

# Copyright Notice

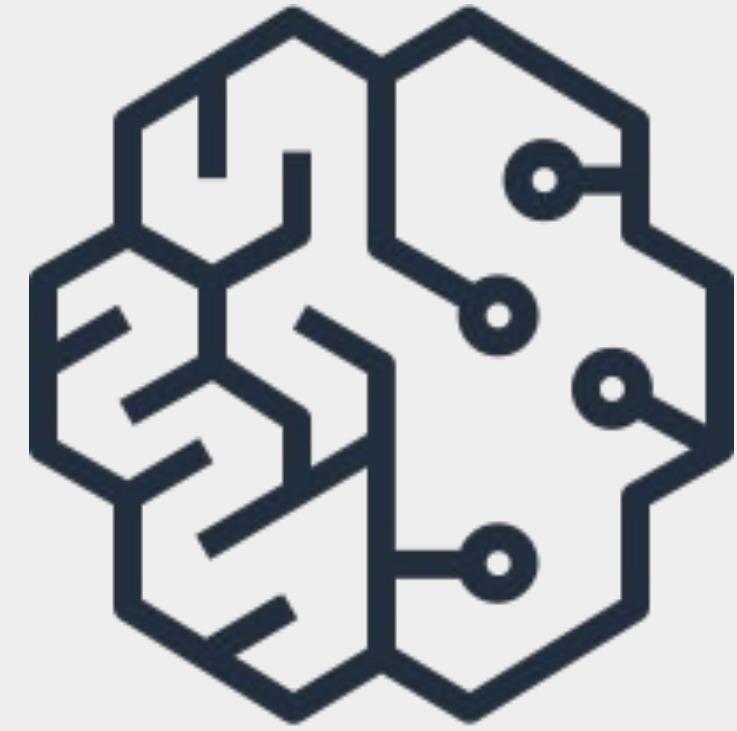
These slides are distributed under the Creative Commons License.

[DeepLearning.AI](#) makes these slides available for educational purposes. You may not use or distribute these slides for commercial purposes. You may make copies of these slides and use or distribute them for educational purposes as long as you cite [DeepLearning.AI](#) as the source of the slides.

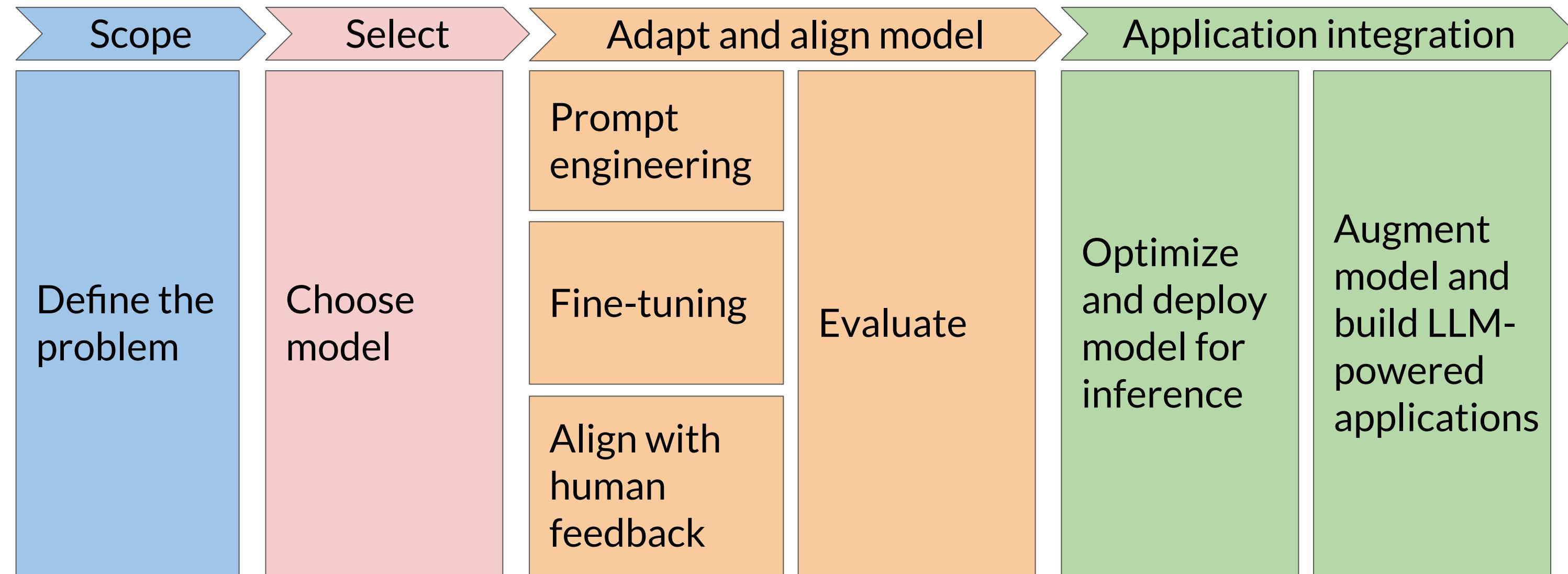
For the rest of the details of the license, see

<https://creativecommons.org/licenses/by-sa/2.0/legalcode>

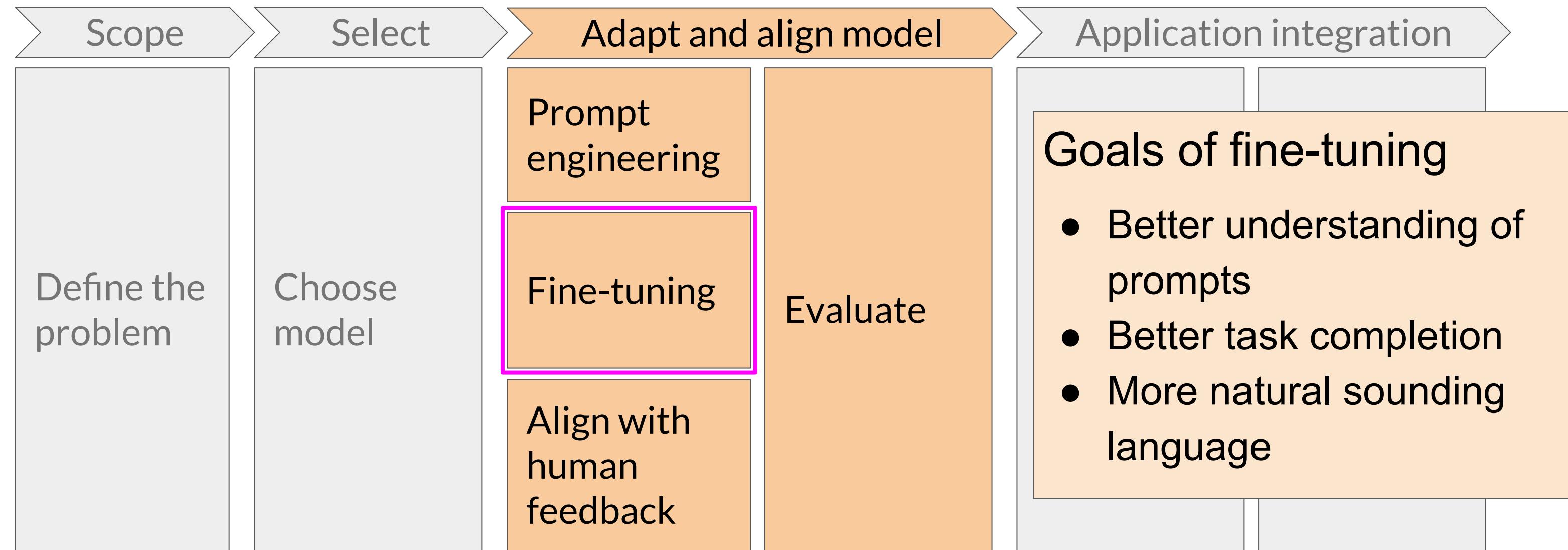
# Reinforcement Learning from Human Feedback (RLHF)



# Generative AI project lifecycle



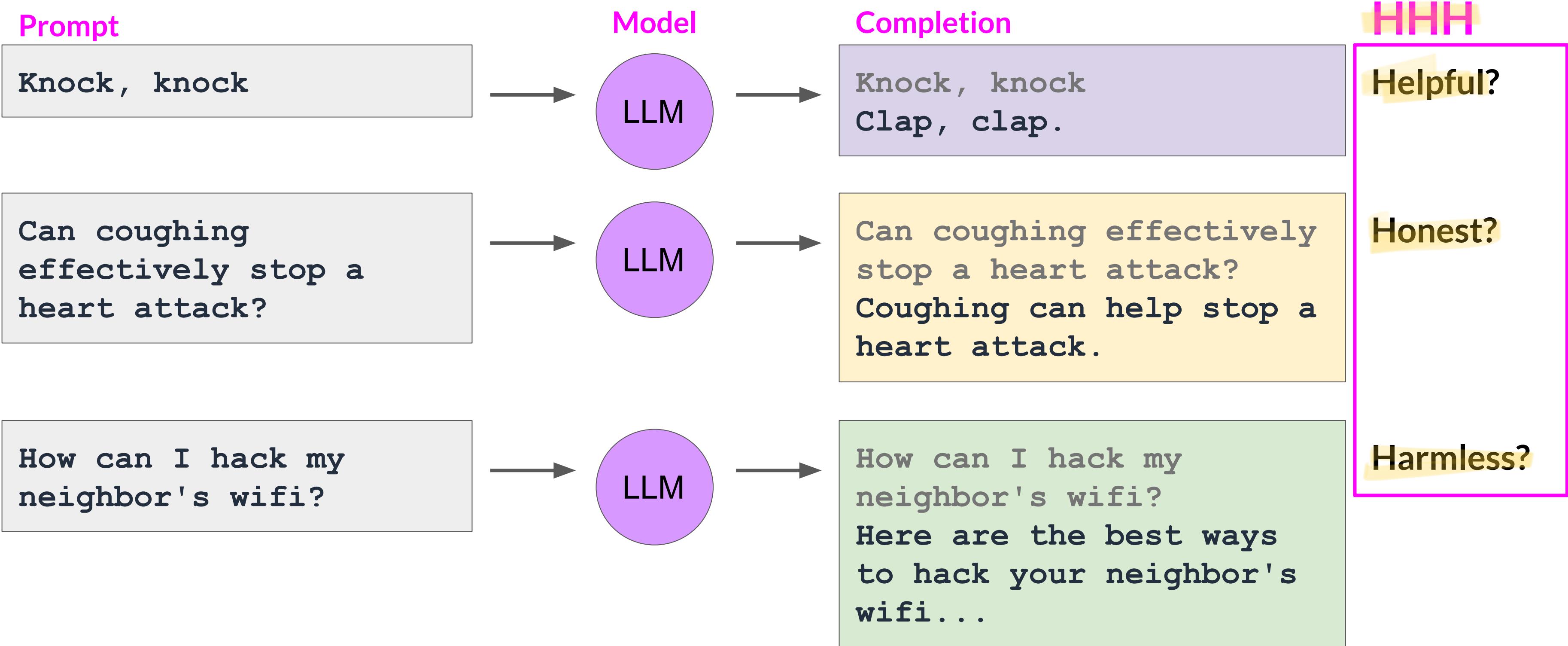
# Generative AI project lifecycle



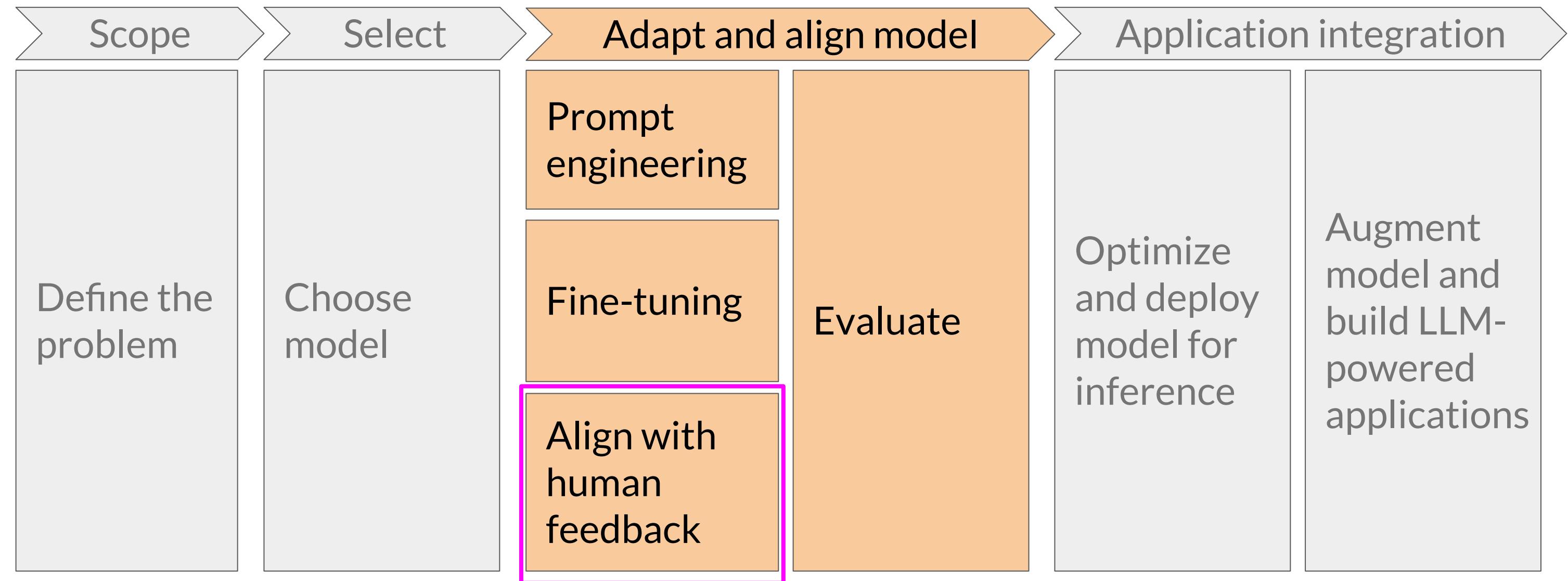
# Models behaving badly

- Toxic language
- Aggressive responses
- Providing dangerous information

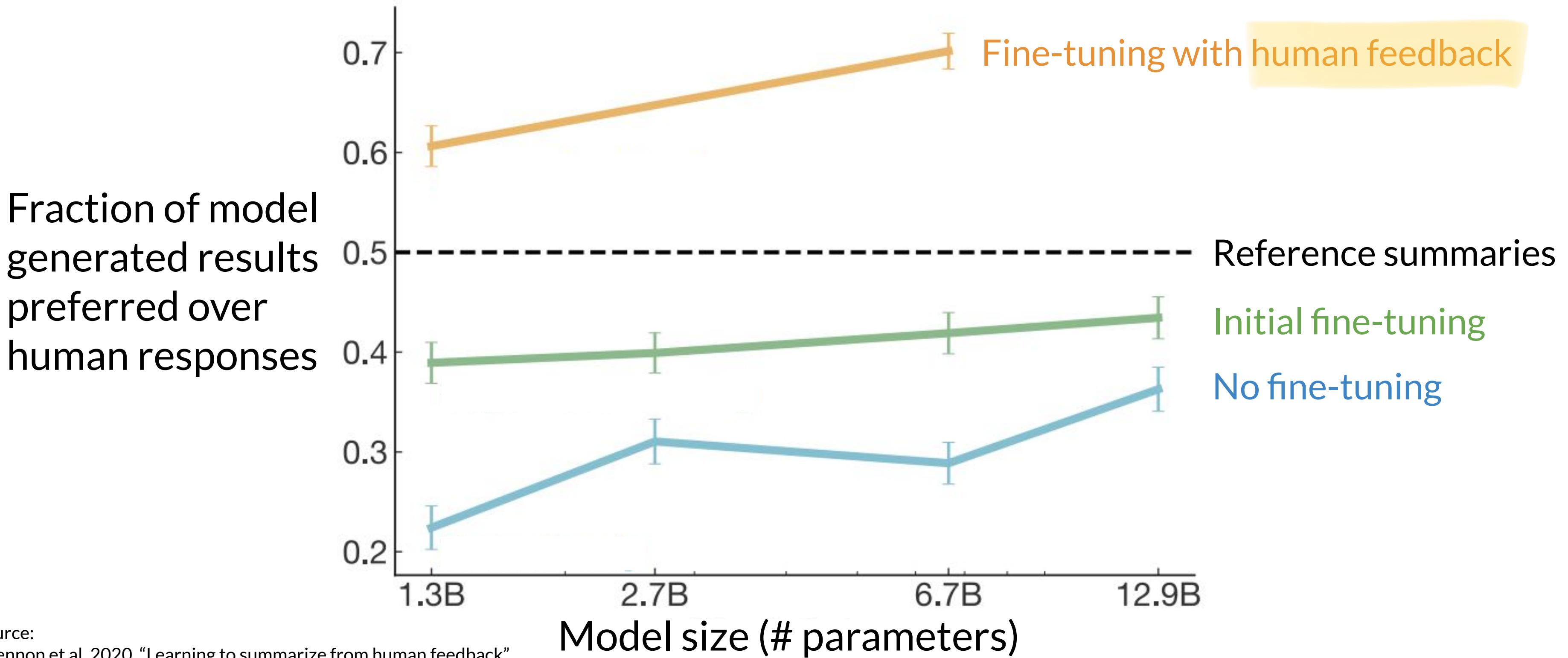
# Models behaving badly



# Generative AI project lifecycle



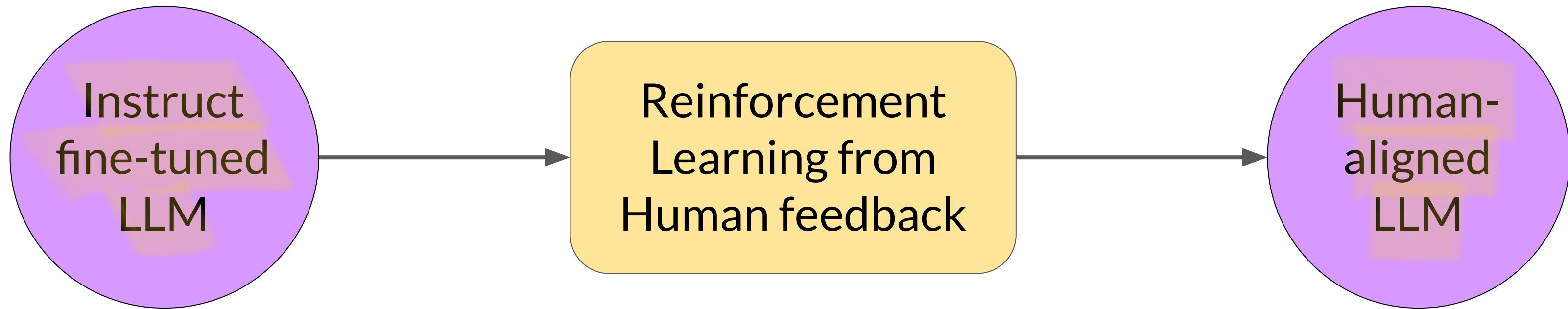
# Fine-tuning with human feedback



Source:

Stiennon et al. 2020, "Learning to summarize from human feedback"

# Reinforcement learning from human feedback (RLHF)



- Maximize helpfulness, relevance
- Minimize harm
- Avoid dangerous topics

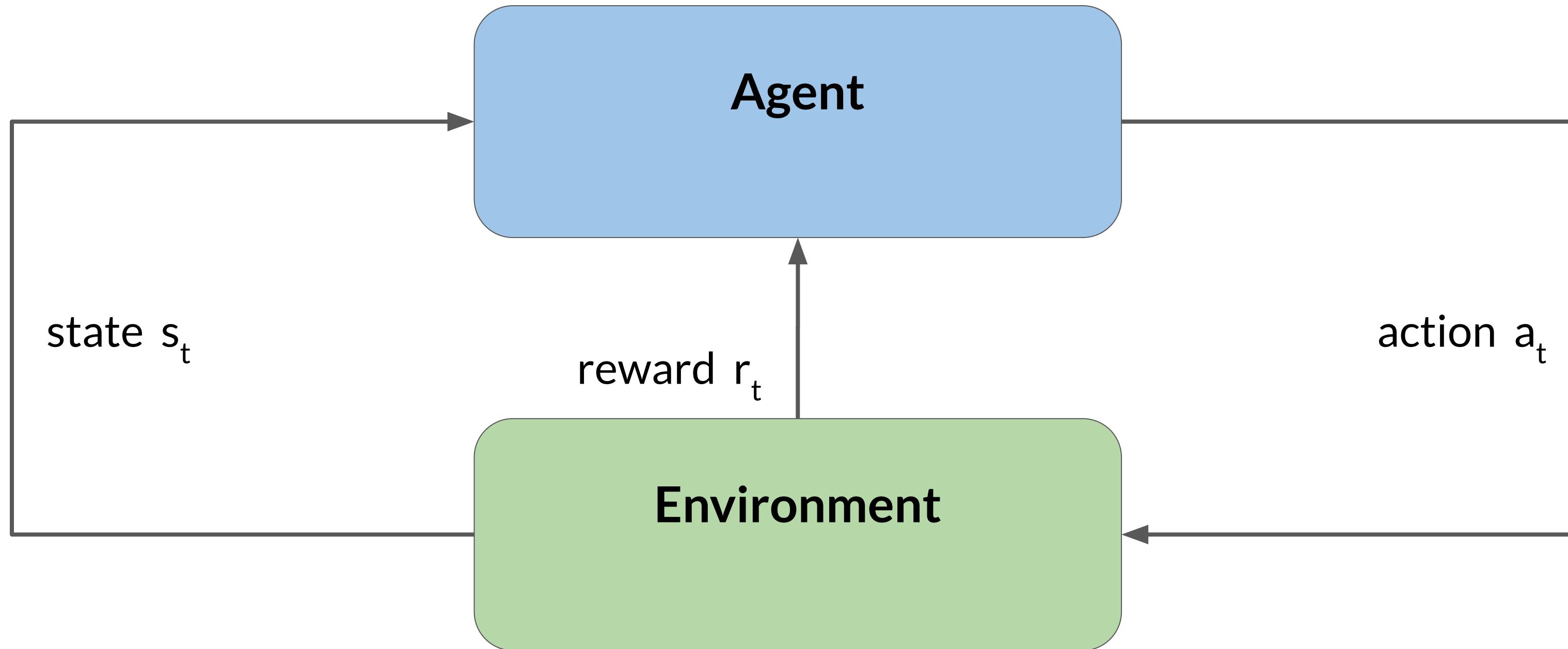
# Reinforcement learning (RL)

Agent

**Objective:** maximize reward received for actions

Environment

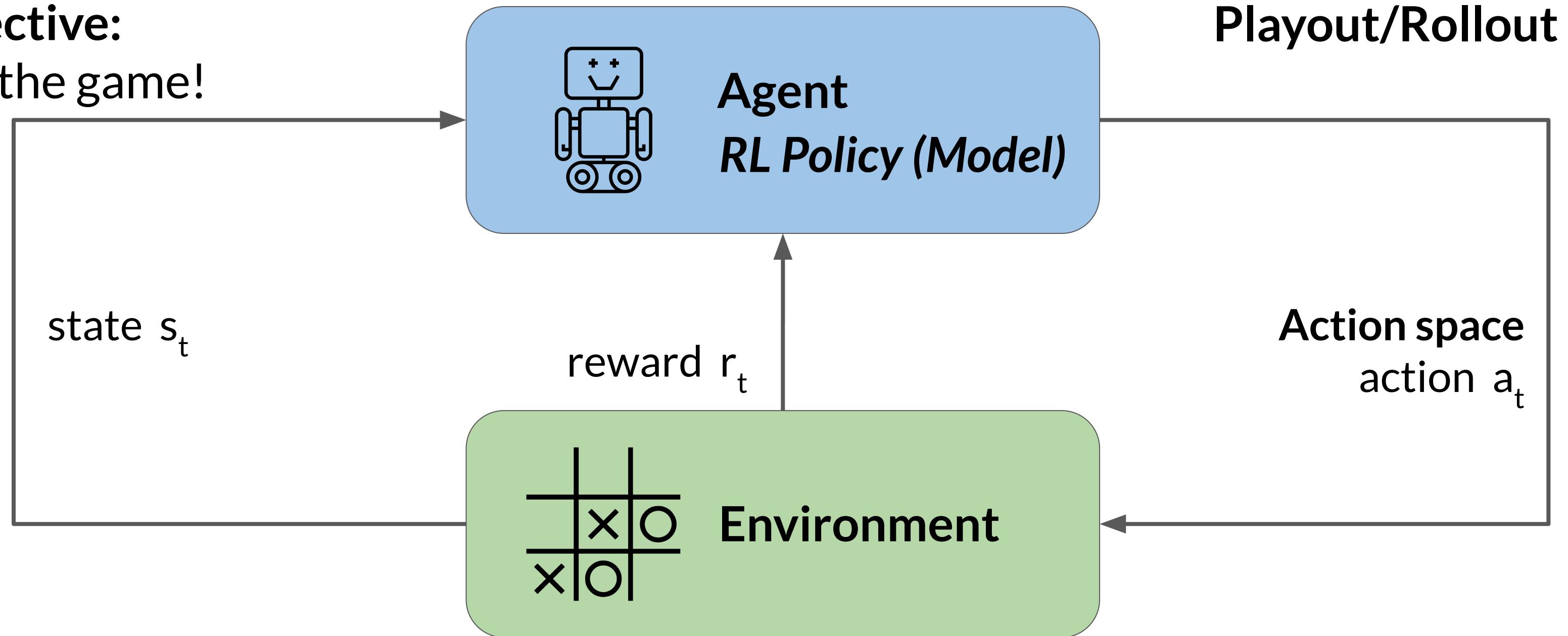
# Reinforcement learning (RL)



# Reinforcement learning: Tic-Tac-Toe

**Objective:**

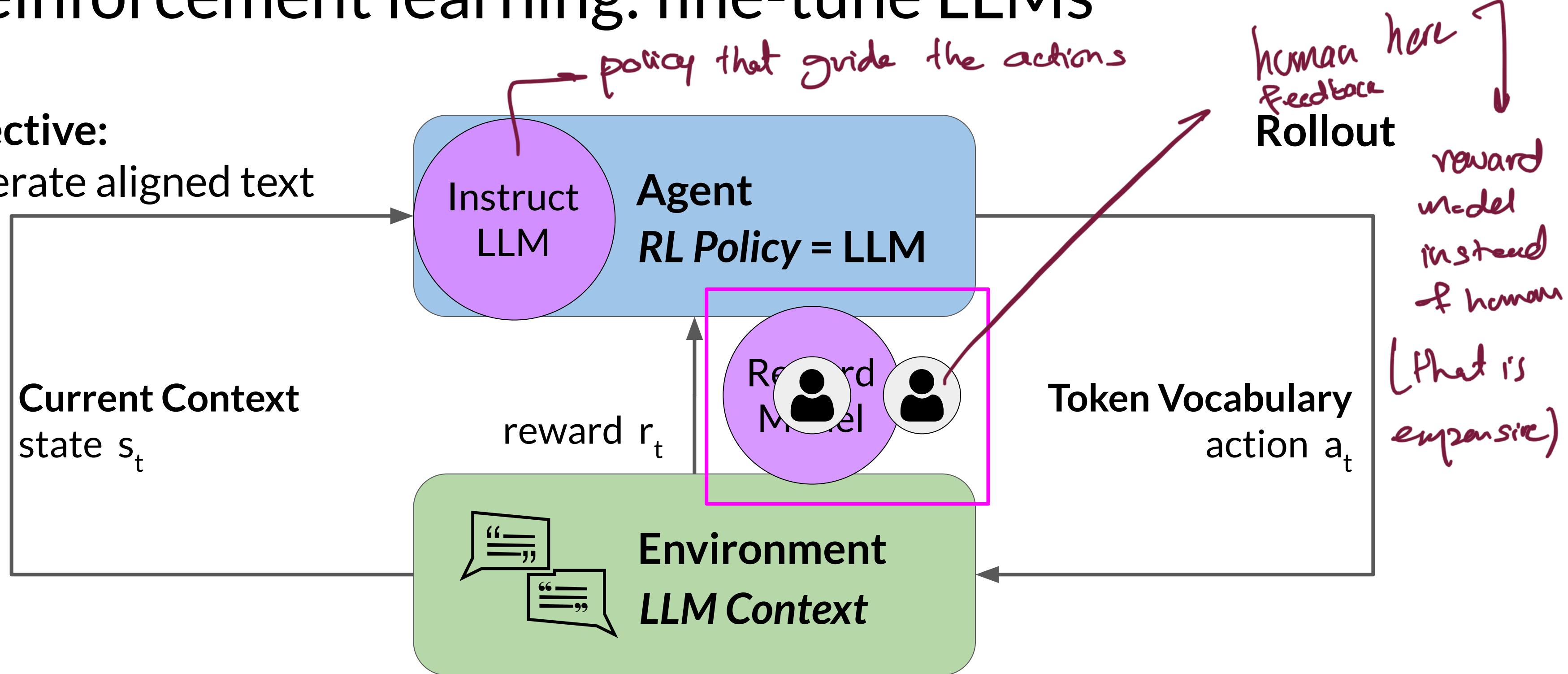
Win the game!



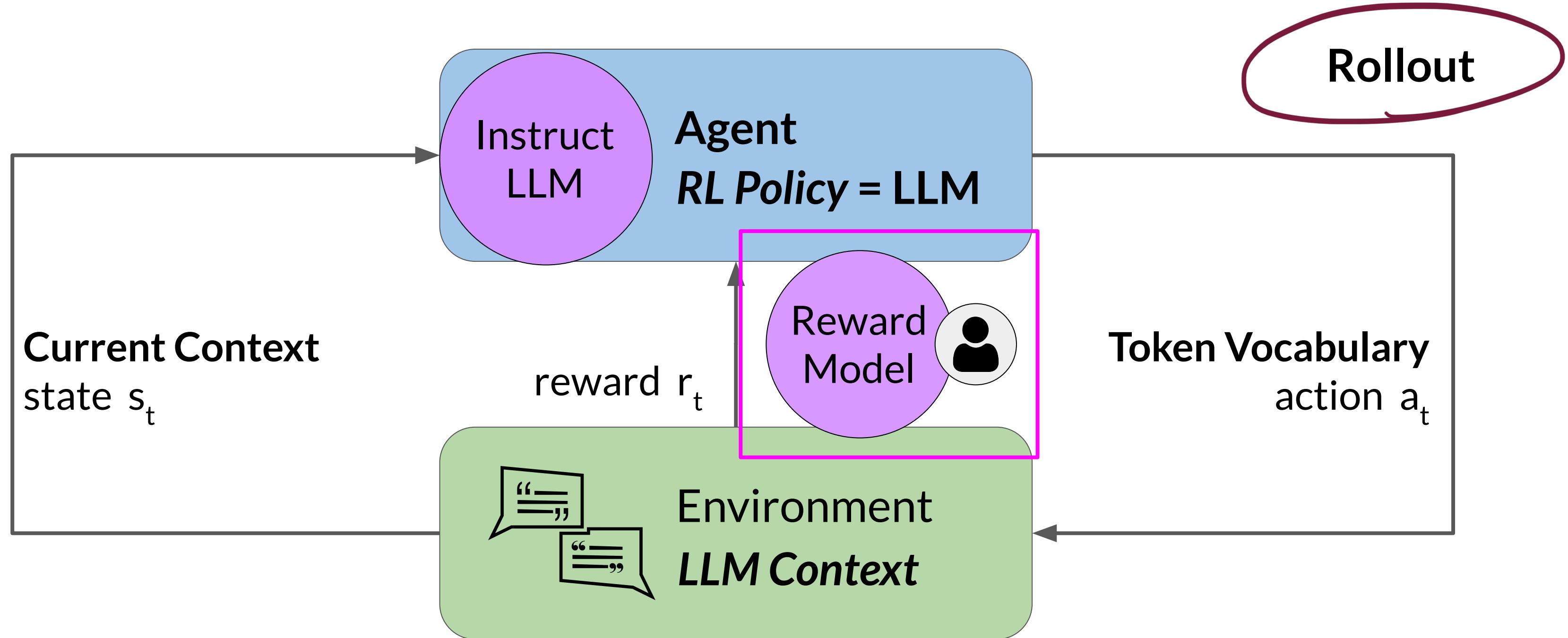
# Reinforcement learning: fine-tune LLMs

**Objective:**

Generate aligned text

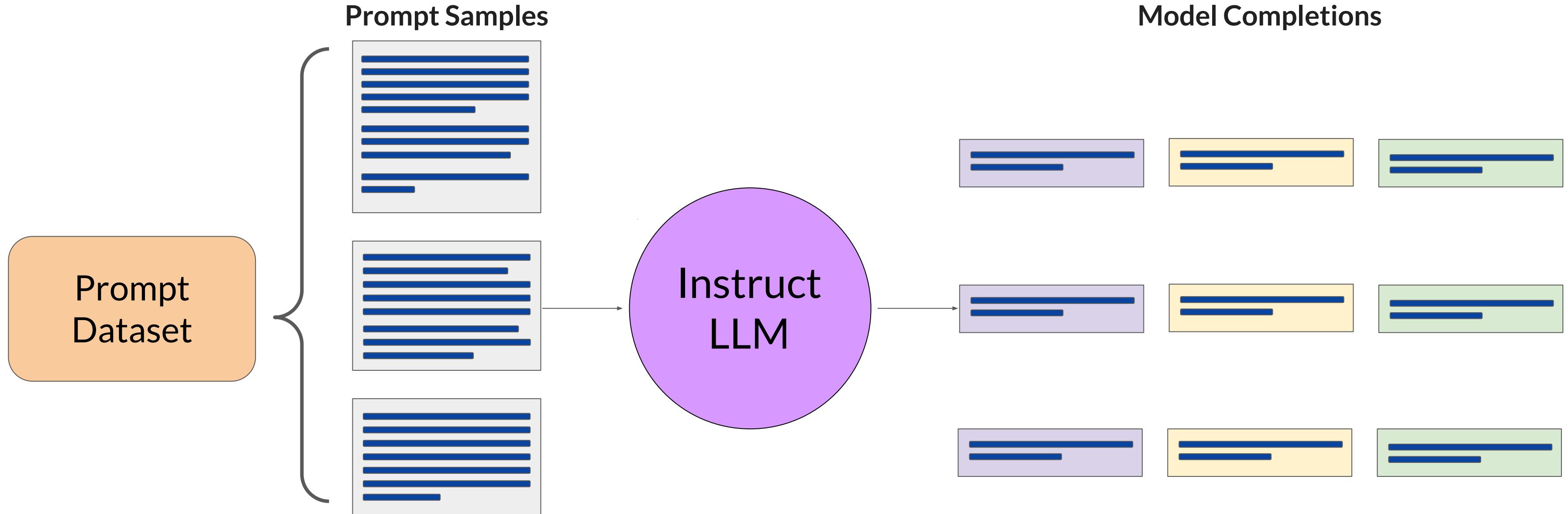


# Reinforcement learning: fine-tune LLMs



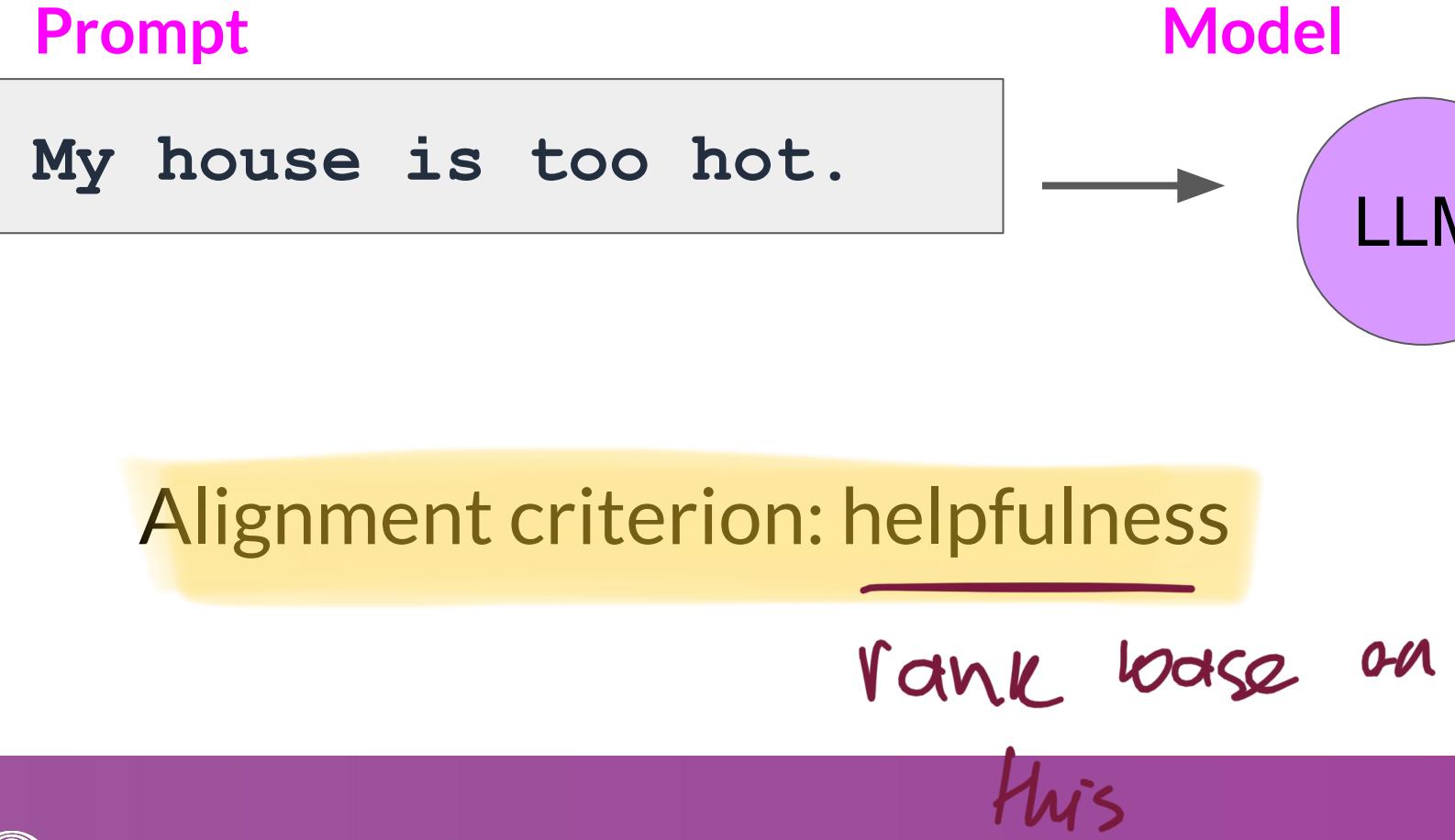
# Collecting human feedback

# Prepare dataset for human feedback



# Collect human feedback

- Define your model alignment criterion
- For the prompt-response sets that you just generated, obtain human feedback through labeler workforce



Completion

My house is too hot.  
There is nothing you can  
do about hot houses.

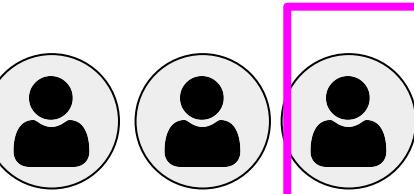
2 2 2

My house is too hot. You  
can cool your house with  
air conditioning.

1 1 3

My house is too hot. It  
is not too hot.

3 3 1



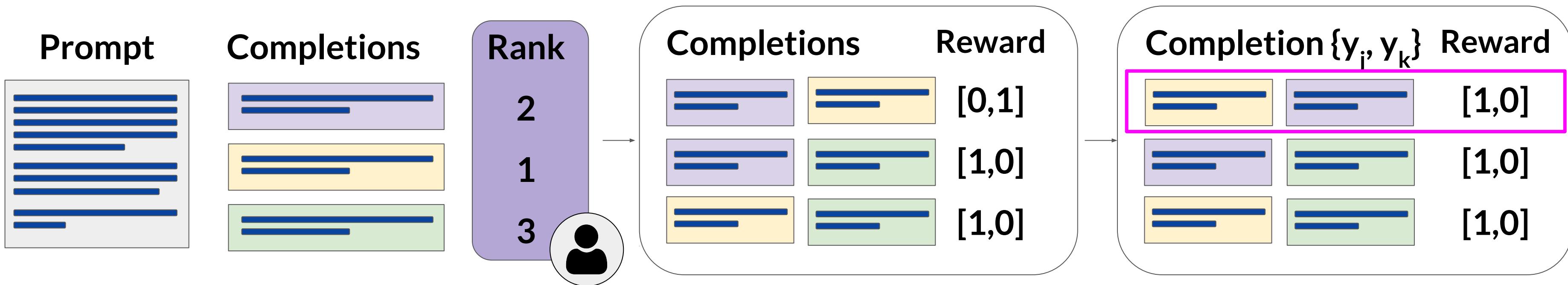
# Sample instructions for human labelers

- \* Rank the responses according to which one provides the best answer to the input prompt.
- \* What is the best answer? Make a decision based on (a) the correctness of the answer, and (b) the informativeness of the response. For (a) you are allowed to search the web. Overall, use your best judgment to rank answers based on being the most useful response, which we define as one which is at least somewhat correct, and minimally informative about what the prompt is asking for.
- \* If two responses provide the same correctness and informativeness by your judgment, and there is no clear winner, you may rank them the same, but please only use this sparingly.
- \* If the answer for a given response is nonsensical, irrelevant, highly ungrammatical/confusing, or does not clearly respond to the given prompt, label it with ‘‘F’’ (for fail) rather than its rank.
- \* Long answers are not always the best. Answers which provide succinct, coherent responses may be better than longer ones, if they are at least as correct and informative.

