LLM Bootcamp 2023

LLMOps: Deployment and Learning in Production

Josh Tobin

April 22, 2023

Outline

- Choosing your base model
- Iteration and prompt management
- Testing
- Deployment
- Monitoring
- Continual improvement and fine-tuning

Outline

- **Choosing your base model**
- Iteration and prompt management
- Testing
- Deployment
- Monitoring
- Continual improvement and fine-tuning

Choosing your base model: TL/DR

- The best model for your use case **depends on tradeoffs** between:
  - Out-of-the-box **quality** for your task
  - Inference **speed** / latency
  - **Cost**
  - **Fine-tuneability** / extensibility
  - Data **security** and license **permissibility**
- Most use cases, most of the time: **start with GPT-4**

Do you want proprietary or open-source?

Proprietary models are better…

- Higher quality today
- Most open source models have licensing friction - may not be suited for commercial use
- Serving open source models introduces infrastructure overhead

… Unless you really need OSS

- Much easier to customize
- Respect data security

On OSS licensing

- **Permissive licenses** like Apache 2.0 let you do more-or-less what you want with the model
- **Restricted licenses** like CC BY-SA 3.0 place restrictions on commercial use, but don't prohibit it. Draw your own conclusions on whether this works for you
- **Non-commercial licenses** like Facebook's proprietary ones or Creative Commons CC BY-NC-SA 4.0 *explicitly prohibit commercial use* and are a bad choice for building apps

Choose your fighter: the proprietary contenders

Model	Params	Context	Training	Quality	Speed	Fine-tuneability
gpt-4	?	8K*	Internet data, Code, Instructions, Human feedback	★★★★	🐇◌◌	✗
gpt-3.5-turbo	175B	4K	Internet data, Code, Instructions, Human feedback	★★★☆	🐇🐇◌	✗
claude	?	9K	Internet data, Code, Instructions, Human feedback	★★★☆	🐇🐇◌	✗
command-xlarge	50B		Internet data, Instructions	★★☆☆	🐇🐇◌	✓
claude-instant	?	9K	Internet data, Code, Instructions, Human feedback	★☆☆☆	🐇🐇🐇	✗
ada,babbage, curie	350M - 7B	2K	Internet data	★☆☆☆	🐇🐇🐇	✓
command-medium	6B		Internet data, Instructions	★☆☆☆	🐇🐇🐇	✓

Training data key
- Internet data
- Code
- Instructions
- Human feedback

* gpt-4 will have 32K context window available, but it's not released at the time of writing

Choose your fighter: the proprietary contenders

Model	Params	Context	Training	Quality	Speed	Fine-tuneability
gpt-4	?	8K*		★★★★		✗
gpt-3.5-turbo	175B	4K		★★★☆		✗
claude	?	9K		★★★☆		✗
command-xlarge	50B			★★☆☆		✓
claude-instant	?	9K		★☆☆☆		✗
ada,babbage, curie	350M - 7B	2K		★☆☆☆		✓
command-medium	6B			★☆☆☆		✓

Highest quality

* gpt-4 will have 32K context window available, but it's not released at the time of writing

Choose your fighter: the proprietary contenders

Model	Params	Context	Training	Quality	Speed	Fine-tuneability
gpt-4	?	8K*		★★★★		✗
gpt-3.5-turbo	175B	4K		★★★☆		✗
claude	?	9K		★★★☆		✗
command-xlarge	50B			★★☆☆		✓
claude-instant	?	9K		★☆☆☆		✗
ada,babbage, curie	350M - 7B	2K		★☆☆☆		✓
command-medium	6B			★☆☆☆		✓

For if you want something faster / cheaper

* gpt-4 will have 32K context window available, but it's not released at the time of writing

Choose your fighter: the proprietary contenders

Model	Params	Context	Training	Quality	Speed	Fine-tuneability
gpt-4	?	8K*		★★★★		✗
gpt-3.5-turbo	175B	4K		★★★☆		✗
claude	?	9K		★★★☆		✗
command-xlarge	50B			★★☆☆		✓
claude-instant	?	9K		★☆☆☆		✗
ada,babbage, curie	350M - 7B	2K		★☆☆☆		✓
command-medium	6B			★☆☆☆		✓

Best OpenAI alternative

* gpt-4 will have 32K context window available, but it's not released at the time of writing

Choose your fighter: the proprietary contenders

Model	Params	Context	Training	Quality	Speed	Fine-tuneability
gpt-4	?	8K*		★★★★		✗
gpt-3.5-turbo	175B	4K		★★★☆		✗
claude	?	9K		★★★☆		✗
command-xlarge	50B			★★☆☆		✓
claude-instant	?	9K		★☆☆☆		✗
ada,babbage, curie	350M - 7B	2K		★☆☆☆		✓
command-medium	6B			★☆☆☆		✓

Best for fine-tuning

* gpt-4 will have 32K context window available, but it's not released at the time of writing

Choose your fighter: the proprietary contenders

Model	Params	Context	Training	Quality	Speed	Fine-tuneability
gpt-4	?	8K*		★★★★		✗
gpt-3.5-turbo	175B	4K		★★★☆		✗
claude	?	9K		★★★☆		✗
command-xlarge	50B			★★☆☆		✓
claude-instant	?	9K		★☆☆☆		✗
ada,babbage, curie	350M - 7B	2K		★☆☆☆		✓
command-medium	6B			★☆☆☆		✓

Best of the fast / cheap models

* gpt-4 will have 32K context window available, but it's not released at the time of writing

Choose your fighter: the proprietary contenders

Model	Params	Context	Training	Quality	Speed	Fine-tuneability
gpt-4	?	8K*		★★★★		✗
gpt-3.5-turbo	175B	4K		★★★☆		✗
claude	?	9K		★★★☆		✗
command-xlarge	50B			★★☆☆		✓
claude-instant	?	9K		★☆☆☆		✗
ada,babbage, curie	350M - 7B	2K		★☆☆☆		✓
command-medium	6B			★☆☆☆		✓

For latency or cost sensitive use cases

* gpt-4 will have 32K context window available, but it's not released at the time of writing

Choose your fighter: the "open source" options

	Model	Params	Context	Training	Quality	License	Notes
Google AI	T5, Flan-T5	12B	2K		★★☆☆	Apache 2.0	
EleutherAI	Pythia, Dolly 2.0	12B	2K		★★☆☆	Apache 2.0 Proprietary	
stability.ai	StableLM / StableLM tuned	7B	4K		★★☆☆	CC BY-SA 4.0 CC BY-NC-SA 4.0	
Meta	LLaMA, Alpaca, Vicuna, Koala	60B	2K		★★☆☆	Proprietary	Chinchilla scaling
Meta	OPT	175B	2K		★☆☆☆	Proprietary	Closest to original GPT-3
BigScience	Bloom	130B	2K		☆☆☆☆	OpenRAIL	
	GLM	130B	2K		★☆☆☆	Proprietary	Masked LM objective

