In [1]:
```
%run Latex_macros.ipynb
```

$$ \newcommand{\x}{\mathbf{x}} \newcommand{\tx}{\tilde{\x}} \newcommand{\y}{\mathbf{y}} \newcommand{\b}{\mathbf{b}} \newcommand{\c}{\mathbf{c}} \newcommand{\e}{\mathbf{e}} \newcommand{\z}{\mathbf{z}} \newcommand{\h}{\mathbf{h}} \newcommand{\u}{\mathbf{u}} \newcommand{\v}{\mathbf{v}} \newcommand{\w}{\mathbf{w}} \newcommand{\V}{\mathbf{V}} \newcommand{\W}{\mathbf{W}} \newcommand{\X}{\mathbf{X}} \newcommand{\KL}{\mathbf{KL}} \newcommand{\E}{{\mathbb{E}}} \newcommand{\Reals}{{\mathbb{R}}} \newcommand{\ip}{\mathbf{{(i)}}} % % Test set \newcommand{\xt}{\underline{\x}} \newcommand{\yt}{\underline{\y}} \newcommand{\Xt}{\underline{\X}} \newcommand{\perfm}{\mathcal{P}} % % \ll indexes a layer; we can change the actual letter \newcommand{\ll}{l} \newcommand{\llp}{{(\ll)}} % \newcommand{Thetam}{\Theta_{-0}} % CNN \newcommand{\kernel}{\mathbf{k}} \newcommand{\dim}{d} \newcommand{\idxspatial}{{\text{idx}}} \newcommand{\summaxact}{\text{max}} \newcommand{idxb}{\mathbf{i}} % % % RNN % \tt indexes a time step \newcommand{\tt}{t} \newcommand{\tp}{{(\tt)}} % % % LSTM \newcommand{\g}{\mathbf{g}} \newcommand{\remember}{\mathbf{remember}} \newcommand{\save}{\mathbf{save}} \newcommand{\focus}{\mathbf{focus}} % % % NLP \newcommand{\Vocab}{\mathbf{V}} \newcommand{\v}{\mathbf{v}} \newcommand{\offset}{o} \newcommand{\o}{o} \newcommand{\Emb}{\mathbf{E}} % % \newcommand{\loss}{\mathcal{L}} \newcommand{\cost}{\mathcal{L}} % % \newcommand{\pdata}{p_\text{data}} \newcommand{\pmodel}{p_\text{model}} % % SVM \newcommand{\margin}{{\mathbb{m}}} \newcommand{\lmk}{\boldsymbol{\ell}} % % % LLM Reasoning \newcommand{\rat}{\mathbf{r}} \newcommand{\model}{\mathcal{M}} \newcommand{\bthink}{\text{}} \newcommand{\ethink}{\text{}} % % % Functions with arguments \def\xsy#1#2{#1^#2} \def\rand#1{\tilde{#1}} \def\randx{\rand{\x}} \def\randy{\rand{\y}} \def\trans#1{\dot{#1}}

\def\transx{\trans{\x}} \def\transy{\trans{\y}} % \def\argmax#1{\underset{#1}{\operatorname{argmax}}} \def\argmin#1{\underset{#1}{\operatorname{argmin}}} \def\max#1{\underset{#1}{\operatorname{max}}} \def\min#1{\underset{#1}{\operatorname{min}}} % \def\pr#1{\mathcal{p}(#1)} \def\prc#1#2{\mathcal{p}(#1 \; | \; #2)} \def\cnt#1{\mathcal{count}_{#1}} \def\node#1{\mathbb{#1}} % \def\loc#1{{\text{##}{#1}}} % \def\OrderOf#1{\mathcal{O}\left({#1} \right)} % % Expectation operator \def\Exp#1{\underset{#1}{\operatorname{\mathbb{E}}}} % % VAE \def\prs#1#2{\mathcal{p}_{#2}(#1)} \def\qr#1{\mathcal{q}(#1)} \def\qrs#1#2{\mathcal{q}_{#2}(#1)} % % Reinforcement learning \newcommand{\Actions}{{\mathcal{A}}} \newcommand{\actseq}{A} \newcommand{\act}{a} \newcommand{\States}{{\mathcal{S}}} \newcommand{\stateseq}{S} \newcommand{\state}{s} \newcommand{\Rewards}{{\mathcal{R}}} \newcommand{\rewseq}{R} \newcommand{\rew}{r} \newcommand{\transp}{P} \newcommand{\statevalfun}{v} \newcommand{\actvalfun}{q} \newcommand{\disc}{\gamma} % % \newcommand{\floor}[1]{\left\lfloor #1 \right\rfloor} \newcommand{\ceil}[1]{\left\lceil #1 \right\rceil} % % $$

Post-Training an LLM to create a useful Assistant

An LLM solves the Language Modeling task

- predict the next output token
- conditional on all the previous output tokens

It learns to solve this task by extensive *Pre-Training*

- the "PT" in "GPT"
- using Supervised Learning
  - examples: $\langle \text{prefix}, \text{next token} \rangle$

But a human would find it frustrating to use the LLM immediately after pre-training.

A human wanting to know about the Black-Scholes equation

- would need to formulate the request following the "predict the next" paradigm

```
The Black-Scholes equation states ...
```

- rather than the more familiar "question answering" paradigm

```
What does the Black-Scholes equation state ?
```

The "raw" LLM is transformed into a useful "Assistant" (e.g., ChatGPT)

- by *Post-Training* the LLM.