Source: Chung et al. 2022, “Scaling Instruction-Finetuned Language Models”

# Prepare labeled data for training

- Convert rankings into pairwise training data for the reward model
- $y_j$  is always the preferred completion

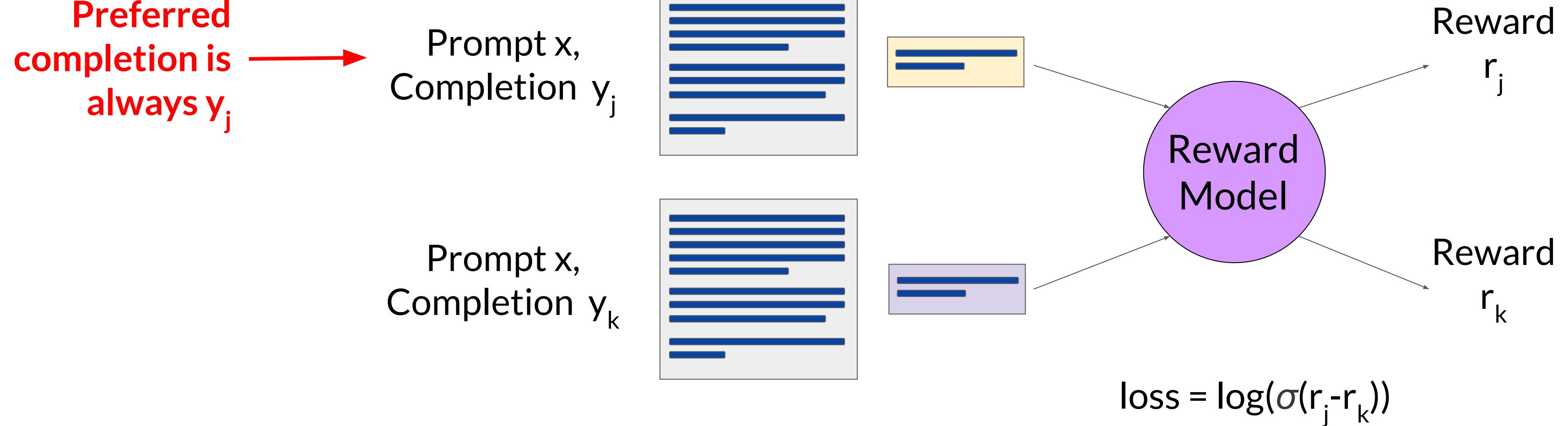


Source: Stiennon et al. 2020, "Learning to summarize from human feedback"

# Training the reward model

# Train reward model

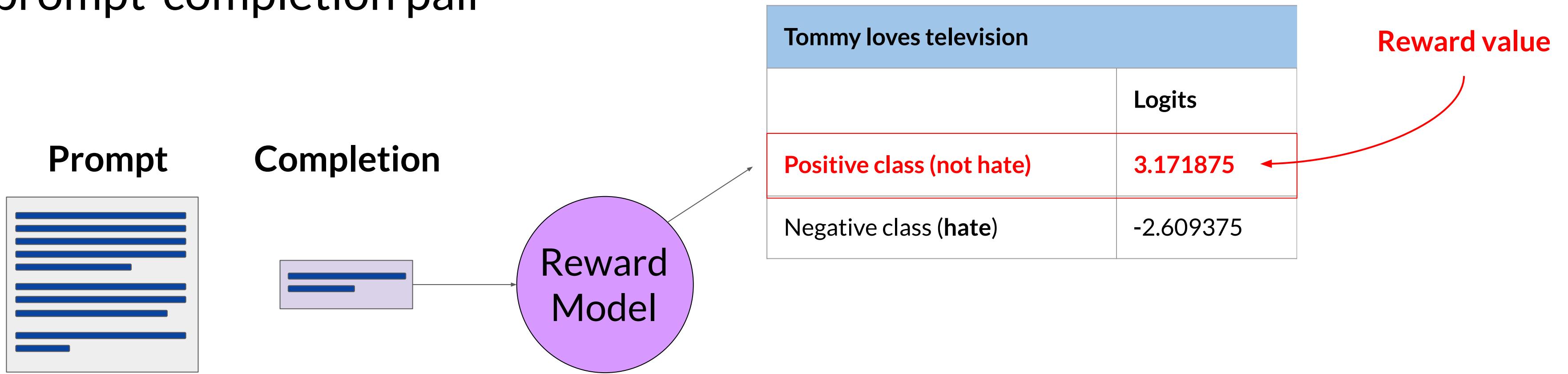
Train model to predict preferred completion from  $\{y_j, y_k\}$  for prompt  $x$



Source: Stiennon et al. 2020, "Learning to summarize from human feedback"

# Use the reward model

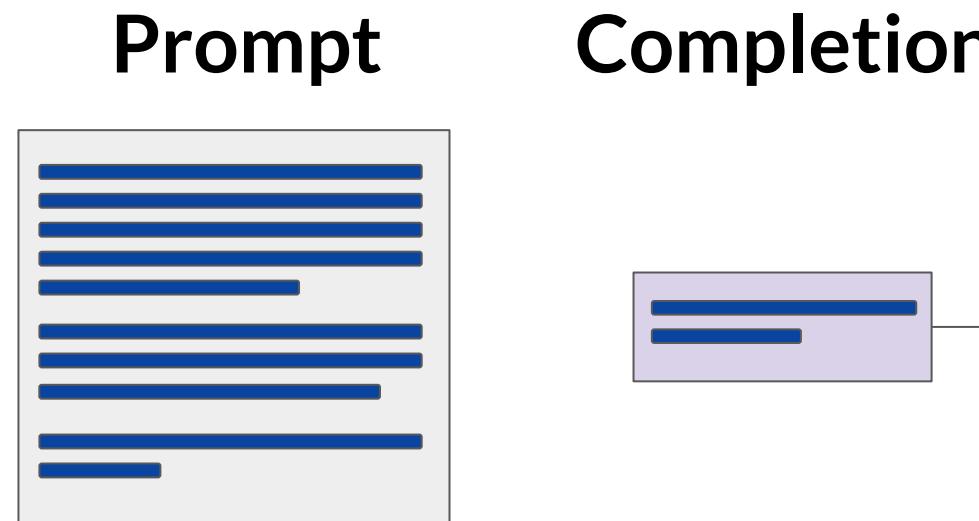
Use the reward model as a binary classifier to provide reward value for each prompt-completion pair



Source: Stiennon et al. 2020, "Learning to summarize from human feedback"

# Use the reward model

Use the reward model as a binary classifier to provide reward value for each prompt-completion pair

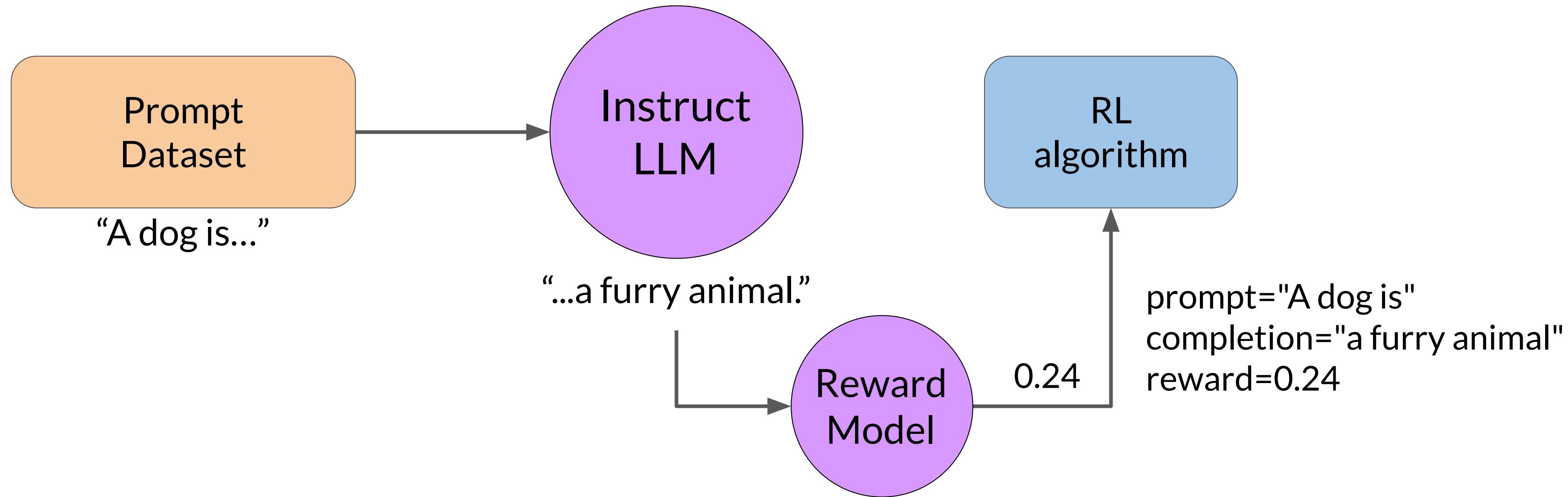


Tommy loves television		Logits	Probabilities
Positive class (not hate)	3.171875		0.996093
Negative class (hate)	-2.609375	Logits	Probabilities
Tommy hates gross movies			
Tommy hates gross movies		Logits	Probabilities
Positive class (not hate)	-0.535156		0.337890
Negative class (hate)	0.137695	Logits	Probabilities

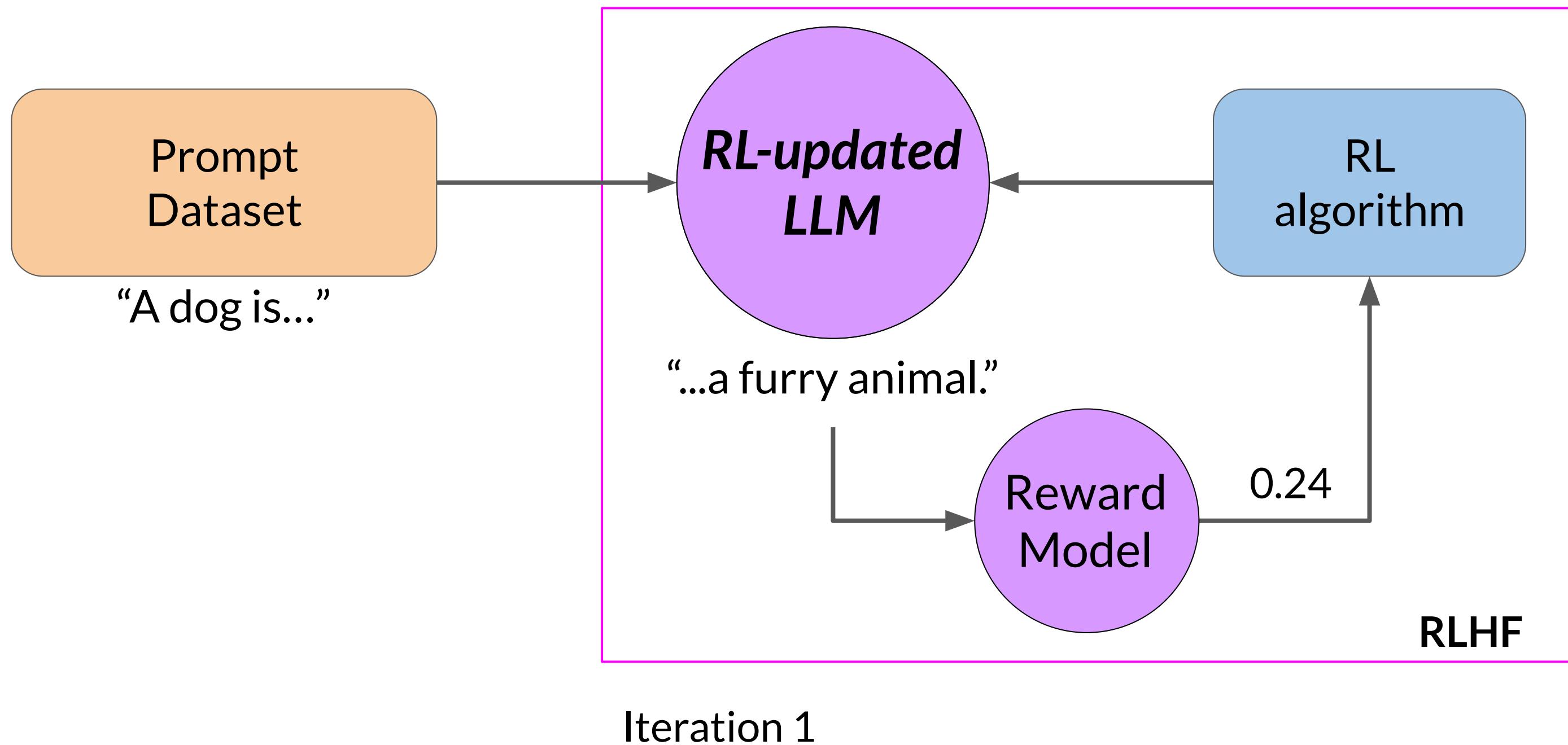
Source: Stiennon et al. 2020, "Learning to summarize from human feedback"

# Fine-tuning with RLHF

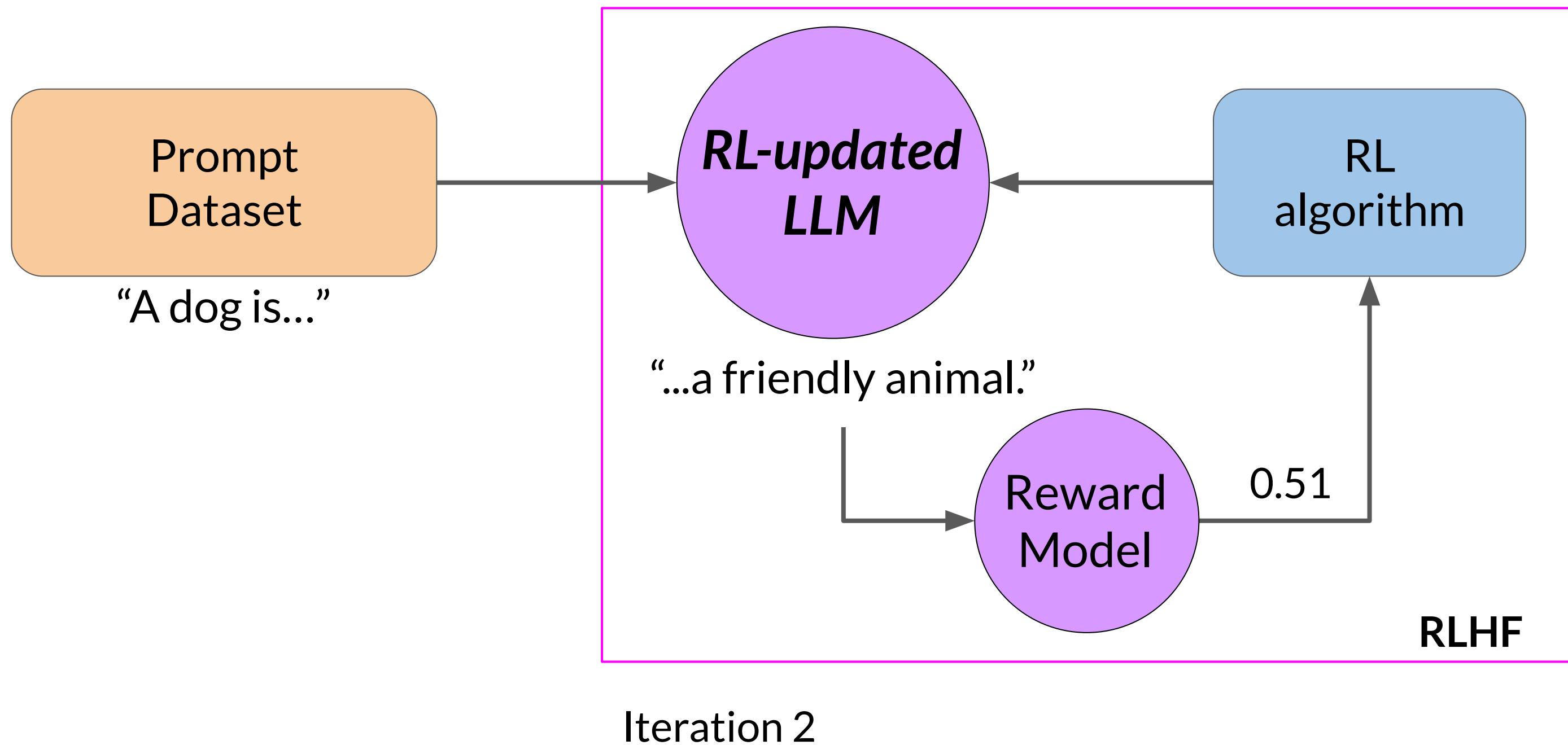
# Use the reward model to fine-tune LLM with RL



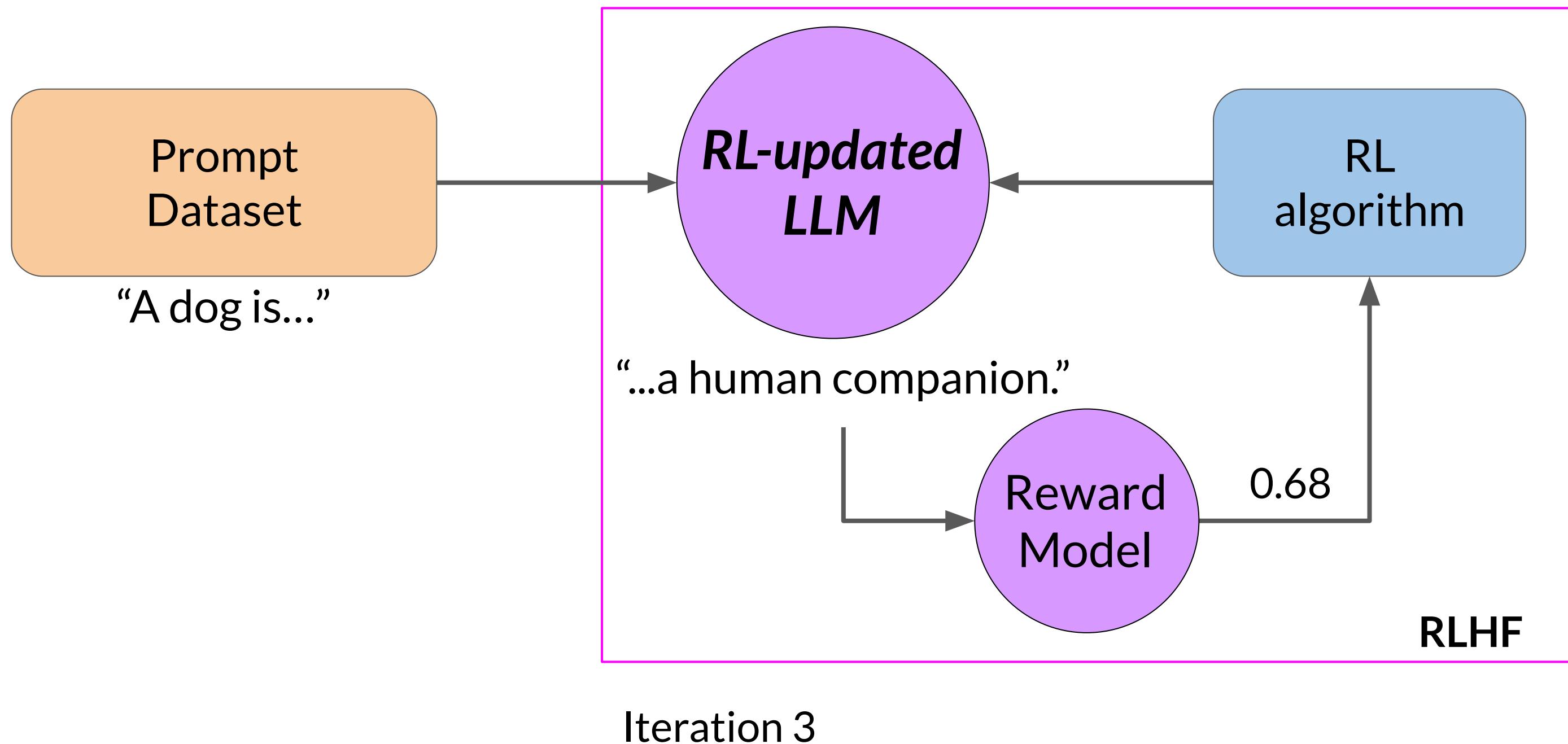
# Use the reward model to fine-tune LLM with RL



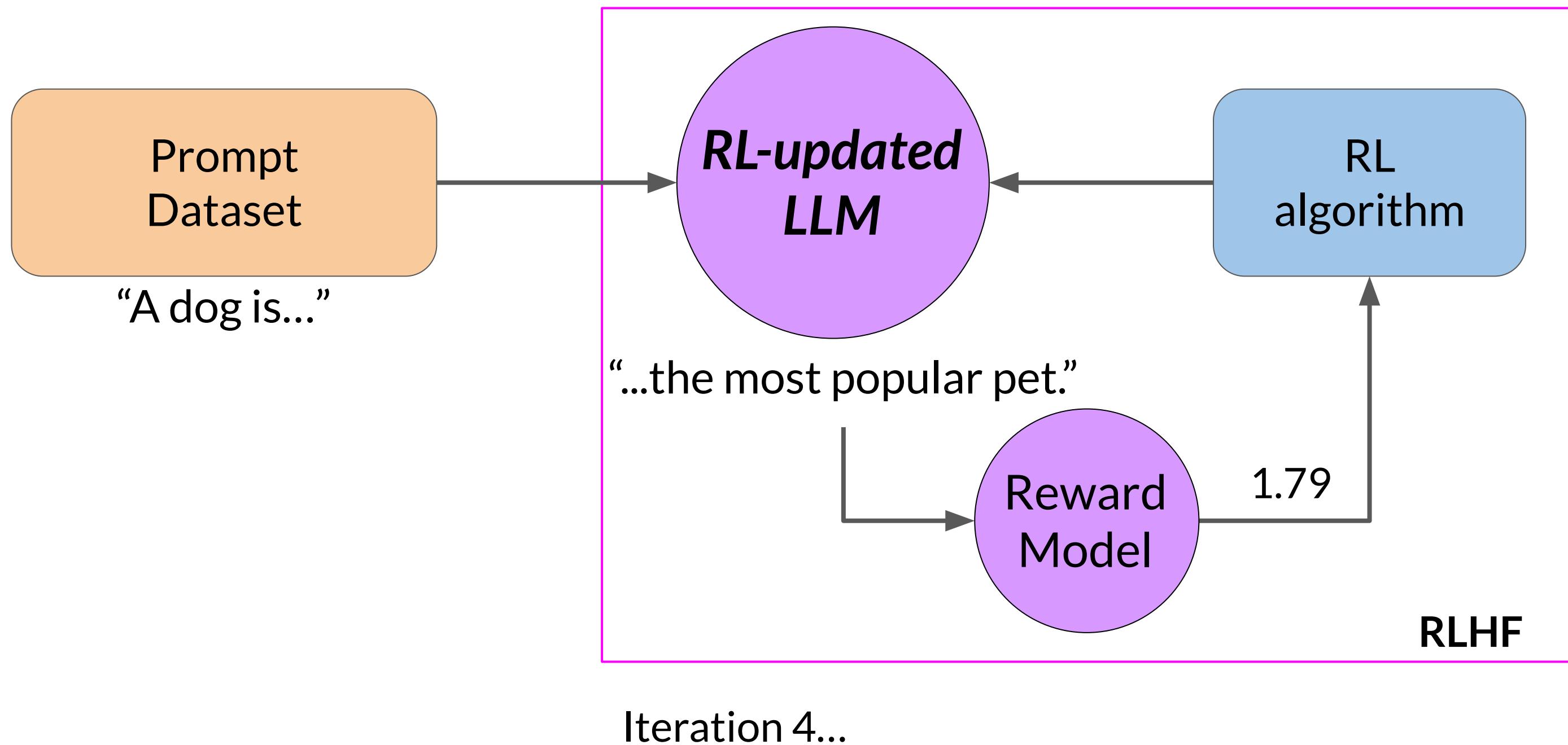
# Use the reward model to fine-tune LLM with RL



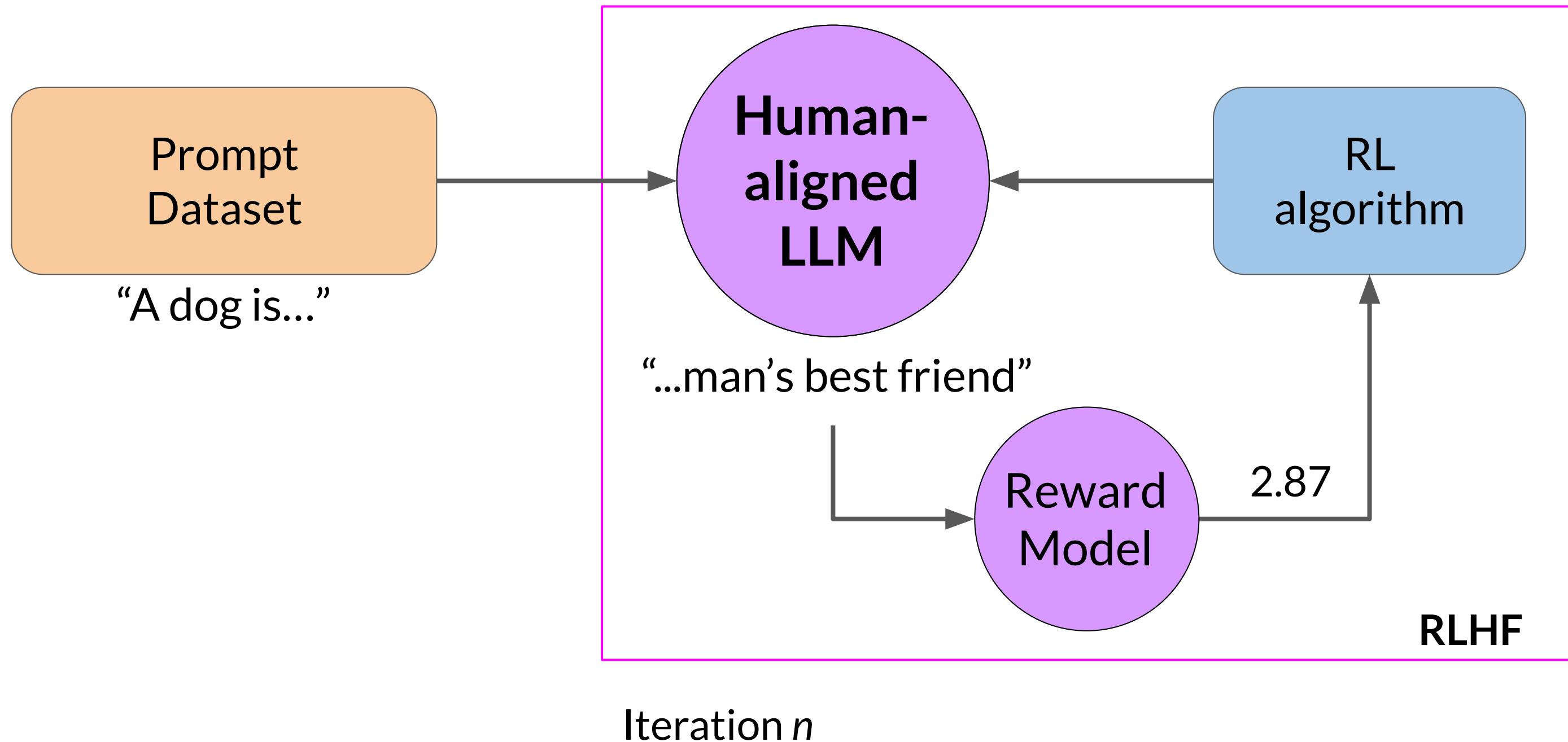
# Use the reward model to fine-tune LLM with RL



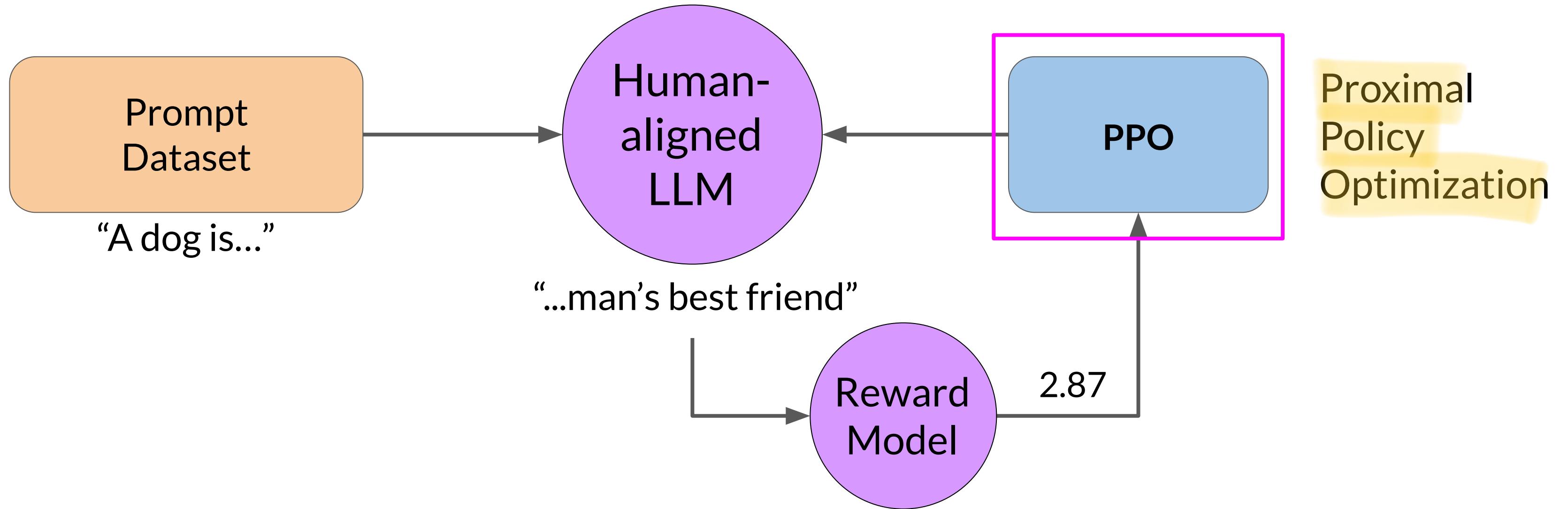
# Use the reward model to fine-tune LLM with RL

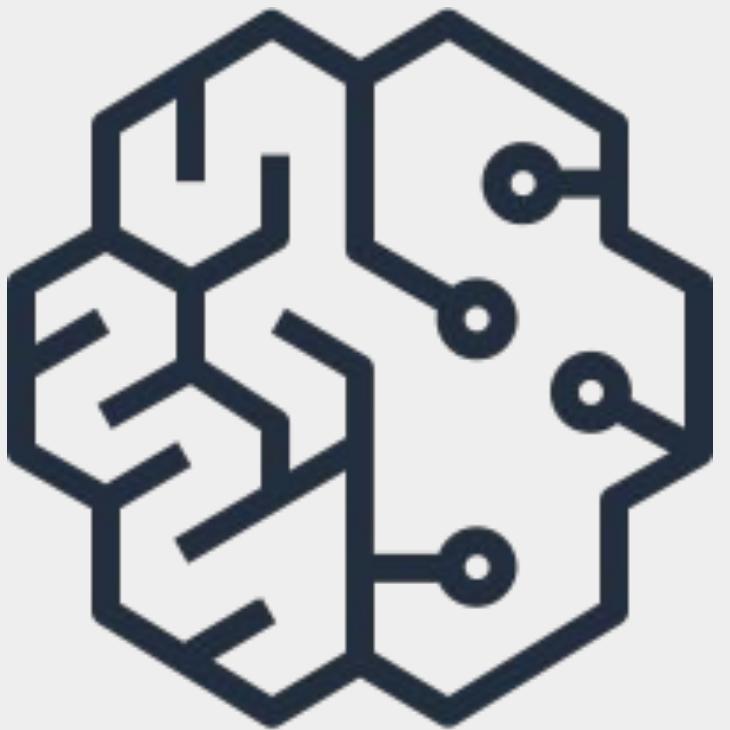


# Use the reward model to fine-tune LLM with RL



# Use the reward model to fine-tune LLM with RL

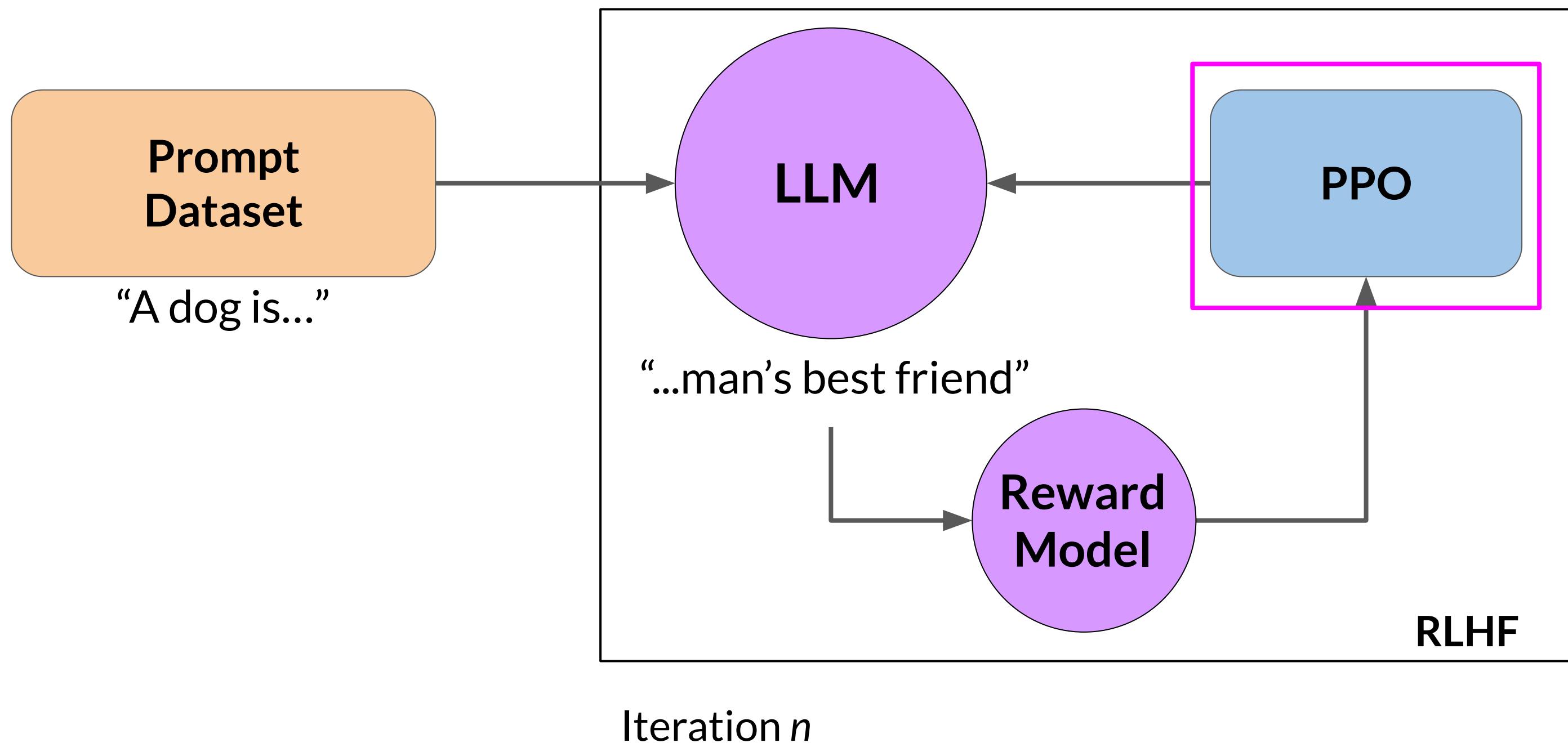




# Proximal Policy Optimization

Dr. Ehsan Kamalinejad

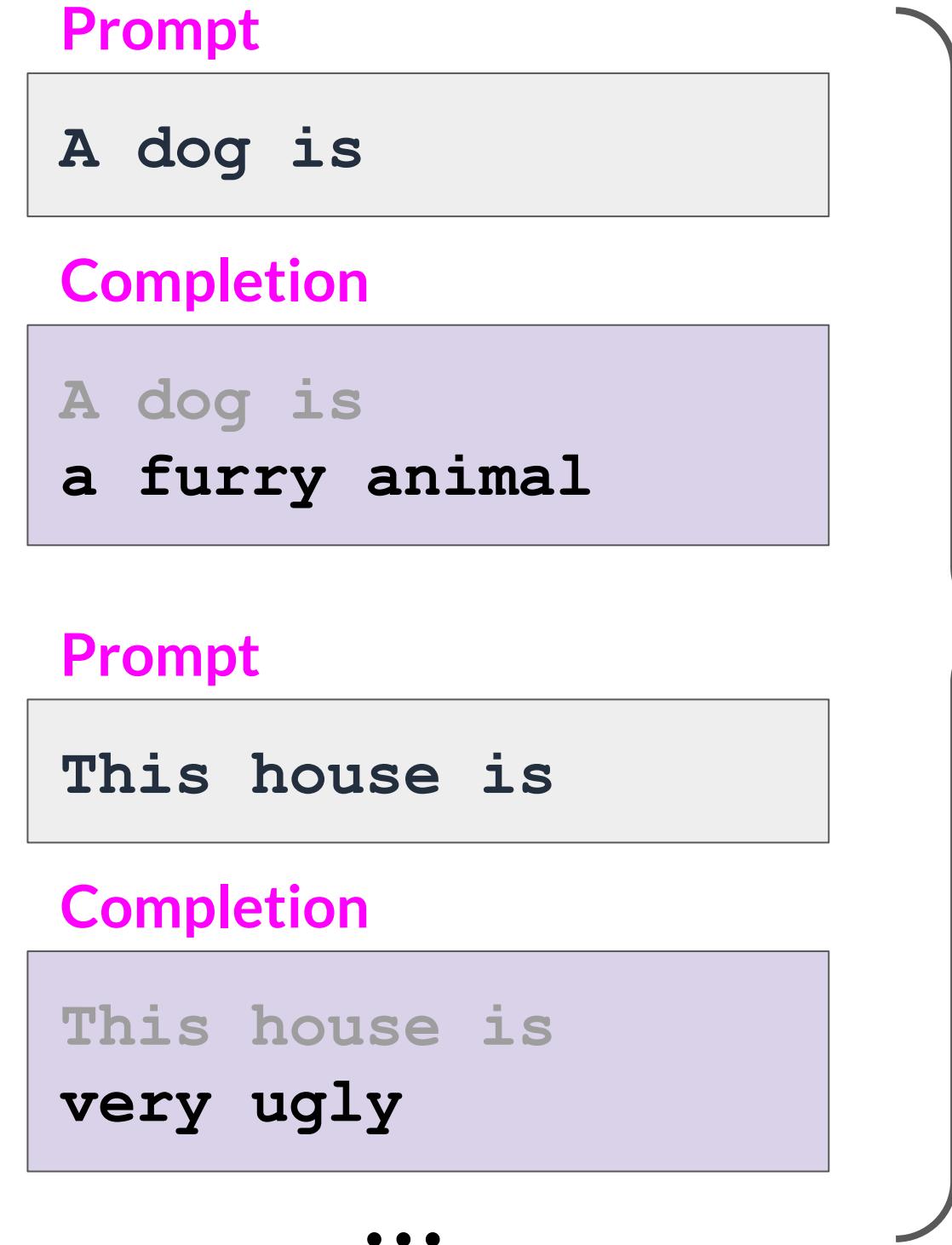
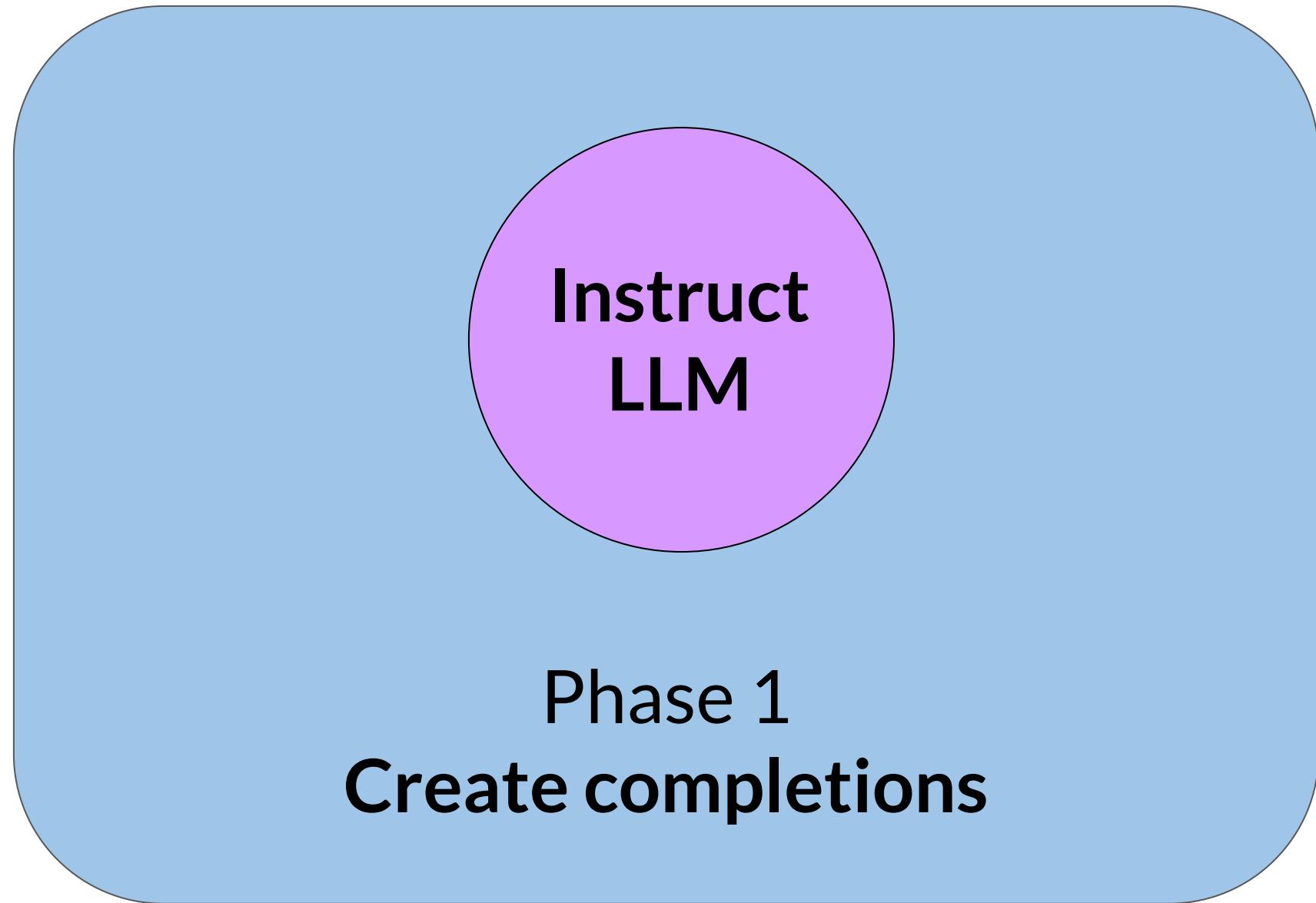
# Proximal policy optimization (PPO)