License key

Non-commercial

Restricted commercial

Permissive

Blue indicates fine-tunes

Choose your fighter: the “open source” options

	Model	Params	Context	Training	Quality	License	Notes
Google AI	T5, Flan-T5	12B	2K		★☆☆☆	Apache 2.0	
EleutherAI	Pythia, Dolly 2.0	12B	2K		★★☆☆	Apache 2.0 Proprietary	
stability.ai	StableLM / StableLM tuned	7B	4K		★★☆☆	CC BY-SA 4.0 CC BY-NC-SA 4.0	
Meta	LLaMA, Alpaca, Vicuna, Koala	60B	2K		★★☆☆	Proprietary	Chinchilla scaling
Meta	OPT	175B	2K		★☆☆☆	Proprietary	Closest to original GPT-3
BigScience	Bloom	130B	2K		☆☆☆☆	OpenRAIL	
	GLM	130B	2K		★☆☆☆	Proprietary	Masked LM objective

Blue indicates fine-tunes

Best bet for decent results w/ permissive license

Choose your fighter: the "open source" options

	Model	Params	Context	Training	Quality	License	Notes
Google AI	T5, Flan-T5	12B	2K		★★☆☆	Apache 2.0	
EleutherAI	Pythia, Dolly 2.0	12B	2K		★★☆☆	Apache 2.0 Proprietary	
stability.ai	StableLM / StableLM tuned	7B	4K		★★☆☆	CC BY-SA 4.0 CC BY-NC-SA 4.0	
Meta	LLaMA, Alpaca, Vicuna, Koala	60B	2K		★★☆☆	Proprietary	Chinchilla scaling
Meta	OPT	175B	2K		★☆☆☆	Proprietary	Closest to original GPT-3
BigScience	Bloom	130B	2K		☆☆☆☆	OpenRAIL	
	GLM	130B	2K		★☆☆☆	Proprietary	Masked LM objective

Recent option; good early reputation for quality

Blue indicates fine-tunes

Choose your fighter: the “open source” options

	Model	Params	Context	Training	Quality	License	Notes
Google AI	T5, Flan-T5	12B	2K		★★☆☆	Apache 2.0	
EleutherAI	Pythia, Dolly 2.0	12B	2K		★★☆☆	Apache 2.0 / Proprietary	
stability.ai	StableLM / StableLM tuned	7B	4K		★★☆☆	CC BY-SA 4.0 / CC BY-NC-SA 4.0	
Meta	LLaMA, Alpaca, Vicuna, Koala	60B	2K		★★☆☆	Proprietary	Chinchilla scaling
Meta	OPT	175B	2K		★☆☆☆	Proprietary	Closest to original GPT-3
BigScience	Bloom	130B	2K		☆☆☆☆	OpenRAIL	
	GLM	130B	2K		★☆☆☆	Proprietary	Masked LM objective

Recent option; likely good alternative to Pythia / Dolly

Blue indicates fine-tunes

Choose your fighter: the “open source” options

	Model	Params	Context	Training	Quality	License	Notes
Google AI	T5, Flan-T5	12B	2K		★★☆☆	Apache 2.0	
EleutherAI	Pythia, Dolly 2.0	12B	2K		★★☆☆	Apache 2.0 Proprietary	
stability.ai	StableLM / StableLM tuned	7B	4K		★★☆☆	CC BY-SA 4.0 CC BY-NC-SA 4.0	
Meta	LLaMA, Alpaca, Vicuna, Koala	60B	2K		★★☆☆	Proprietary	Chinchilla scaling
Meta	OPT	175B	2K		★☆☆☆	Proprietary	Closest to original GPT-3
BigScience	Bloom	130B	2K		☆☆☆☆	OpenRAIL	
	GLM	130B	2K		★☆☆☆	Proprietary	Masked LM objective

Ecosystem / performance pick if you don’t need commercial use

Blue indicates fine-tunes

Choose your fighter: the “open source” options

	Model	Params	Context	Training	Quality	License	Notes
Google AI	T5, Flan-T5	12B	2K		★☆☆☆	Apache 2.0	
EleutherAI	Pythia, Dolly 2.0	12B	2K		★★☆☆	Apache 2.0 Proprietary	
stability.ai	StableLM / StableLM tuned	7B	4K		★★☆☆	CC BY-SA 4.0 CC BY-NC-SA 4.0	
Meta	LLaMA, Alpaca, Vicuna, Koala	60B	2K		★★☆☆	Proprietary	Chinchilla scaling
Meta	OPT	175B	2K		★☆☆☆	Proprietary	Closest to original GPT-3
BigScience	Bloom	130B	2K		☆☆☆☆	OpenRAIL	
	GLM	130B	2K		★☆☆☆	Proprietary	Masked LM objective

For if you want to do research on something like 2019 GPT-3

Blue indicates fine-tunes

Choose your fighter: the “open source” options

	Model	Params	Context	Training	Quality	License	Notes
Google AI	T5, Flan-T5	12B	2K		★★☆☆	Apache 2.0	
EleutherAI	Pythia, Dolly 2.0	12B	2K		★★☆☆	Apache 2.0 Proprietary	
stability.ai	StableLM / StableLM tuned	7B	4K		★★☆☆	CC BY-SA 4.0 CC BY-NC-SA 4.0	
Meta	LLaMA, Alpaca, Vicuna, Koala	60B	2K		★★☆☆	Proprietary	Chinchilla scaling
Meta	OPT	175B	2K		★☆☆☆	Proprietary	Closest to original GPT-3
BigScience	Bloom	130B	2K		☆☆☆☆	OpenRAIL	
	GLM	130B	2K		★☆☆☆	Proprietary	Masked LM objective

Not worth considering

Blue indicates fine-tunes

How to assess the performance of LLMs?

- The **only** way to know which LLM will work best is to evaluate it **on your task**
  - More on this later!
- Benchmarks *can* be helpful, but are also misleading

Choosing your base model: recommendations

- Most projects should start with GPT-4
  - This will give you a proof of concept about the feasibility of your task
  - Metaphor: “prototype in Python”
- If cost or latency is a factor, consider “downsizing”
  - GPT-3.5 and Claude are good choices and comparable in performance
  - If you want to go even faster / cheaper, any provider will do, but Anthropic’s option is the most “modern”
- If you need to fine-tune, consider Cohere
- Today, only use OSS if you really need it
  - OSS is progressing fast and should be a viable option soon

Outline

- Choosing your base model
- **Iteration and prompt management**
- Testing
- Deployment
- Monitoring
- Continual improvement and fine-tuning

As you work on your prompts and chains, how to 'save your work'?

Why does this matter?

“Traditional” deep learning in 2015

- Every time I train a model, I write the hyperparameters in a spreadsheet
- Save the trained model in a file on my laptop
- No way to reproduce experiments, share work with the team, etc

Prompt engineering today

- Every time I change my prompt, I play around with it in a playground
- Old prompts are lost to time
- No way to reproduce experiments, share work with the team, etc

“Traditional” deep learning today

- Every time I run model.train(), I automatically get a log of the experiment run that’s comparable, shareable, and fully reproducible

?