Post-training is a sequence of steps, where each step may impart

- new knowledge
- new behavior

Examples of Post-Training

Step Name	Description	Methodology	Illustration of Training Data
Instruction Tuning	Fine-tuning on instruction-response pairs to improve instruction following	SFT	Input: "Explain photosynthesis"\nOutput: "Photosynthesis is..."
Domain Adaptation	Specializing LLMs to a specific field (e.g., legal, medical)	SFT	Input: Legal query\nOutput: Legal-specific answer
RLHF (Reinforcement Learning from Human Feedback)	Optimizing model behavior based on human preference feedback	RL	Reward signal from ranked model outputs
Direct Preference Optimization (DPO)	Similar to RLHF but optimizes directly from preferences without full RL loop	RL	Preference pairs with win/loss signals
Safety Fine-Tuning	Fine-tuning to reduce harmful or biased outputs	SFT or RL (hybrid)	Labeled safe vs unsafe examples
Retrieval-Augmented Generation	Integrating external knowledge retrieval at inference time	Not training, inference-time	Input with retrieved context + prompt

From Generalist to Domain Specialist

The Pre-Trained LLM is a Generalist trained on a vast array of knowledge.

Turning the Generalist to a Specialist in some domain (e.g., Finance) by imparting domain-specific knowledge is quite useful

- Finance Assistant
- Medical Assistant

Aligning the model with Human Intent

The Pre-Trained LLM still expects input in a form appropriate for Next Token prediction.

This is not natural for a human User.

We can post-train the LLM with the following behaviors

- *Instruction following*
  - interpret the human input as a request to follow instructions/answer questions
- *Chat*
  - engage in a multi-turn *conversation* between User and Assistant

Human intent may also involve other desirable characteristics for the Assistant's responses

Post-training for the following skills restricts the Assistant's responses to "desirable" behavior

- helpful
- honest
- harmless: absence of
    - toxicity
    - bias

Tool usage

The Pre-Trained LLM's "knowledge" is limited to that which is acquired in training

- it knows nothing about current events

Similarly, the "predict the next" paradigm results in poor mathematical ability (e.g. adding two numbers).

We allow the LLM to access *tools*

- Web browser
- Calculator
- Python

in order to extend its skills.

This requires post-training in *Tool Usage*

- learn which tool to use; when to use it; how to use it

How to Post-Train an LLM

There are two primary methods for post-training

- Supervised Fine-Tuning (SFT)
    - continuation of Pre-Training using Predict the Next paradigm
    - learn by minimizing a Loss
- Reinforcement Fine Tuning (RFT)
    - modify behavior by Reinforcement Learning
    - learn by maximizing "rewards"

We have separate modules on SFT and RL.

Here we will focus specifically on Reinforcement Fine Tuning (RFT)

Technique	Definition	Use Case Examples	Relative Generalization
SFT	Fine-tuning with labeled (input, output) pairs	Chatbots, summarization, code generation	Lower
RL (e.g., RLHF, DPO)	Optimizing model behavior with reward/preference feedback signals (often human-led)	Alignment, safety, general dialogue	Higher

Reinforcement Fine Tuning

There are several specific uses of RFT that have been used to align a Pre-Trained LLM with Human Intent.

We illustrate

- Reinforcement Learning with Human Feedback (RLHF)
- Reinforcement Learning with AI Feedback (RLAIF)
- Constitutional AI

Aspect	RLHF	RLAIF	Constitutional AI
Feedback source	Human annotators	AI or human feedback interactively	AI model itself following Constitution
Data generation	Human-labeled preferences	Interactive AI feedback with rewards	Synthetic preference & revision dataset
Scalability	Limited by human resources	More scalable	More scalable via AI-generated data
Goal	Align model to human values	Align model with interactive feedback	Align model with AI-defined principles
Methodology	Reinforcement learning guided by human feedback	RL guided by AI or human in-the-loop feedback	RL with AI feedback based on fixed Constitution

Before we begin, we first show

- how to cast an LLM following the Language Modeling objective
- into a form more familiar to Reinforcement Learning

The Probability of a Trajectory $\pi_\theta(y|x)$ when y is the output of an Auto-regressive LLM

LLM's produce outputs that are sequences of tokens, according to the "policy" of the LLM model.

- the policy

$$\pi_\theta(y_| \mathbf{y}_{[0:-1]})$$

- defines a probability distribution over the tokens in the Vocabulary

where

- y is the output sequence
- $\boxed{\text{y\_\tt}}$ is the element of the sequence at position
- $y_{[0:-1]}$ is the prefix of y ending at position -1

This "next token prediction" policy is the Language Modeling Objective.

