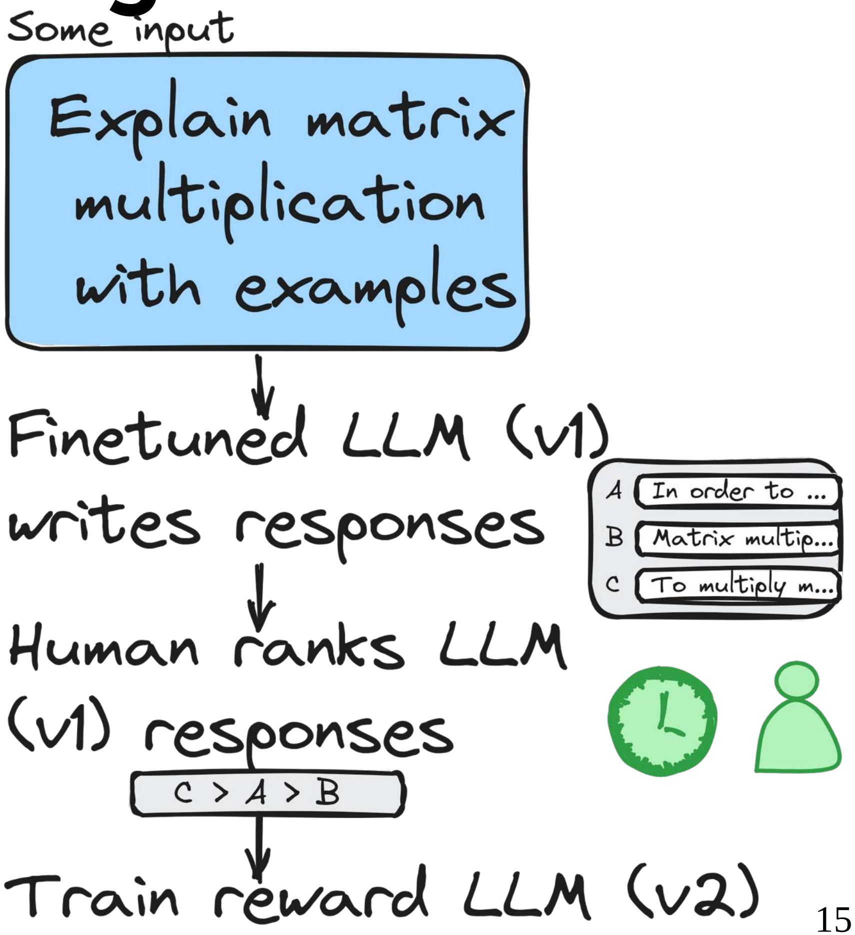
We utilize this collection of data to finetune the pre-existing base model in a guided manner.



RLHF pipeline: Training reward LLM

In RLHF pipeline step 2, we utilize the finetuned model via supervised training to construct a reward model for the next step.
This involves generating multiple responses for each prompt and having individuals rank them according to preference.

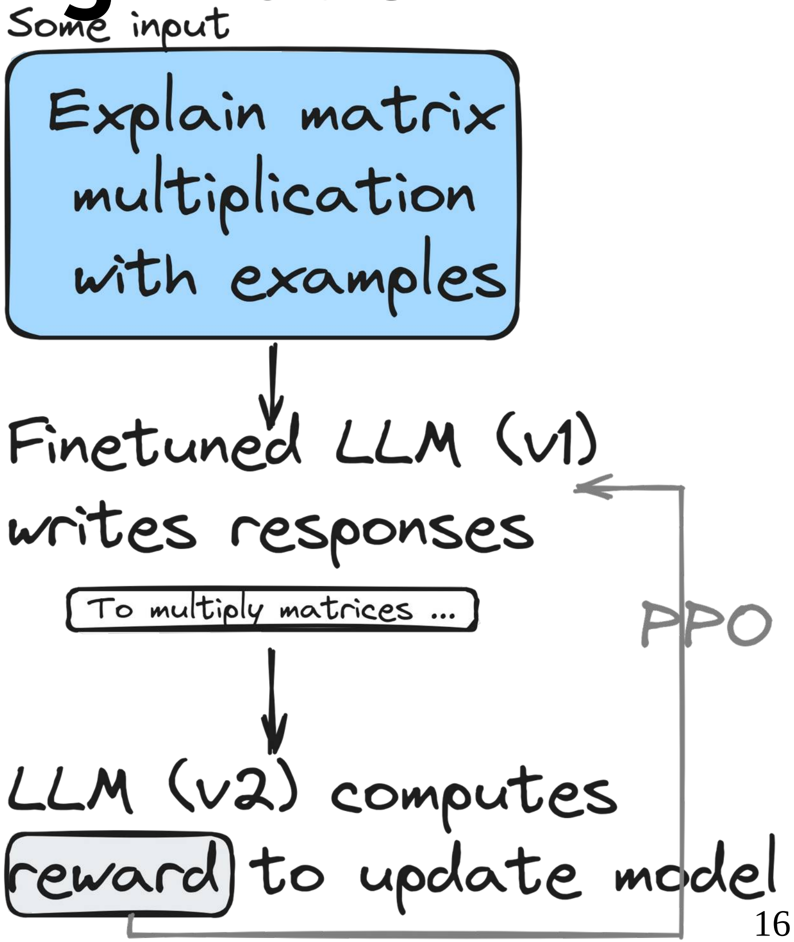
To transform the model from RLHF pipeline step 1 to a reward model, we replace its output layer (the next-token layer) with a regression layer that has a single output node.