Does prompt engineering need better experiment tools?

Why was experiment management so impactful in deep learning?

- You constantly need to go back and check old experiments, because:
  - Experiments take a long time to run, so it's important to be able to "go back" and refresh state
  - You often run many experiments in parallel
  - You run a ton of them, so you often end up repeating yourself if not careful

Prompt engineering today does not have the same dynamic

- Experiments are quick — more like writing code than training a model
- Experimentation is usually sequential
- Most of the time, experimentation is limited

Robust automatic evaluation could change this!

Three levels of prompt/chain tracking

- Level 1: do nothing (e.g., just make prompts in the OpenAI playground)
  - Good enough for v0, not what you want for building apps
- Level 2: track prompts in git
  - What you should do most of the time
- Level 3: track prompts in a specialized tool
  - For running parallel evals, decoupling prompt changes from deploys, or involving non-technical stakeholders

What to look for in a specialized prompt tracking tool?

- Decoupled from git
- Logged prompts are executable both in code and UI
- Connected to execution visualizations
- Deploy directly from the tool

Experiment management tools are moving into prompts

W&B

comet

mlflow

Watch this space!

Prompt management: recommendations

- Manage your prompts and chains in git
- If you
  - (i) collaborate with non-technical stakeholders, or
  - (ii) automate your eval
- Then it's worth trying one of the experiment mgt tools (or keeping an eye out for new ones 😉)

Outline

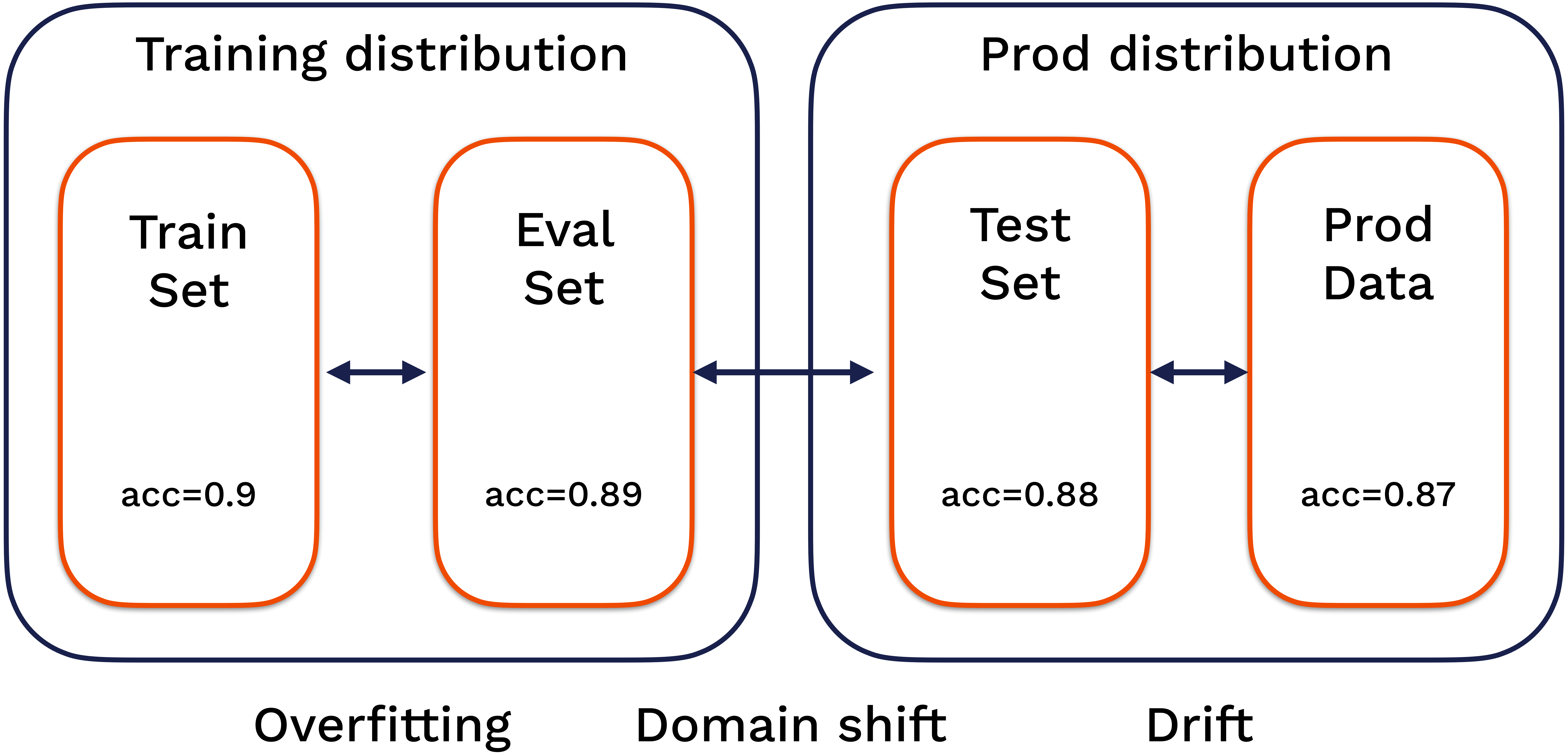
- Choosing your base model
- Iteration and prompt management
- **Testing**
- Deployment
- Monitoring
- Continual improvement and fine-tuning

How do you measure whether your new model / prompt is better than the old one?

Why does this matter?

- LLMs make **tons of mistakes**
- Just because your new prompt looks better on a few examples does not mean that it's **better in general**
  - **Super common to improve in one way and get worse in another!**
- If people rely on your model, they're **trusting you to maintain performance on their task**

Testing ML models: the old-school way



Why doesn't this work for LLMs?

Training distribution

Prod distribution

You don't have access to the training distribution

Why doesn't this work for LLMs?

Training distribution

Prod distribution

Production distribution is *always* different than training

Why doesn't this work for LLMs?