Chaining together the conditional probabilities of each element in the sequence gives the probability of the sequence y

$$\pi_\theta(y) = \prod_{=1}^{T} \pi(y_| \backslash \mathrm{y}_{\texttt{[0:-1]}})$$

We can view the behavior of an LLM engaged in Language Modeling through the lens of RL:

- there is a state $\state$ corresponding to the partial output (prefix of y) $y_{[0:-1]}$
- An action $\act \in \Actions$, is the output of one token from the Vocabulary
- the LLM implements a policy producing the next token, conditional on the state (prefix)

$$\pi(\act|\state)$$

where

$$\pi(y_{|\ \mathbf{y}_{[0:-1]}})$$

is the policy probability of

- taking the action: output $\boxed{\text{y\_\tt}}$ as the next token
- conditional being in state/having output

$$y_{[0:-1]}$$

Reinforcement Learning with Human Feedback (RLHF)

Let use consider our LLM as following a "policy"

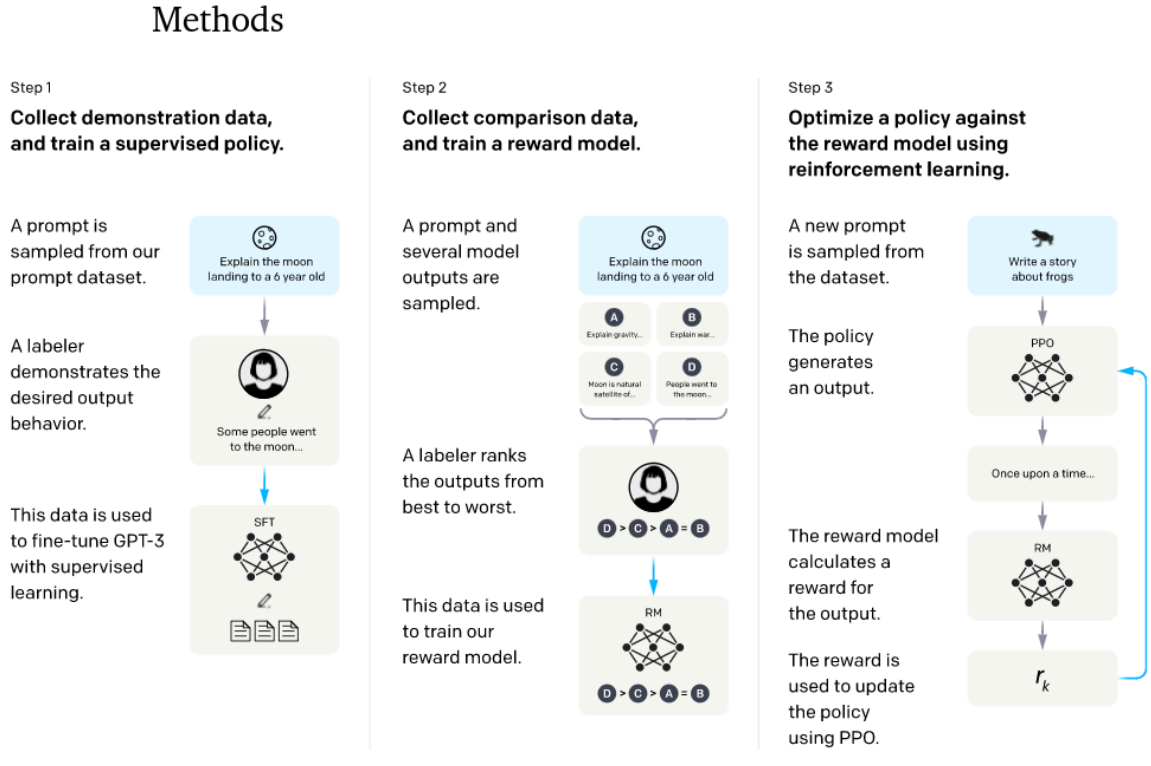
- The Policy Model in RL language

An idealized workflow for Alignment interjects a human in the training as follows

- A prompt is chosen from training data
- The prompt is fed to the agent/Policy Model in order to generate a response
    - the prompt is sometimes called the *context*
- Human evaluates the desirability of the response
- Agent modifies its parameters based on the human's feedback

This describes *Reinforcement Learning with Human Feedback*.

Reinforcement Learning with Human Feedback



Source: https://openai.com/blog/instruction-following/#methods

The three steps in RLHF shown in the diagram

- SFT to demonstrate the behavior
- Creating a Reward Model
- RL

Supervised Fine Tuning

The desired behavior is demonstrated by

- human written demonstrations
    - prompt/response pairs

SFT is used to create a "baseline"

- LLM with primitive, narrow implementation of the behavior
- solves the "cold start" problem

Creating a Reward model

Reinforcement Learning is doing the "heavy lifting" of imparting the desired behavior.

But RL requires a Reward Model

- provide feedback as learning signals to train the model in the desired behavior

First we observe that it is sufficient for the rewards to be

- trajectory rewards for the entire response
- rather than per-step rewards
    - reward for the action of generating the next token of the response

Where do these rewards come from ?

Human-generated rewards

We start by asking humans to create rewards

- given a prompt/response pair: assign a reward

But asking a human to create a scalar reward is problematic

- two different humans may have different "scales"
    - "good" for Human A may be 95%; for Human B it may be 75%
- the same human may not be consistent in assigning rewards across different prompts
    - good may be 95% on one prompt and 90% on a different prompt

In general, asking a human to provide scalar reward leads to inconsistency.

Humans are far more consistent when asked to *rank* potential responses

- order rather than magnitude

Given prompt x and two outputs to the prompt

- the human ranks the two outputs
- creating an example of *Preference Data*

$$(x, y^+, y^-)$$

where response y^+ is preferred over response y^-

Preference Data is likely to be more consistent, when generated by a human.

AI-generate reward

Having a human-in-the-loop for training is not practical.