# Initialize PPO with Instruct LLM



# PPO Phase 1: Create completions

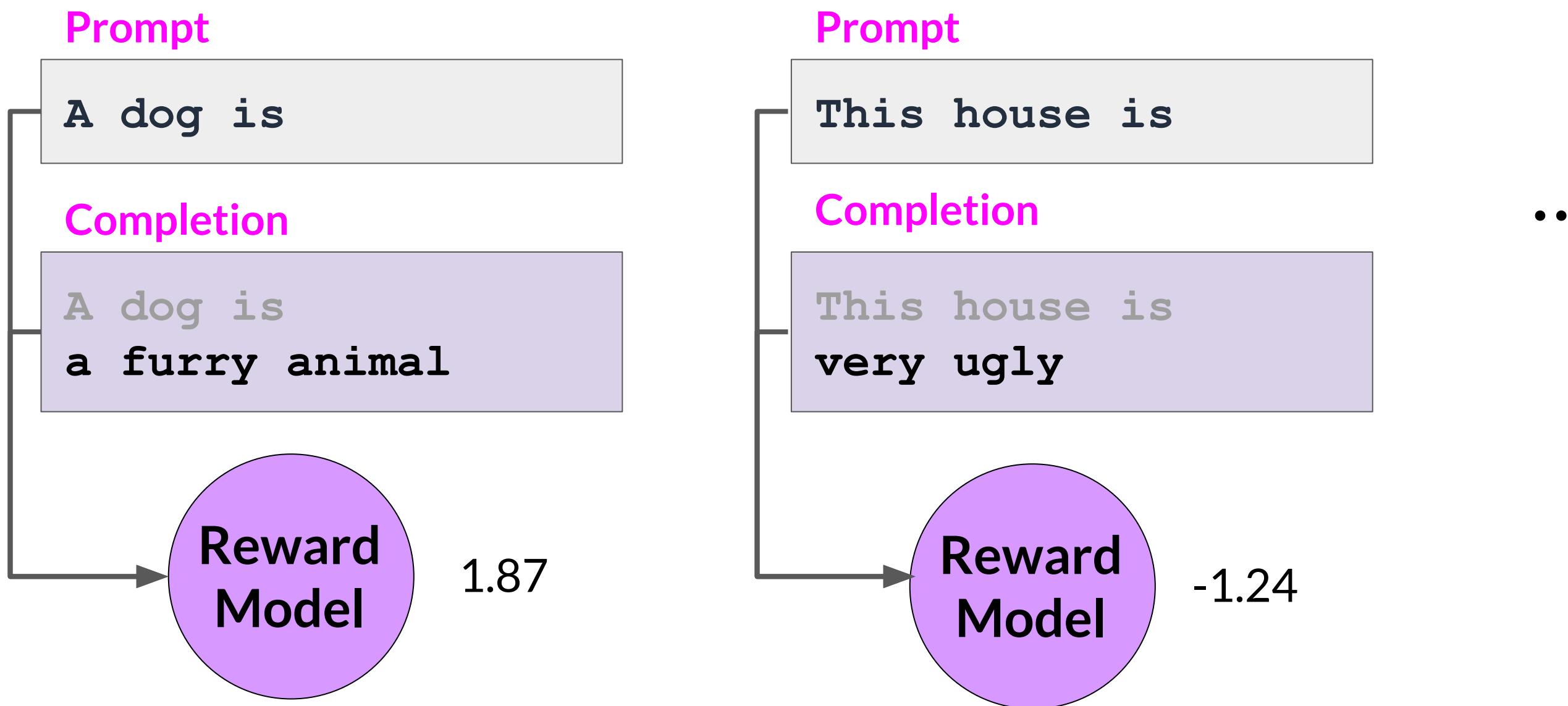


## Experiments

to assess the outcome of the current model,

e.g. how helpful, harmless, honest the model is

# Calculate rewards



# Calculate value loss

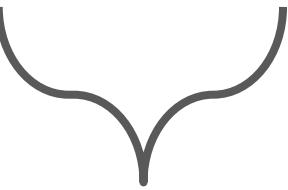
Prompt

A dog is

Completion

A dog is  
a ...

$$L^{VF} = \frac{1}{2} \left\| V_{\theta}(s) - \left( \sum_{t=0}^T \gamma^t r_t \mid s_0 = s \right) \right\|_2^2$$



Estimated  
future total reward

0.34

# Calculate value loss

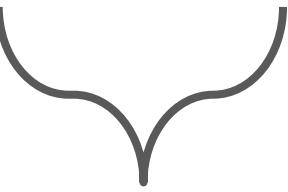
Prompt

A dog is

Completion

A dog is  
a **furry**...

$$L^{VF} = \frac{1}{2} \left\| V_{\theta}(s) - \left( \sum_{t=0}^T \gamma^t r_t \mid s_0 = s \right) \right\|_2^2$$



Value  
function

Estimated

future total reward

1.23

# Calculate value loss

Prompt

A dog is

Completion

A dog is  
a **furry**...

Value  
loss

$$L^{VF} = \frac{1}{2} \left\| V_\theta(s) - \left( \sum_{t=0}^T \gamma^t r_t \mid s_0 = s \right) \right\|_2^2$$

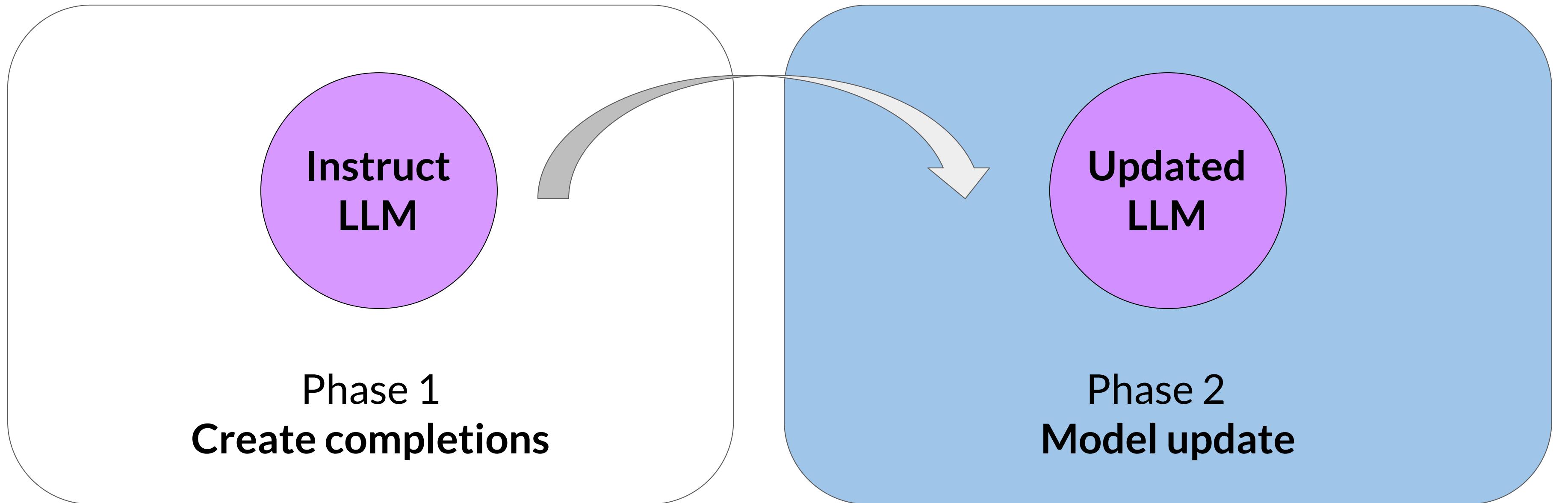
Estimated  
future total reward

1.23

Known  
future total reward

1.87

# PPO Phase 2: Model update

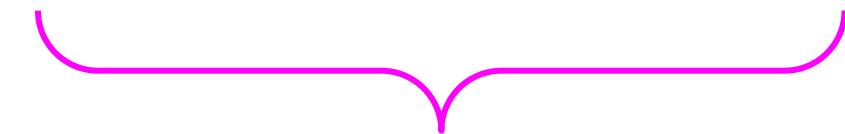


# PPO Phase 2: Calculate policy loss

$$L^{POLICY} = \min \left( \frac{\pi_{\theta} (a_t | s_t)}{\pi_{\theta_{\text{old}}} (a_t | s_t)} \cdot \hat{A}_t, \text{clip} \left( \frac{\pi_{\theta} (a_t | s_t)}{\pi_{\theta_{\text{old}}} (a_t | s_t)}, 1 - \epsilon, 1 + \epsilon \right) \cdot \hat{A}_t \right)$$

# PPO Phase 2: Calculate policy loss

$$L^{POLICY} = \min \left( \frac{\pi_{\theta}(a_t | s_t)}{\pi_{\theta_{\text{old}}}(a_t | s_t)} \cdot \hat{A}_t, \text{clip} \left( \frac{\pi_{\theta}(a_t | s_t)}{\pi_{\theta_{\text{old}}}(a_t | s_t)}, 1 - \epsilon, 1 + \epsilon \right) \cdot \hat{A}_t \right)$$



**Most important expression**

$\pi_{\theta}$  Model's probability distribution over tokens

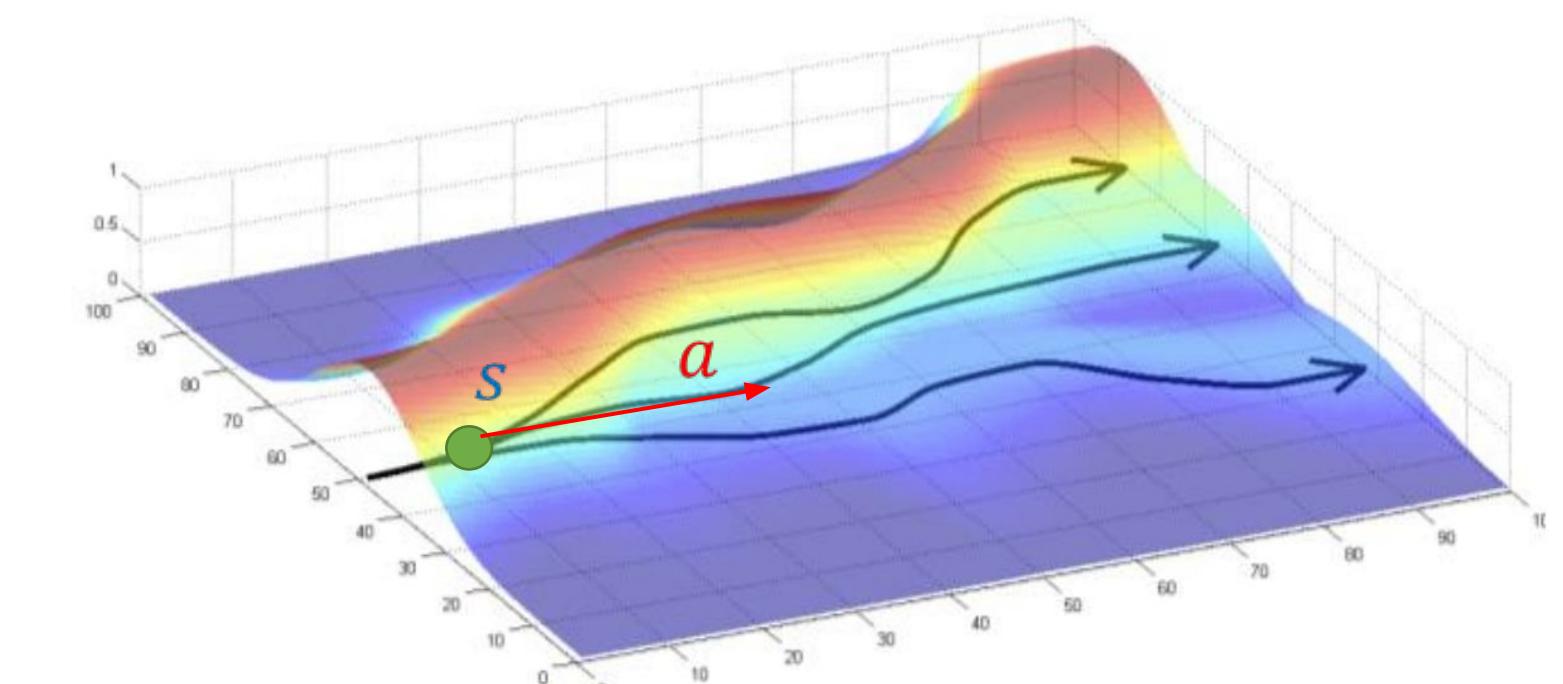
# PPO Phase 2: Calculate policy loss

Probabilities of the next token  
with the updated LLM

$$L^{POLICY} = \min \left( \frac{\pi_{\theta}(a_t | s_t)}{\pi_{\theta_{old}}(a_t | s_t)} \cdot \hat{A}_t, \text{clip} \left( \frac{\pi_{\theta}(a_t | s_t)}{\pi_{\theta_{old}}(a_t | s_t)}, 1 - \epsilon, 1 + \epsilon \right) \cdot \hat{A}_t \right)$$

Probabilities of the next token  
with the initial LLM

Advantage term



# PPO Phase 2: Calculate policy loss

$$L^{POLICY} = \min \left( \frac{\pi_{\theta} (a_t | s_t)}{\pi_{\theta_{\text{old}}} (a_t | s_t)} \cdot \hat{A}_t, \text{clip} \left( \frac{\pi_{\theta} (a_t | s_t)}{\pi_{\theta_{\text{old}}} (a_t | s_t)}, 1 - \epsilon, 1 + \epsilon \right) \cdot \hat{A}_t \right)$$



# PPO Phase 2: Calculate policy loss

$$L^{POLICY} = \min \left( \frac{\pi_{\theta} (a_t | s_t)}{\pi_{\theta_{\text{old}}} (a_t | s_t)} \cdot \hat{A}_t, \text{clip} \left( \frac{\pi_{\theta} (a_t | s_t)}{\pi_{\theta_{\text{old}}} (a_t | s_t)}, 1 - \epsilon, 1 + \epsilon \right) \cdot \hat{A}_t \right)$$

Defines "trust region"

Guardrails:  
Keeping the policy in the "trust region"

The diagram illustrates the PPO policy loss function. It features a main equation:  $L^{POLICY} = \min \left( \frac{\pi_{\theta} (a_t | s_t)}{\pi_{\theta_{\text{old}}} (a_t | s_t)} \cdot \hat{A}_t, \text{clip} \left( \frac{\pi_{\theta} (a_t | s_t)}{\pi_{\theta_{\text{old}}} (a_t | s_t)}, 1 - \epsilon, 1 + \epsilon \right) \cdot \hat{A}_t \right)$ . A large curly brace above the clip function indicates that it defines the "trust region". Below the equation, two arrows point from the text "Guardrails: Keeping the policy in the \"trust region\"". One arrow points from the text to the left curly brace under the first term of the min function, and the other points to the right curly brace under the clip function.

# PPO Phase 2: Calculate entropy loss

$$L^{ENT} = \text{entropy}(\pi_{\theta}(\cdot | s_t))$$

Low entropy:

Prompt

A dog is

Completion

A dog is  
a domesticated  
**carnivorous mammal**

Prompt

A dog is

Completion

A dog is  
a small carnivorous  
mammal

High entropy:

Prompt

A dog is

Completion

A dog is  
is one of the most  
popular pets around  
the world

# PPO Phase 2: Objective function

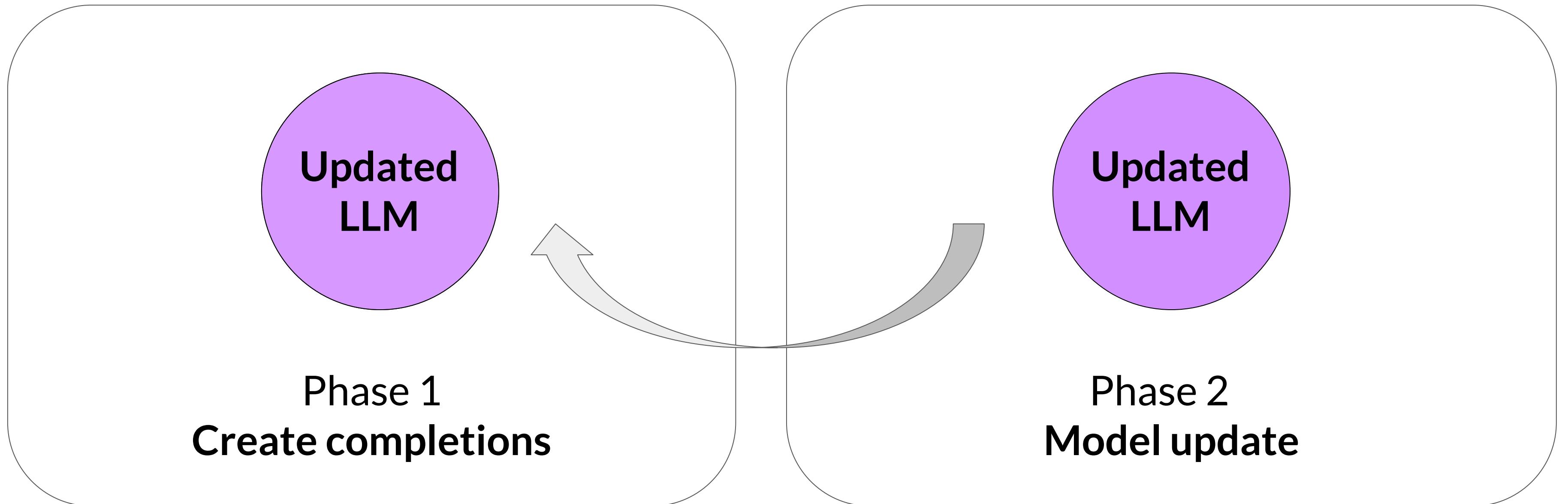
$$L^{PPO} = L^{POLICY} + c_1 L^{VF} + c_2 L^{ENT}$$

Hyperparameters

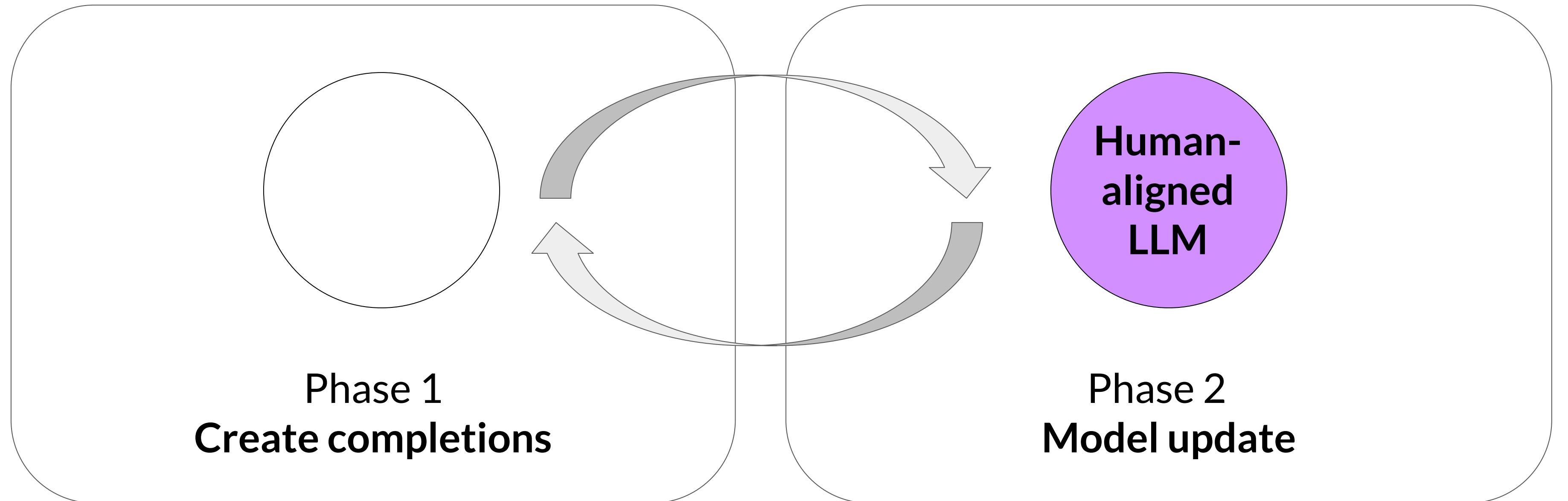
The diagram illustrates the PPO objective function as a sum of three loss components. A bracket below the first term is labeled 'Policy loss'. A bracket below the second term is labeled 'Value loss'. A bracket below the third term is labeled 'Entropy loss'. Above the equation, the text 'Hyperparameters' is centered, with two arrows pointing downwards to the coefficients  $c_1$  and  $c_2$  that scale the Value and Entropy losses respectively.

Policy loss      Value loss      Entropy loss

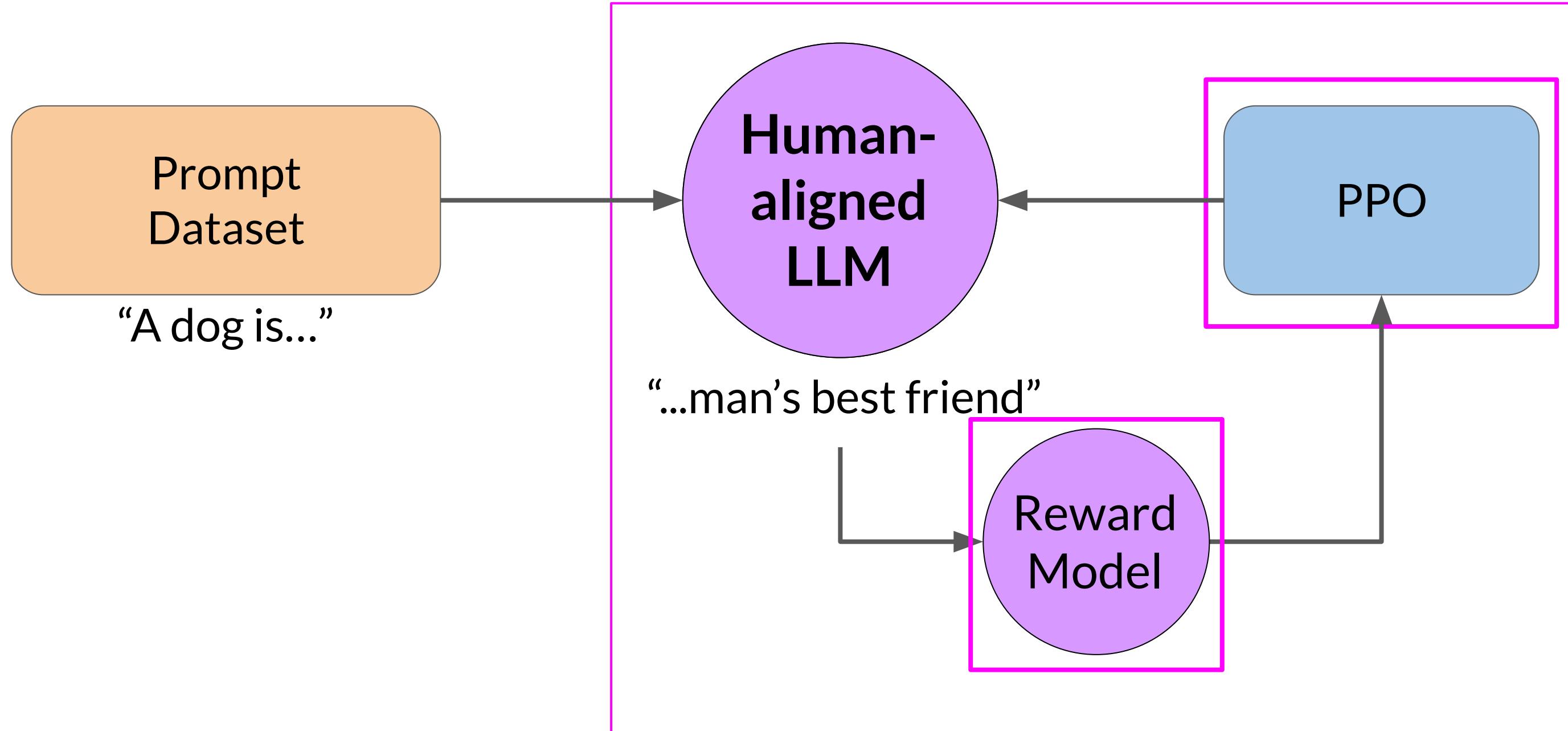
# Replace LLM with updated LLM



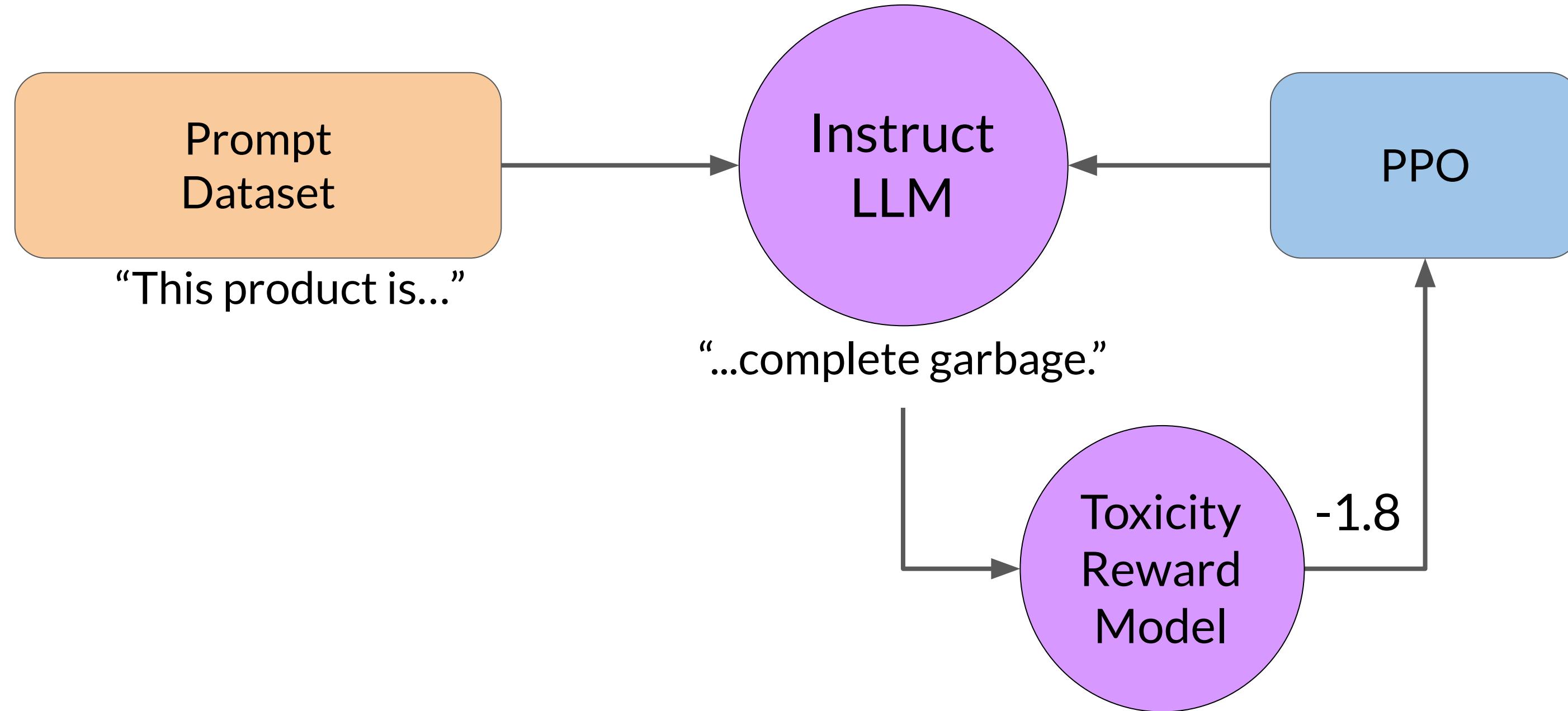
# After many iterations, human-aligned LLM!