Traditional ML

pred=["cat", "dog", "dog", "cat", "dog", "cat", "dog", "dog", "cat", "dog"]
label=["dog", "dog", "dog", "cat", "dog", "cat", "dog", "dog", "cat", "dog"]

acc=0.9

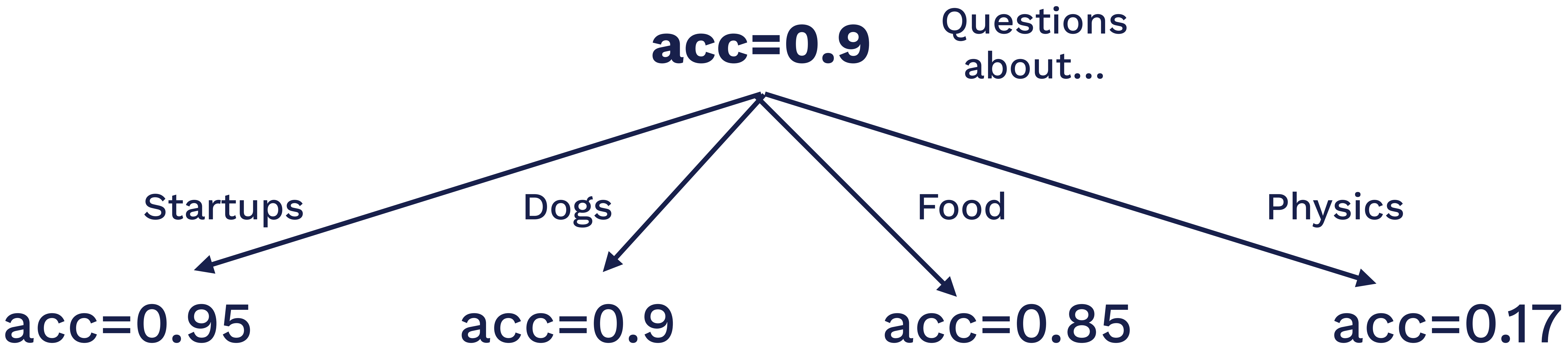
Generative

pred=["this is an image of a tabby cat"]
label=["photograph of a cat"]

What metric?

It's hard to define quantitative metrics

Why doesn't this work for LLMs?



It's hard to summarize a diverse set of inputs and tasks with a single number

Why doesn’t this work for LLMs?

- Trained on the internet ➡️ there is always drift, it doesn’t matter as much
- Qualitative ➡️ Hard-to-measure success
- Diversity of behaviors ➡️ aggregate metrics don’t work

How to think about testing LLMs

What data?

What metric(s)?

Building an evaluation dataset for *your* task

1. Start incrementally
2. Use your LLM to help
3. Add more data as you roll out
4. Toward “test coverage” for AI?

1. Start incrementally

Write a short story about {subject}

subject=dogs

- Start by evaluating ad-hoc

subject=linkedin

subject=hats

1. Start incrementally

- Start by evaluating ad-hoc
- As you find "interesting" evaluation examples, organize them into a small dataset
    - To evaluate, run your model on every example on the dataset

Write a short story about {subject}

<u>dataset</u>

[{"subject": "dogs"},
{"subject": "linkedin"},
{"subject": "hats"}]

1. Start incrementally

- Start by evaluating ad-hoc
- As you find "interesting" evaluation examples, organize them into a small dataset

Write a short story about {subject}

dataset

[{"subject": "dogs"},
{"subject": "linkedin"},
{"subject": "hats"}]

What makes an "interesting" example?

- Hard
- Different

2. Use your LLM to help

Auto-evaluator 🧠📝

PineappleExpress808 / auto-evaluator Public

LLMs can help you generate test cases!

```
You are a smart assistant designed to help high school teachers come up with reading
comprehension questions.
Given a piece of text, you must come up with a question and answer pair that can be used to test
a student's reading comprehension abilities.
When coming up with this question/answer pair, you must respond in the following format:

```


{{
    "question": “$YOUR_QUESTION_HERE",
    "answer": "$THE_ANSWER_HERE"
}}

```


Everything between the ``` must be valid json.
```

3. Add more data as you roll out

Hard data

- What do your users dislike?
- What do your annotators dislike?
- What does another model dislike?

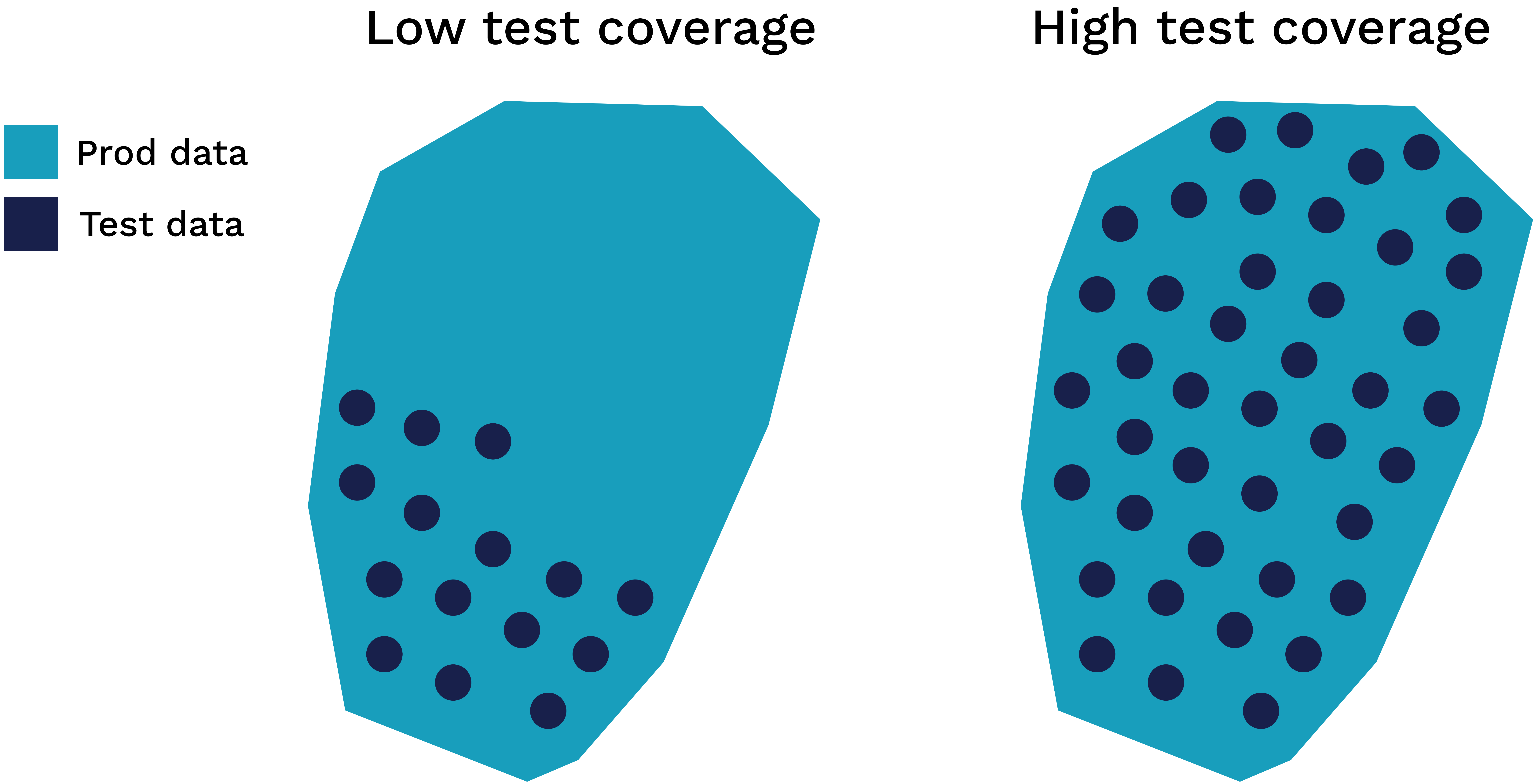
Different data

- Outliers relative to your current eval set
- Underrepresented topics, intents, documents, etc

4. Toward “test coverage” for AI?

- This process feels arbitrary
- Is there any way we can quantify the “quality” of our test set?