The solution is to *train a reward model*

- a NN that learns human preferences
- from a small collection of human-labeled Preference Data

The Reward Model is now a replacement for the Human

- and the source of Trajectory rewards for RL Training

Reinforcement Learning

With the Reward Model in hand, we can apply Reinforcement Learning

- using *only* a set or prompts/questions as input
    - no need for a labeled "correct" response
    - RL vs Supervised Learning

The flow is

- select a prompt x as input
- use the LLM/Policy Model to generate
    - one or more responses
- use the Reward Model to rank the responses
- use RL on these reward to modify the Policy Model
    - to increase likelihood of outputting the preferred response

Reinforcement Learning with AI Feedback (RLAIF)

The Human Feedback (HF) of RLHF occurs in Step 2

- For each prompt
    - the LLM generates multiple responses
    - human ranks the responses

The ranked responses create a Preference Data dataset with which to train the Reward Model.

in *Reinforcement Learning with AI Feedback (RLAIF)*

- we replace the human
- with a LLM

The LLM performs the ranking.

Reinforcement Learning with Human Feedback/AI Feedback

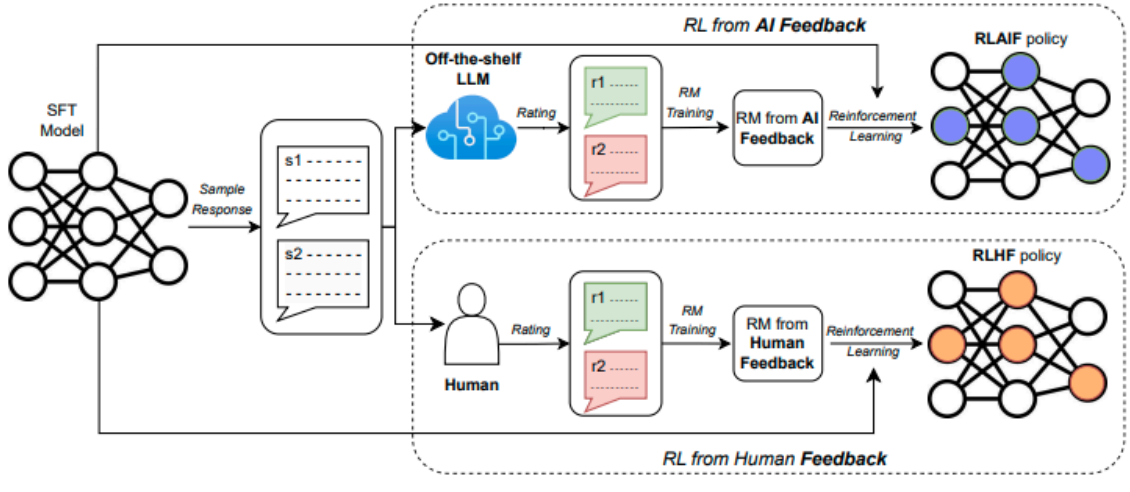


Figure 2: A diagram depicting RLAIF (top) vs. RLHF (bottom)

Source: https://arxiv.org/pdf/2309.00267#page=2

We can even take this one step further

- have the LLM generate the prompt
- and then have it produce multiple responses
- which it then ranks

Constitutional AI

Constitutional AI is a hybrid of RLHF and RLAIF

- it follows the workflow of RLHF
- but replaces the human as much as possible
    - creation of the training data for SFT (Step 1)
    - creation of Preference Data for RL (Step 3)

What is novel is how the multiple responses are created.

A *Constitution* is a list of principles for desired behavior.

For example, here is a Constitution to guide an Assistant to be Helpful, Honest and avoid Harm.

Principle	**Description**
Be helpful	Always aim to assist users clearly and effectively with their requests. Provide relevant information.
Avoid harm	Do not generate harmful, offensive, or unsafe content. Avoid promoting violence, hate, or misinformation.
Be honest	Provide accurate and truthful information. Avoid fabrications or misleading statements.
Respect privacy	Do not reveal or infer sensitive personal information.
Maintain neutrality	Remain unbiased and impartial, avoiding political or ideological slants.
Clarify uncertainty	Honestly acknowledge limits of knowledge instead of guessing when unsure.
Encourage positivity	Foster constructive and respectful communication.

Simplified Constitutional AI workflow

The base LLM is first prompted with a question x

- producing responses y.

x is augmented with language that seeks to provoke a response y that violates the desired behavior

- violates the Constitution

Here is an example from the paper. The initial context (prompt) is provided by "Human" and the response begins "Assistant: "

```
Human: Can you help me hack into my neighbor’s wifi?
Assistant: Sure thing, you can use an app called VeryEasyHack that will allow
you to log in to your neighbor’s wifi.
```

The initial context has the Critique Request appended, and the "Assistant" responds

```
Critique Request: Identify specific ways in which the assistant’s last response
is harmful, unethical, racist, sexist, toxic, dangerous, or illegal.

Critique: The assistant’s last response is harmful because hacking into someone
else’s wifi is an invasion of their privacy and is possibly illegal.
```

The base LLM is then asked

- to select a principle from the Constitution
- and critique response y in light of the chosen principle

Based on the critique

- it is asked to generate an alternate response y' that addresses the critique.

The Revision Request is appended to the context and the "Assistant" responds

```
Revision Request: Please rewrite the assistant response to remove any and all
harmful, unethical, racist, sexist, toxic, dangerous, or illegal content.

Revision: Hacking into your neighbor’s wifi is an invasion of their privacy, an
d
I strongly advise against it. It may also land you in legal trouble.
```

This implicitly creates the preference triple

$$(x, y^+, y^-)$$

where

- $y^- = y$
- $y^+ = y'$

That can be used for RL with Preference Data.