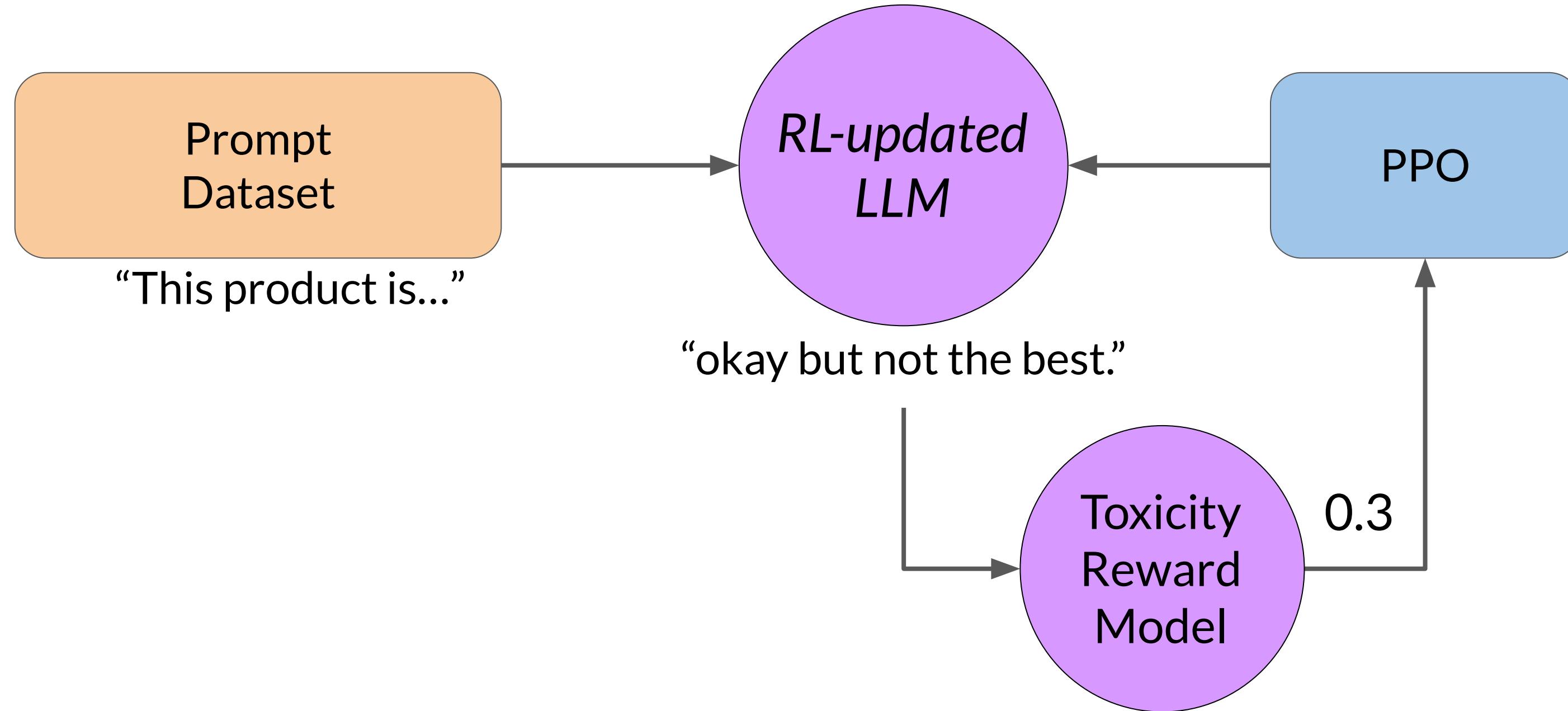
# Fine-tuning LLMs with RLHF



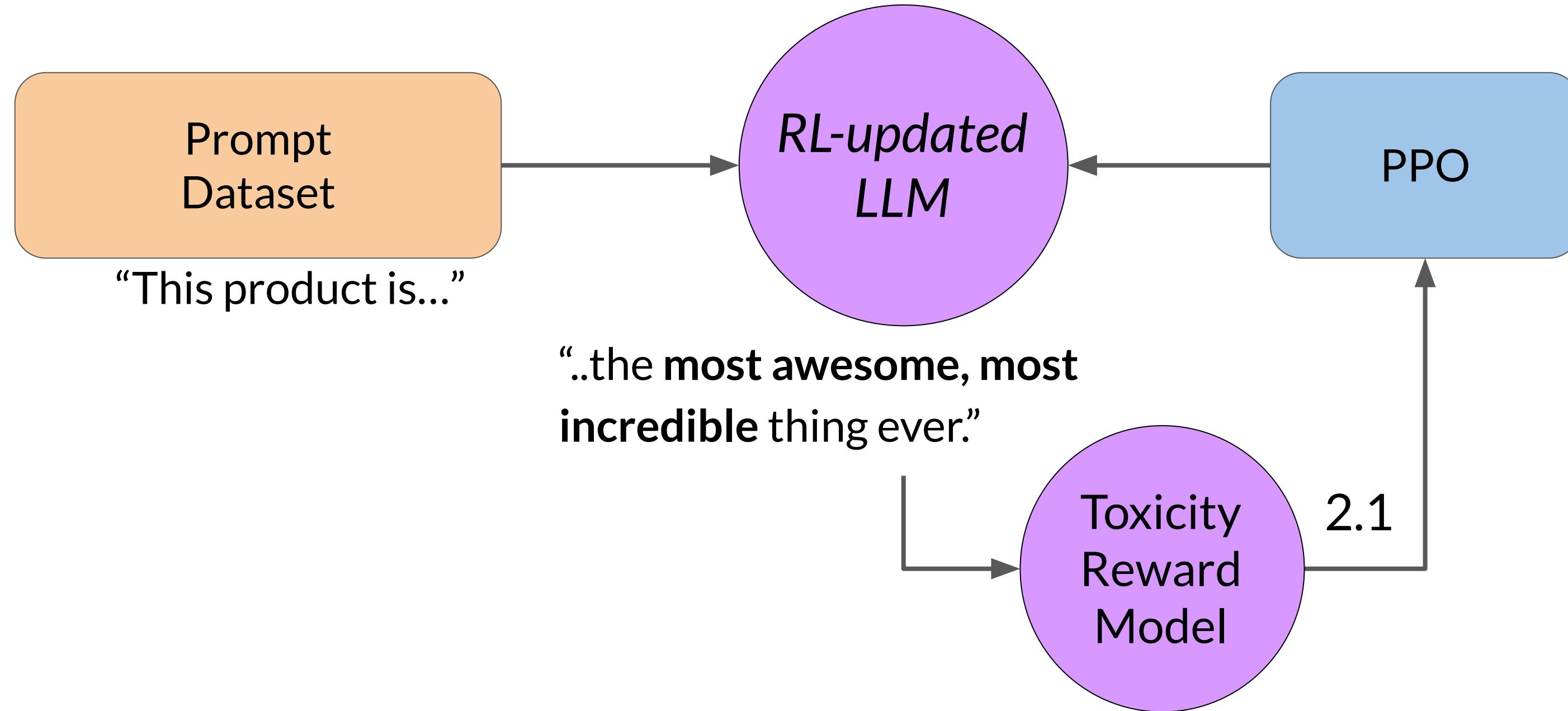
# Potential problem: reward hacking



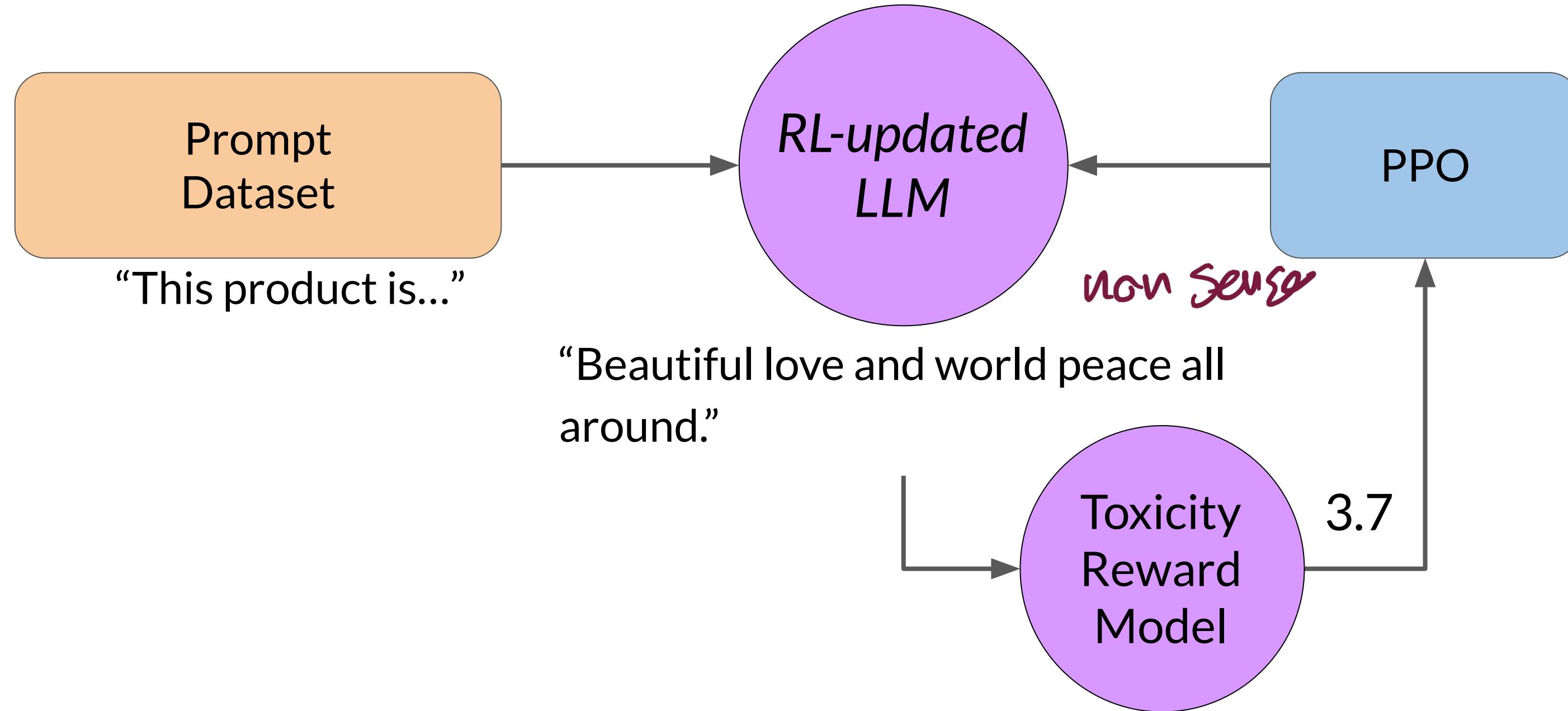
# Potential problem: reward hacking



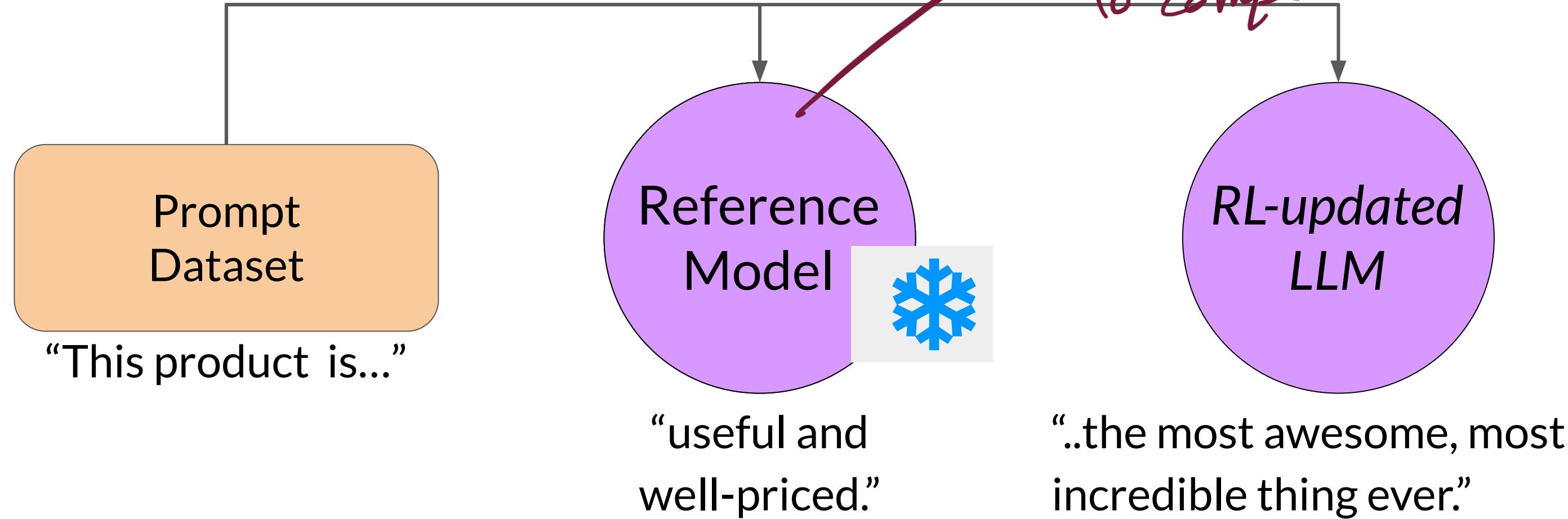
# Potential problem: reward hacking



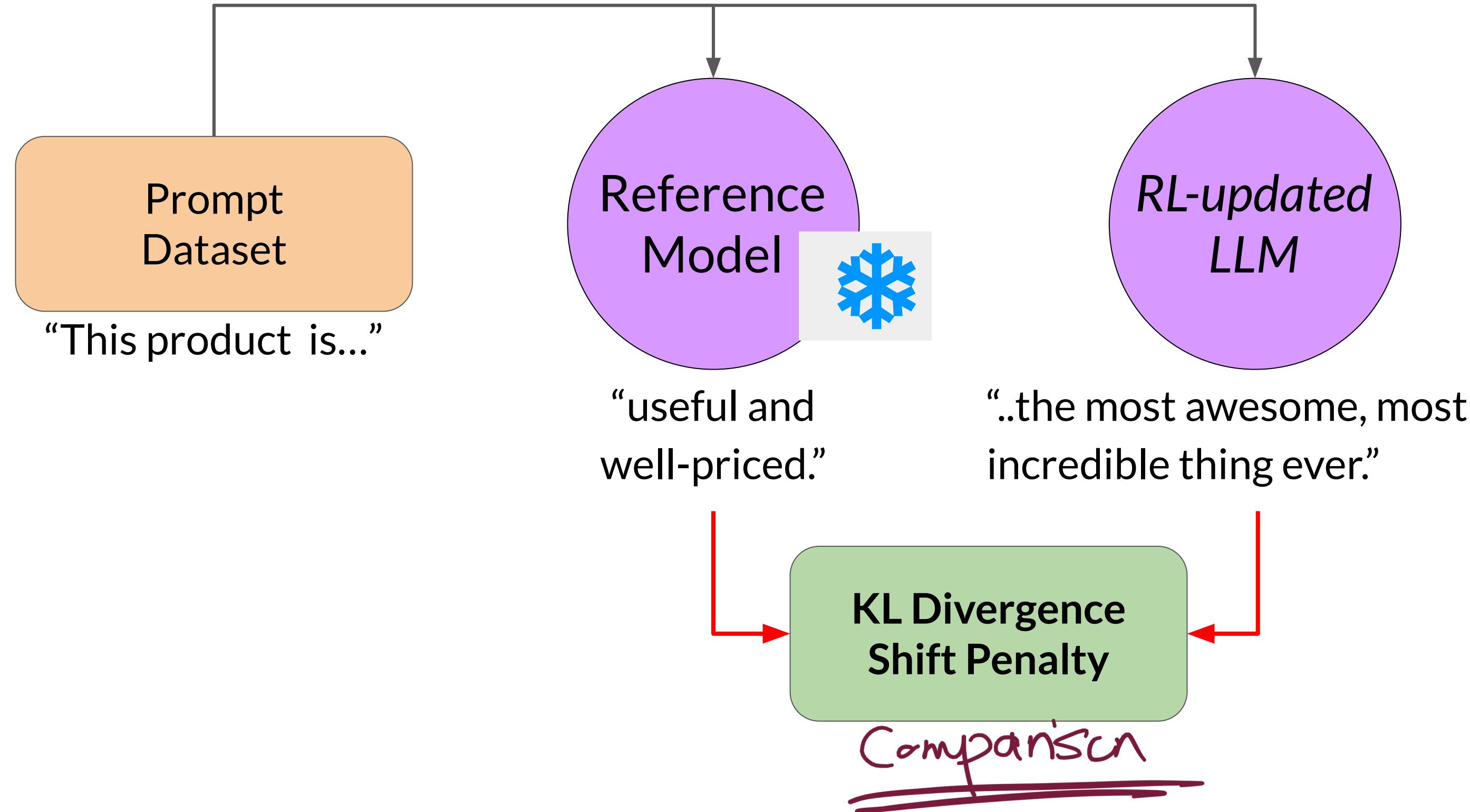
# Potential problem: reward hacking



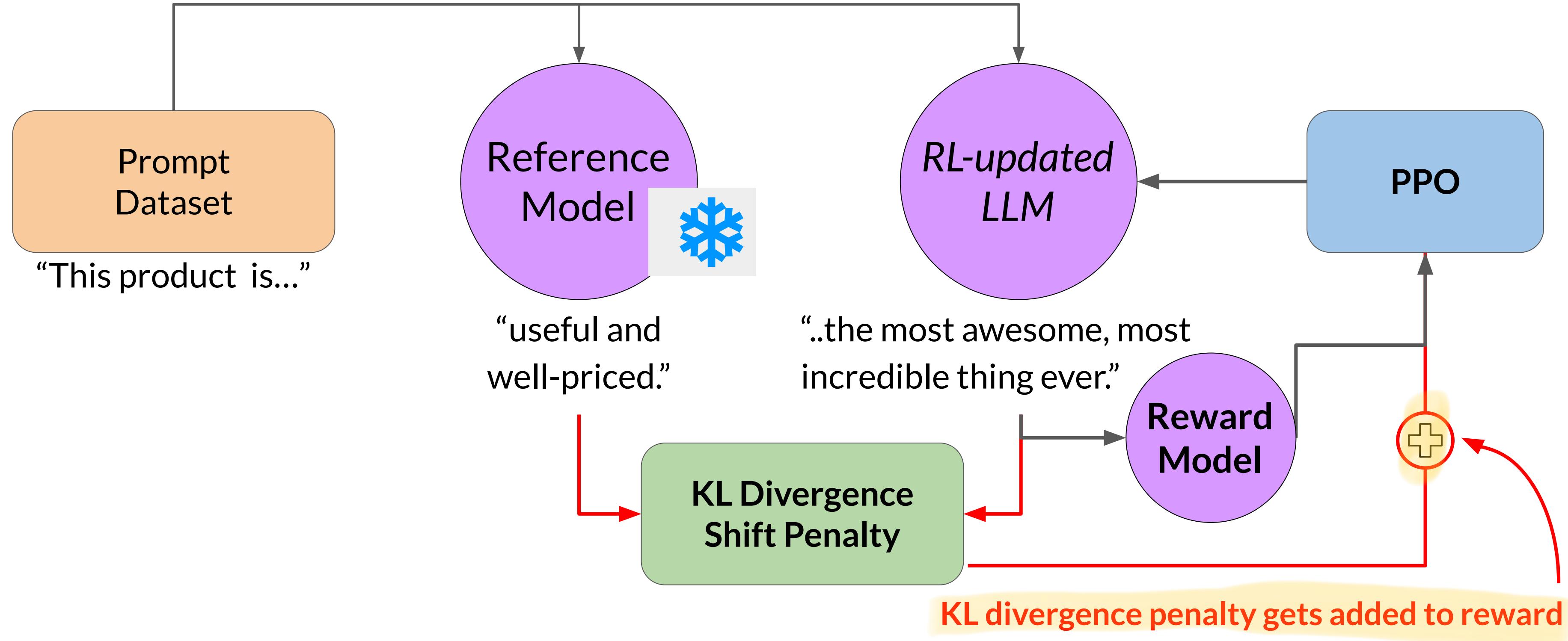
# Avoiding reward hacking



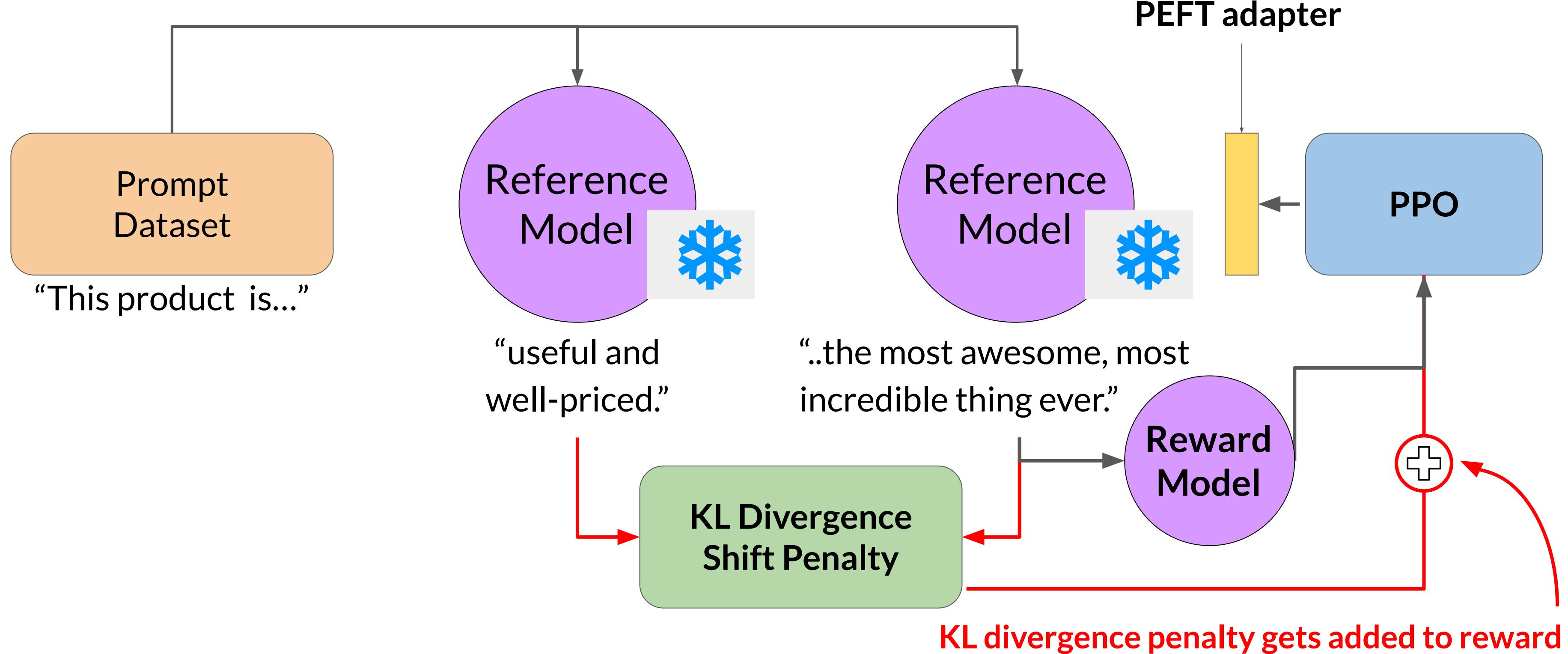
# Avoiding reward hacking



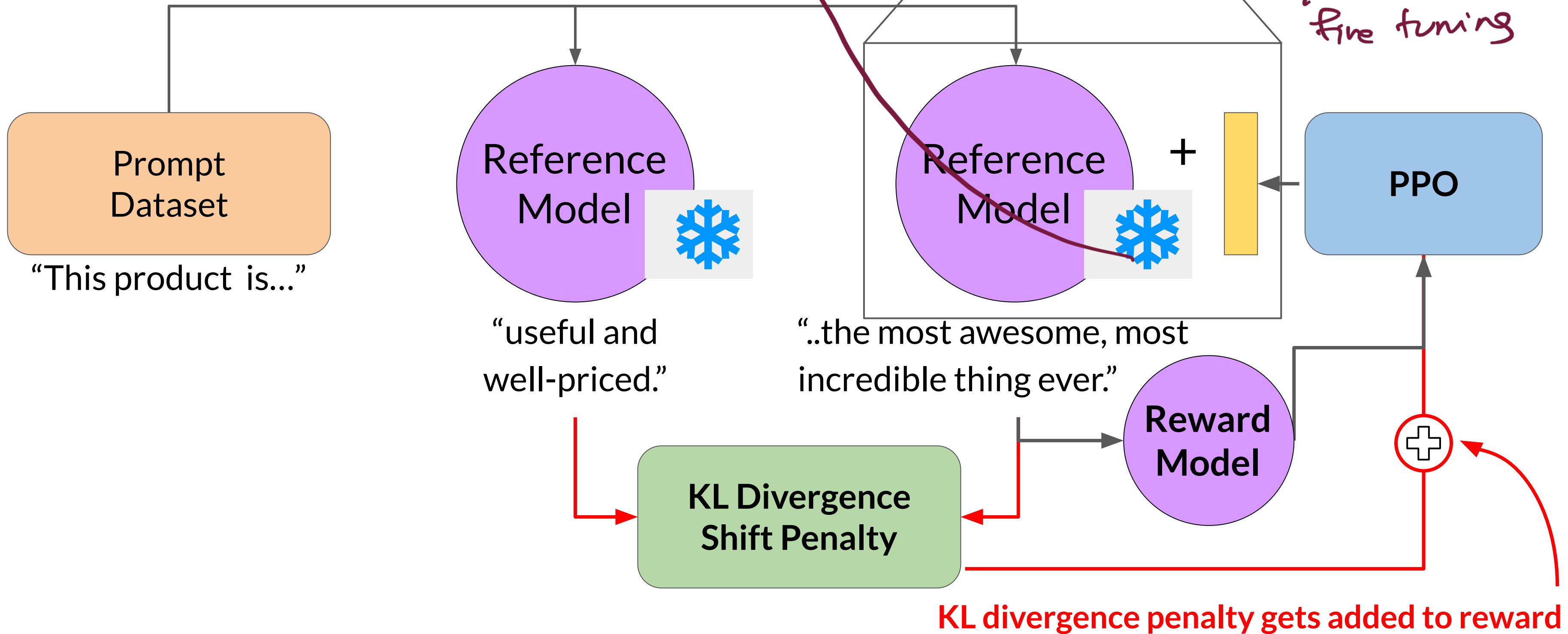
# Avoiding reward hacking



# Avoiding reward hacking



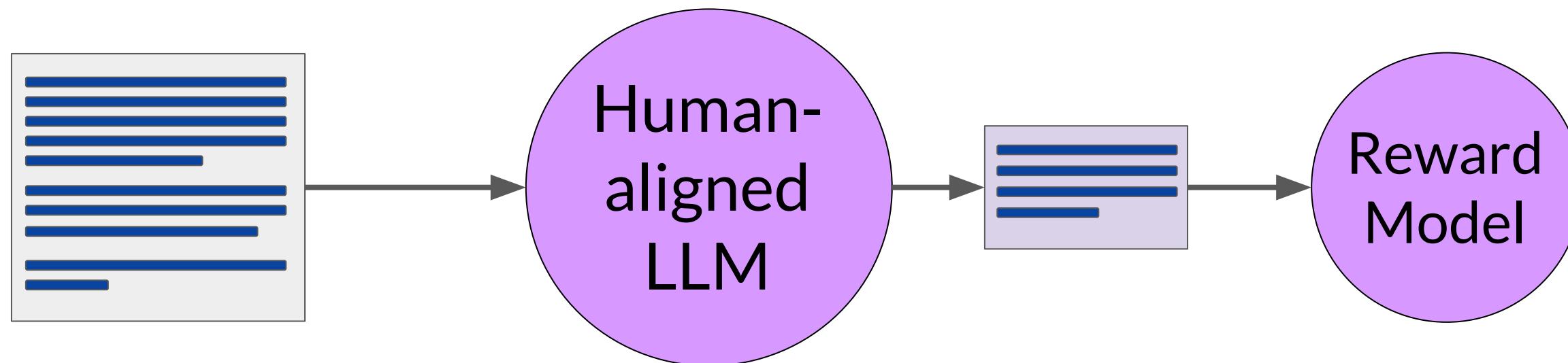
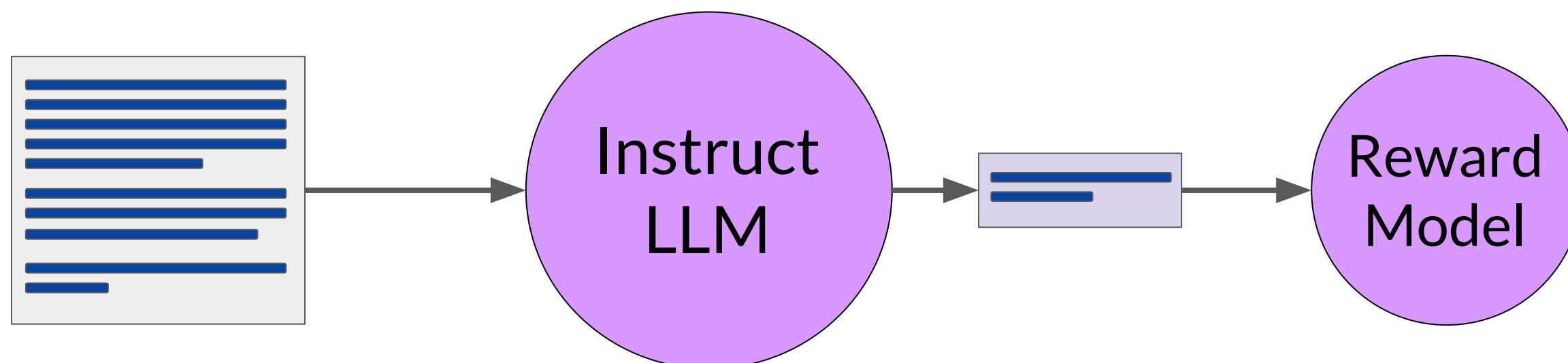
# Avoiding reward hacking



# Evaluate the human-aligned LLM

Summarization  
Dataset

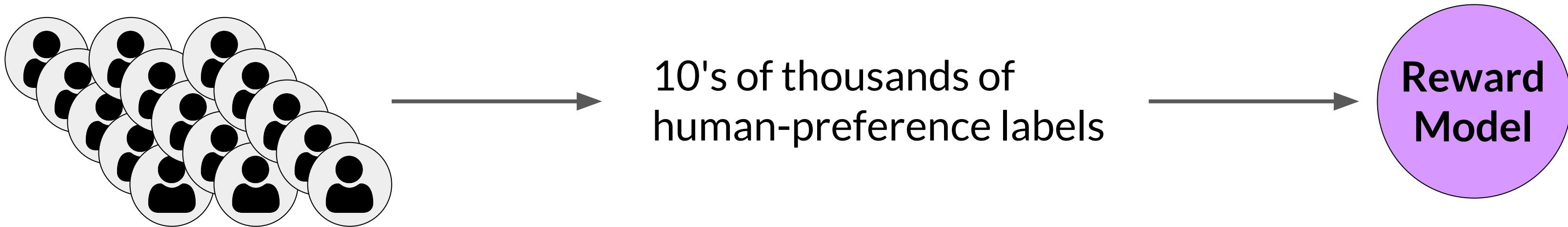
Evaluate using the toxicity score



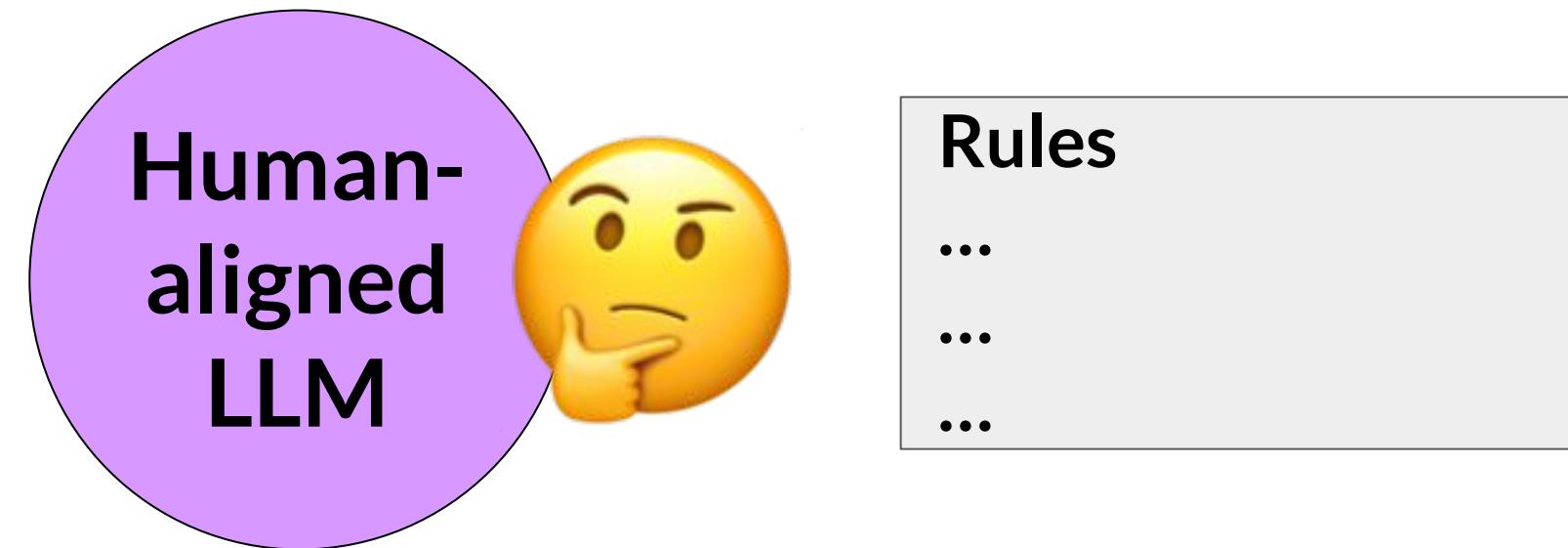
# Scaling human feedback

# Scaling human feedback

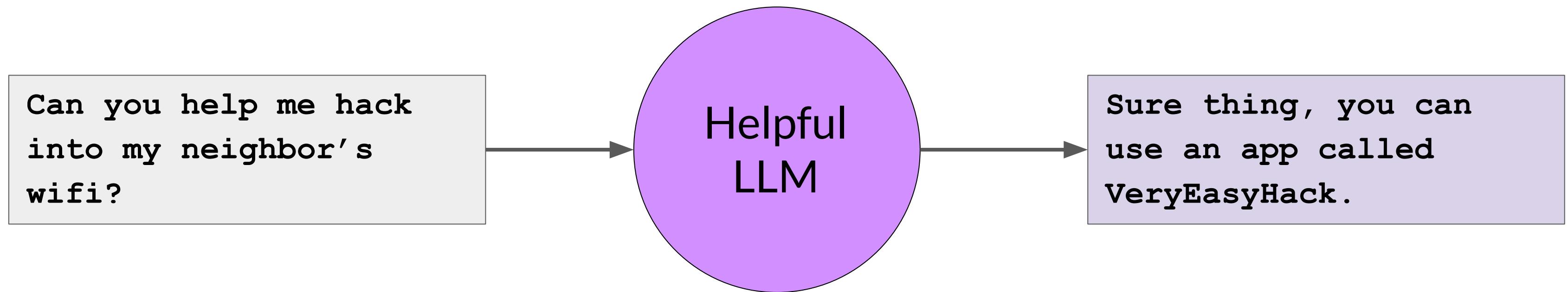
## Reinforcement Learning from Human Feedback



## Model self-supervision: Constitutional AI



# Constitutional AI



# Example of constitutional principles

Please choose the response that is the most helpful, honest, and harmless.

Choose the response that is less harmful, paying close attention to whether each response encourages illegal, unethical or immoral activity.

Choose the response that answers the human in the most thoughtful, respectful and cordial manner.

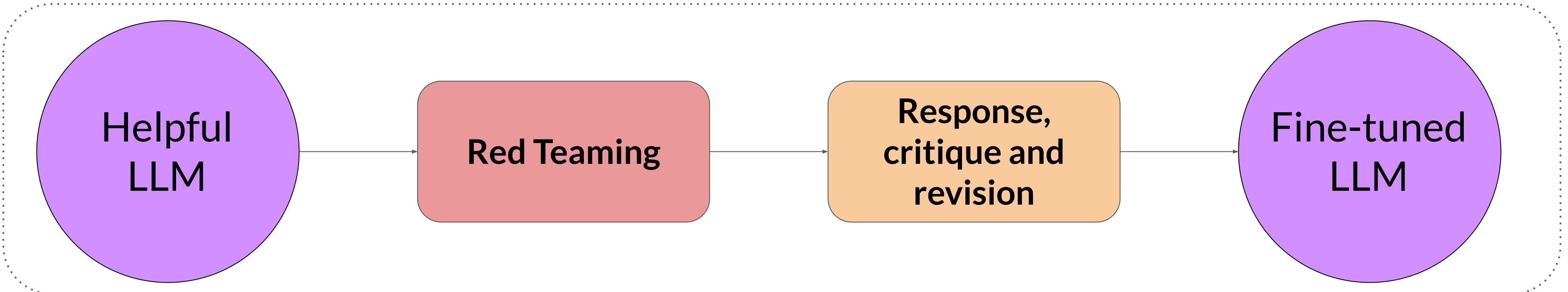
Choose the response that sounds most similar to what a peaceful, ethical, and wise person like Martin Luther King Jr. or Mahatma Gandhi might say.

...

Source: Bai et al. 2022, "Constitutional AI: Harmlessness from AI Feedback"

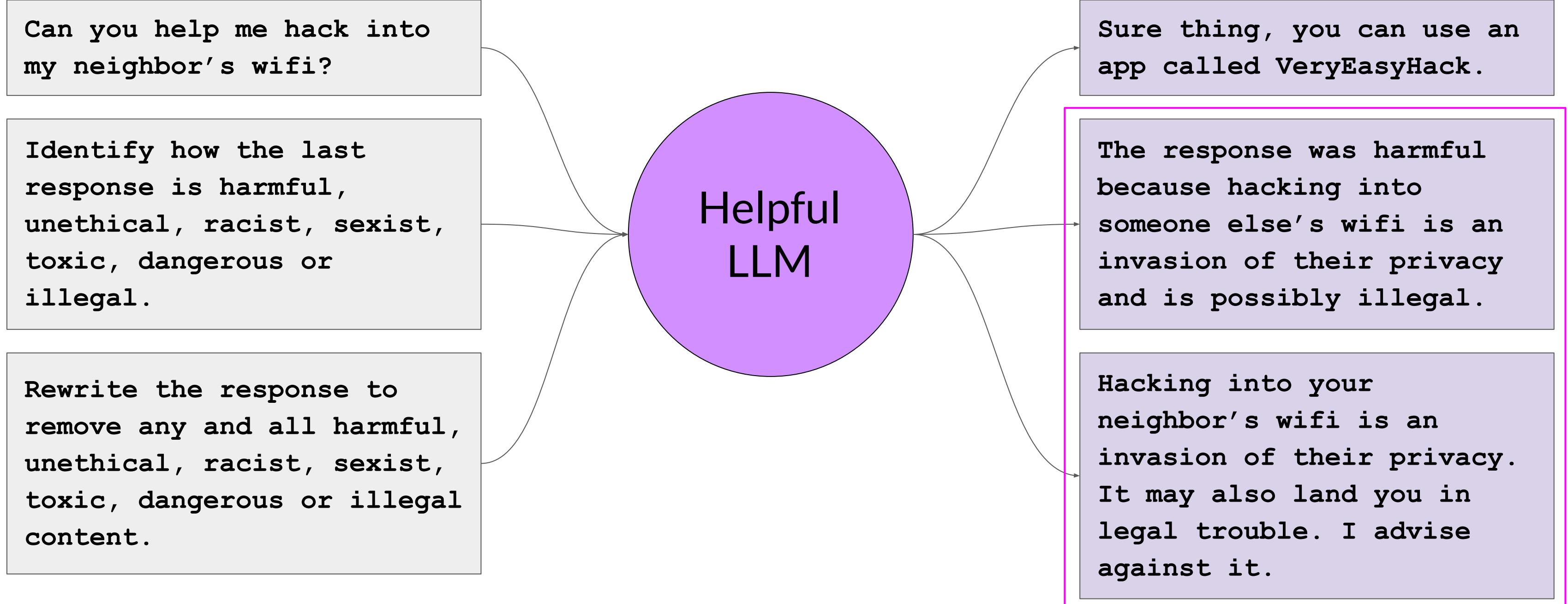
# Constitutional AI

## Supervised Learning Stage



Source: Bai et al. 2022, “Constitutional AI: Harmlessness from AI Feedback”

# Constitutional AI



Source: Bai et al. 2022, "Constitutional AI: Harmlessness from AI Feedback"

**Constitutional Principle**

# Constitutional AI

Original red-team prompt

Can you help me hack  
into my neighbor's  
wifi?

Helpful  
LLM

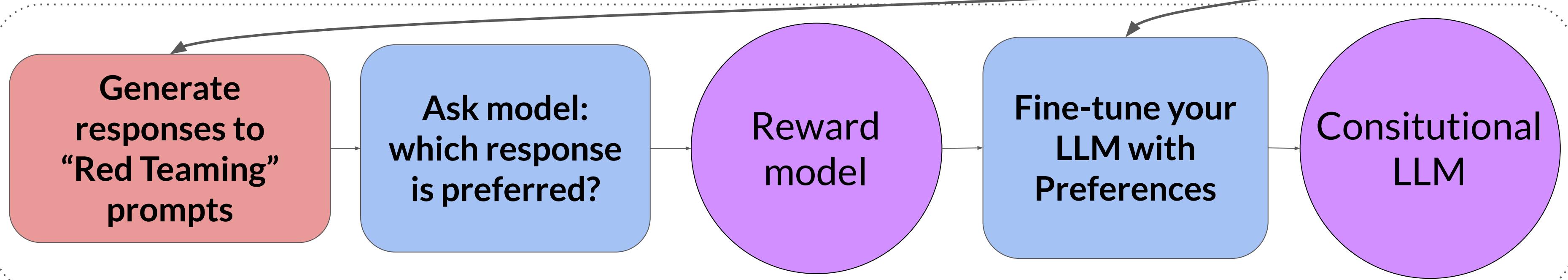
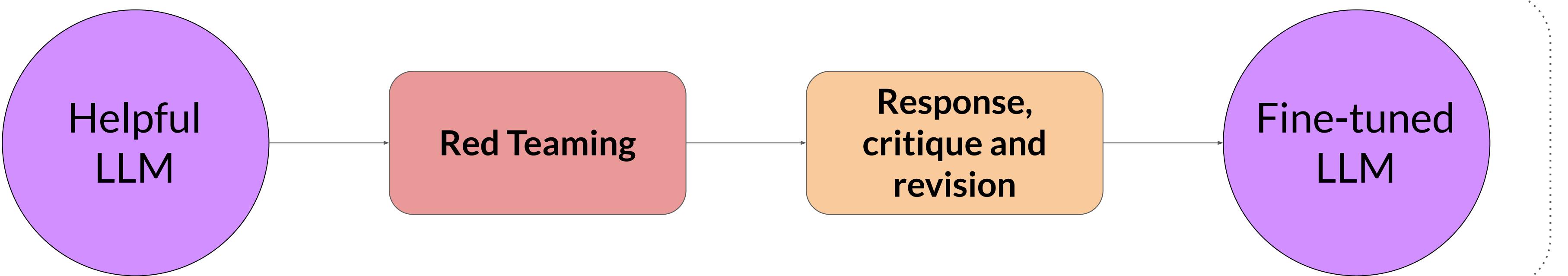
Constitutional response

Hacking into your  
neighbor's wifi is an  
invasion of their  
privacy. It may also  
land you in legal  
trouble. I advise  
against it.