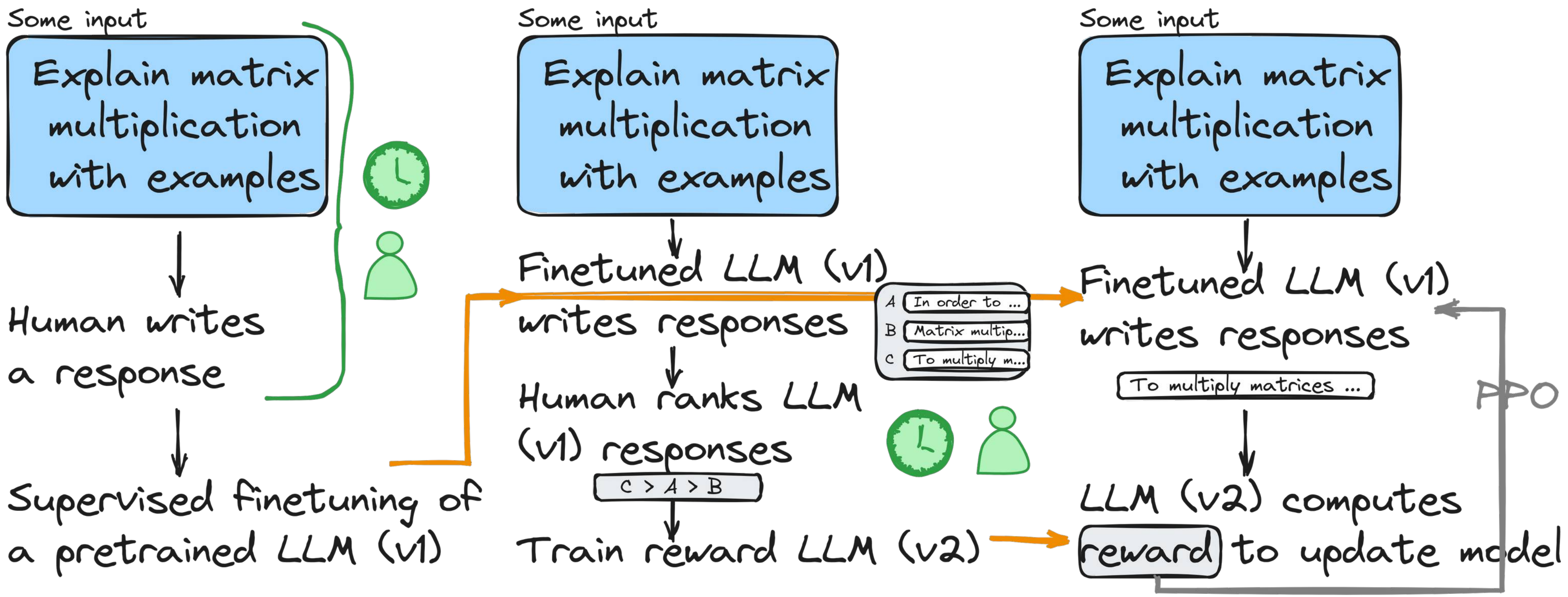
RLHF pipeline: Updating model

In the 3rd and last step of the RLHF pipeline, we employ the reward model (v2) to further finetune the previous model that underwent supervised finetuning (v1).

We adjust the v1 model using proximal policy optimization (PPO) guided by the reward scores obtained from the reward model we established in RLHF pipeline step 2.