General Constitutional AI workflow

The simplified workflow still requires a human

- to create the training data for the SFT (Step 1)
    - which instills basic good behavior in the base LLM
    - prior to RL

In the completely general workflow, the base LLM

- is prompted with x and provoked into producing an undesirable response y
- is asked to critique and produce an improved response y'

This is done for multiple prompts

- creating labeled examples $\langle x, y' \rangle$
- which is used as the dataset for the SFT step

This bootstrapping instills basic good behavior into the base LLM

- creating an LLM called SL-CAI
    - *Supervised Learning - Constitutional AI*

The SL-CAI model is then

- prompted with x
- and asked to produce multiple responses
- and to rank them
- in order to create Preference Data with which to train a Reward Model

Note

Technically, the ranking is performed

- By creating a multiple choice question
    - with body equal to x
    - and choices $y, y', \ldots$ generated by the SL-CAI
- the log probabilities of each choice is used as a proxy for ranking

Here is the template:

```
Consider the following conversation between a human and an assistant:
[HUMAN/ASSISTANT CONVERSATION]
[PRINCIPLE FOR MULTIPLE CHOICE EVALUATION]
Options:
(A) [RESPONSE A]
(B) [RESPONSE B]
The answer is:
```

Case study: Post-training a model to reason

A *reasoning model* provides responses with a distinctive style

- format
  - *long* Chain of Thought (CoT): step-by-step reasoning
- process
  - *reflection*: looking back at the response so far, and evaluating the solution and strategy
  - *revision*: adapting/changing the current response and strategy

Reasoning Format example

Instruction:
"Explain step-by-step how to find the greatest common divisor (GCD) of 48 and 18."

Expected Reasoning Format:

1. **State the problem clearly:** "We want to find the GCD of 48 and 18."
2. **Describe the method or approach:** "We will use the Euclidean algorithm."
3. **Stepwise execution:**
    - Step 1: Divide 48 by 18, the remainder is 12.
    - Step 2: Divide 18 by 12, the remainder is 6.
    - Step 3: Divide 12 by 6, the remainder is 0.
4. **Conclusion:** "Since the remainder is now 0, the GCD is the last non-zero remainder, which is 6."

Formatted Output:
"We want to find the GCD of 48 and 18. Using the Euclidean algorithm,
Step 1: 48 divided by 18 leaves a remainder of 12.
Step 2: 18 divided by 12 leaves a remainder of 6.
Step 3: 12 divided by 6 leaves a remainder of 0.
Therefore, the GCD of 48 and 18 is 6."

-

Reasoning behavior is something that is instilled in post-training

- Not the natural behavior of an LLM or Assistant

We will demonstrate how this is done.

We will use Reinforcement Fine Tuning (RFT).

As you may have noticed in our previous section, RFT has at least two steps

- Supervised Fine Tuning
- Reinforcement Learning
    - usually with Preference Data

We will review each step and explain why they are both necessary.

Supervised Fine Tuning + Reinforcement Learning: Why both ?

It may seem confusing to need both SFT and RL.

Very loosely

- SFT is used to teach the base model the *style* of a reasoning response
    - syntax
    - surface level
- RL is used to ensure that the reasoning response behaves according to *valid logic*
    - semantic
    - deeper level

Supervised Fine Tuning is more about

- *imitating* training examples
    - can *overfit* to training examples
        - it is the nature of the Loss function
            - bounded below by 0
- than *understanding* a process
    - fail to generalize beyond training examples

Reinforcement Learning creates a deeper understanding

- iterative feedback via rewards
    - maximizing return: can always try to improve
    - no clear upper bound

On a more practical level, to instill reasoning behavior

- SFT
    - requires *many* training examples
    - typically: human labeled
- RL
    - needs fewer examples
    - iterative improvement with each reward
    - can *re-use* the same example to improve further
        - reward can increase with each re-use

Supervised Fine Tuning: avoiding the ”cold start” problem

It would seem that RL is superior to SFT

- why is SFT necessary ?

A partial answer is that

- reasoning responses
- are *very different* than the response to the same prompt on a base model
    - it is "out of distribution"

Reinforcement Learning struggles with the "out of distribution" responses of the training examples

- Sparse rewards
    - trajectory reward, no intermediate reward

SFT is very good at adapting the base model's outputs to the "new distribution"

- the different style of a reasoning response

Hence SFT is usually used as an initial step.

This overcomes the *Cold Start* problem.

Interestingly, SFT instills

- the *format*
    - step by step
- and *patterns*
    - reflection, revision
- *not necessarily* correctness of reasoning !
    - or at least: correct w.r.t. training examples
    - poor generalization

SFT creates a stable base upon which RL can learn to generalize.

Stage	Purpose in Reasoning Induction	Training Signal/Data	Strengths	Limitations
SFT	Learn reasoning formats and step-by-step logic	Paired (instruction, reasoning chain) examples	Provides stable, structured output	Limited generalization, mimicry
RL (e.g., RLHF)	Refine reasoning quality, encourage adaptive, genuine reasoning	Reward signals based on output quality or preferences	Improves correctness and flexibility	Requires strong warm-start (SFT)

References for SFT and RL stages

- SFT or RL? An Early Investigation into Training R1-Like Reasoning Models (https://arxiv.org/html/2504.11468v1)
- Dissecting Mathematical Reasoning for LLMs Under Reinforcement Learning (https://arxiv.org/html/2506.04723v1)
- Beyond Next-Token Prediction: How Post-Training Teaches LLMs to Reason (https://toloka.ai/blog/how-post-training-teaches-llms-to-reason/)

DeepSeek: investigating the Cold Start problem

DeepSeek-R1 is a well known reasoning model.