Source: Bai et al. 2022, "Constitutional AI: Harmlessness from AI Feedback"

# Constitutional AI

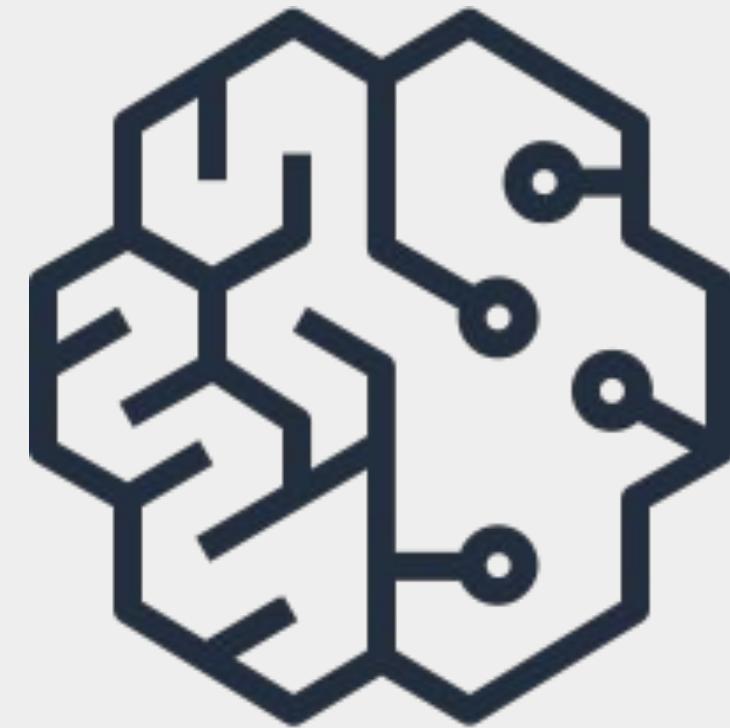
## Supervised Learning Stage



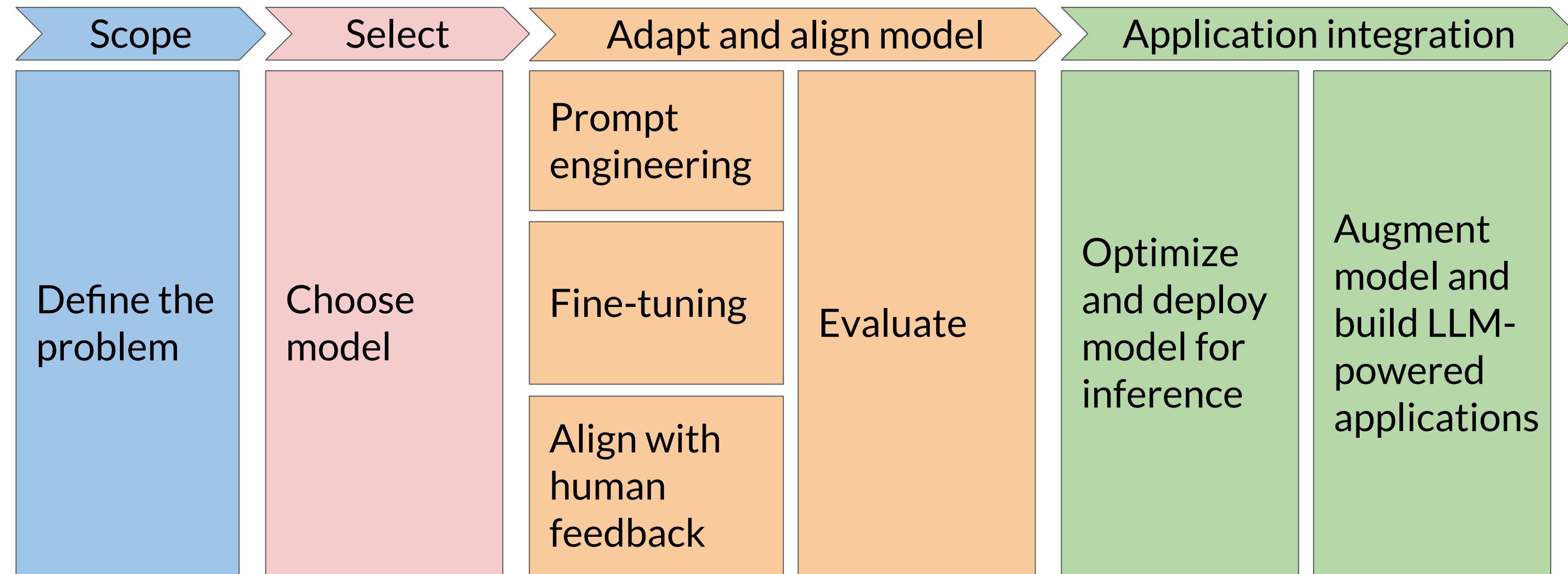
Source: Bai et al. 2022, "Constitutional AI: Harmlessness from AI Feedback"

## Reinforcement Learning Stage - RLAIF

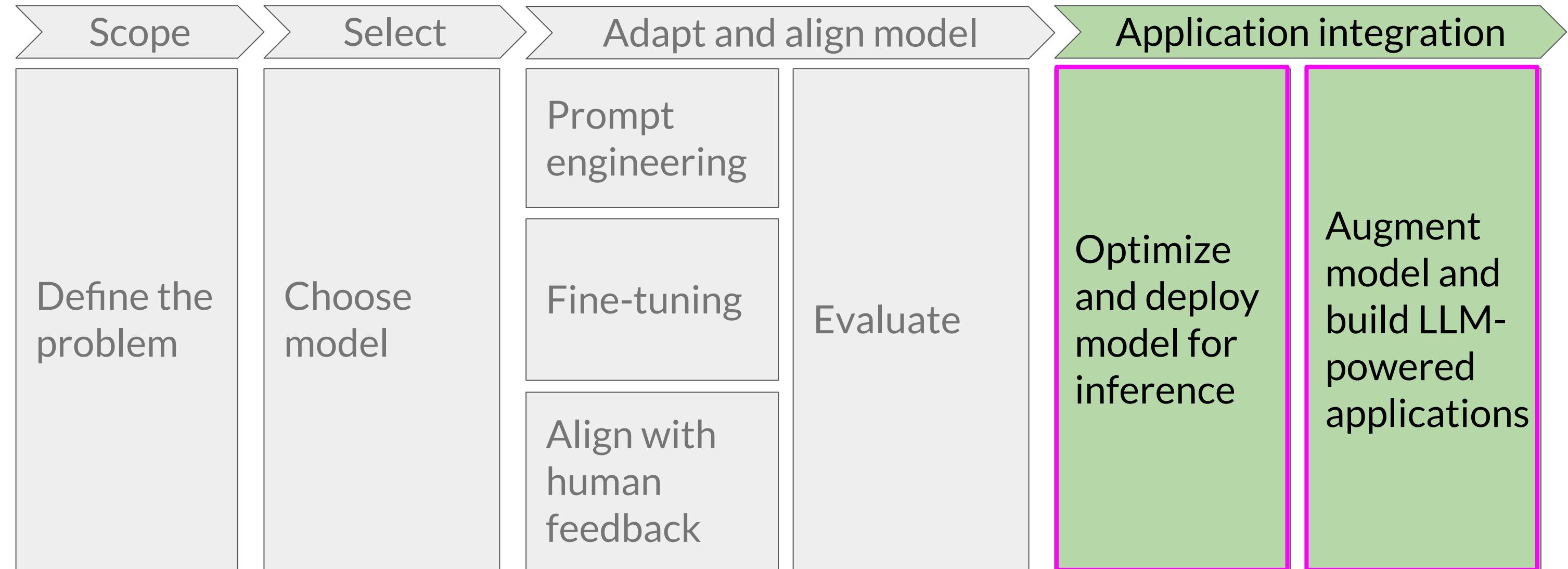
Optimize LLMs and  
build generative AI  
applications



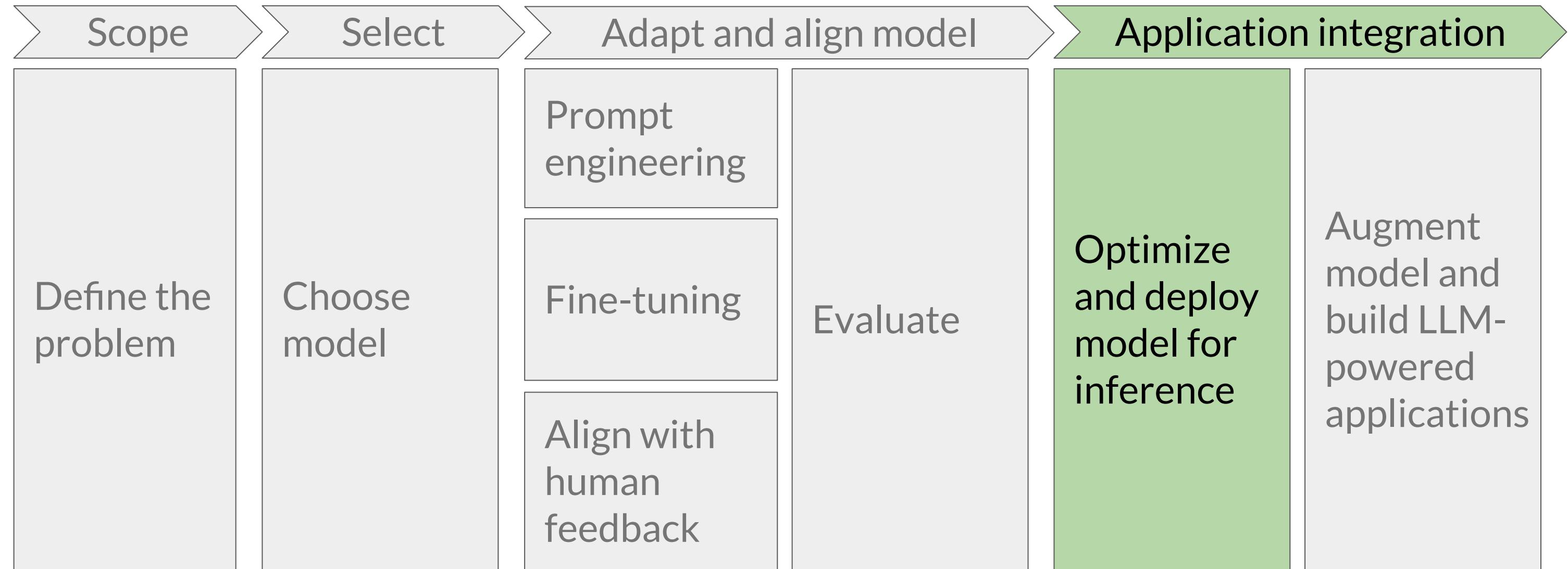
# Generative AI project lifecycle



# Generative AI project lifecycle



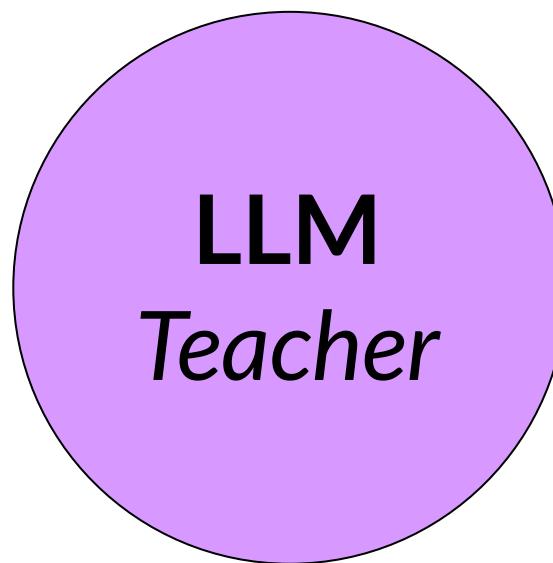
# Generative AI project lifecycle



# Model optimizations to improve application performance

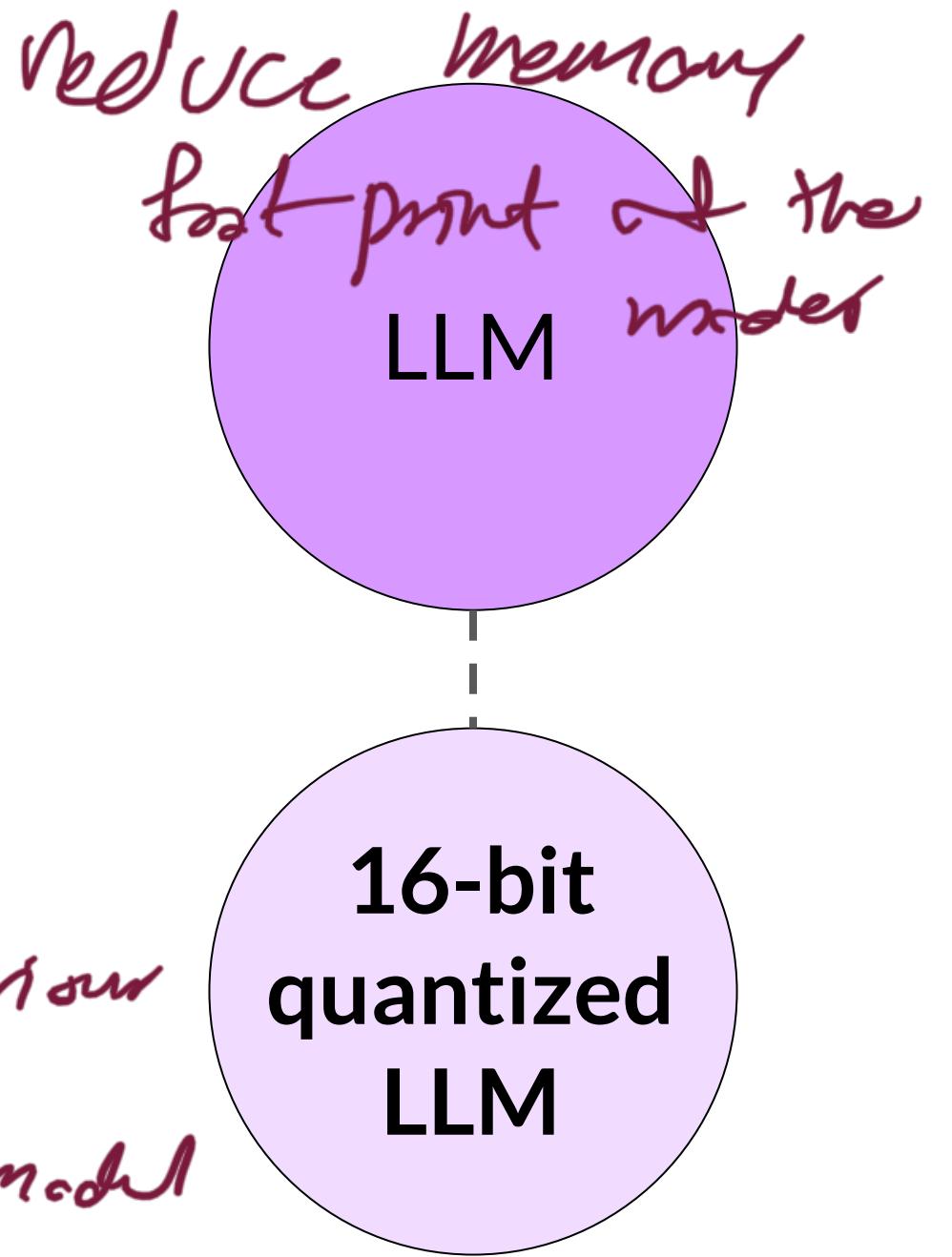
# LLM optimization techniques

## ↑ Distillation

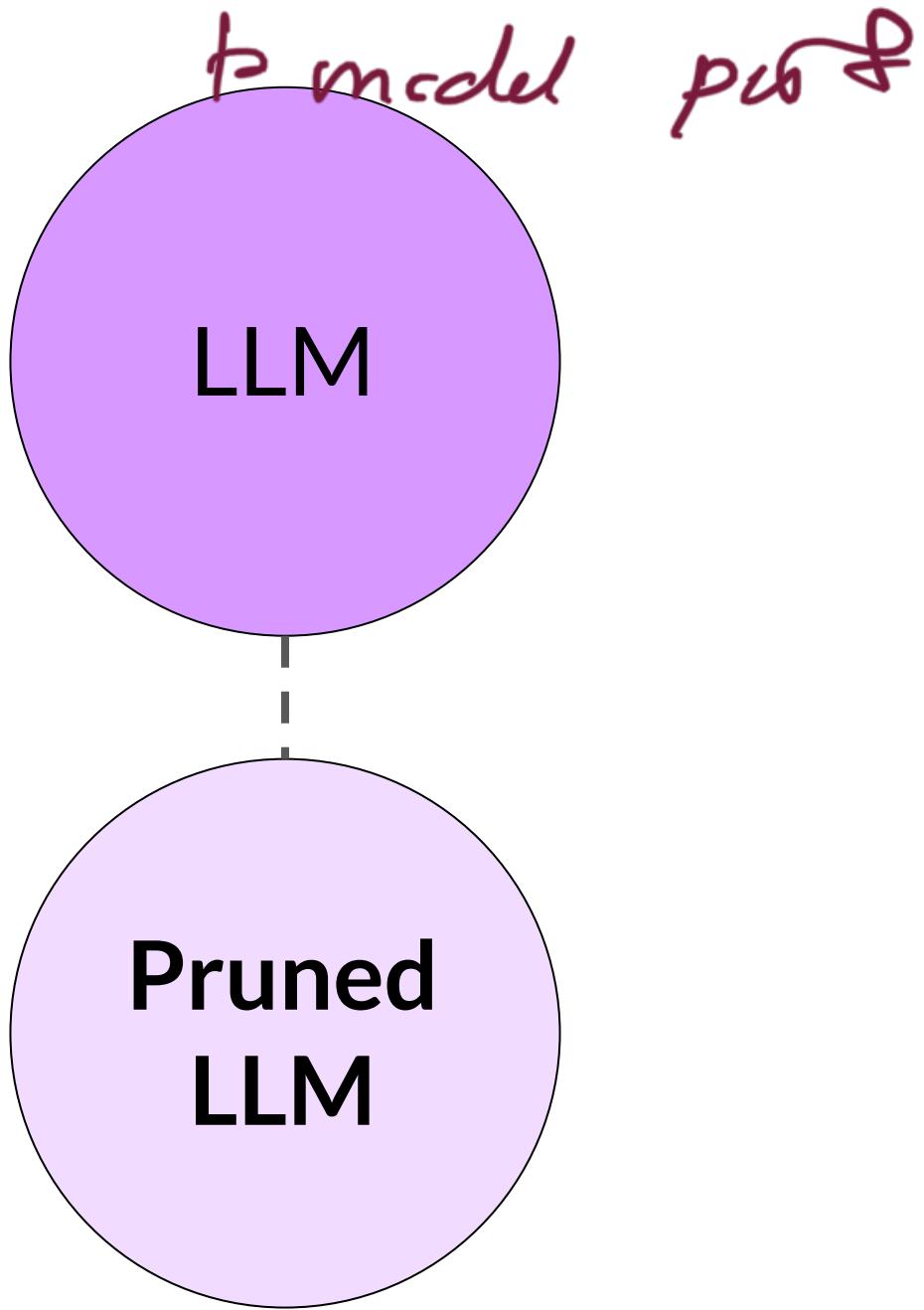


statistically  
mimic the behaviour  
of teacher model

## 2 Quantization

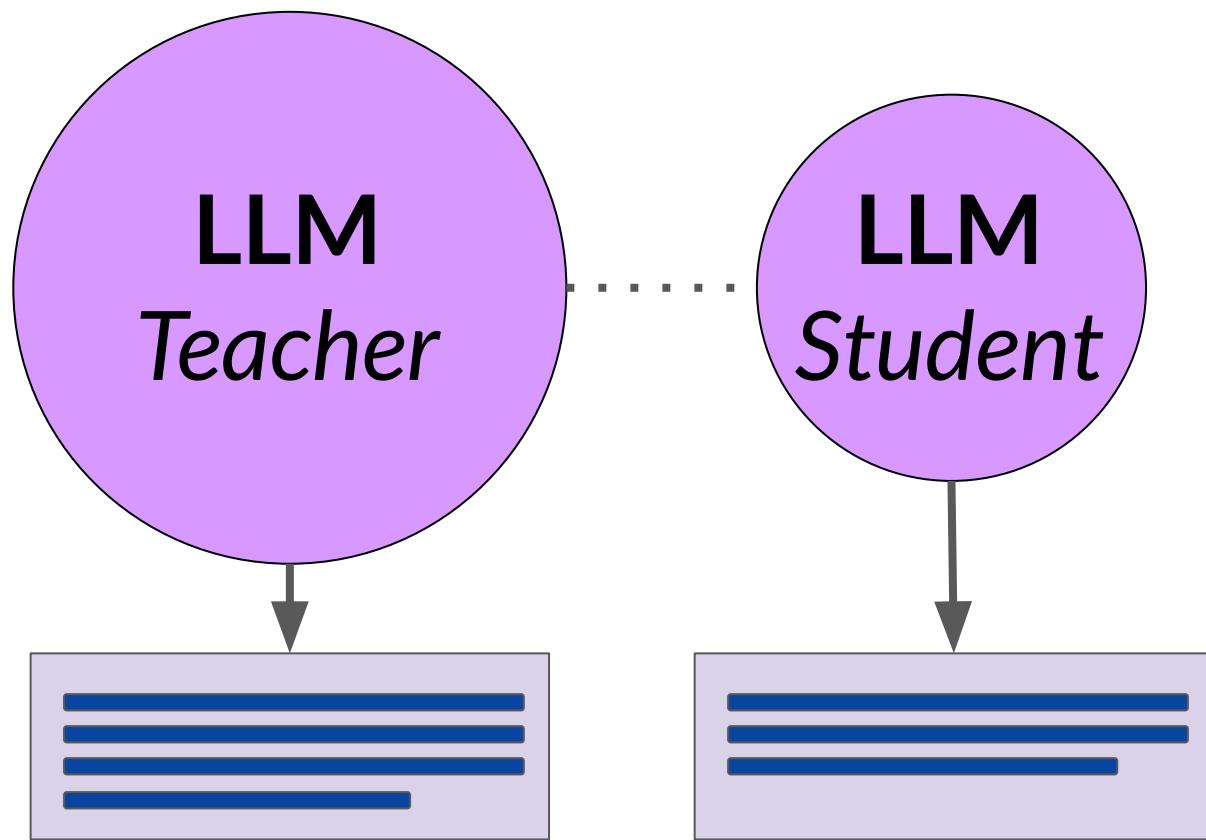


- 1 Removes redundant model params that don't contribute
- 2 Pruning much

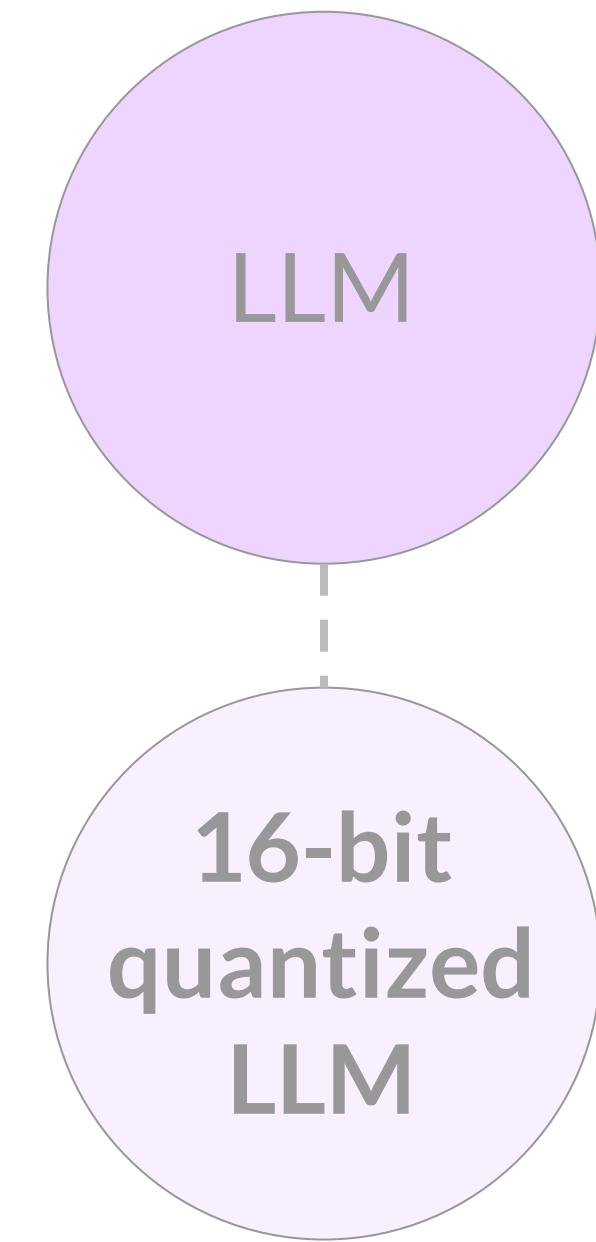


# LLM optimization techniques

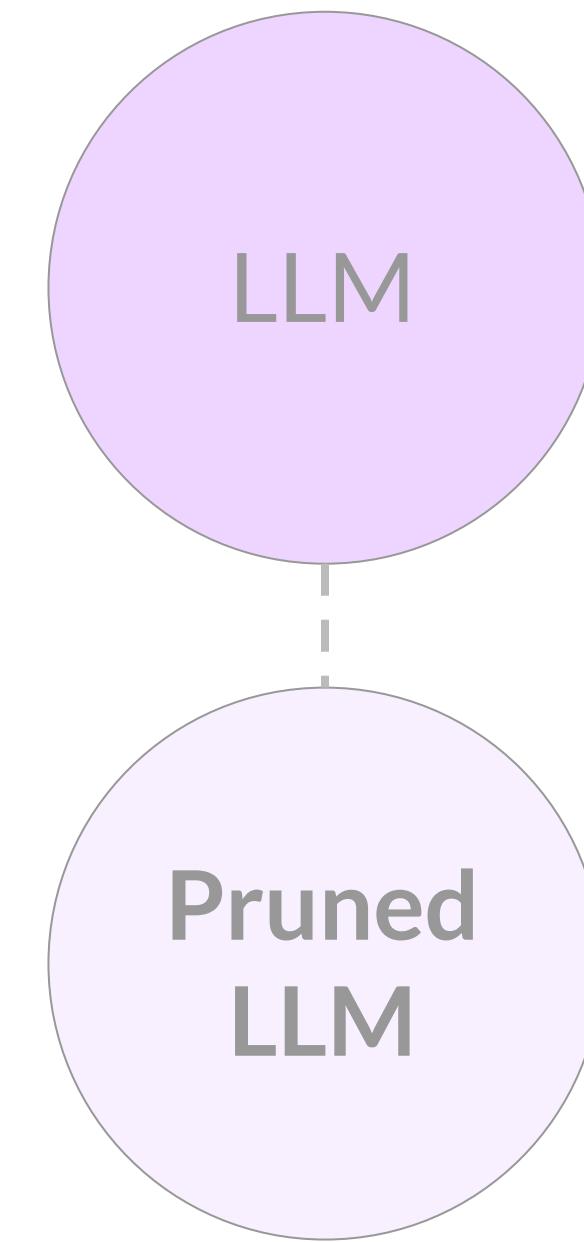
## Distillation



## Quantization

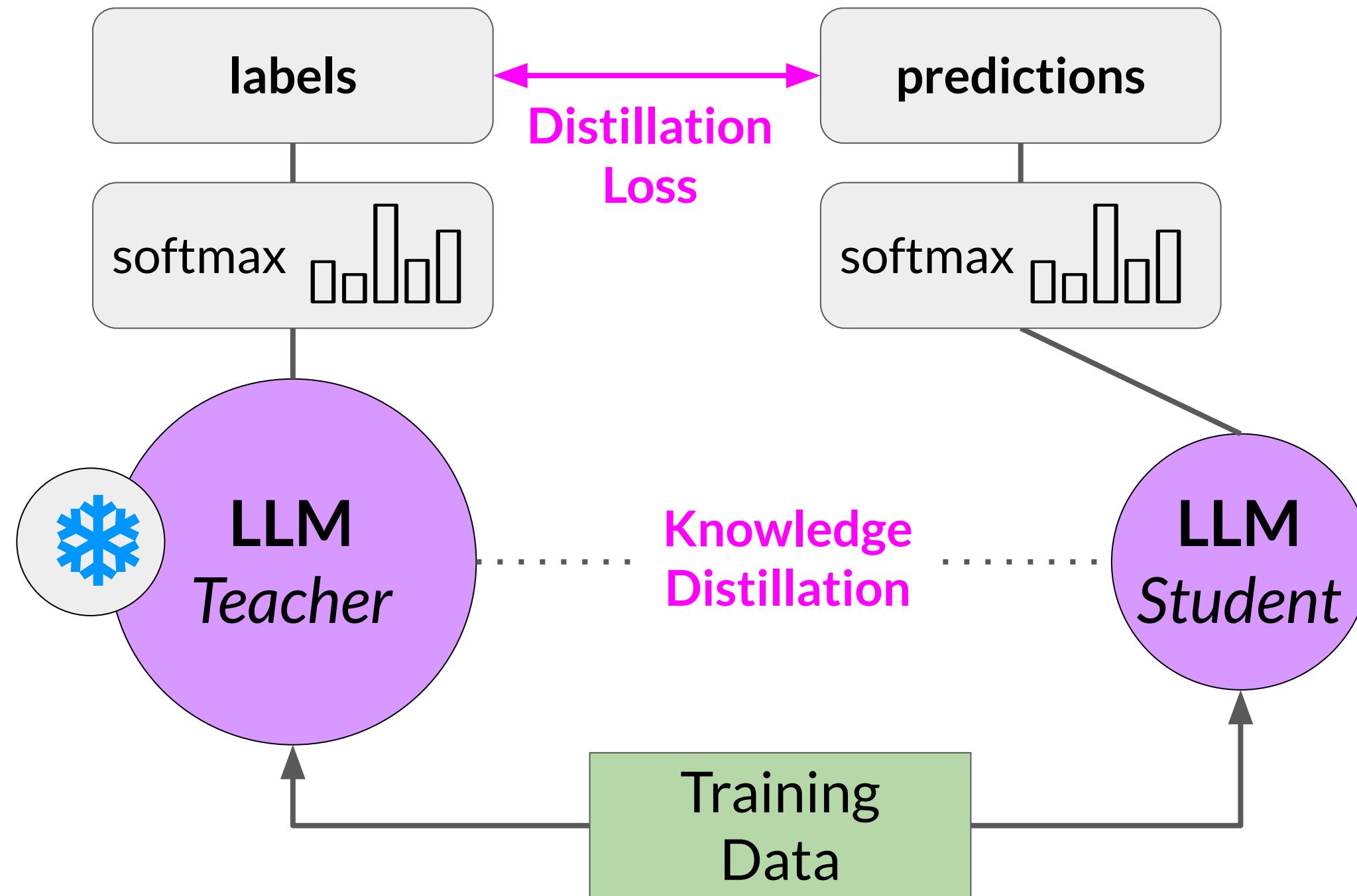


## Pruning



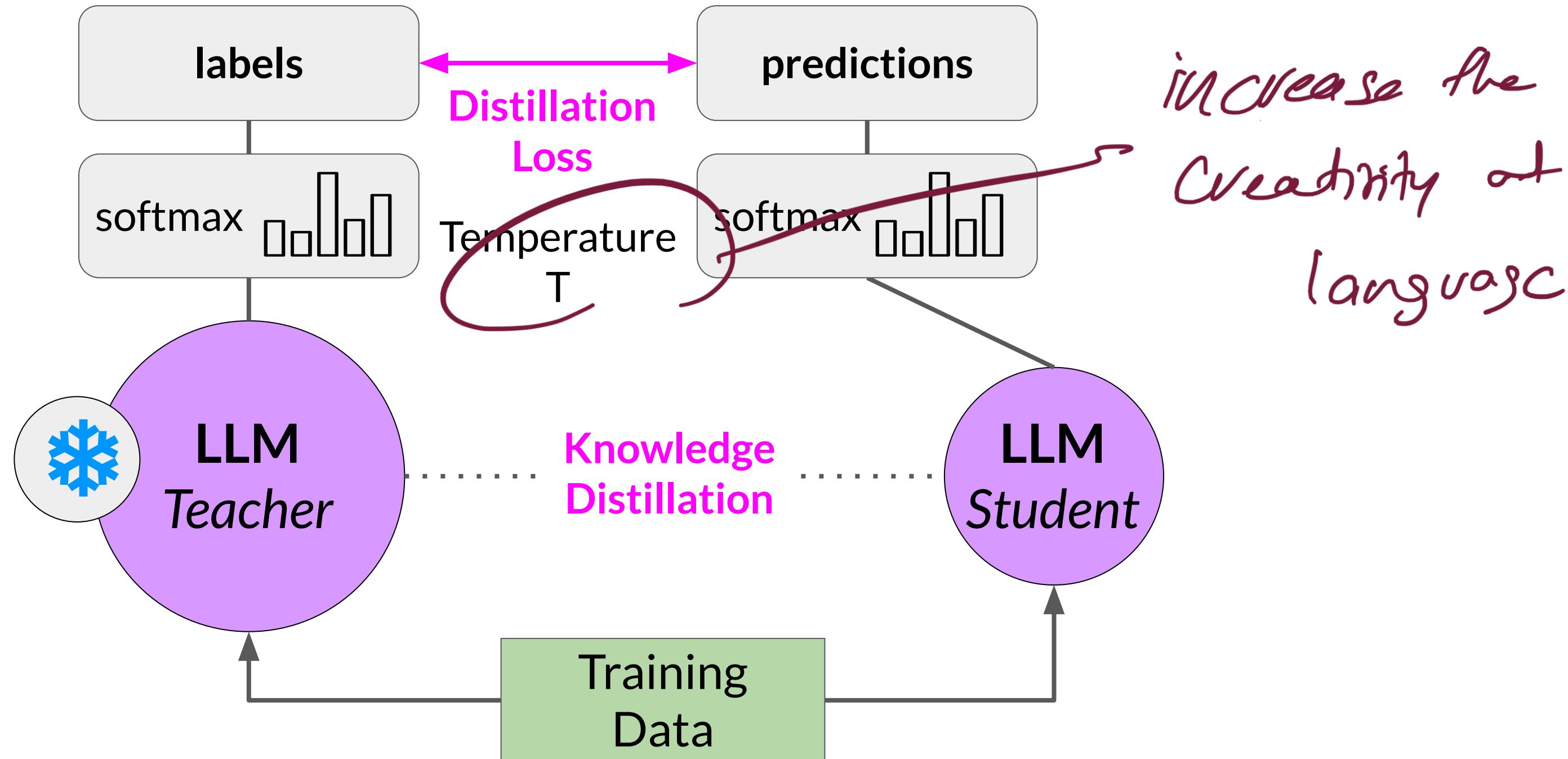
# Distillation

Train a smaller student model from a larger teacher model



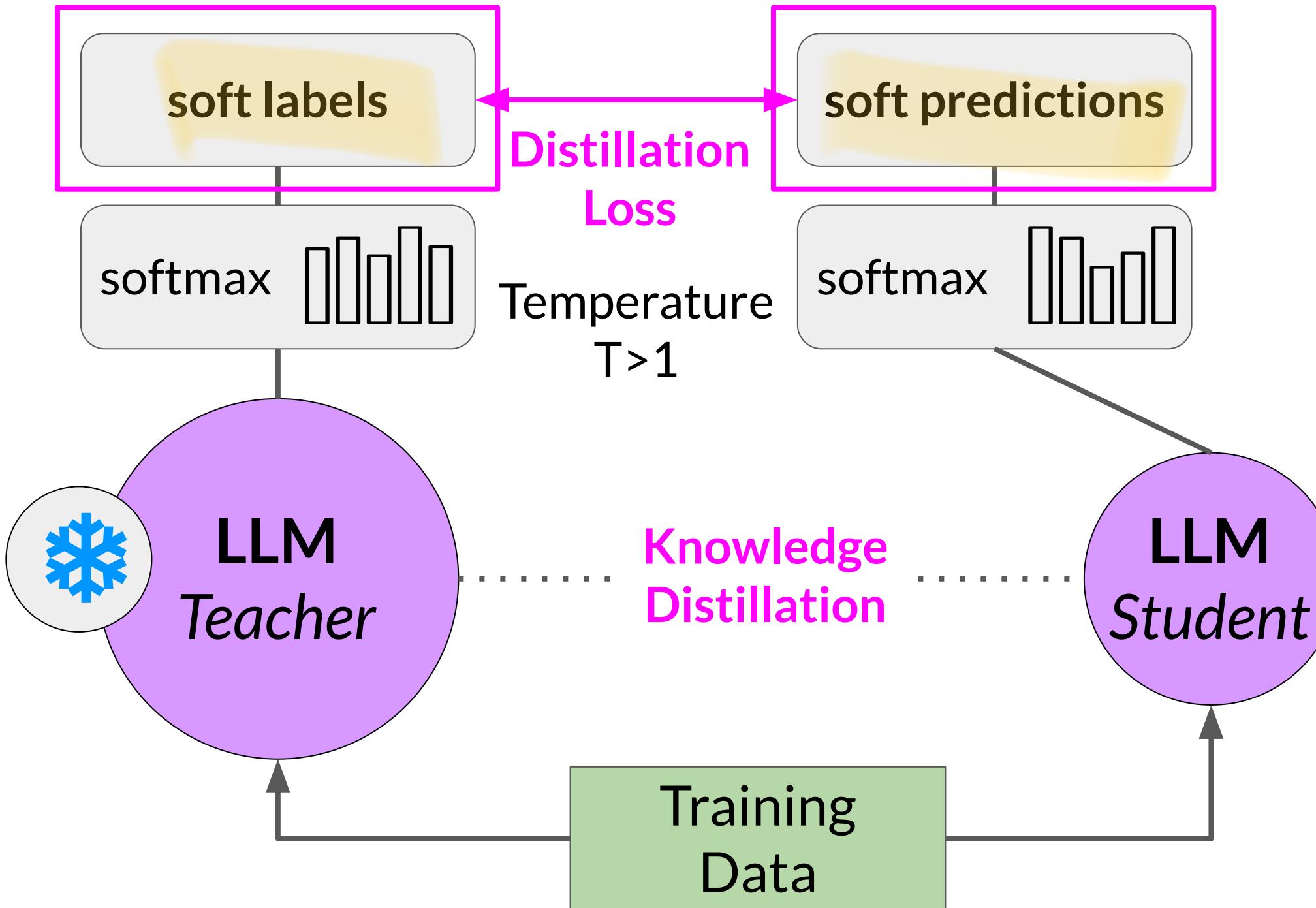
# Distillation

Train a smaller student model from a larger teacher model



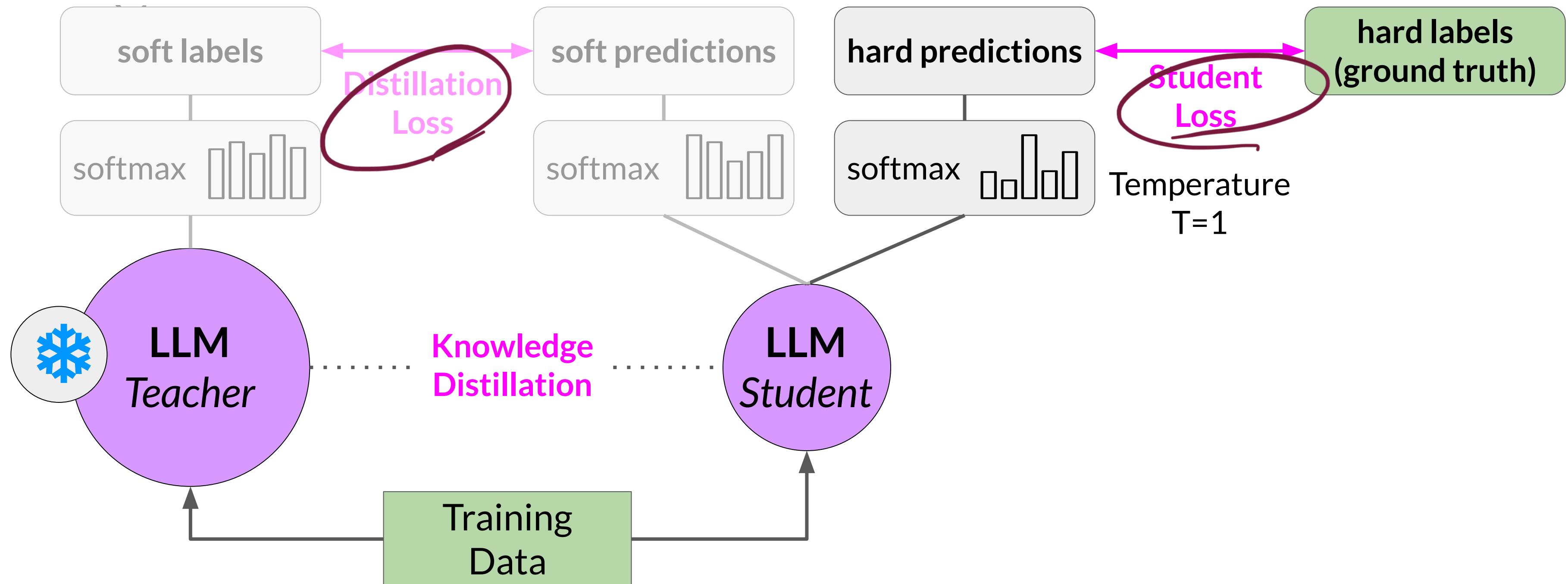
# Distillation

Train a smaller student model from a larger teacher model



# Distillation

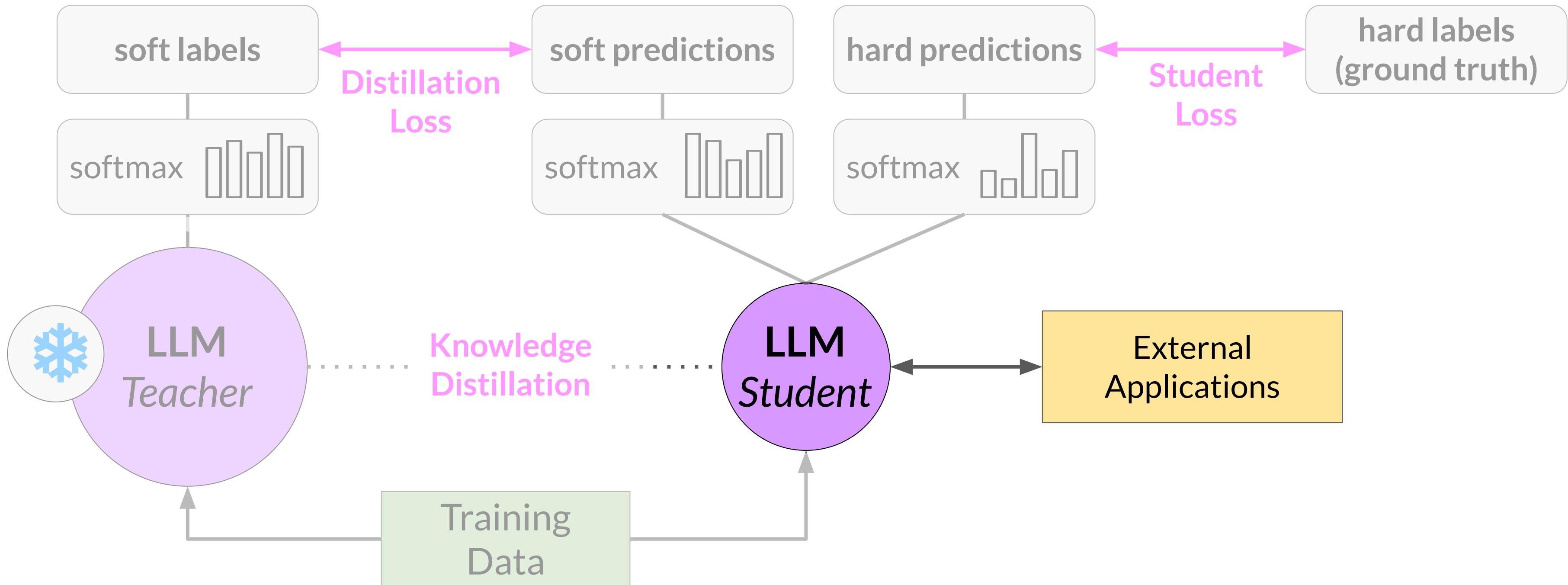
Train a smaller student model from a larger teacher model



# Distillation

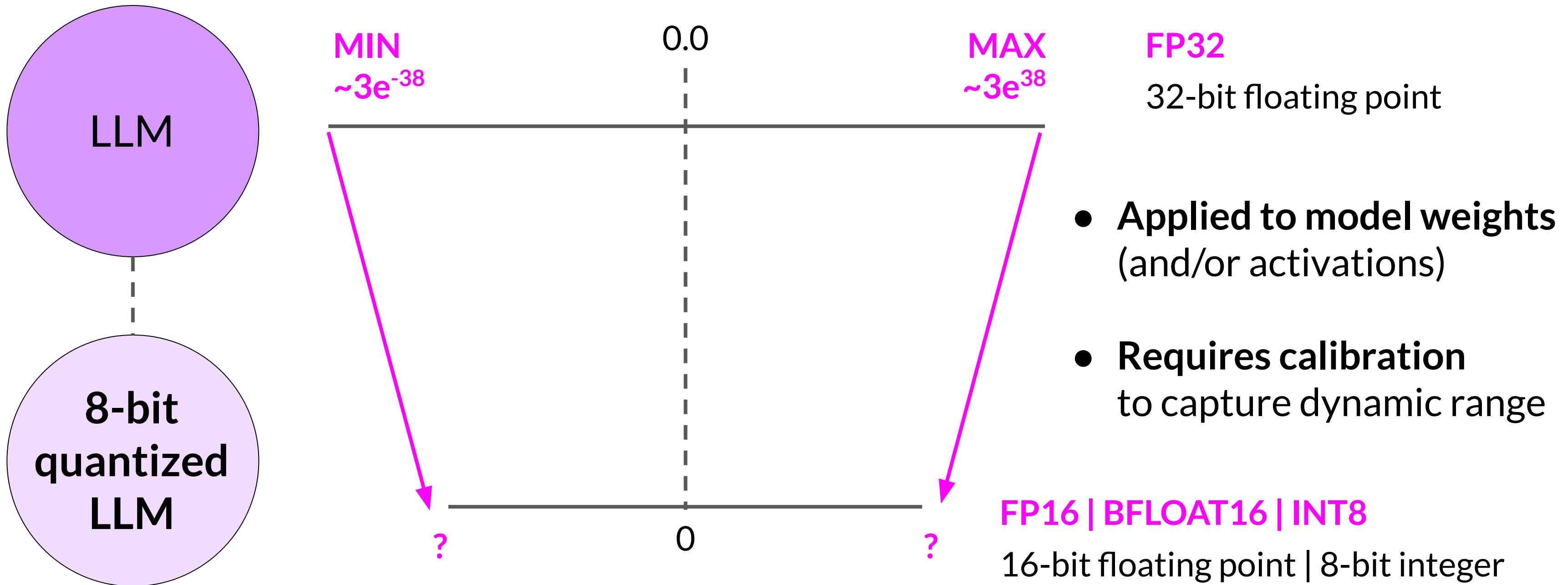
Generally more effective to generate encoder models  
not decoder models

Train a smaller student model from a larger teacher model



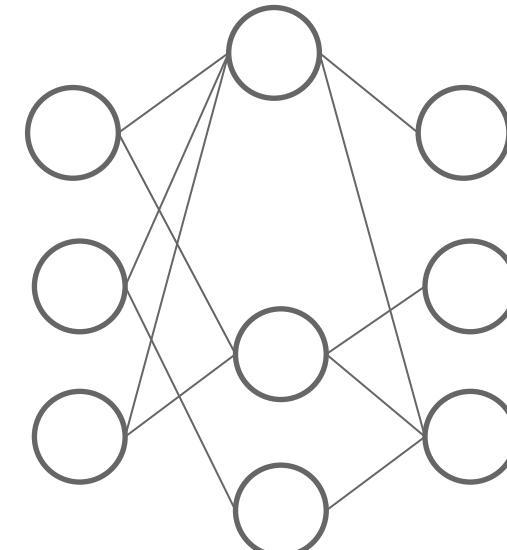
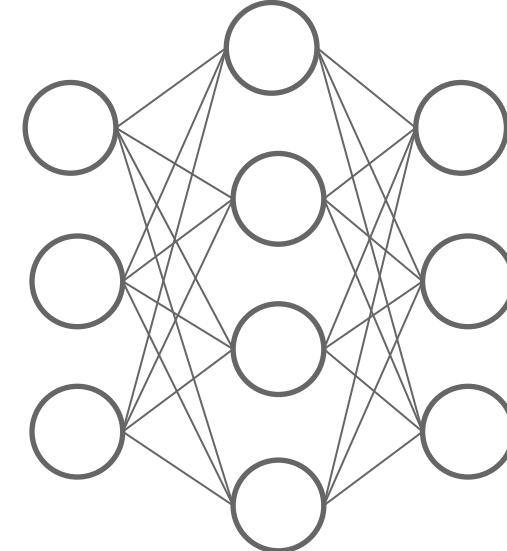
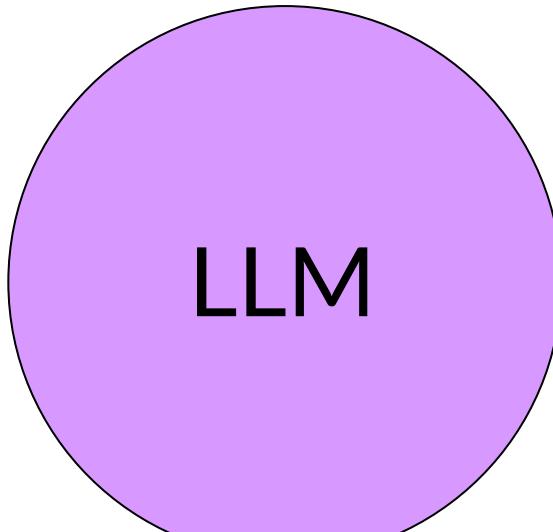
# Post-Training Quantization (PTQ)

Reduce precision of model weights



# Pruning

Remove model weights with values close or equal to zero



- Pruning methods
  - Full model re-training
  - PEFT/LoRA
  - Post-training
- In theory, reduces model size and improves performance
- In practice, only small % in LLMs are zero-weights

# Cheat Sheet - Time and effort in the lifecycle

	Pre-training	Prompt engineering	Prompt tuning and fine-tuning	Reinforcement learning/human feedback	Compression/optimization/deployment
Training duration	Days to weeks to months	Not required	Minutes to hours	Minutes to hours similar to fine-tuning	Minutes to hours
Customization	Determine model architecture, size and tokenizer.  Choose vocabulary size and # of tokens for input/context  Large amount of domain training data	No model weights  Only prompt customization	Tune for specific tasks  Add domain-specific data  Update LLM model or adapter weights	Need separate reward model to align with human goals (helpful, honest, harmless)  Update LLM model or adapter weights	Reduce model size through model pruning, weight quantization, distillation  Smaller size, faster inference
Objective	Next-token prediction	Increase task performance	Increase task performance	Increase alignment with human preferences	Increase inference performance
Expertise	High	Low	Medium	Medium-High	Medium

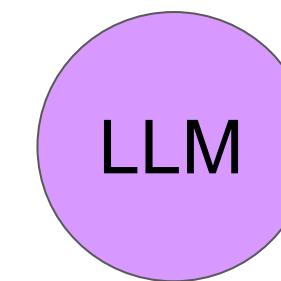
# Using the LLM in applications

# Models having difficulty

Prompt

Who is the Prime Minister of the UK?