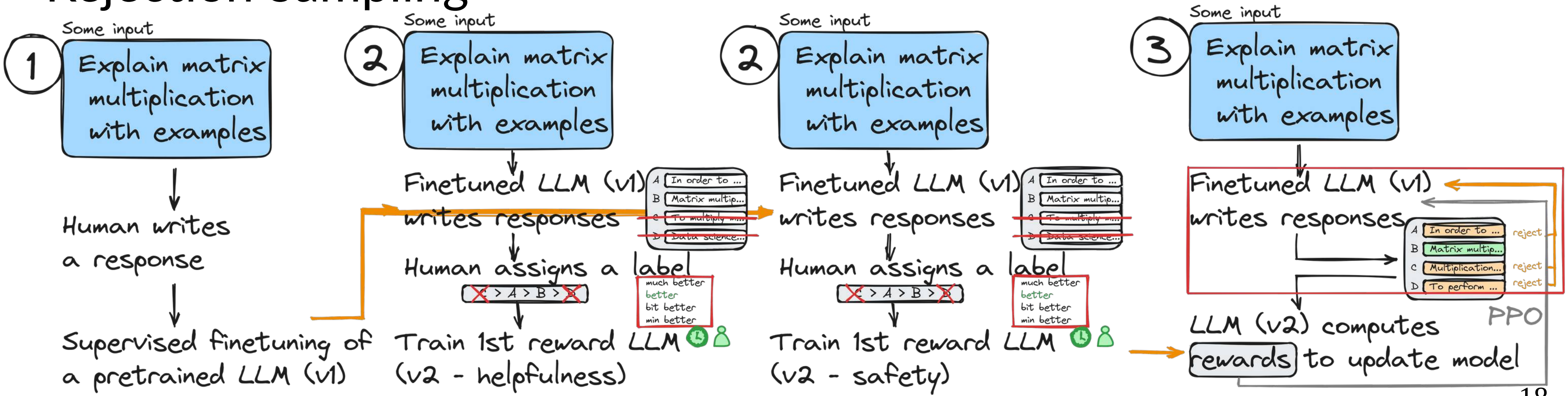
(Instruct)GPT RLHF pipeline



LLAMA RLHF pipeline

The Meta AI Llama 2 model, while using a similar RLHF approach to InstructGPT, introduces several noteworthy differences:

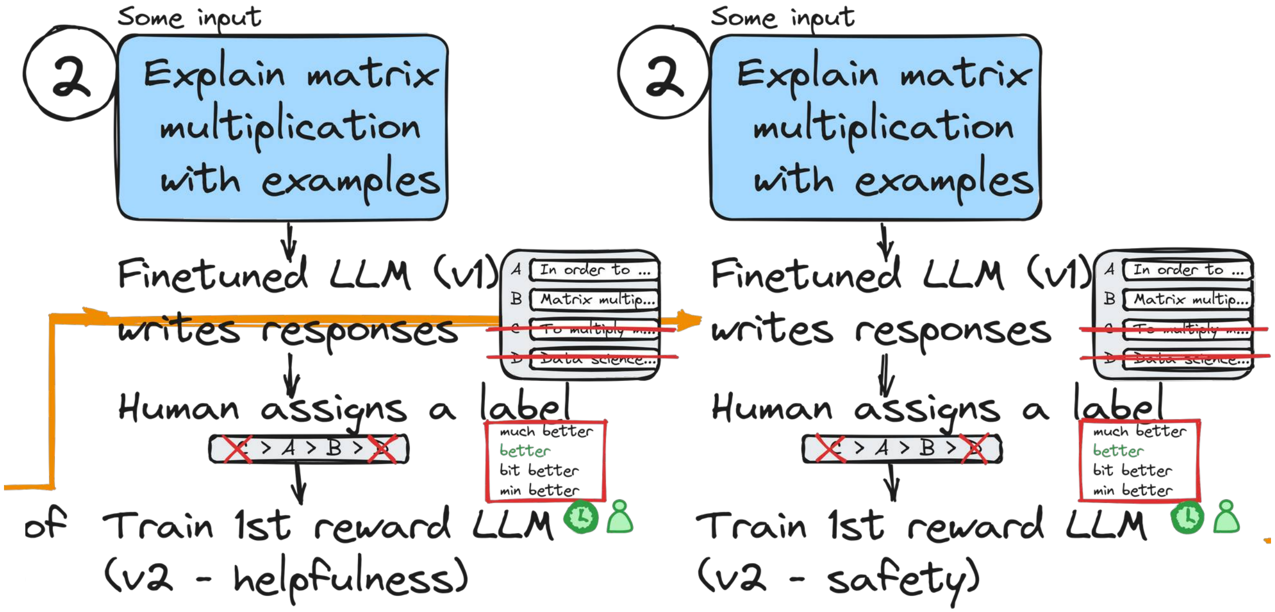
- Two Reward Models
- Margin Loss
- Rejection Sampling



LLAMA RLHF pipeline

Two Reward Models: Llama2 employs two reward models, one focused on helpfulness and the other on safety. The final optimization of the model is based on a combination of these two scores.