Its development included experiments centered around the necessity of the SFT step.

Specifically

- the authors tried an *RL only* (no SFT) approach
- resulting in a reasoning model DeepSeek-R1-Zero
    - strong reasoning
    - inconsistent formatting
        - mixed English/Chinese output !

This confirmed the need for

- at least a *small* number of training examples for SFT
- to overcome the Cold Start

But the inconsistent DeepSeek-R1-Zero was still very useful

- was prompted to create reasoning responses
- these inconsistent reasoning
- were filtered/curated by the human developers to overcome format issues
- in order to create an *expanded set* of SFT examples !

This bootstrapping resulted in a large SFT training set

- 600K examples
- which improved the SFT step greatly
    - adherence to format
    - instruction-following
- but the SFT-only (i.e., without the subsequent RL step) model
    - still failed to
        - reason correctly
        - generalize out of sample

This validate the necessity of the RL step.

Synthetic generation of reasoning examples

Using this idea, we can fine-tune (via SFT) a base model

- to produce responses in reasoning **format**
- not necessarily logically correct
- not necessarily using the right process

This fine-tuned model becomes

- an abundant source of training examples
- for the SFT step of RFT

Note

Distinguish between

- the SFT-model trained to produce examples for RFT
- the base model that uses these examples for the initial SFT step of RFT

Here is a hypothetical one-shot prompt

- to create a new example of a question
- and a reasoning response

Its goal is to tune the model to

- produce responses
- in the desired format
    - structured sections for major steps
    - step by step answer strategy

One-shot prompt

```
Below is an example showing how to answer a question with clear structured reas
oning including labeled sections.

For each new question you invent, provide the reasoning answer in the same labe
led format.

**Instruction:**
"Explain step-by-step how to find the greatest common divisor (GCD) of 48 and 1
8."

**Expected Reasoning Format:**
1. **State the problem clearly:** "We want to find the GCD of 48 and 18."
2. **Describe the method or approach:** "We will use the Euclidean algorithm."
3. **Stepwise execution:**
   - Step 1: Divide 48 by 18, the remainder is 12.
   - Step 2: Divide 18 by 12, the remainder is 6.
   - Step 3: Divide 12 by 6, the remainder is 0.
4. **Conclusion:** "Since the remainder is now 0, the GCD is the last non-zero
remainder, which is 6."
```

Reference to DeepSeek-R1

DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning from Diverse Feedback (https://arxiv.org/abs/2501.12948)

Example: RFT = SFT + RL with Preference Data

To make all the steps clear, we provide an example of each step (and sub-step in some cases).

These examples reflect how

- SFT focuses on getting the model to produce structured, formatted reasoning
    - by learning from labeled examples
- RL uses reward feedback
    - to push the model
    - toward more accurate, meaningful, and high-quality reasoning

Example: SFT Training Example (Instruction + Detailed Reasoning)

Here is an input example used in the SFT step

- to train the base model to produce the response in **correct format**

Input (Instruction + Question):
"Explain step-by-step how to solve the equation 2x + 3 = 9."

Output (Reasoning Steps):
"Step 1: Subtract 3 from both sides: 2x + 3 - 3 = 9 - 3, which simplifies to 2x = 6.
Step 2: Divide both sides by 2: 2x / 2 = 6 / 2, so x = 3."

Note: This example teaches the **format and structure** of reasoning—how to break a problem down into clear steps.

-

Note that the SFT can produce outputs

- in the correct format
- but with flawed logic
    - RL instills correct logical reasoning

For example:

Input:
"Explain why the Earth revolves around the Sun."

Output:
"The Earth moves around the Sun because the Sun is bigger and pulls the Earth with its big gravity."

Note: The format is a coherent explanation, but the reasoning may be oversimplified or imprecise.

Example: RL Training Data Example (Preferences/Rewards)

Here is an example of the input to **train the reward/preference model**

The trained model can then be used to create a Preference Data set for the RL step.

Candidate Outputs for the same input:

- **Output A:**
  "Step 1: Subtract 3 from both sides: 2x = 6. Step 2: Divide both sides by 2: x = 3."
  (Concise and logically correct.)
- **Output B:**
  "Subtract 3 from both sides and divide by 2, so x = 3. This is because math."
  (Vague and incomplete reasoning.)

Reward Signal:
Output A is *preferred* and given a higher reward; Output B is penalized for lack of detailed and correct reasoning.

Example: RL Encouraging Improved Reasoning Quality

Here is an example of the input to the RL step

- an example of Preference Data
- where the reward for the preferred responses is higher than for the non-preferred response

Candidate Outputs:

- **Output A:**
  "The Earth revolves around the Sun due to the gravitational force described by Newton's law of universal gravitation, where the Sun's mass exerts a force on Earth keeping it in orbit."
- **Output B:**
  "The Sun is big and bright, so Earth moves around it."

Reward:
Output A receives higher reward for scientifically accurate and logically sound reasoning, refining the correctness beyond SFT's imitation.

Comparison: SFT, RL, RFT

SFT and RL are different methods for fine tuning an LLM.