4. Toward "test coverage" for AI?



4. Toward “test coverage” for AI?

Test coverage and distribution shift are similar

- Distribution shift measures ***how far the test distribution is away from a reference distribution*** and is used to see if data is changing
- Test coverage measures ***how well your eval set covers your production data*** and is used to find more helpful eval data

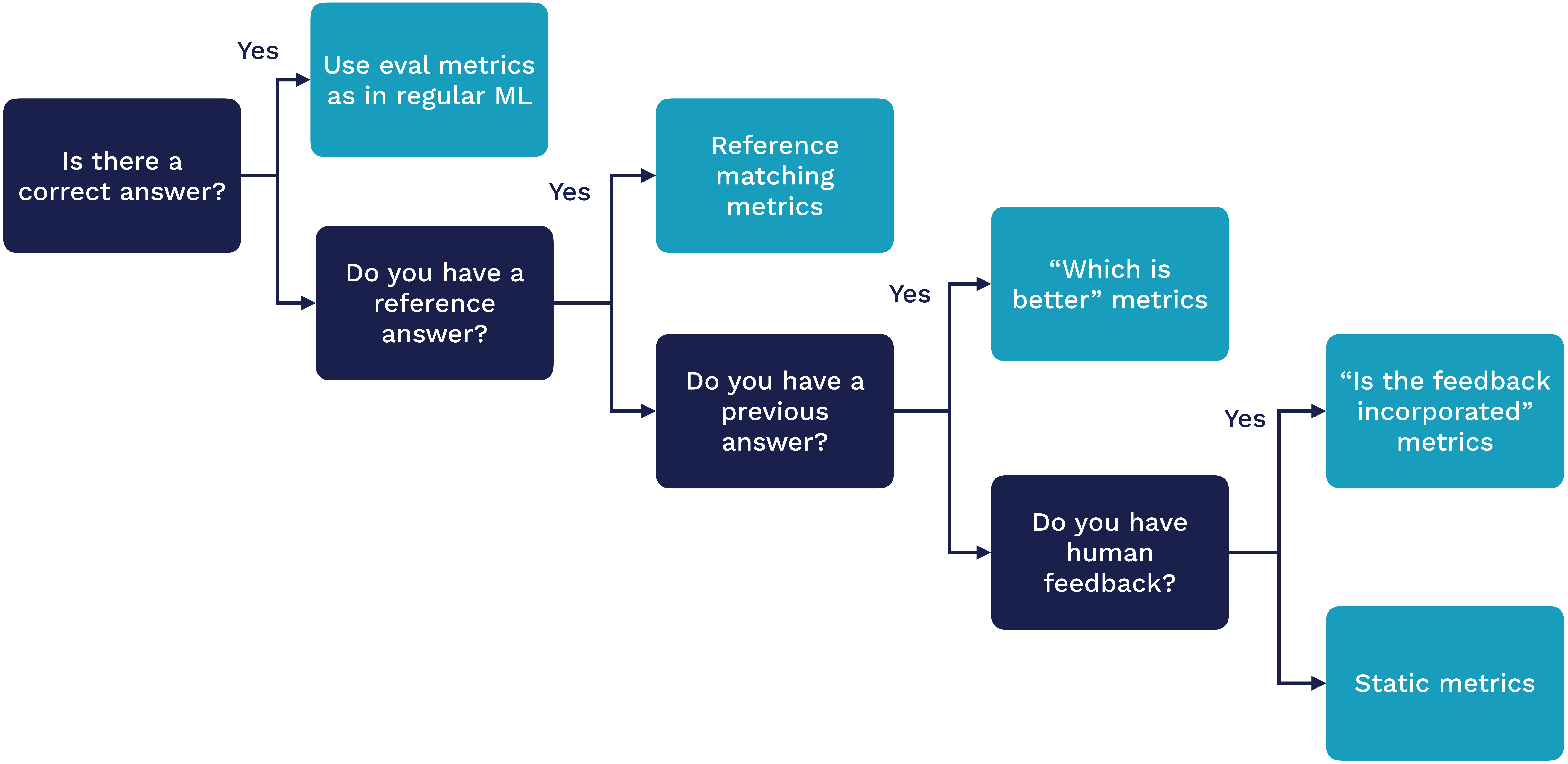
4. Toward “test coverage” for AI?

Is this enough?

- What if “hard” data is underrepresented?
- What if online and offline metrics don’t match?

➡️ also would need a notion of *test reliability* that measures diff between offline and online performance

Evaluation metrics for LLMs



Evaluation metrics for LLMs

- Regular eval metrics
  - Accuracy, etc
- Reference matching metrics
  - Semantic similarity
  - Ask another LLM, “are these two answers factually consistent”, etc
- “Which is better” metrics
  - Ask an LLM which of the two answers is better, according to any criteria you want
- “Is the feedback incorporated” metrics
  - Ask an LLM whether the new answer incorporates the feedback on the old answer
- Static metrics
  - Verify the output has the right structure (e.g., is JSON formatted)
  - Ask a model to grade the answer (e.g., on a scale 1-5)

Key idea: using LLMs to evaluate other LLMs

Can you evaluate LLMs automatically?

- Automatic eval can unlock parallel experimentation
- You still probably do some manual checks
  - What kind of feedback? Thumbs up / down, written feedback, corrected answer?

Outline

- Choosing your base model
- Iteration and prompt management
- Testing
- **Deployment**
- Monitoring
- Continual improvement and fine-tuning

Overview

- Just call the API from your frontend
- Where it becomes more complicated
  - If you have significant logic beyond the API call (complicated prompt construction, complicated chains).
    - Might want to isolate as a service
- Deploying open-source LLMs is a whole other thing

Deploying OSS LLMs

- Beyond our scope
- Some references below

https://fullstackdeeplearning.com/course/2022/lecture-5-deployment/

https://blog.replit.com/llm-training

Improving the output of LLMs in production

- Self-critique
  - Ask an LLM "is this the right answer"
- Sample many times, choose the best option
- Sample many times, ensemble

Guardrails.ai

Note: Guardrails is an alpha release, so expect sharp edges and bugs.

🛤️ What is Guardrails?

Guardrails is a Python package that lets a user add structure, type and quality guarantees to the outputs of large language models (LLMs). Guardrails:

- does pydantic-style validation of LLM outputs. This includes semantic validation such as checking for bias in generated text, checking for bugs in generated code, etc.
- takes corrective actions (e.g. reasking LLM) when validation fails,
- enforces structure and type guarantees (e.g. JSON).