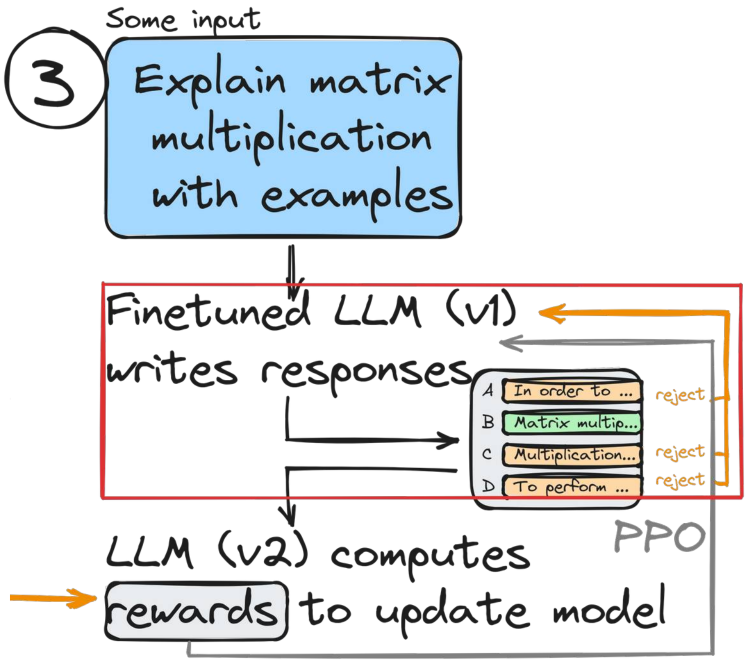
Margin Loss: In the process of ranking model responses, Llama2 introduces a "margin" label, which measures the gap between preferences. This margin parameter helps refine the ranking loss calculation.