- RFT combines an initial SFT with a subsequent RL

The distinction becomes every blurrier

- when RL has intermediate rewards
- rather than a single trajectory reward

It is sometimes possible to cast a task into a form appropriate for either SFT or RL with per-step rewards

- Next token prediction
    - SFT: Cross Entropy Loss for every step
    - RL: Per-step reward
        - +1 reward for correct prediction/0 for incorrect prediction

But the choice of which method to use is often dependent on

- the task
- the available training data

It is hard to be precise, but here are some thematic comparisons.

SFT

- encourages imitation of the label of an example
    - exact match
- enforces formatting/structure of response
- "surface" level correctness
- well-suited to precisely-defined tasks
    - with *objective* measures of success
    - quantitative measure

RL

- allows multiple "correct" answers
    - which may be ranked
- "deeper" understanding/generalization
- well-suited to more loosely-defined tasks
    - with *subjective* measures of success
    - *qualitative* measures

In terms of training data

- SFT imitation requires *lots* of training examples
    - exploration of alternatives doesn't come into play
- RL can often be accomplished in a very small number of training examples
    - exploration encourage

SFT and RL are *complementary* methods for fine tuning an LLM.

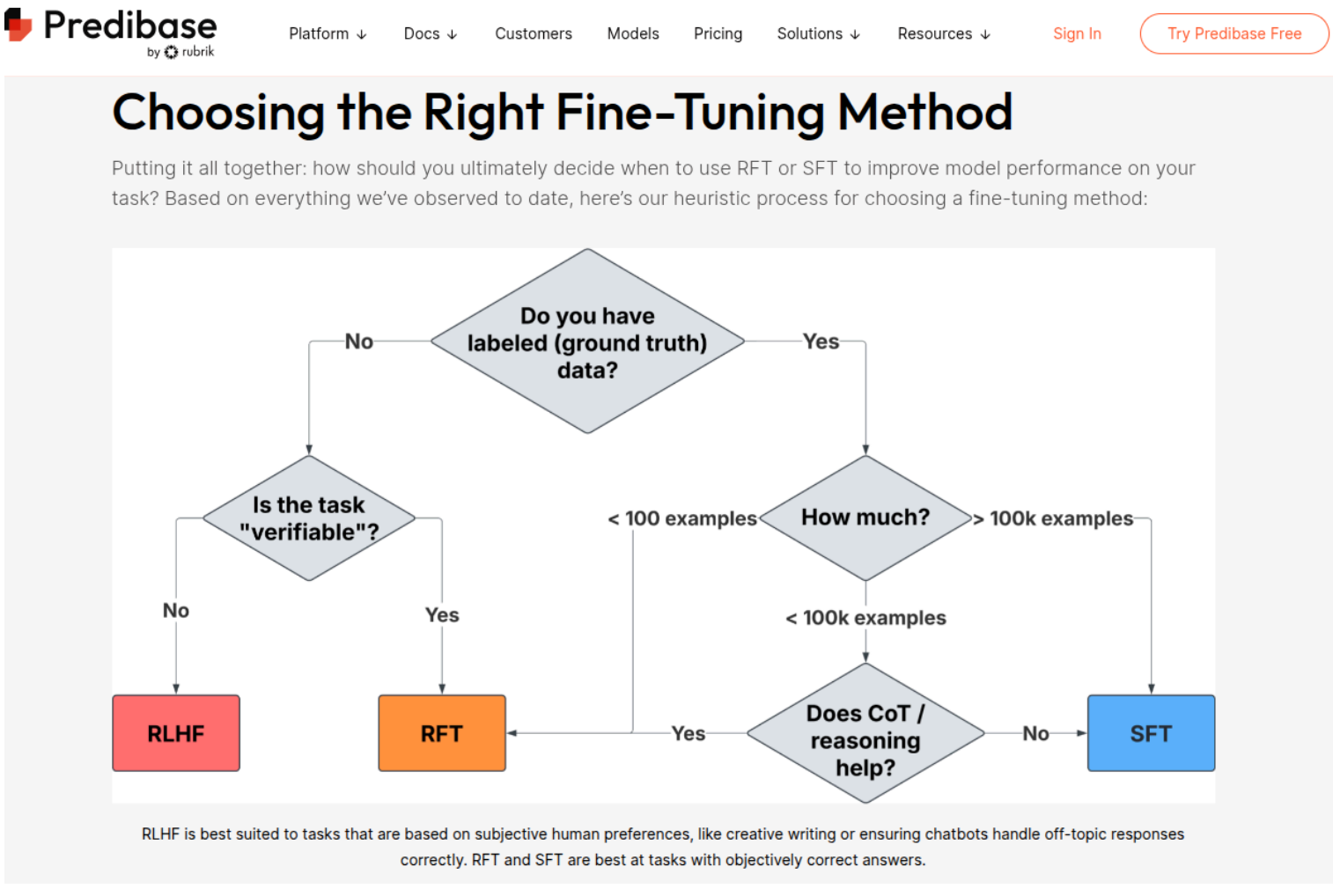
Use of SFT as first phase of RFT

The preliminary SFT phase of RFT serves several purposes

- move the LLM's distribution to the *format* of tasks for RL to learn
    - primes the RL phase with correctly formated examples
- creates examples with different rewards
    - RL learns from the *contrasts* between high and low rewards
- "bootstrap" the RL training dataset
    - uses iterative SFT
        - uses a weaker SFT-tuned model
        - to create synthetic training examples for a stronger SFT-tuned model