Outline

- Choosing your base model
- Iteration and prompt management
- Testing
- Deployment
- **Monitoring**
- Continual improvement and fine-tuning

Most important signals

- Outcomes and end-user feedback
- Model performance metrics (if applicable)
- Proxy metrics
- Measuring what actually goes wrong

https://gantry.io/blog/youre-probably-monitoring-your-models-wrong/

Gathering feedback from users

- Good feedback = low-friction, high signal
  - Best = part of the user's workflow
- "Accept changes" pattern
- "Thumbs up / down" pattern
- The role of longer-form feedback

What actually goes wrong with LLMs?

- Most common: often UI stuff
  - Latency especially
- Incorrect answers / “hallucinations”
- Long-winded answers
- Too many “dodged” questions
- Prompt injection attacks
- Toxicity, profanity

Outline

- Choosing your base model
- Iteration and prompt management
- Testing
- Deployment
- Monitoring
- **Continual improvement and fine-tuning**

Use user feedback to...

Make the prompt better

Fine-tune the model

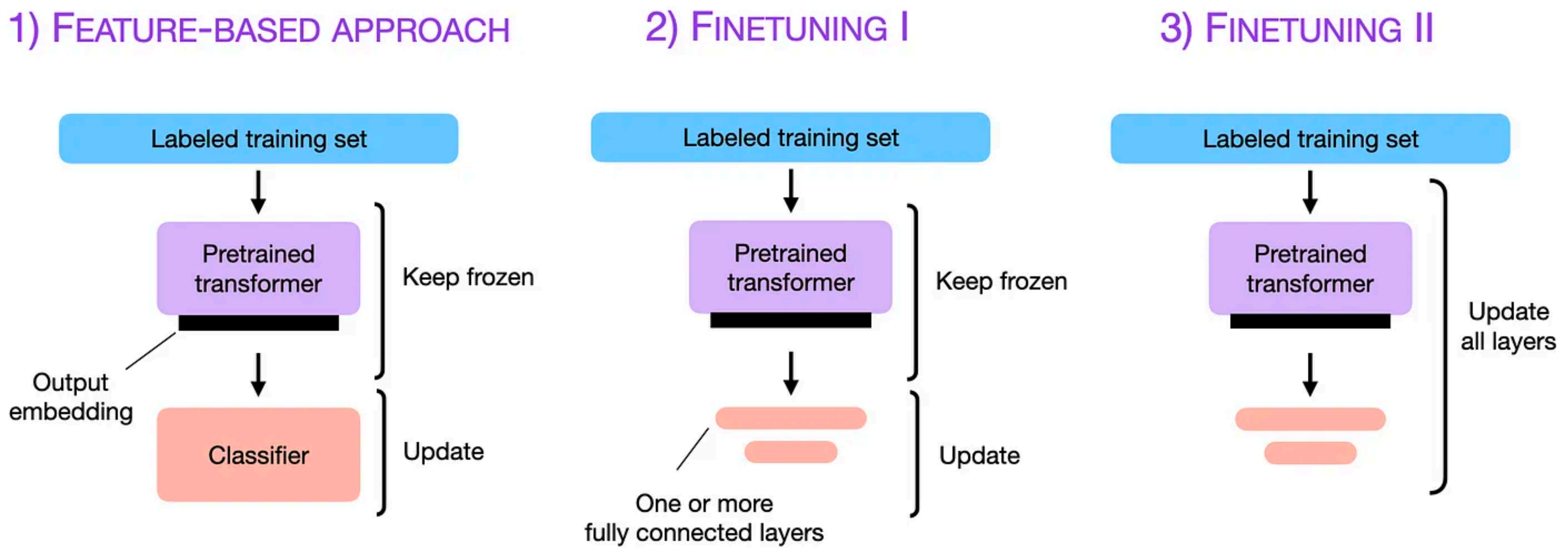
Using user feedback to make the prompt better

- Find **themes** in user feedback that are not addressed by the model
  - Often, found by humans
- Adjust the prompt to account for the theme by
  - Doing prompt engineering
  - Changing the context
- *Open question: can this be automated?*

Fine-tuning LLMs

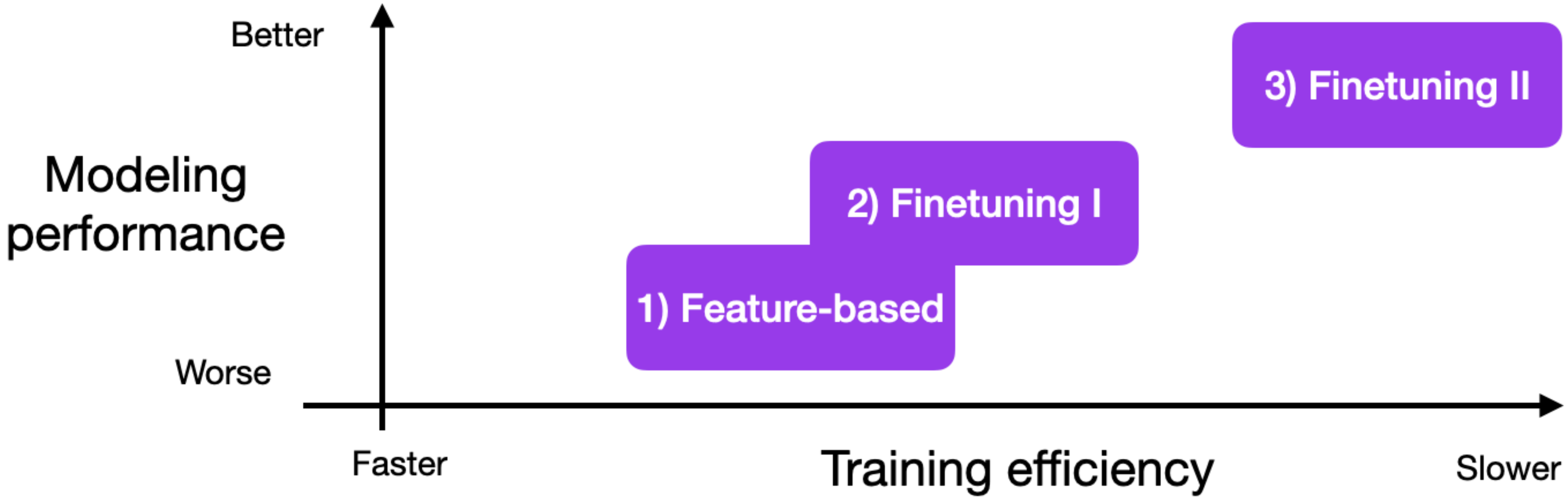
- Supervised fine-tuning
  - If you want to adapt the model to your specific task, and in-context learning isn't working well
  - Or, if you have a lot of data
  - Or, if you want to save cost by building something smaller / cheaper
- Fine-tuning from human feedback
  - Not many companies do this on their own today — technically complex / expensive