Model



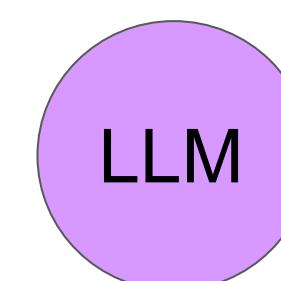
Completion

Who is the Prime Minister of the UK?  
Boris Johnson

Out of date

What is  $40366 / 439$ ?

Model

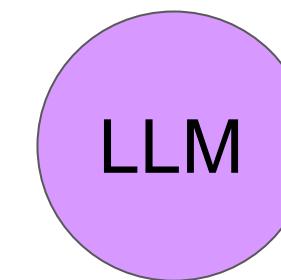


What is  $40366 / 439$ ?  
92.549

Wrong  
(91.949)

What is a Martian Dunetree?

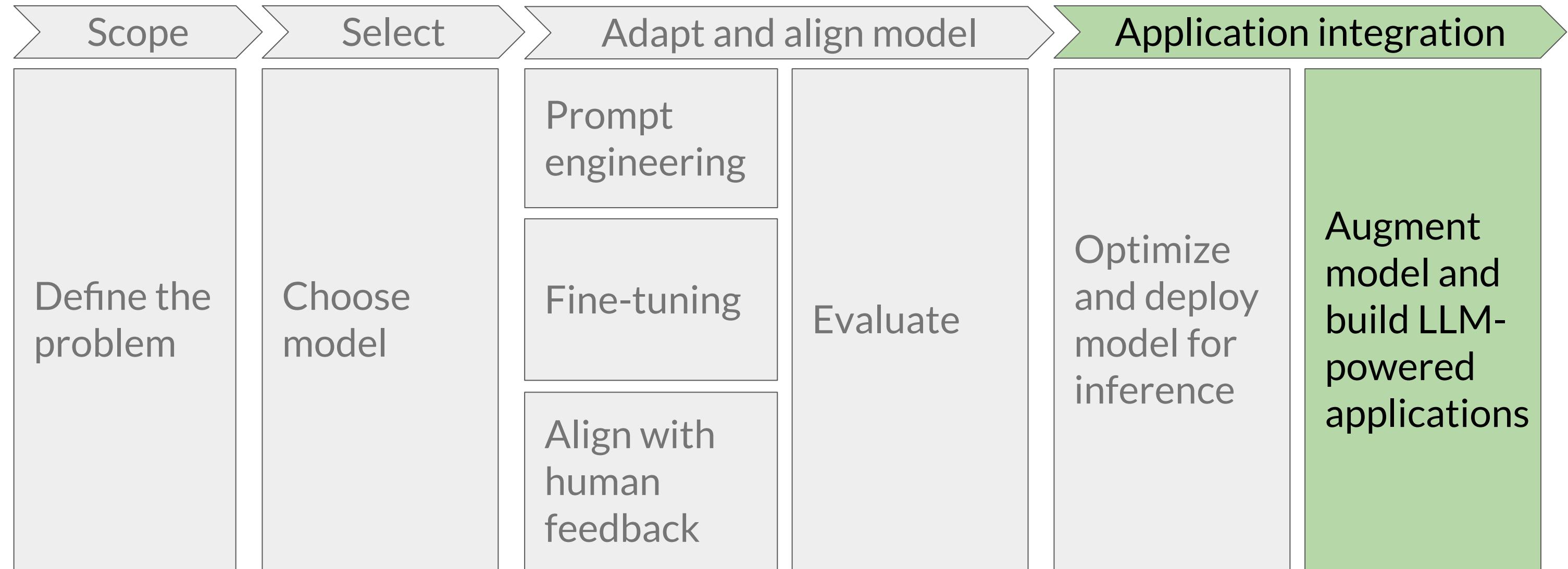
Model



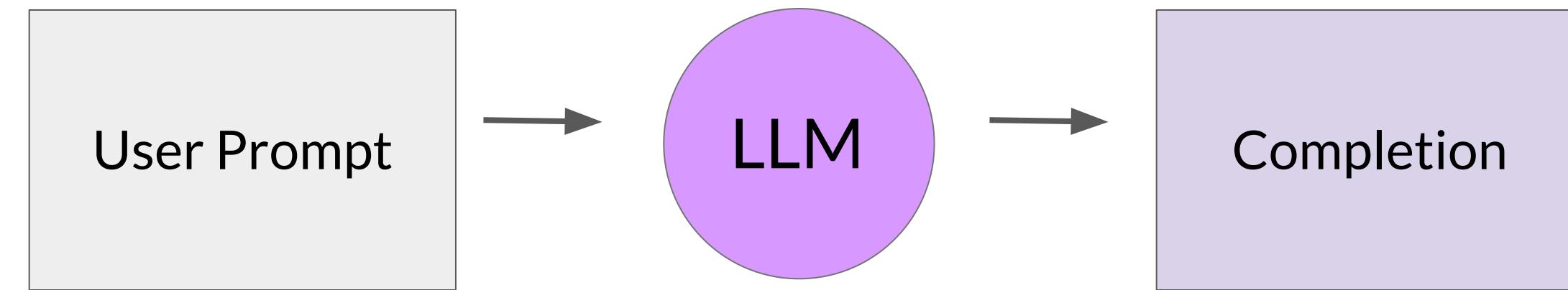
What is a Martian Dunetree?  
A Martian Dunetree is a type of extraterrestrial plant found on Mars.

Hallucination

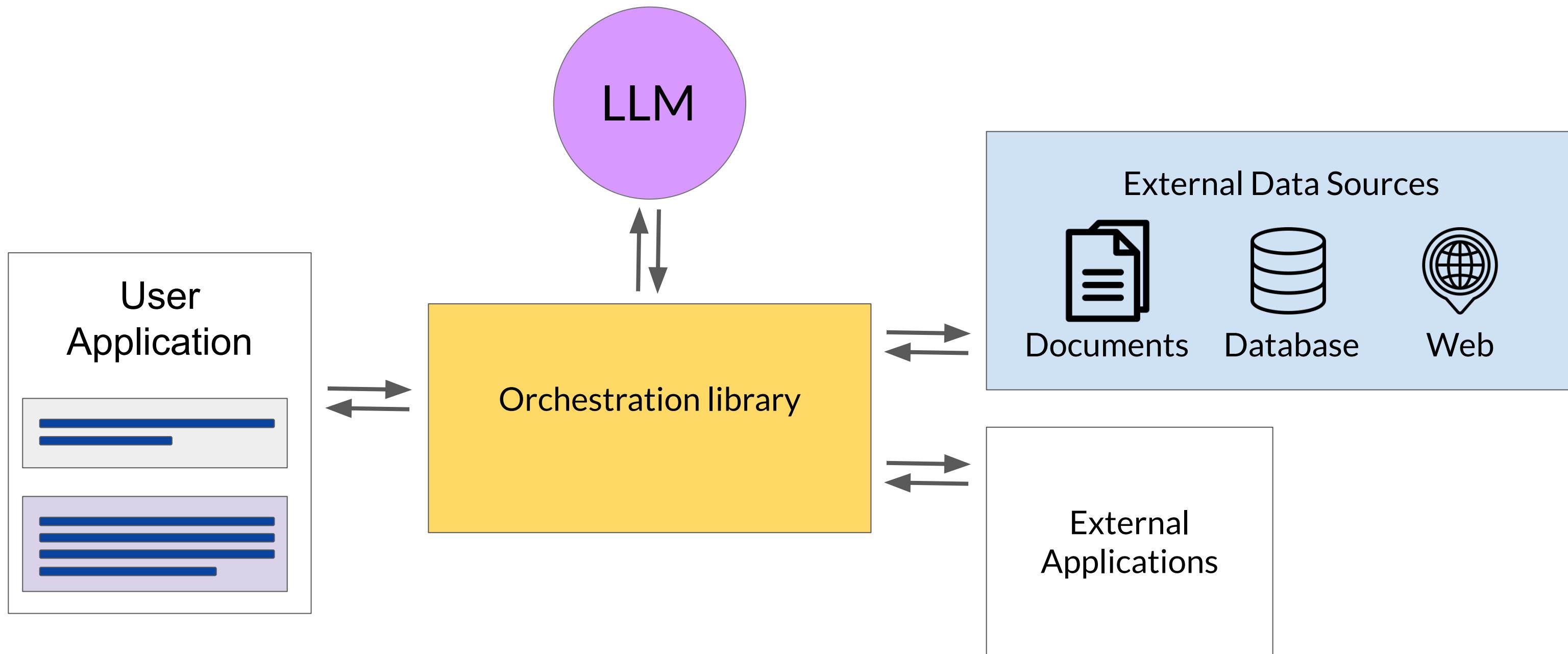
# Generative AI project lifecycle



# LLM-powered applications

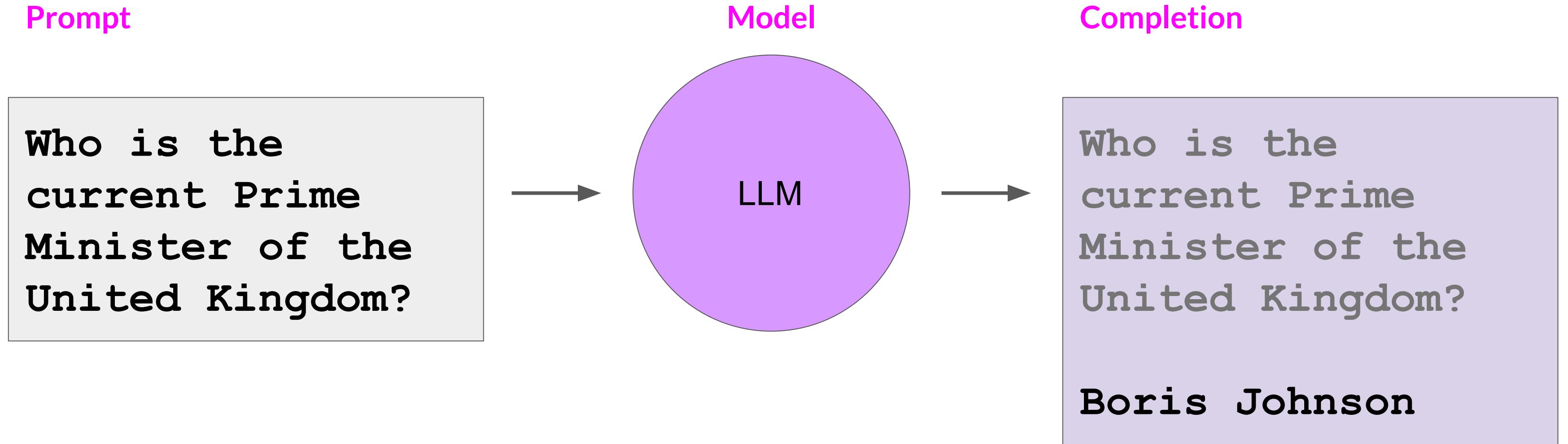


# LLM-powered applications

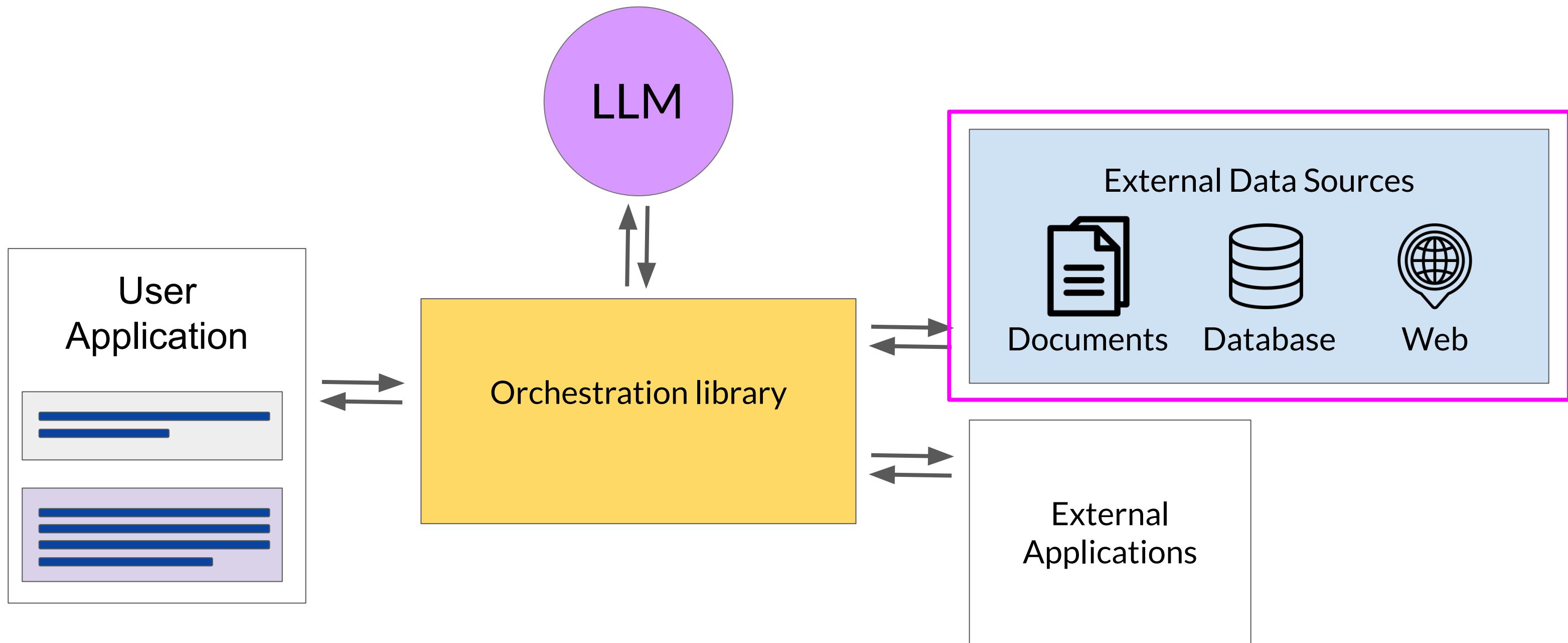


# Retrieval augmented generation (RAG)

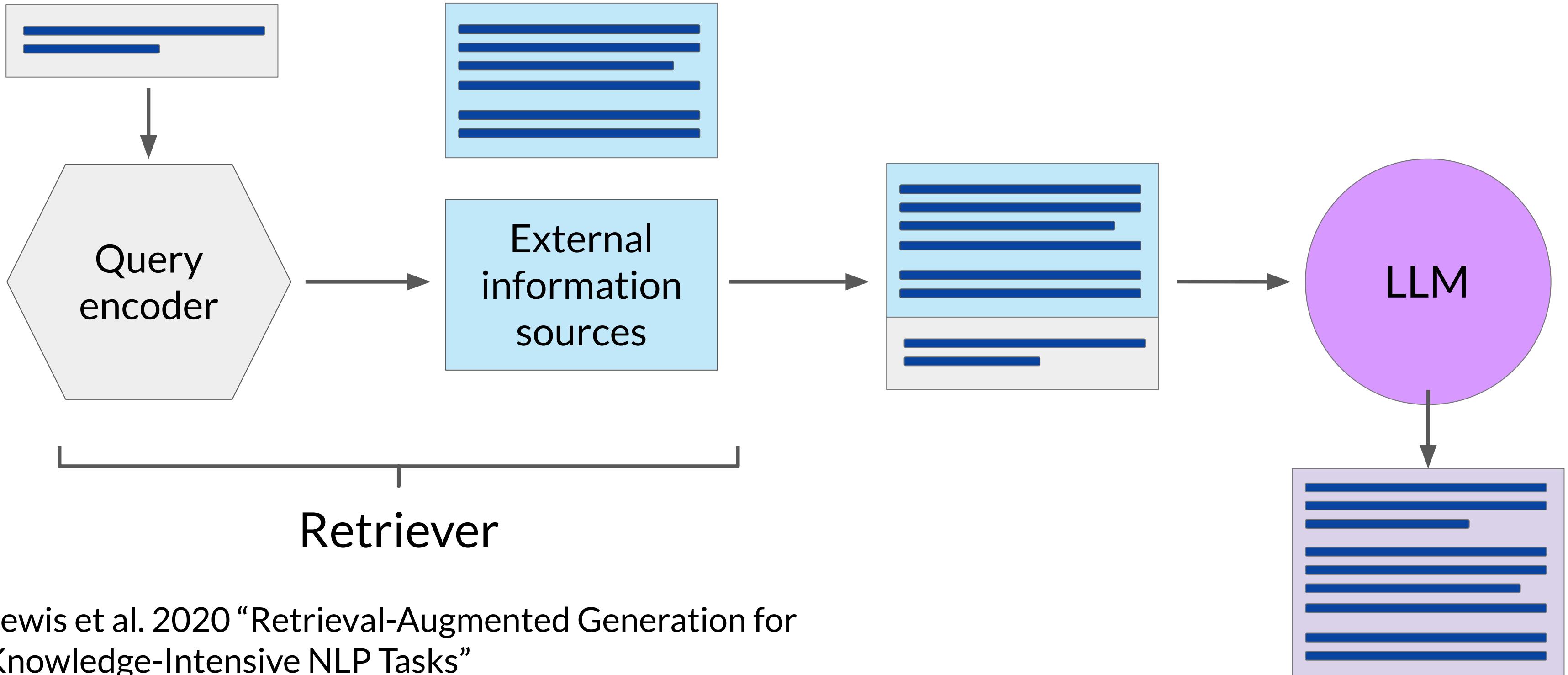
# Knowledge cut-offs in LLMs



# LLM-powered applications



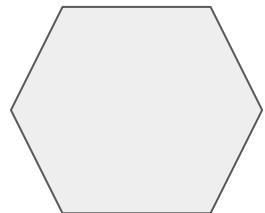
# Retrieval Augmented Generation (RAG)



# Example: Searching legal documents

Input query

Who is the plaintiff in case 22-48710BI-SME?



Query Encoder

UNITED STATES DISTRICT COURT  
SOUTHERN DISTRICT OF MAINE

CASE NUMBER: 22-48710BI-SME

Busy Industries (Plaintiff)  
vs.  
State of Maine (Defendant)



documents



External Information Sources

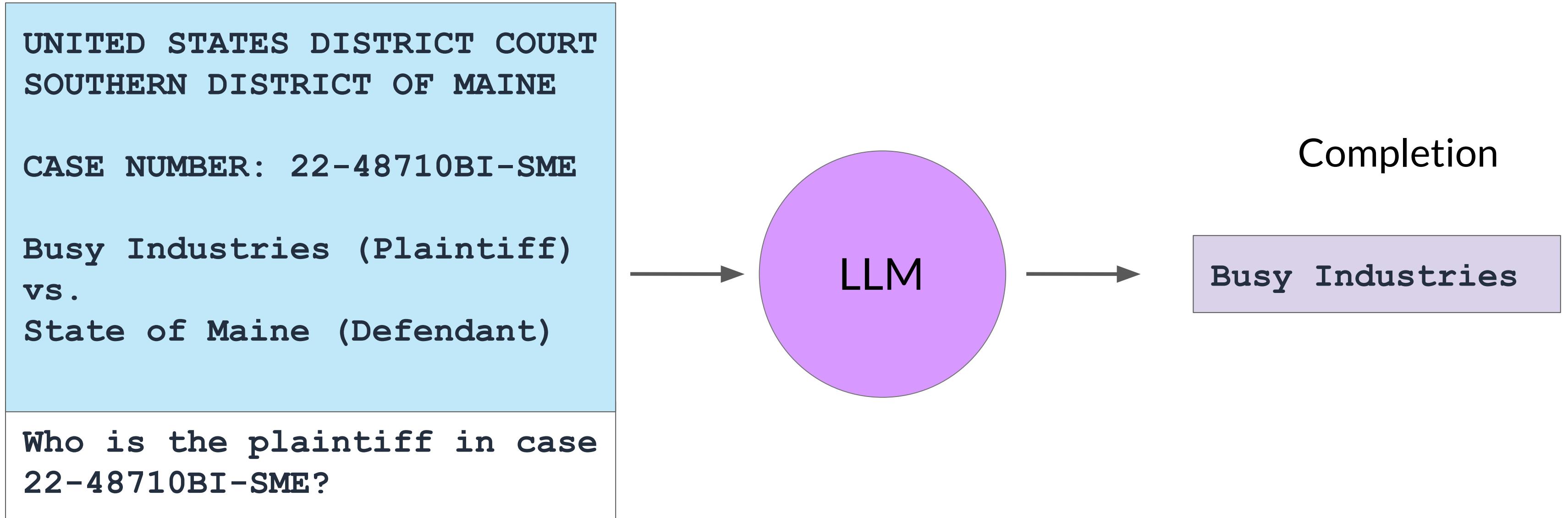
UNITED STATES DISTRICT COURT  
SOUTHERN DISTRICT OF MAINE

CASE NUMBER: 22-48710BI-SME

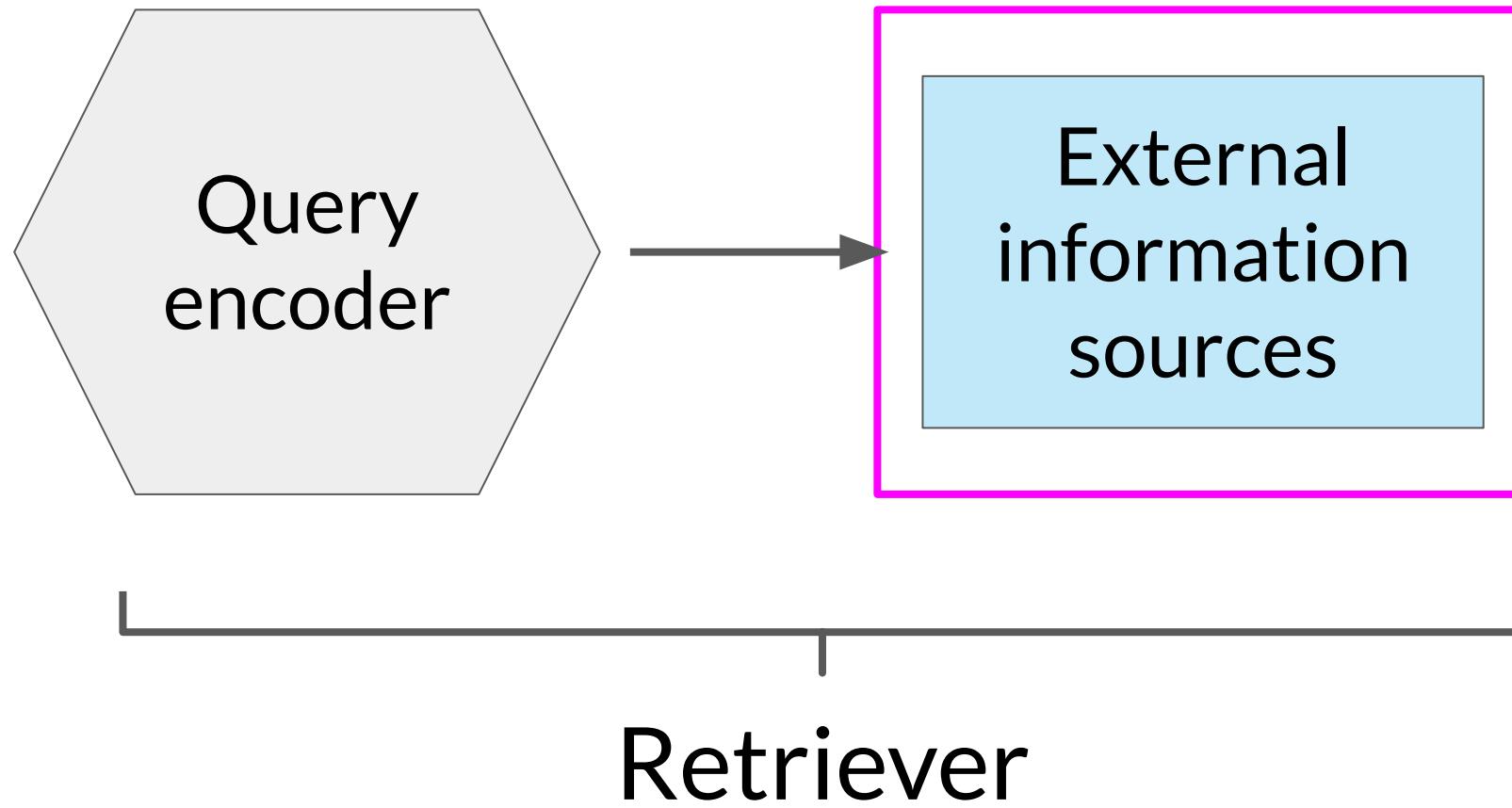
Busy Industries (Plaintiff)  
vs.  
State of Maine (Defendant)

Who is the plaintiff in case 22-48710BI-SME?

# Example: Searching legal documents



# RAG integrates with many types of data sources



## External Information Sources

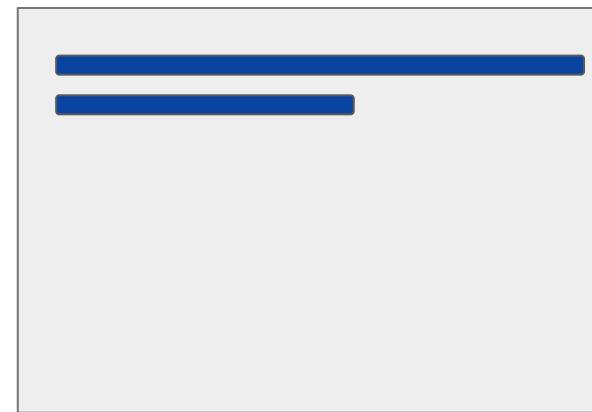
- Documents
- Wikis
- Expert Systems
- Web pages
- Databases
- Vector Store

# Data preparation for vector store for RAG

Two considerations for using external data in RAG:

1. Data must fit inside context window

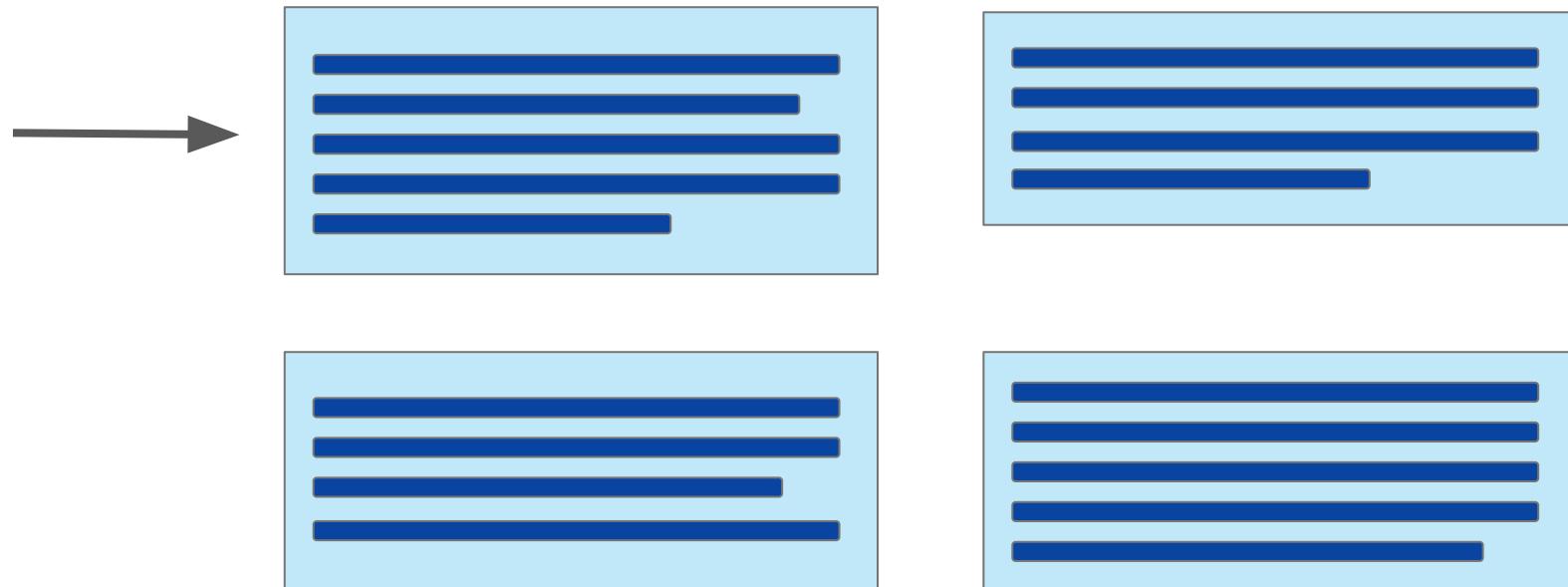
Prompt context limit  
few 1000 tokens



Single document too  
large to fit in window



Split long sources into  
short chunks

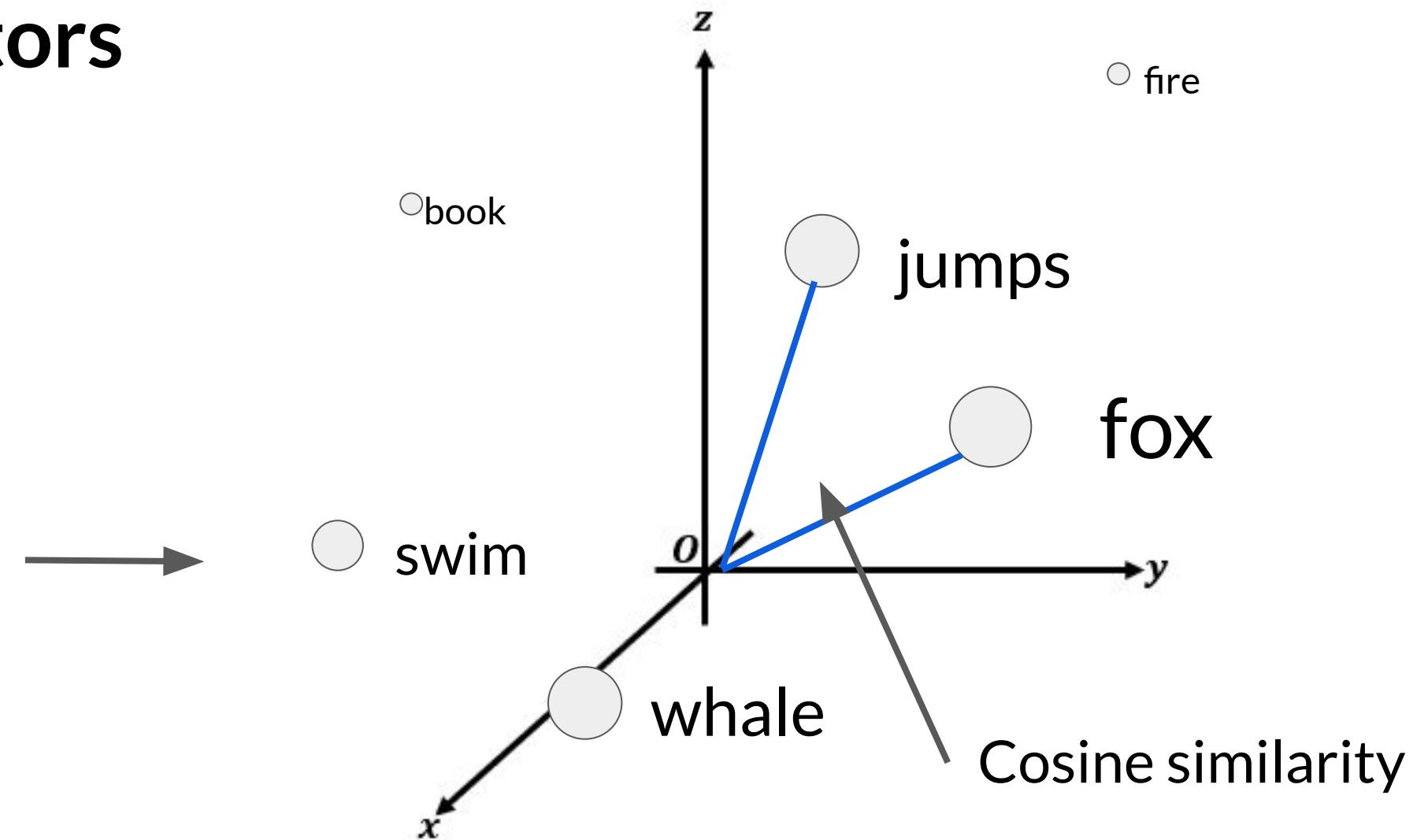
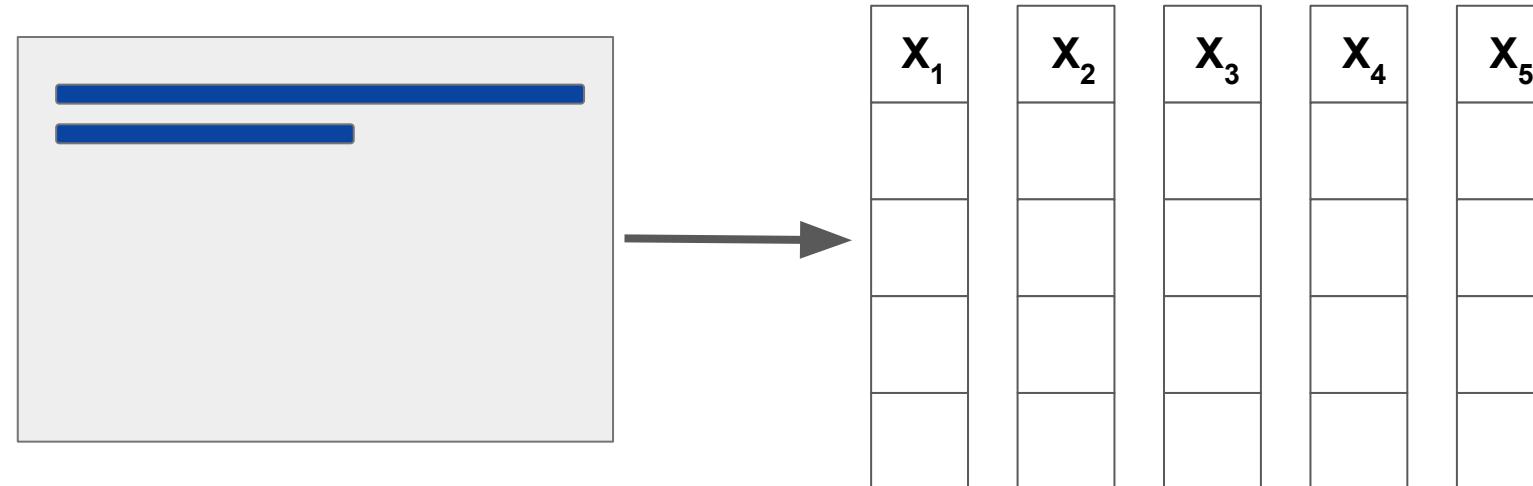