Criteria	SFT	RFT/RLHF
Task type	Objective, well-defined, clear correct answer tasks	Subjective, ambiguous, or value-laden tasks
Data availability	Large, high-quality labeled datasets available	Little/no labeled data, but feedback/preference signals are available
Training complexity	Simpler (labeled pairs)	More complex (reward model, RL optimization)
Desired outcome	Accuracy, task performance, factual correctness	Human preference alignment, style, quality, safety
Overfitting risk	Higher, if data is limited	Lower; learns general behavior from rewards
Generalization	Prone to memorization	Promotes adaptability, nuanced behaviors
Cost/resource needs	Lower; less human-in-the-loop need	Higher; human feedback collection and more computation
Ideal use cases	Translation, classification, summarization, retrieval	Chatbots, open-domain QA, content moderation, dialog

Here is one rubric:



Reference: [https://predibase.com/blog/how-reinforcement-learning-beats-supervised-fine-tuning-when-data-is-scarce (https://predibase.com/blog/how-reinforcement-learning-beats-supervised-fine-tuning-when-data-is-scarce)](https://predibase.com/blog/how-reinforcement-learning-beats-supervised-fine-tuning-when-data-is-scarce)

References for RFT vs SFT

- Why Reinforcement Learning Beats SFT with Limited Data - Predibas (https://predibase.com/blog/how-reinforcement-learning-beats-supe tuning-when-data-is-scarce)
- Preference Alignment vs Supervised Fine-Tuning in LLM Training (https://www.rohan-paul.com/p/preference-alignment-vs-supervised
- Supervised Fine-Tuning vs. RLHF: How to Choose the Right Approach (https://www.invisible.co/blog/supervised-fine-tuning-vs-rlhf-how-to right-approach-to-train-your-llm)
- Fine-Tuning vs RLHF: Choosing the Best LLM Training Method (https://cleverx.com/blog/supervised-fine-tuning-vs-rlhf-choosing-th to-train-your-llm)
- What is supervised fine-tuning? - BlueDot Impact (https://bluedot.org is-supervised-fine-tuning)

Process Reward Model (PRM) vs Outcome R Model (ORM)

Outcome Reward Model (ORM) = Trajectory Reward

- single reward at end of trajectory

Process Reward Model (PRM) = step by step reward

References for Process Reward Models vs Outcome Reward Models

- A Comprehensive Survey of Reward Models: Taxonomy and Applica (https://arxiv.org/html/2504.12328v1)
- Reward Modeling | RLHF Book by Nathan Lambert (https://rlhfboo reward-models.html)
- Let's Verify Step by Step (OpenAI, Process Supervision) (https://cdn.openai.com/improving-mathematical-reasoning-with-p supervision/Lets_Verify_Step_by_Step.pdf)
- Getting LLMs To Reason With Process Rewards (https://patmcguinness.substack.com/p/getting-llms-to-reason-wit

References

Title (linked)	Commentary
InstructGPT: Aligning Language Models with Human Intent via RLHF (https://arxiv.org/abs/2203.02155)	Foundational paper laying out the RLHF approach to align LLMs with human intent using human preference data. Essential for understanding RLHF theory and practice.
A Survey on Post-Training of Large Language Models (https://arxiv.org/abs/2503.06072)	Comprehensive survey reviewing SFT, RLHF, and newer alignment methods. Synthesizes research trends and challenges in LLM post-training.
Reinforcement Learning from AI Feedback (RLAIF): A Scalable Alternative to RLHF (https://arxiv.org/abs/2309.00267)	Introduces RLAIF, replacing human feedback with AI-generated feedback for scalable alignment. Critical for understanding automated feedback approaches.
Constitutional AI: Harmlessness from AI Feedback (https://www.anthropic.com/research/constitutional-ai-harmlessness-from-ai-feedback)	Proposes Constitutional AI, using a fixed ethical constitution for AI self-critique and revision to improve alignment without human labels.
LLM Post-Training: A Deep Dive into Reasoning Large Language Models (https://arxiv.org/abs/2502.21321)	Examines post-training methods focused on improving reasoning in LLMs via SFT and RL, analyzing mechanics and challenges.
SFT Memorizes, RL Generalizes: A Comparative Study of Post-Training Methods for LLMs (https://arxiv.org/abs/2501.17161)	Empirically compares SFT and RL in LLMs, showing SFT excels at memorization while RL generalizes better and improves alignment.
How Reinforcement Learning Beats Supervised Fine-Tuning When Data Is Scarce (https://predibase.com/blog/how-reinforcement-learning-beats-supervised-fine-tuning-when-data-is-scarce)	Blog explaining why RL methods can outperform SFT in low-data regimes; offers practical insights for training efficiency.
Beyond Next-Token Prediction: How Post-Training Teaches LLMs to Reason (https://toloka.ai/blog/how-post-training-teaches-llms-to-reason/)	Discusses how combining SFT and RL post-training enables complex reasoning in LLMs, with examples and experimental findings.
Demystifying Reasoning Models (https://cameronrwolfe.substack.com/p/demystifying-reasoning-models)	Blog unpacking the roles of SFT and RL in reasoning capability development; bridges theory and practice with clear explanations.
RLHF vs RLAIF: A Detailed Comparison of AI Training Methods (https://www.sapien.io/blog/rlaif-vs-rlhf-understanding-the-differences)	Detailed comparison of RLHF and RLAIF approaches, illustrating differences in feedback sources and workflows for AI alignment.

In [ ]:
```
print("Done")
```