Supervised fine-tuning of LLMs

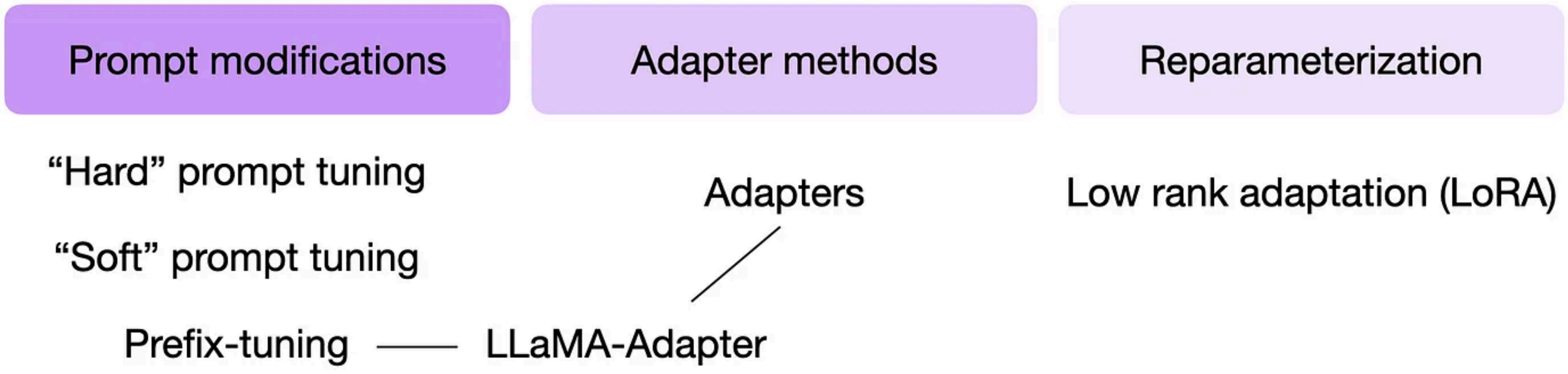


https://magazine.sebastianraschka.com/p/finetuning-large-language-models

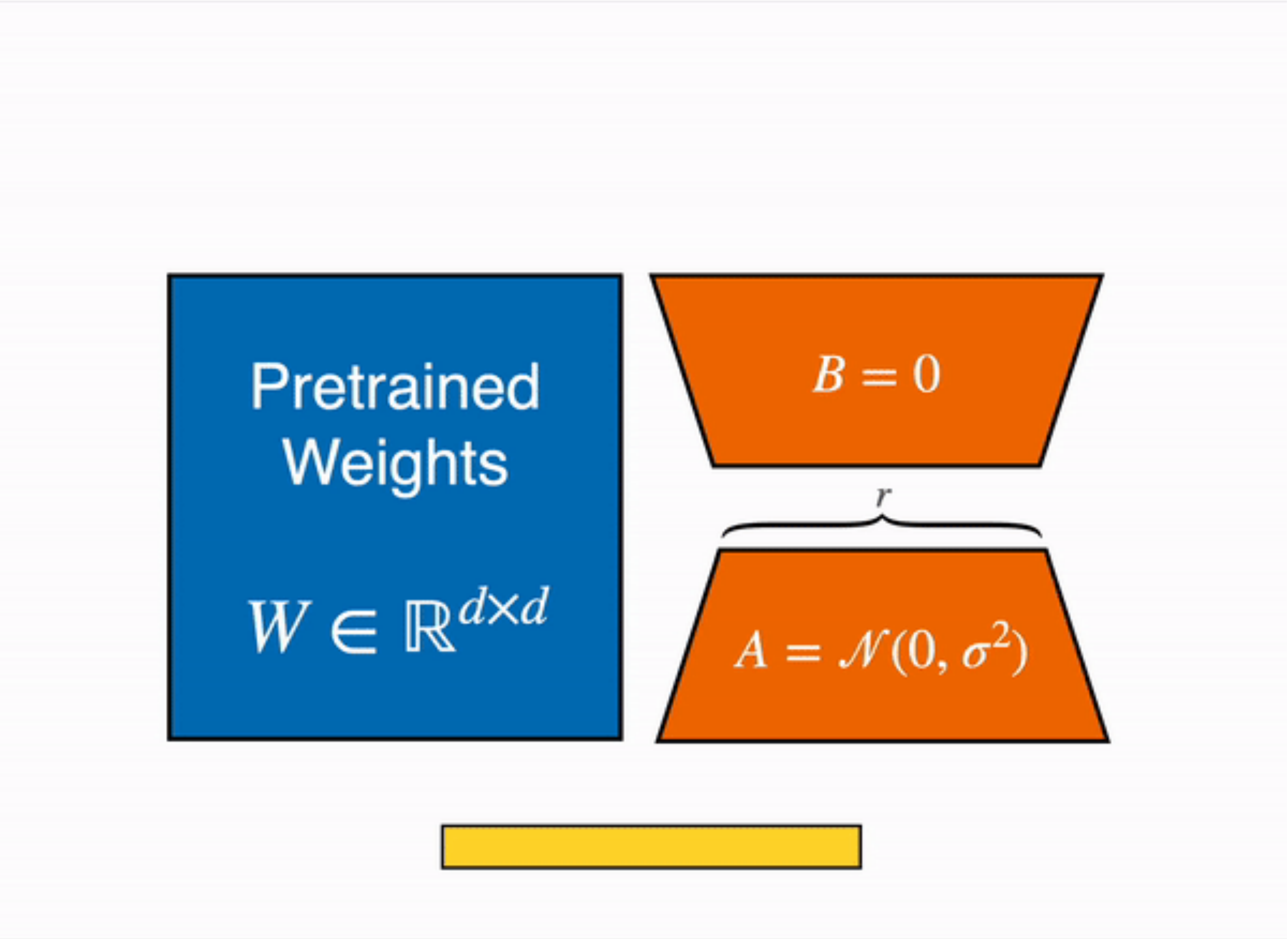
Supervised fine-tuning of LLMs



Parameter-efficient fine tuning



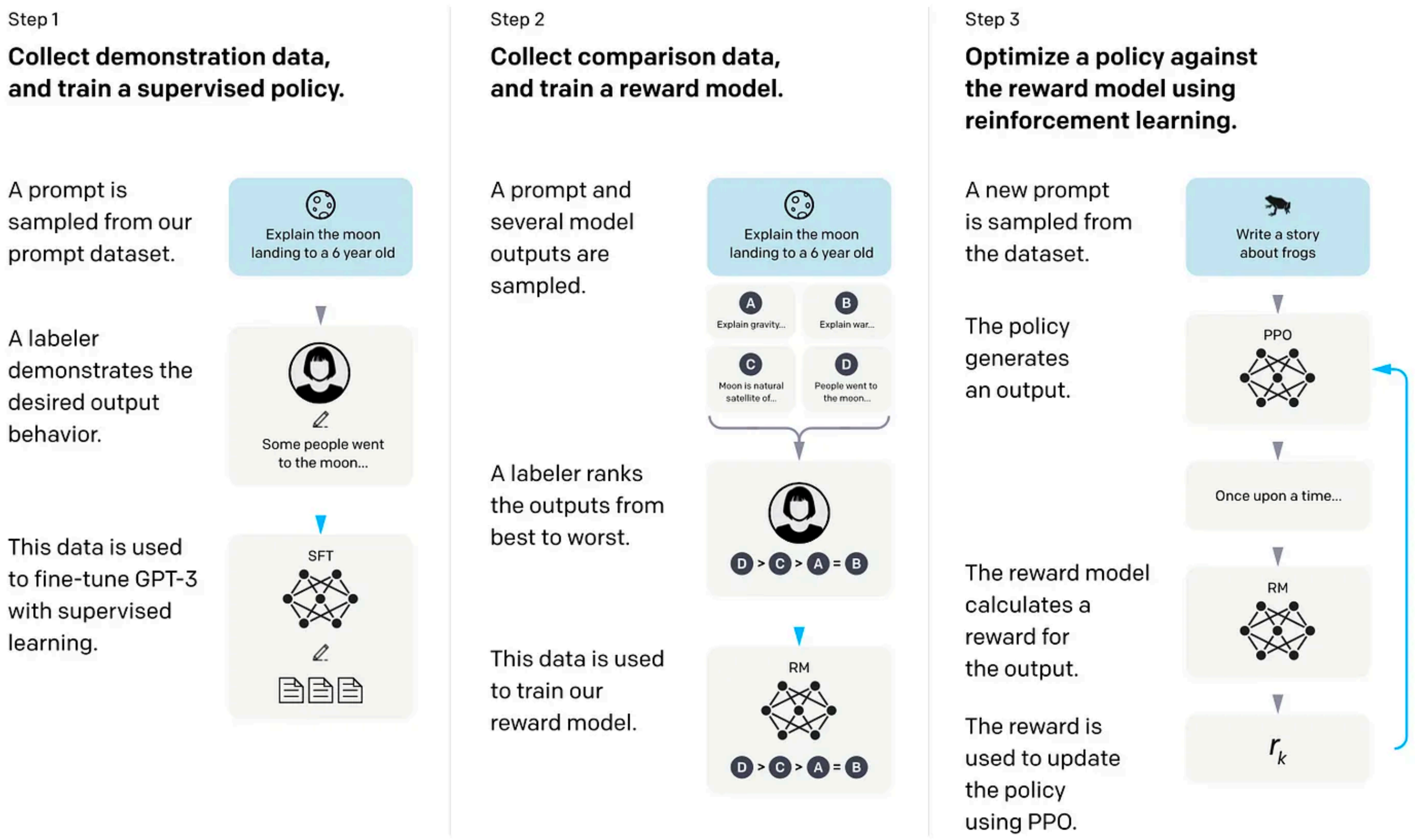
Parameter-efficient fine tuning: LoRA



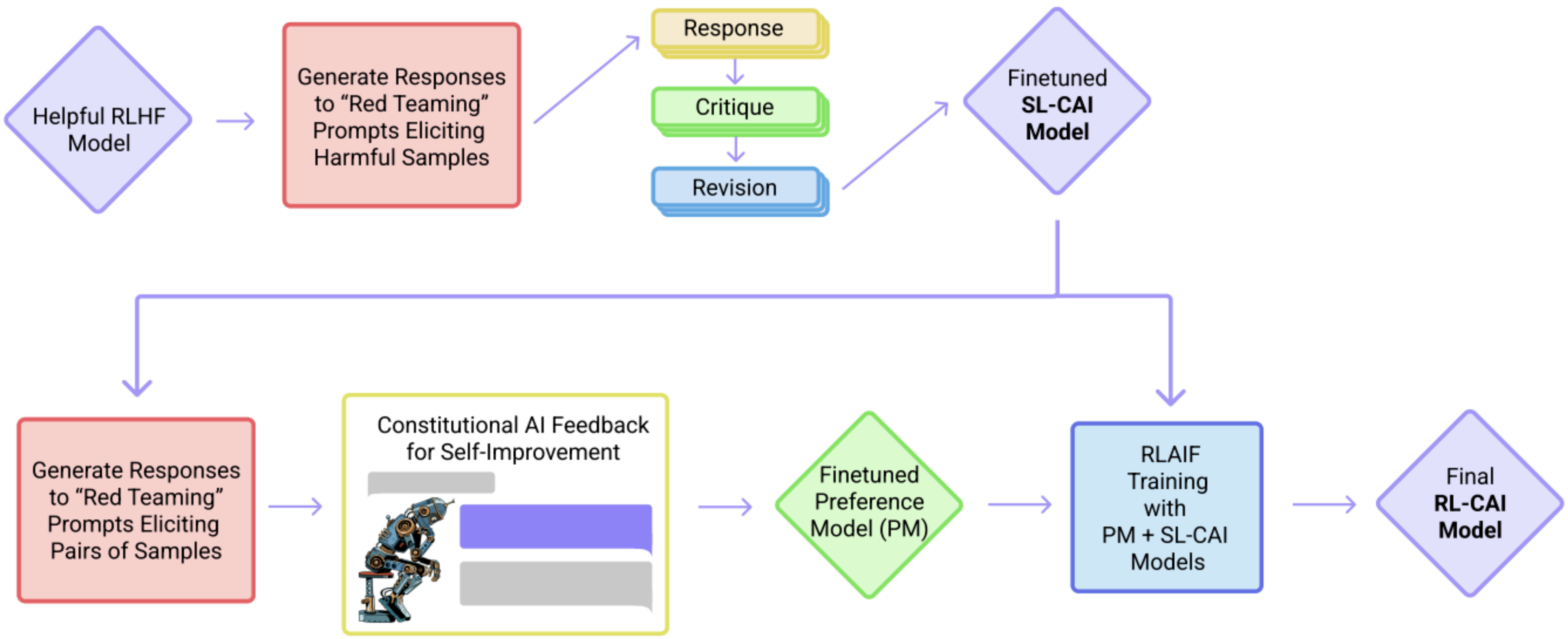
https://huggingface.co/blog/stackllama

https://arxiv.org/abs/2106.09685

Reinforcement learning from human feedback



Reinforcement learning from AI feedback



https://arxiv.org/abs/2212.08073

Fine-tuning LLMs: recommendations

- You probably don’t need to do this if you use GPT-4
- Reasons to fine-tune:
  - You need to use smaller models
  - You have a lot of data and retrieval isn’t working well
- Low-rank updates / parameter efficient tuning might make this more accessible
- RLHF / RLAIF is still difficult today

Conclusion: test-driven development for LLMs