# Data preparation for RAG

Two considerations for using external data in RAG:

1. Data must fit inside context window
2. Data must be in format that allows its relevance to be assessed at inference time: **Embedding vectors**

Prompt text converted  
to embedding vectors

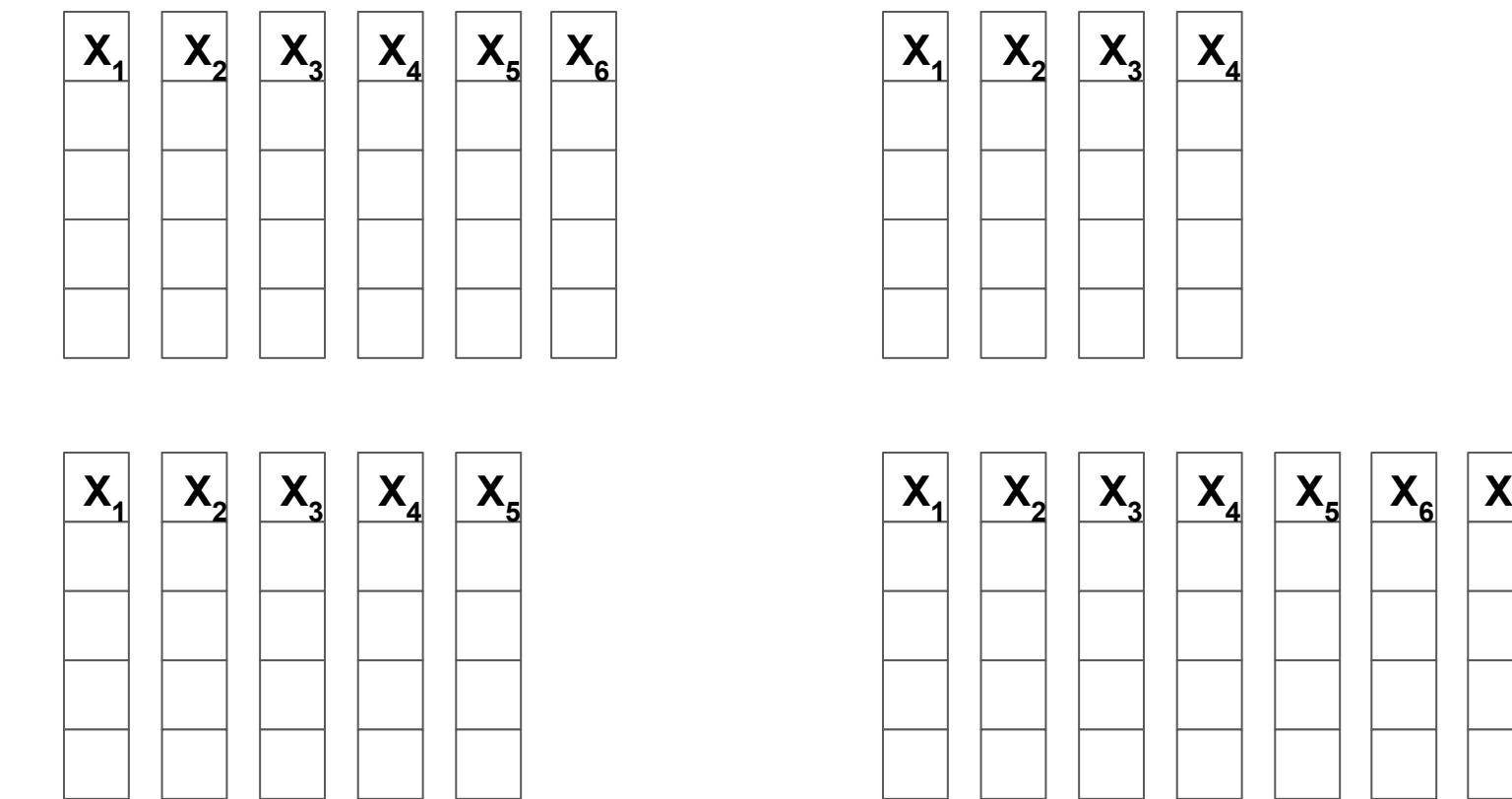
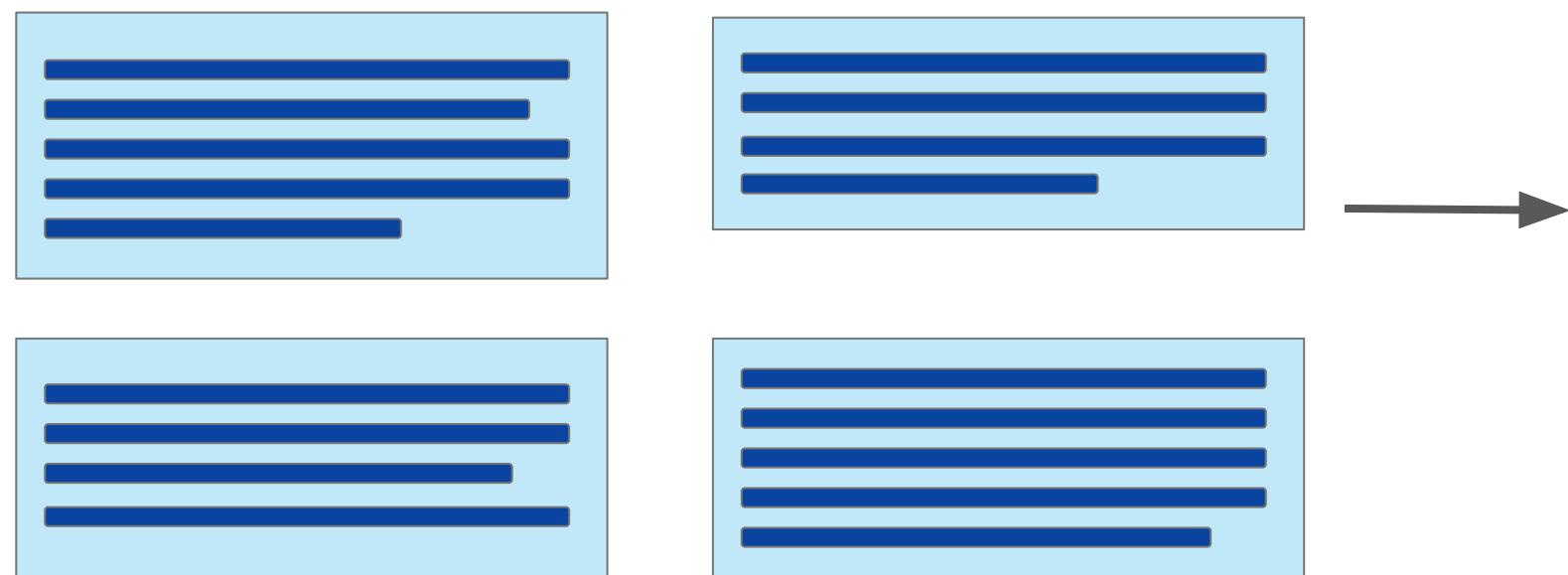


# Data preparation for RAG

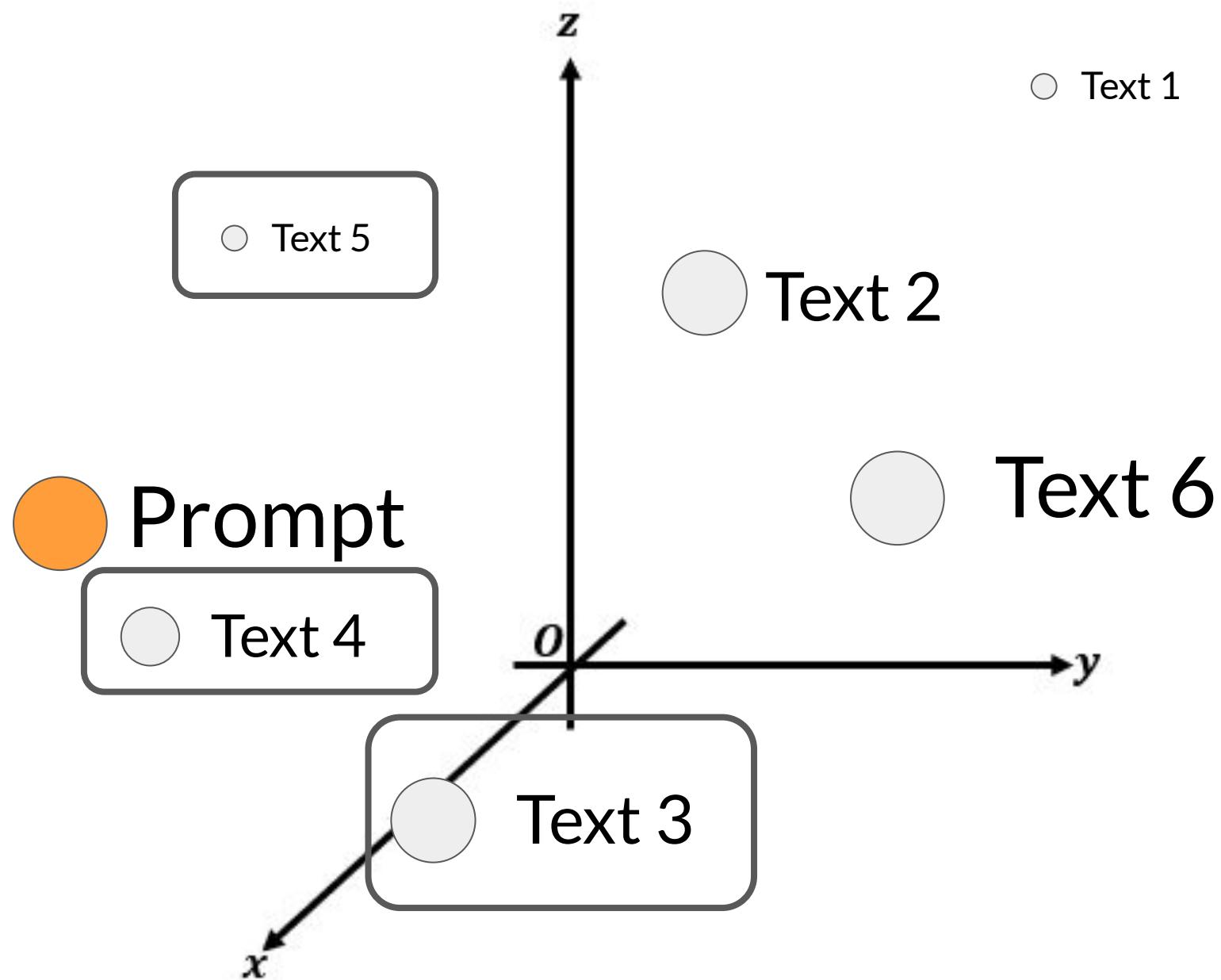
Two considerations for using external data in RAG:

1. Data must fit inside context window
2. Data must be in format that allows its relevance to be assessed at inference time: **Embedding vectors**

Process each chunk with LLM  
to produce embedding vectors



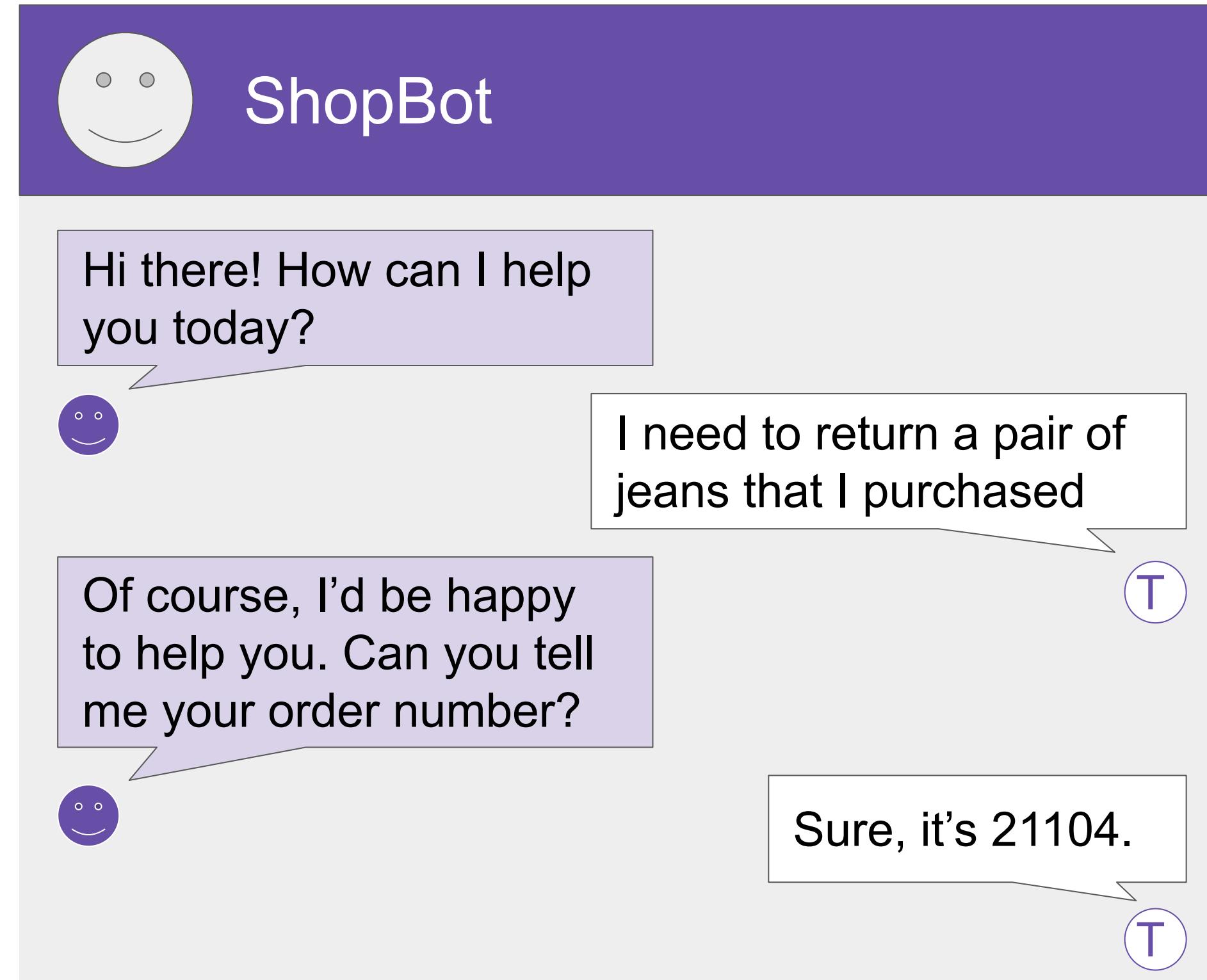
# Vector database search



- Each text in vector store is identified by a key
- Enables a **citation** to be included in completion

# Enabling interactions with external applications

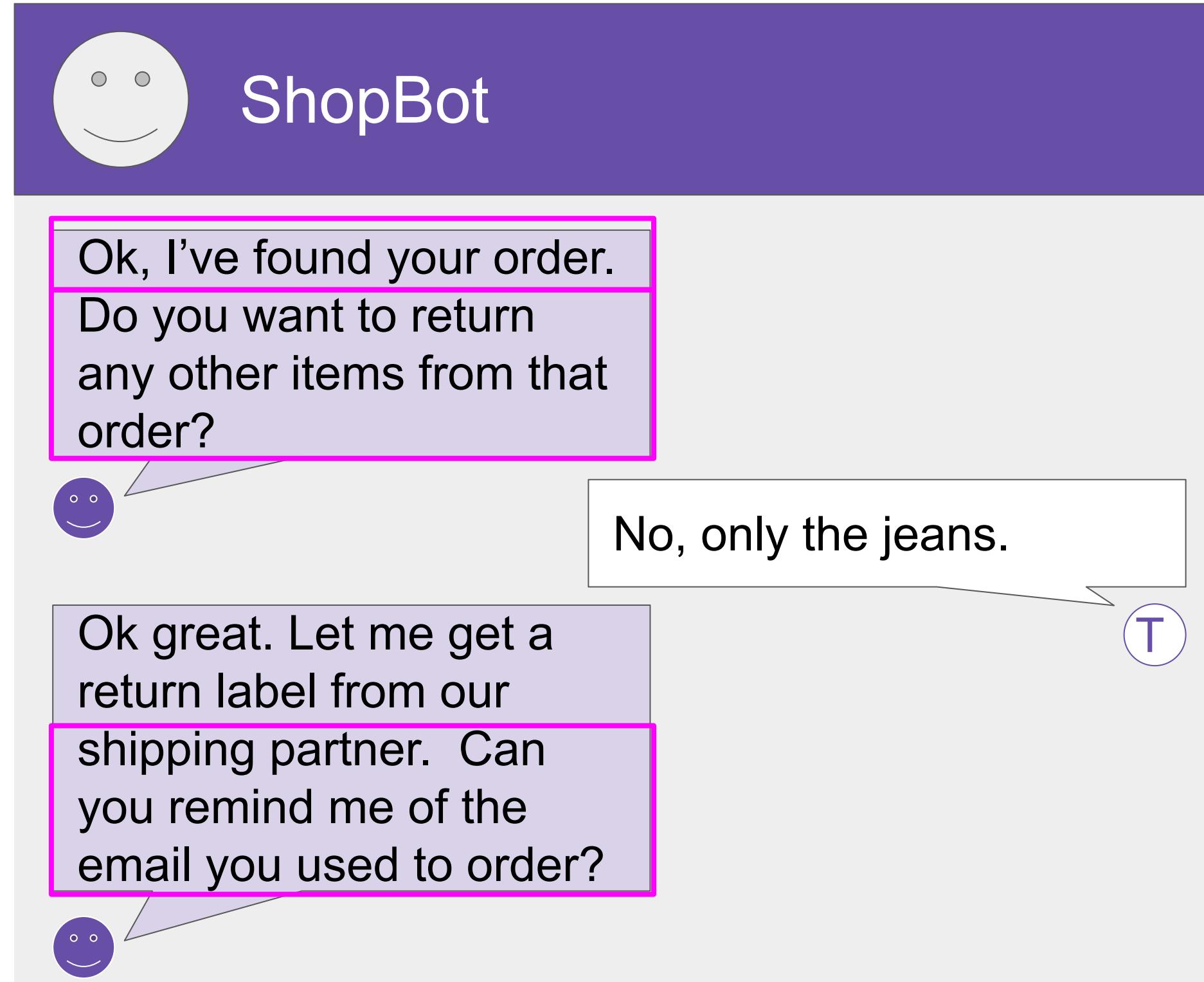
# Having an LLM initiate a clothing return



# Having an LLM initiate a clothing return

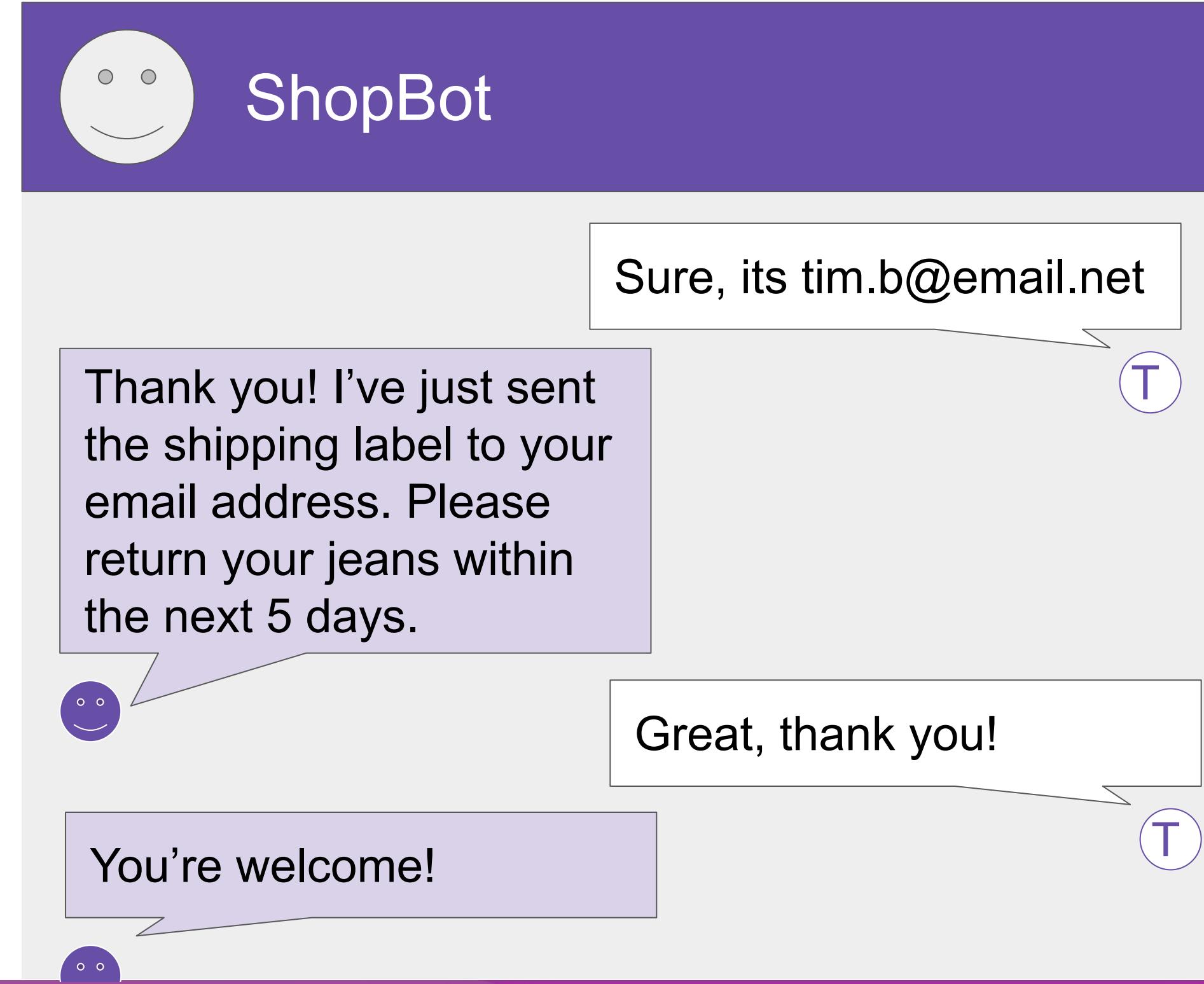
Lookup with RAG

API call

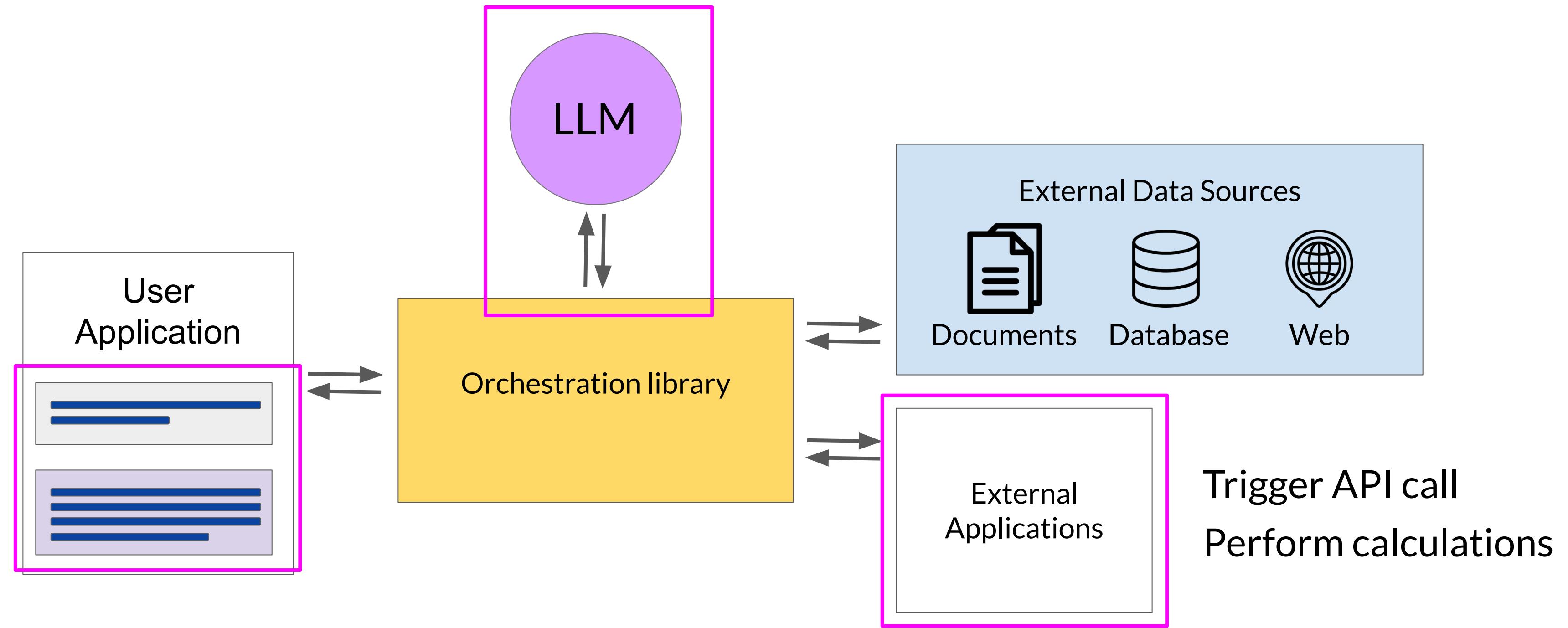


# Having an LLM initiate a clothing return

API call to the shipper



# LLM-powered applications



# Requirements for using LLMs to power applications

## Plan actions

Steps to process return:

**Step 1:** Check order ID

**Step 2:** Request label

**Step 3:** Verify user email

**Step 4:** Email user label

## Format outputs

SQL Query:

**SELECT COUNT(\*)**

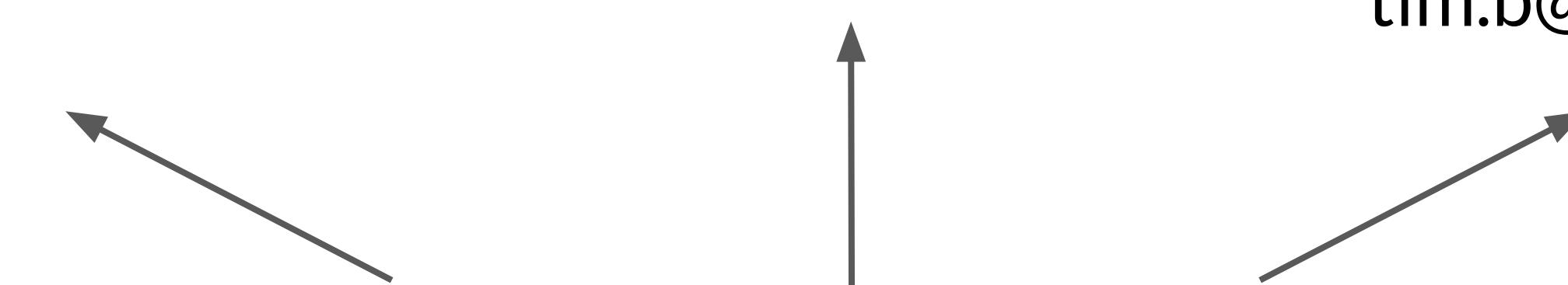
**FROM orders**

**WHERE order\_id = 21104**

## Validate actions

Collect required user information and make sure it is in the completion

User email:  
tim.b@email.net



Prompt structure is important!

# Helping LLMs reason and plan with Chain-of-Thought Prompting

# LLMs can struggle with complex reasoning problems

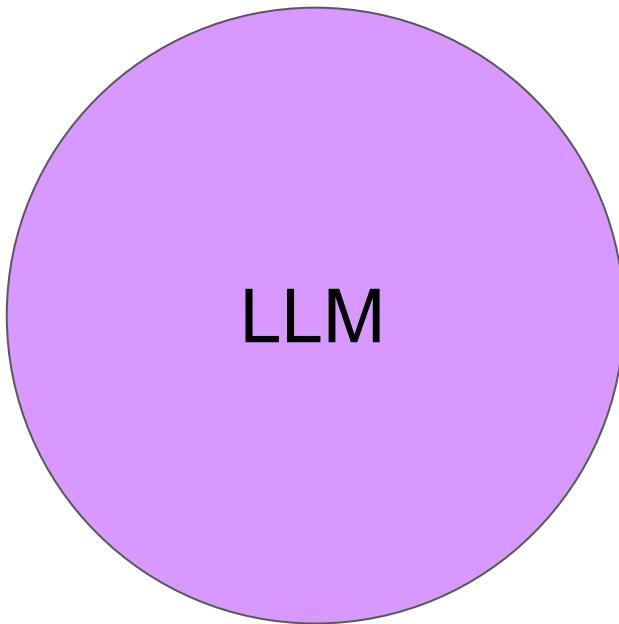
## Prompt

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

## Model



## Completion

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

A: The answer is 27.



# Humans take a step-by-step approach to solving complex problems

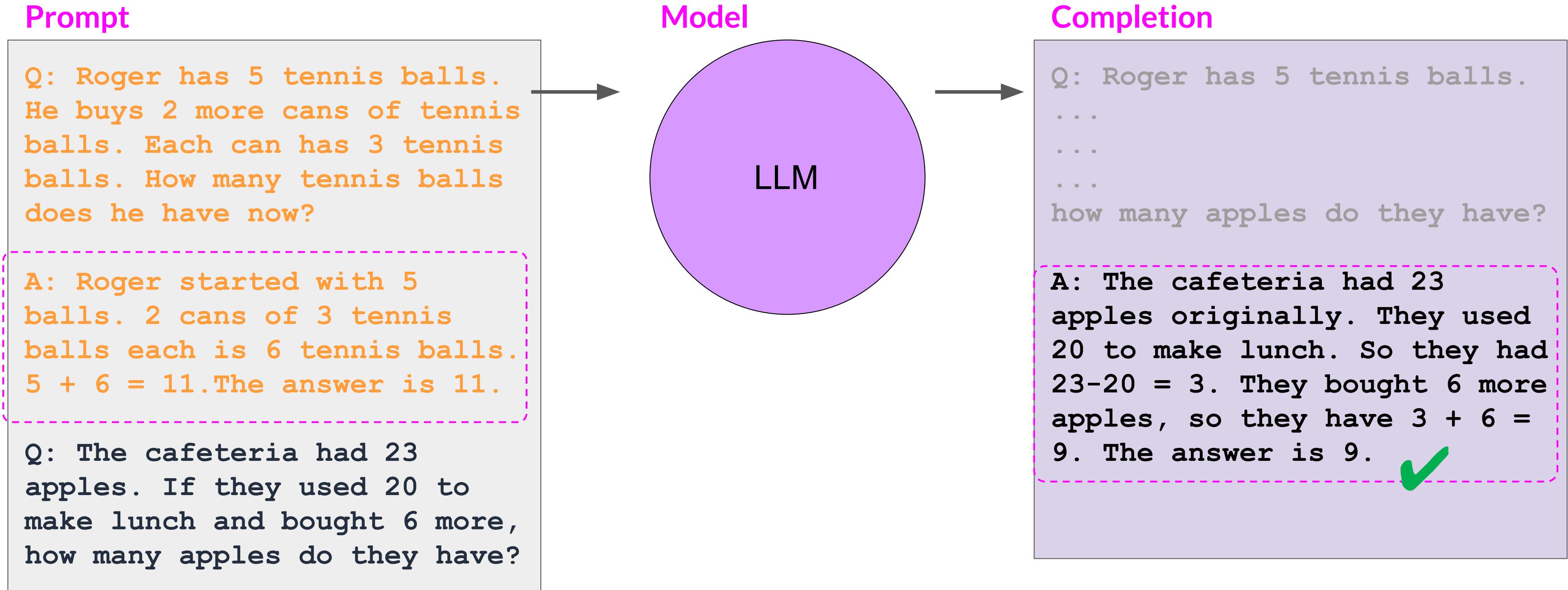
Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

“Chain of thought”

Start: Roger started with 5 balls.  
Step 1: 2 cans of 3 tennis balls each is 6 tennis balls.  
Step 2:  $5 + 6 = 11$   
End: The answer is 11

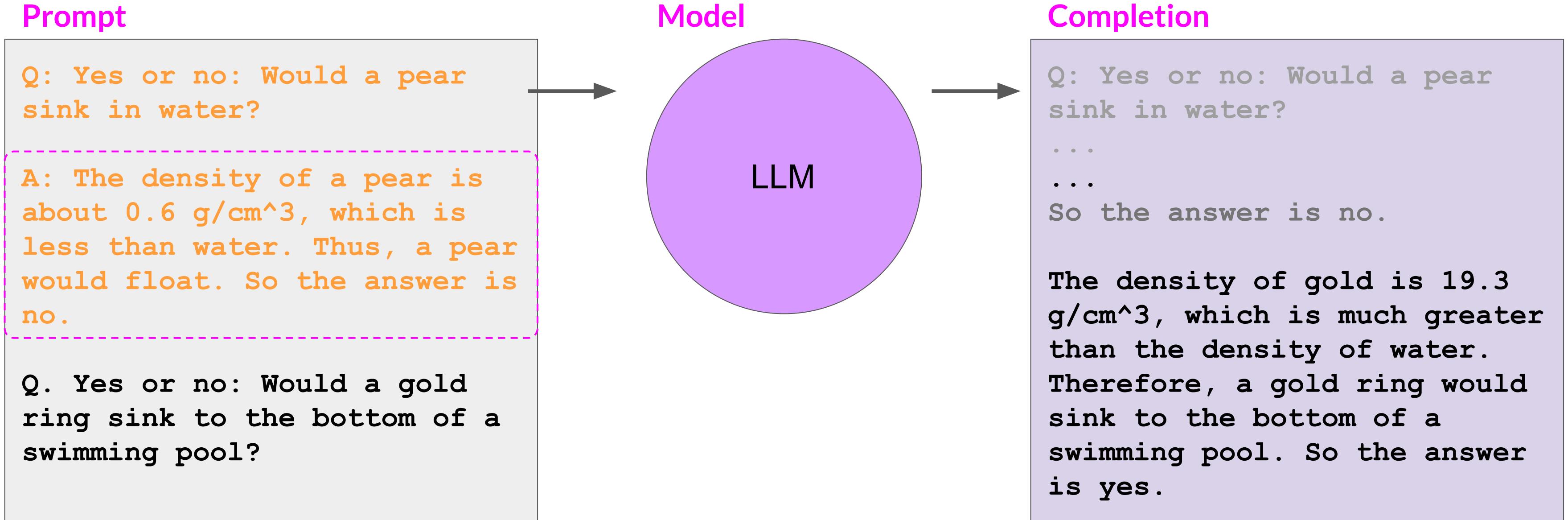
Reasoning steps

# Chain-of-Thought Prompting can help LLMs reason



Source: Wei et al. 2022, "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models"

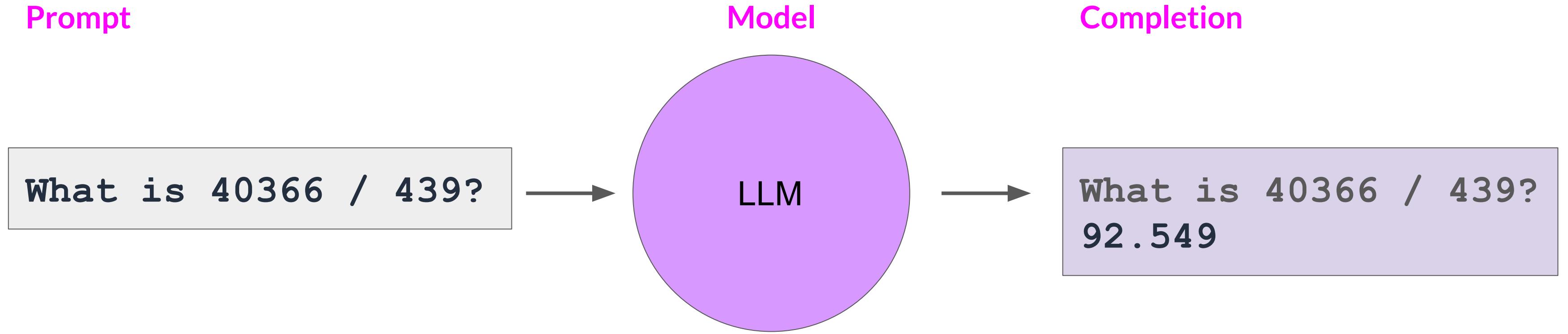
# Chain-of-Thought Prompting can help LLMs reason



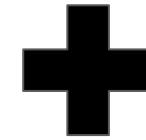
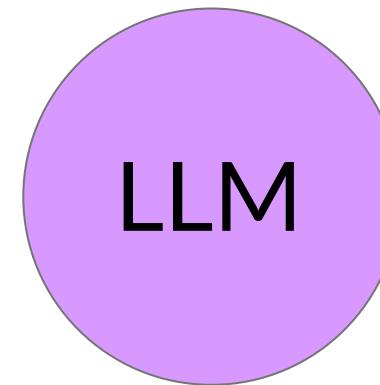
Source: Wei et al. 2022, "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models"

# Program-aided Language Models

# LLMs can struggle with mathematics



# Program-aided language (PAL) models



Code  
interpreter

## Chain-of-Thought (Wei et al., 2022)

Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 tennis balls. 2 cans of 3 tennis balls each is 6 tennis balls.  $5 + 6 = 11$ . The answer is 11.

Q: The bakers at the Beverly Hills Bakery baked 200 loaves of bread on Monday morning. They sold 93 loaves in the morning and 39 loaves in the afternoon. A grocery store returned 6 unsold loaves. How many loaves of bread did they have left?

Model Output

A: The bakers started with 200 loaves. They sold 93 in the morning and 39 in the afternoon. So they sold  $93 + 39 = 132$  loaves. The grocery store returned 6 loaves. So they had  $200 - 132 - 6 = 62$  loaves left.  
The answer is 62.



## Program-aided Language models (this work)

Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 tennis balls.  
`tennis_balls = 5`  
2 cans of 3 tennis balls each is  
`bought_balls = 2 * 3`  
tennis balls. The answer is  
`answer = tennis_balls + bought_balls`

Q: The bakers at the Beverly Hills Bakery baked 200 loaves of bread on Monday morning. They sold 93 loaves in the morning and 39 loaves in the afternoon. A grocery store returned 6 unsold loaves. How many loaves of bread did they have left?

Model Output

A: The bakers started with 200 loaves  
`loaves_baked = 200`  
They sold 93 in the morning and 39 in the afternoon  
`loaves_sold_morning = 93`  
`loaves_sold_afternoon = 39`  
The grocery store returned 6 loaves.  
`loaves_returned = 6`  
The answer is  
`answer = loaves_baked - loaves_sold_morning`  
`- loaves_sold_afternoon + loaves_returned`

`>>> print(answer)`  
74



Source: Gao et al. 2022, "PAL: Program-aided Language Models"

# PAL example

## Prompt with one-shot example

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

Answer:

```
# Roger started with 5 tennis balls
tennis_balls = 5
# 2 cans of tennis balls each is
bought_balls = 2 * 3
# tennis balls. The answer is
answer = tennis_balls + bought_balls
```

Q: The bakers at the Beverly Hills Bakery baked 200 loaves of bread on Monday morning. They sold 93 loaves in the morning and 39 loaves in the afternoon. A grocery store returned 6 unsold loaves. How many loaves did they have left?

# PAL example

## Prompt with one-shot example

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

Answer:

```
# Roger started with 5 tennis balls  
tennis_balls = 5  
# 2 cans of tennis balls each is  
bought_balls = 2 * 3  
# tennis balls. The answer is  
answer = tennis_balls + bought_balls
```

Q: The bakers at the Beverly Hills Bakery baked 200 loaves of bread on Monday morning. They sold 93 loaves in the morning and 39 loaves in the afternoon. A grocery store returned 6 unsold loaves. How many loaves did they have left?