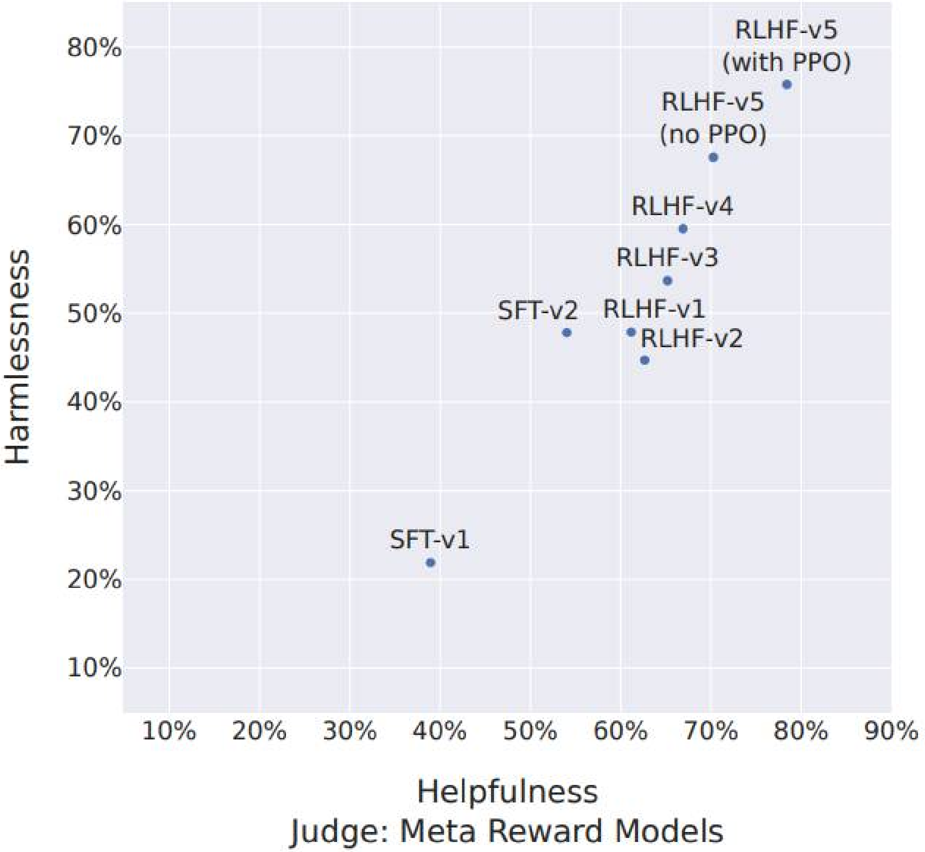
LLAMA RLHF pipeline

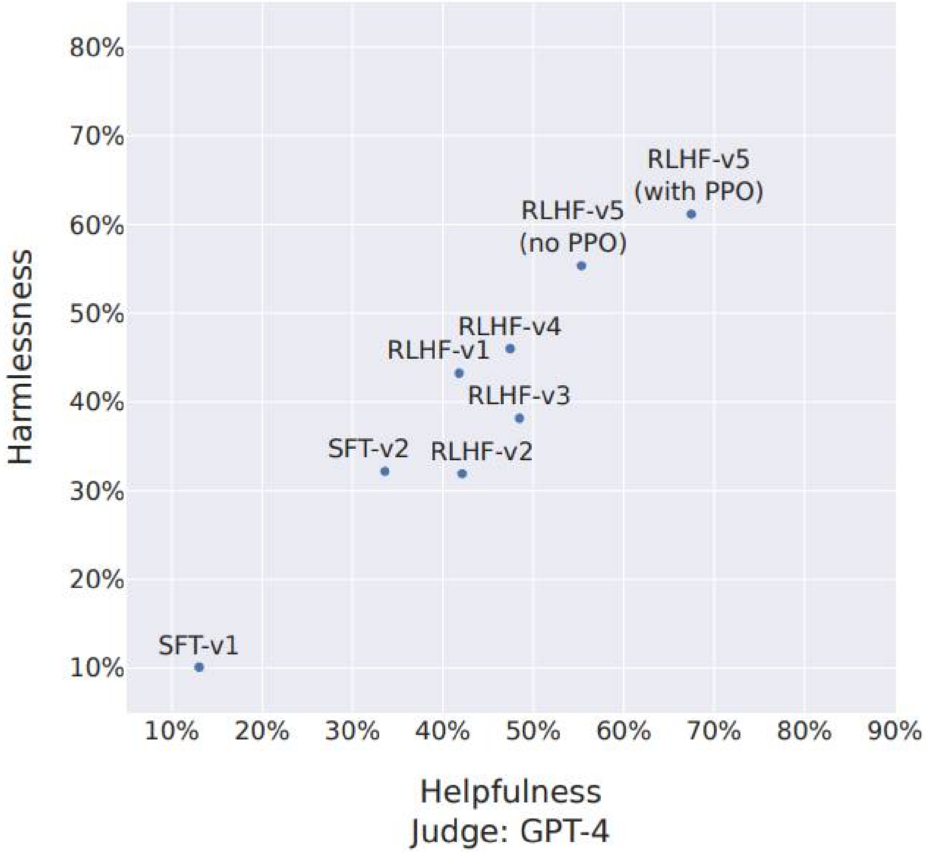
Rejection Sampling: Llama2 uses an iterative RLHF approach, creating multiple versions of the model (from RLHF-V1 to RLHF-V5). In addition to PPO, they employ rejection sampling.

This technique involves generating multiple outputs and selecting the one with the highest reward for gradient updates during optimization. It's different from PPO, which updates based on a single sample.



RLHF Pipelines Comaprison





https://arxiv.org/pdf/2307.09288.pdf

RLHF Alternatives

Constitutional AI: Harmlessness from AI Feedback
Researchers introduce a self-training method based on a set of rules provided by humans.

The Wisdom of Hindsight Makes Language Models Better Instruction Followers
The study introduces HIR (Hindsight Instruction Labeling), a two-step method involving prompt sampling and training, which effectively converts cases where the Language Model deviates from instructions

RLAIF: Scaling Reinforcement Learning from Human Feedback with AI Feedback
The study on RLAIF demonstrates that ratings used for reward model training in RLHF can be generated by an LLM rather than solely relying on human input.

Reinforced Self-Training (ReST) for Language Modeling
ReST is a method that aligns language models with human preferences through a sampling-based approach, iteratively training on progressively higher-quality subsets to enhance its reward function

Github repo

Notebooks and pdfs

https://www.linkedin.com/in/reznikovivan/
https://github.com/IvanReznikov