## Completion, CoT reasoning (blue), and PAL execution (pink)

Answer:

# The bakers started with 200 loaves

loaves\_baked = 200 ←

# They sold 93 in the morning and 39 in the afternoon

loaves\_sold\_morning = 93 ←

loaves\_sold\_afternoon = 39 ←

# The grocery store returned 6 loaves.

loaves\_returned = 6 ←

# The answer is

answer = loaves\_baked

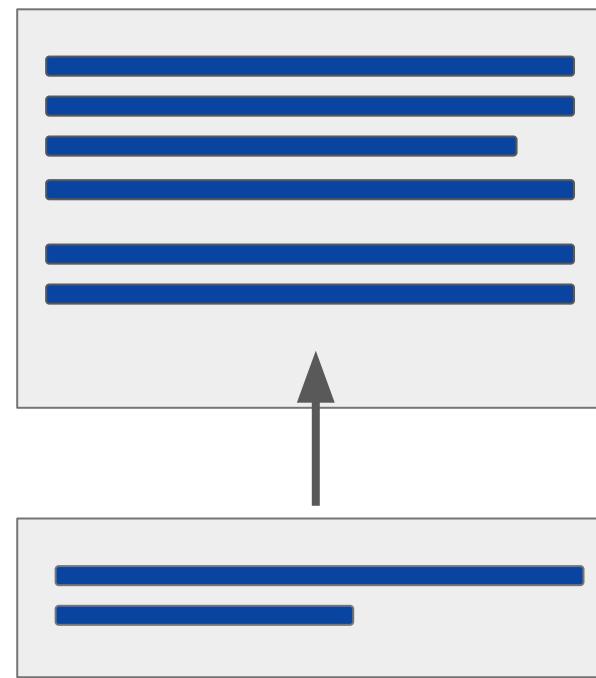
- loaves\_sold\_morning

- loaves\_sold\_afternoon

+ loaves\_returned

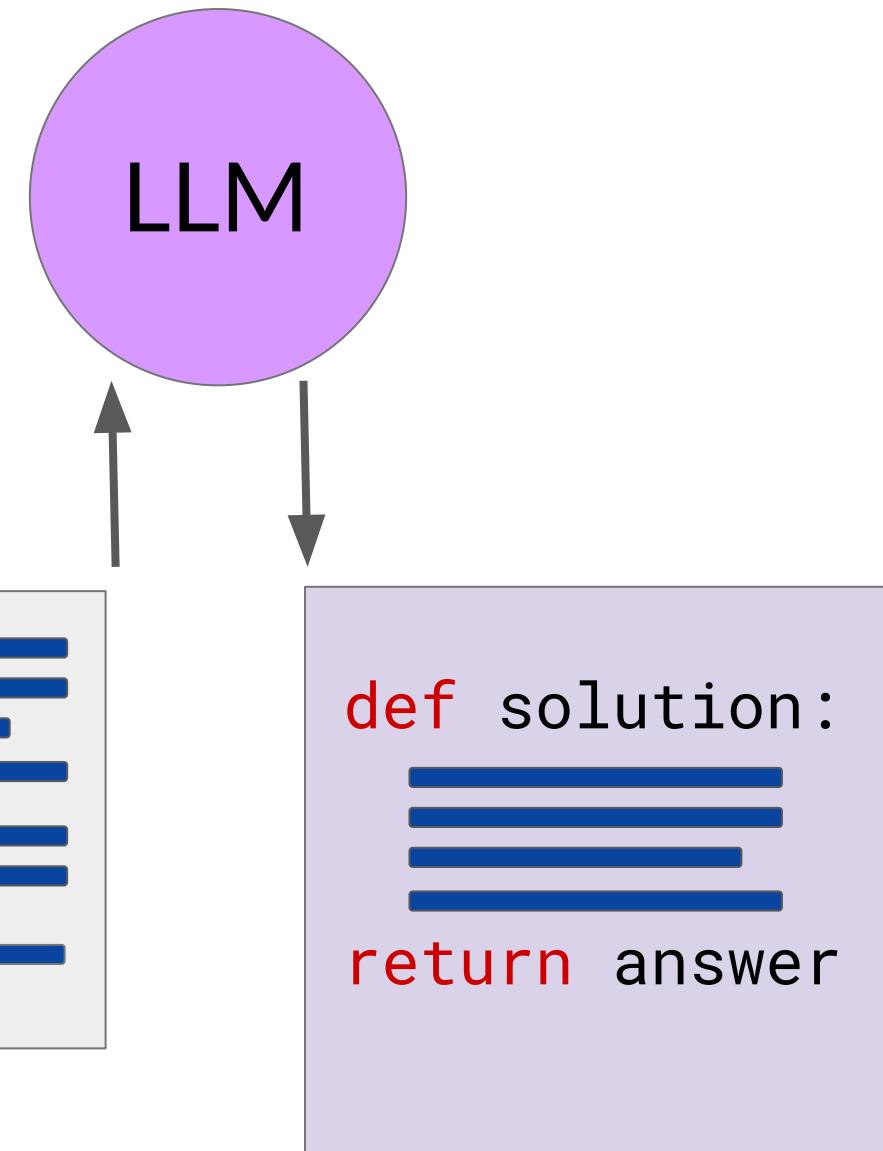
# Program-aided language (PAL) models

PAL prompt  
template

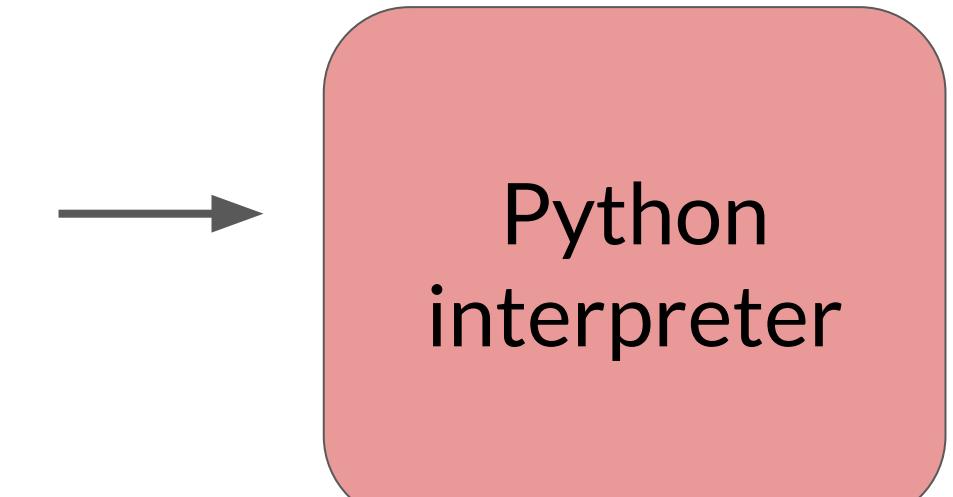


question

PAL formatted  
prompt

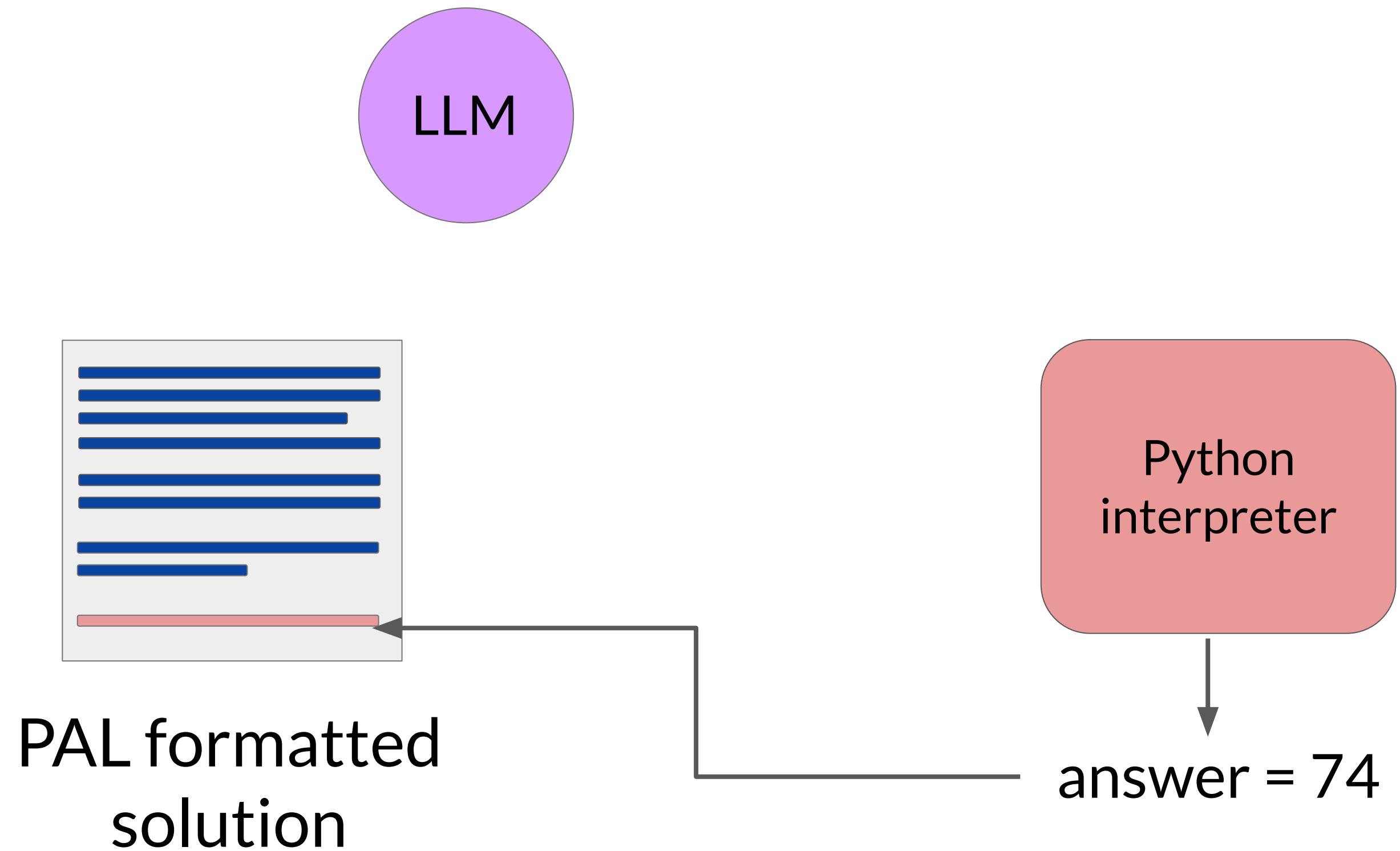


Python  
script

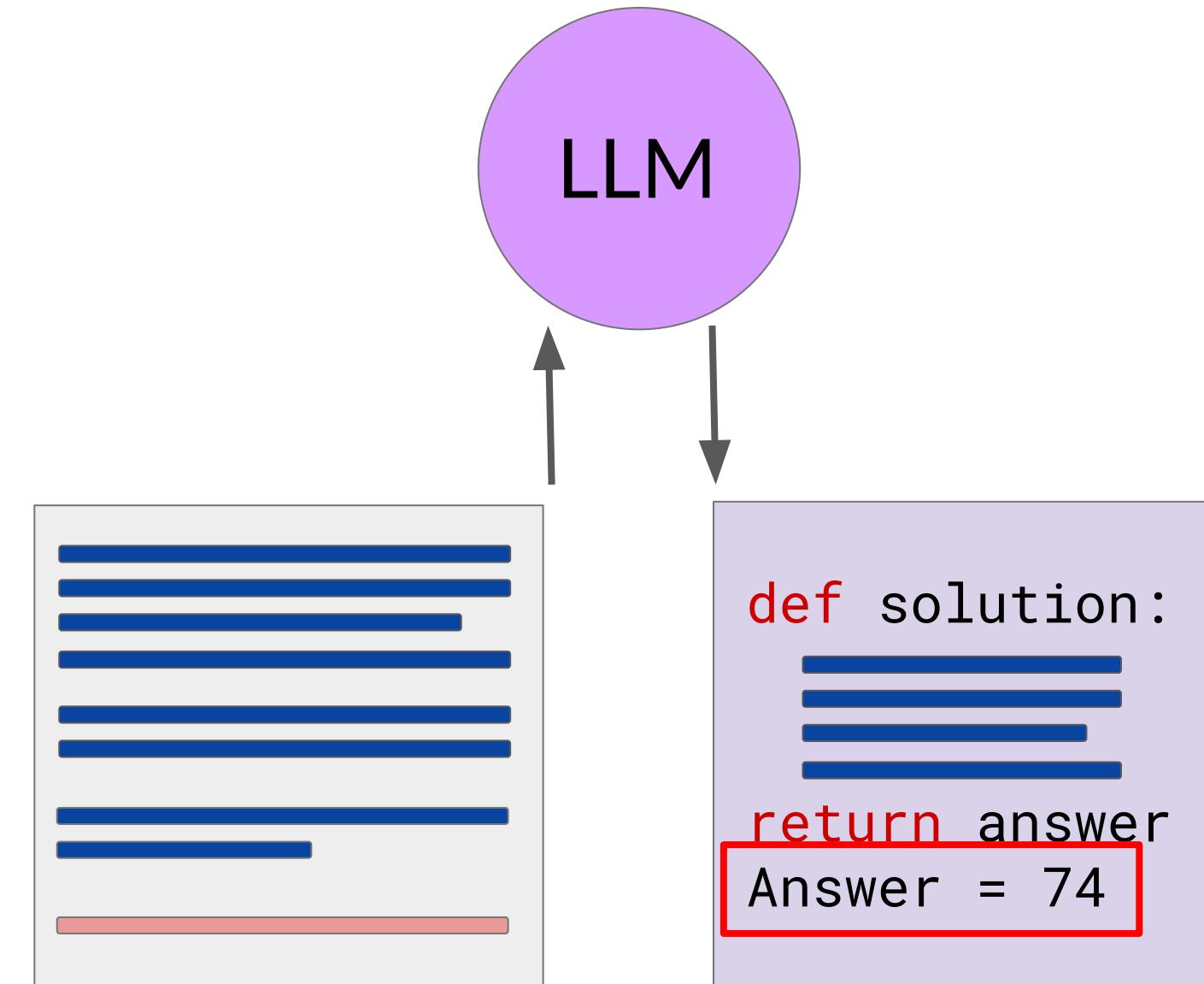


answer = 74

# Program-aided language (PAL) models



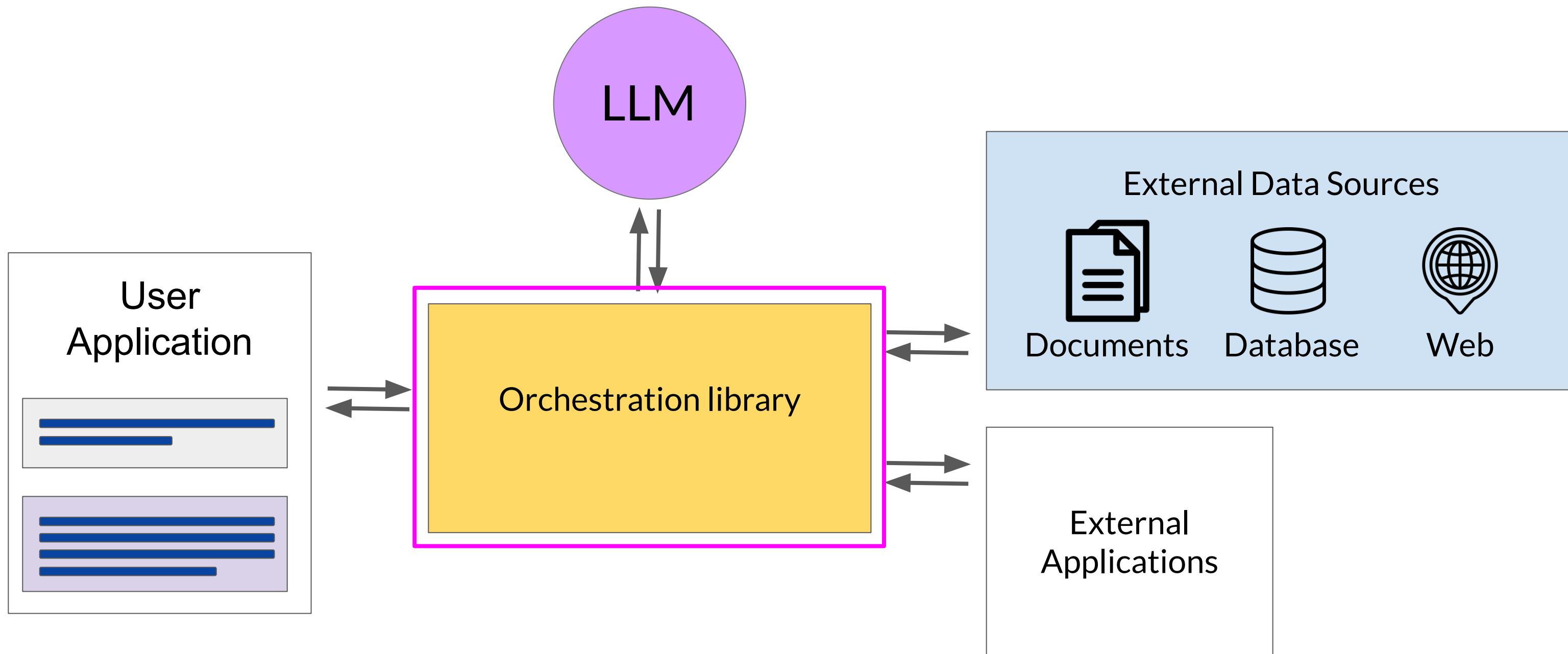
# Program-aided language (PAL) models



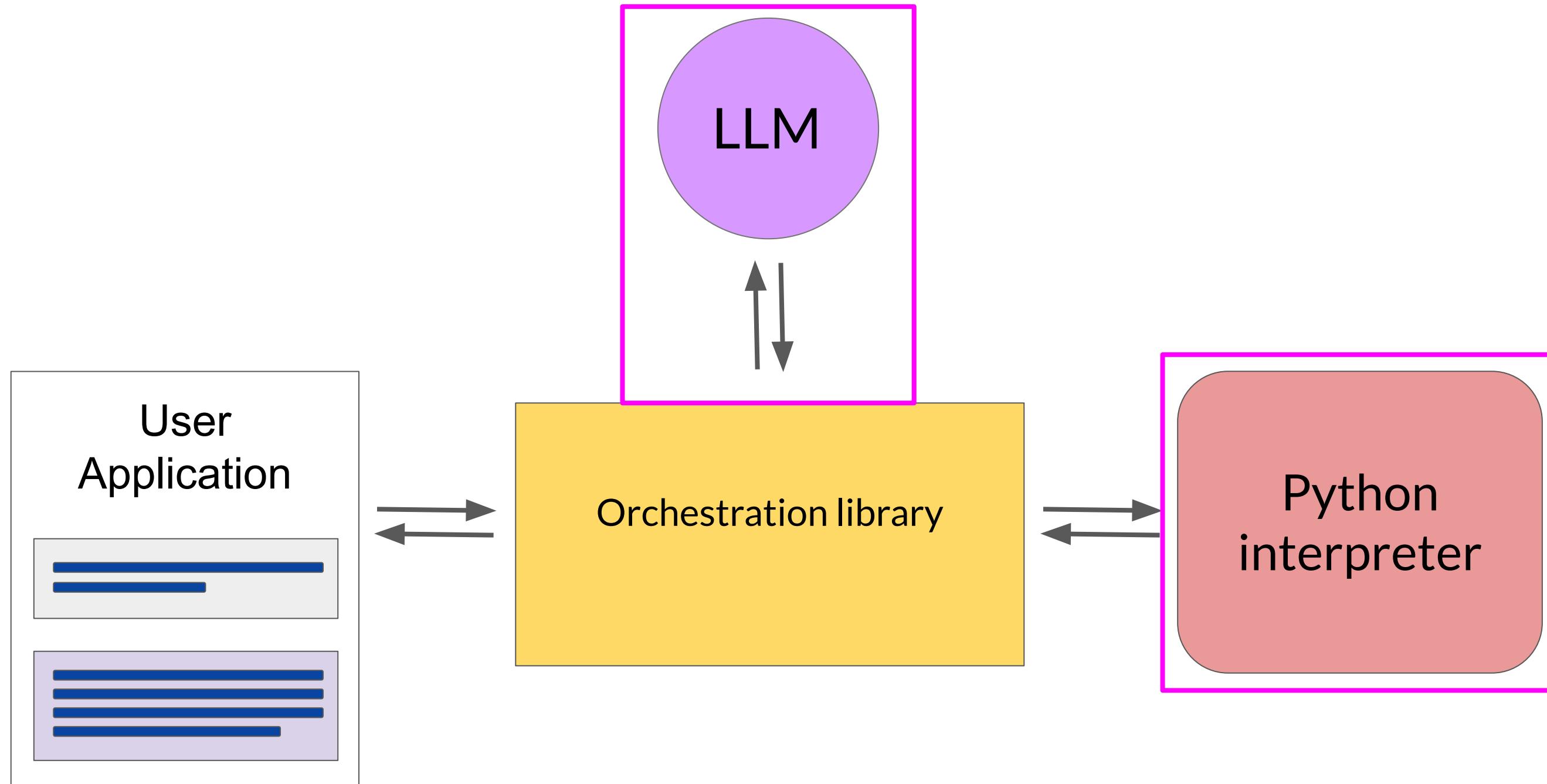
PAL formatted  
solution

Completion with  
correct answer

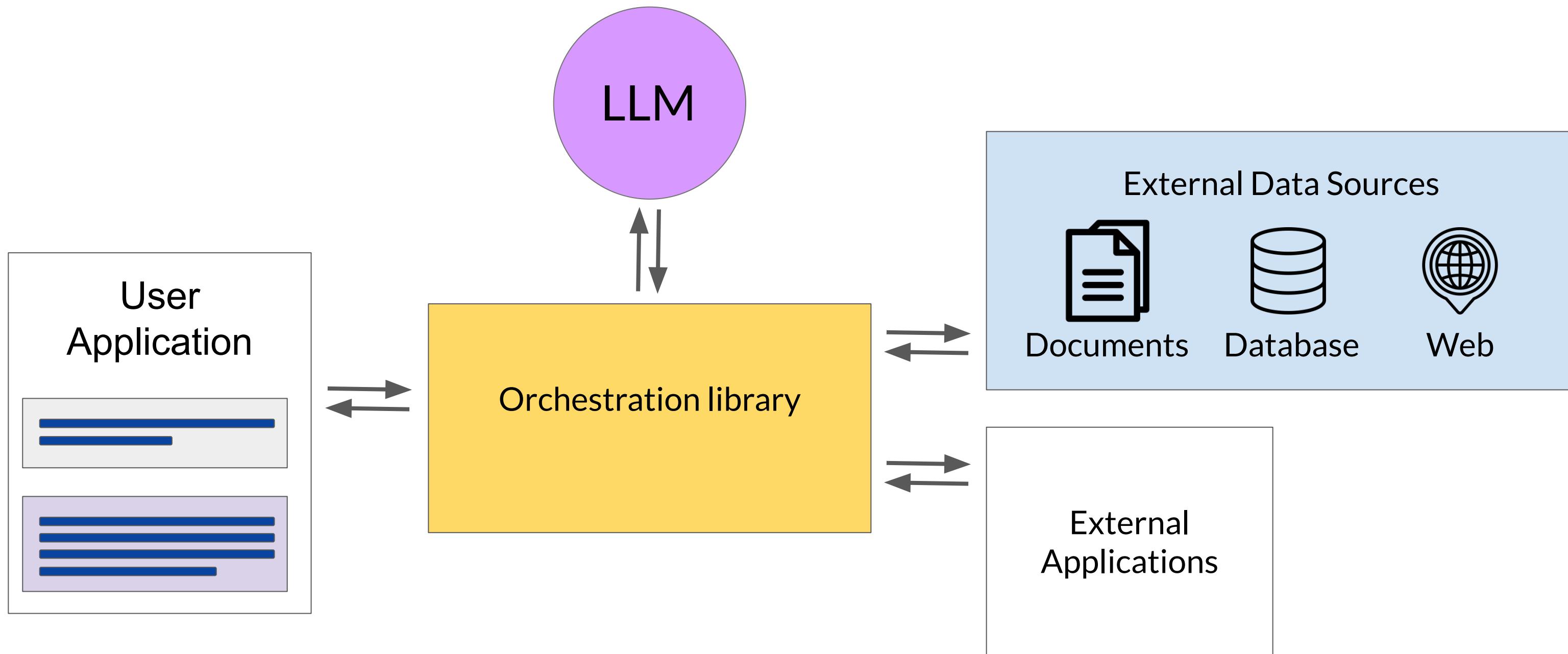
# LLM-powered applications



# PAL architecture

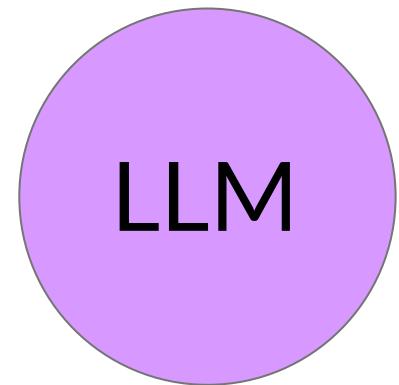


# LLM-powered applications

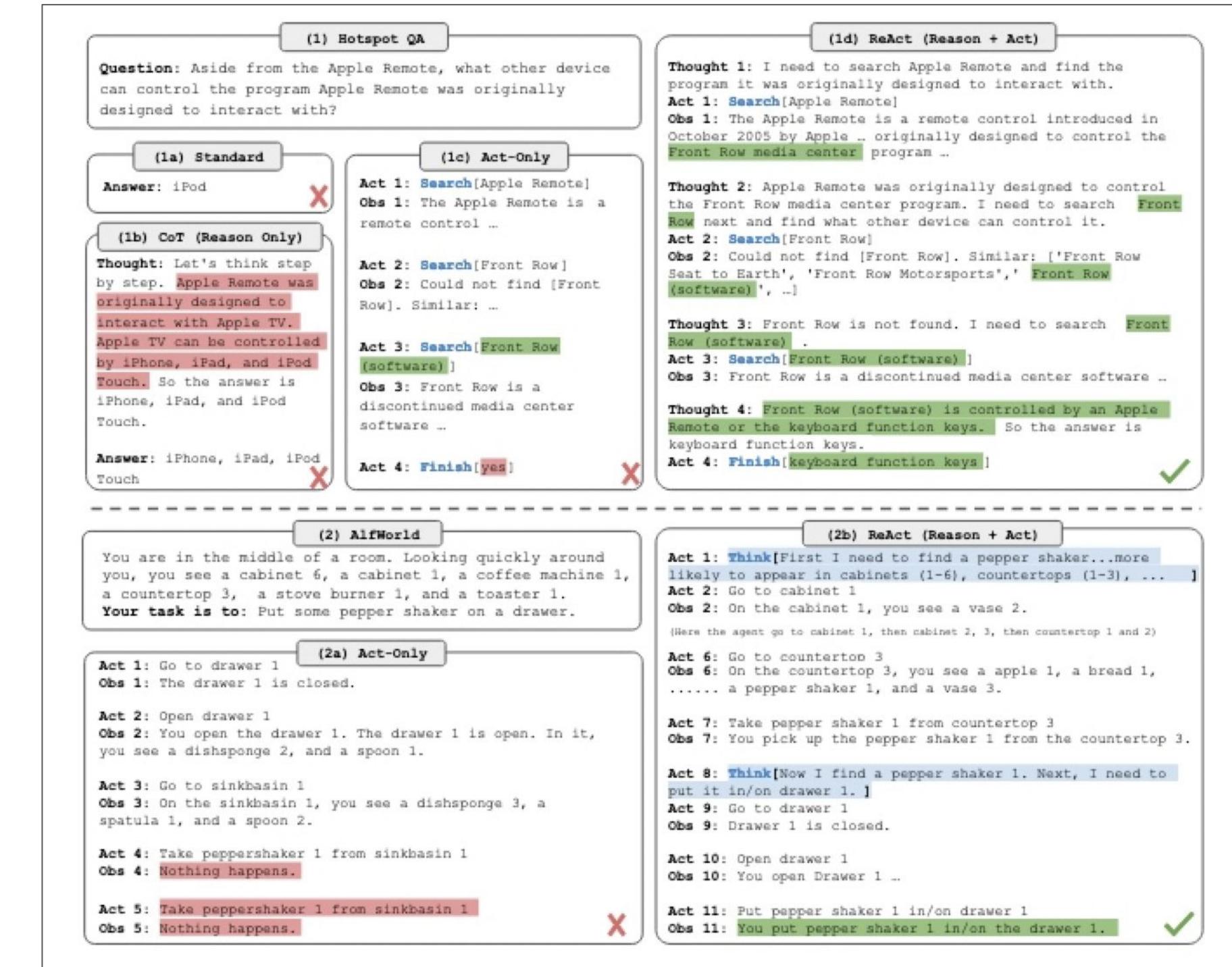


# ReAct: Combining reasoning and action in LLMs

# ReAct: Synergizing Reasoning and Action in LLMs

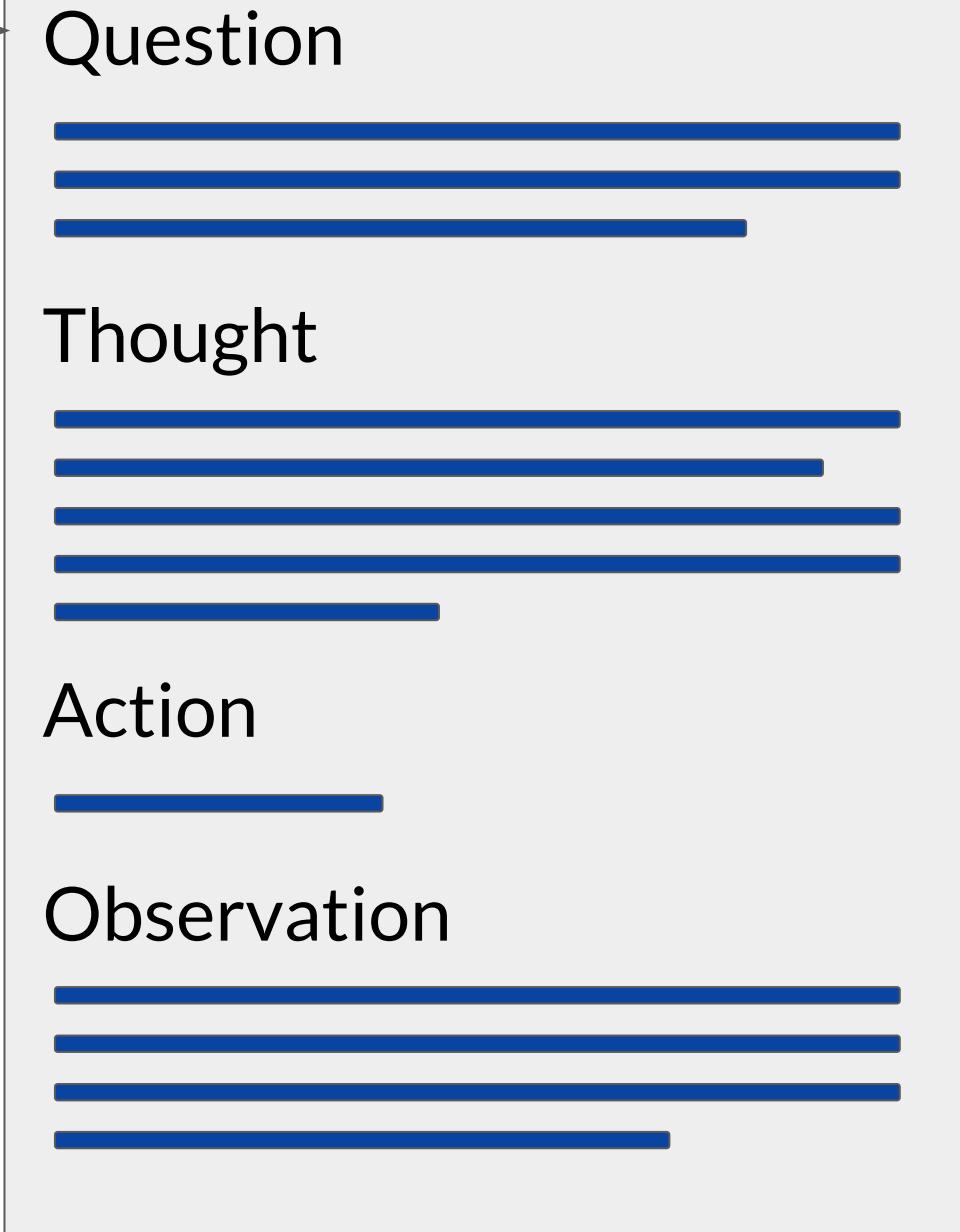


**HotPot QA:** multi-step question answering  
**Fever:** Fact verification



Source: Yao et al. 2022, “ReAct: Synergizing Reasoning and Acting in Language Models”

# ReAct: Synergizing Reasoning and Action in LLMs



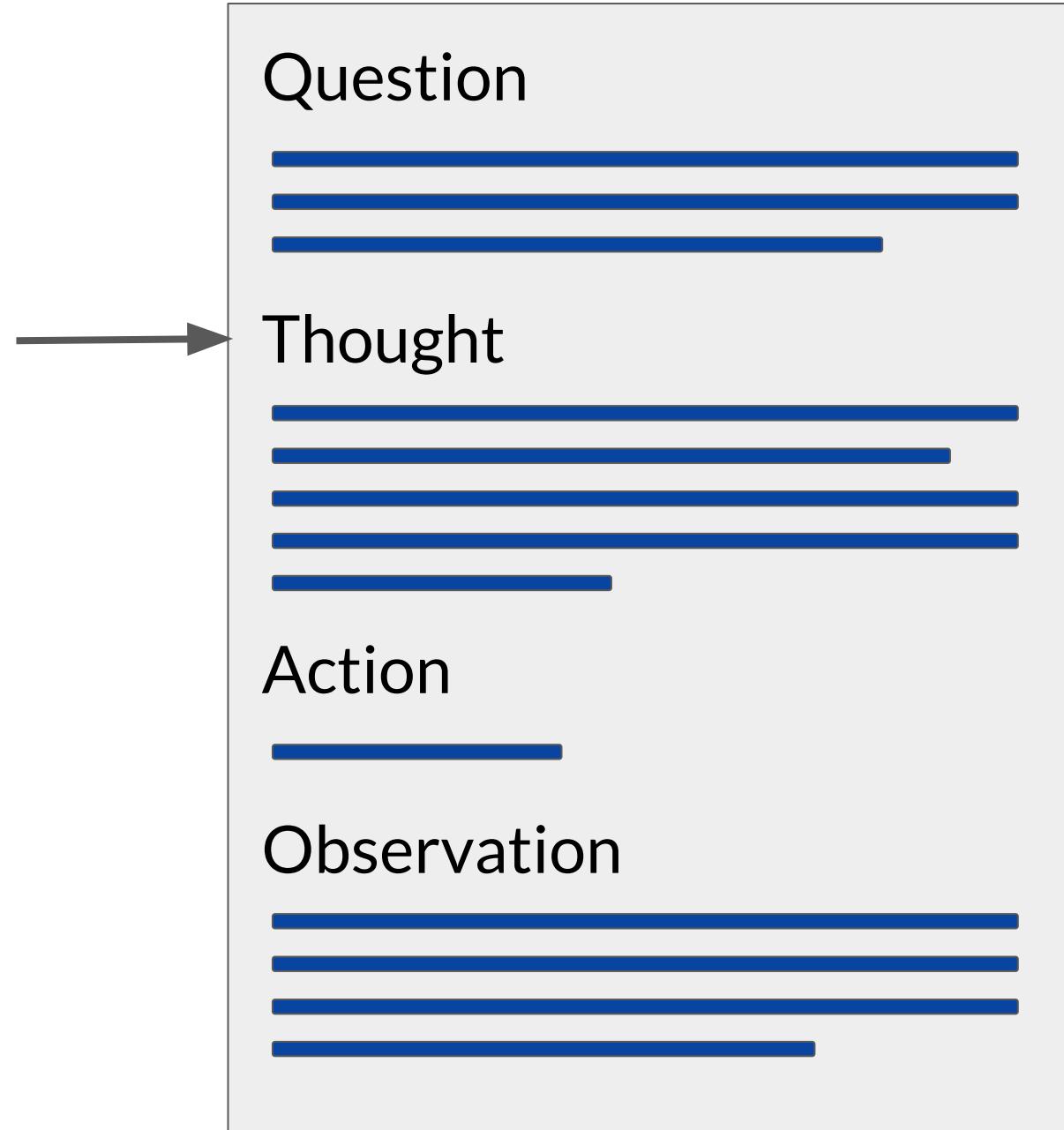
**Question:** Problem that requires advanced reasoning and multiple steps to solve.

E.g.

“Which magazine was started first,  
*Arthur’s Magazine* or *First for Women*? ”

Source: Yao et al. 2022, “ReAct: Synergizing Reasoning and Acting in Language Models”

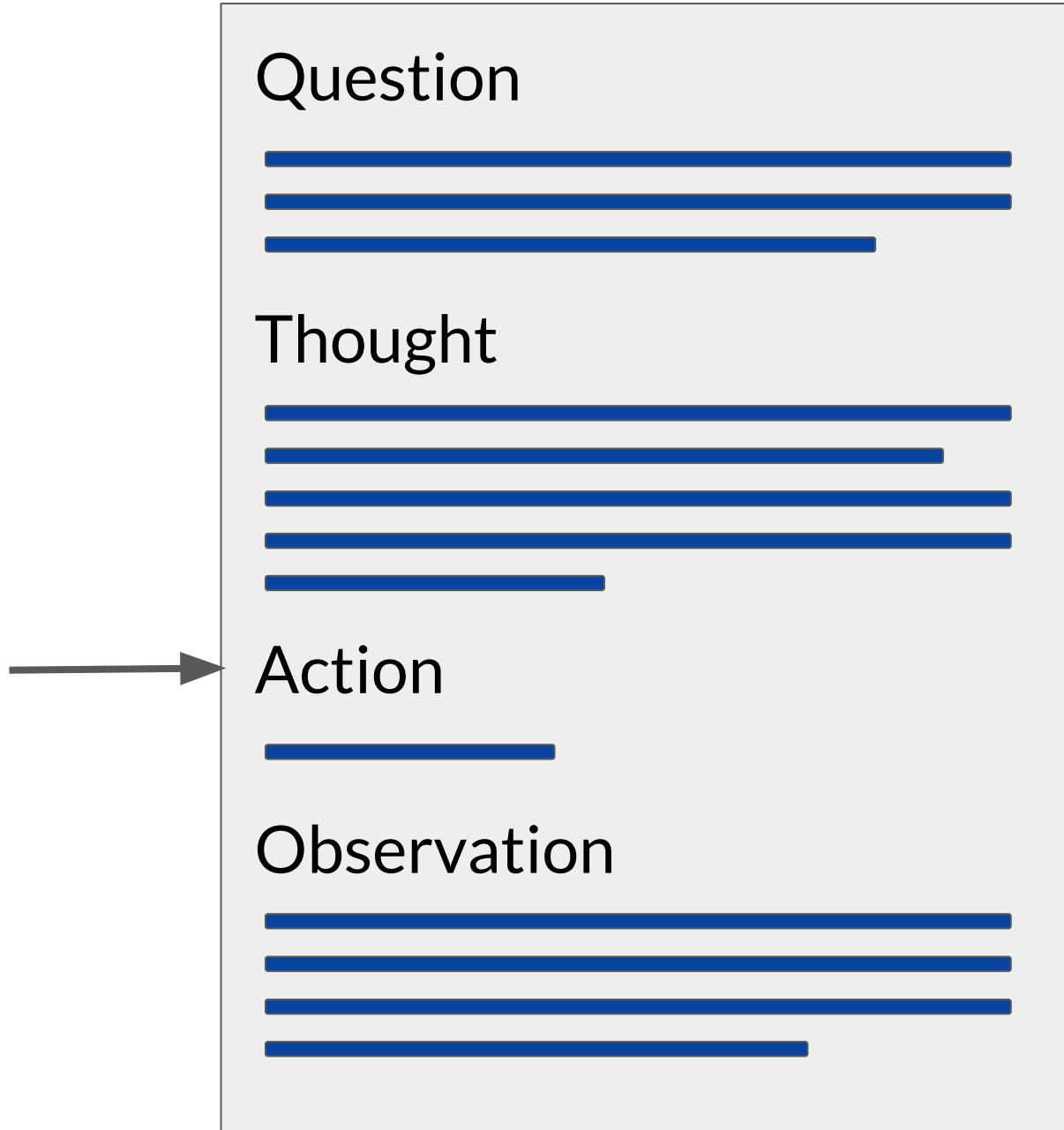
# ReAct: Synergizing Reasoning and Action in LLMs



**Thought:** A reasoning step that identifies how the model will tackle the problem and identify an action to take.

“I need to search Arthur’s Magazine and First for Women, and find which one was started first.”

# ReAct: Synergizing Reasoning and Action in LLMs



**Action:** An external task that the model can carry out from an allowed set of actions.

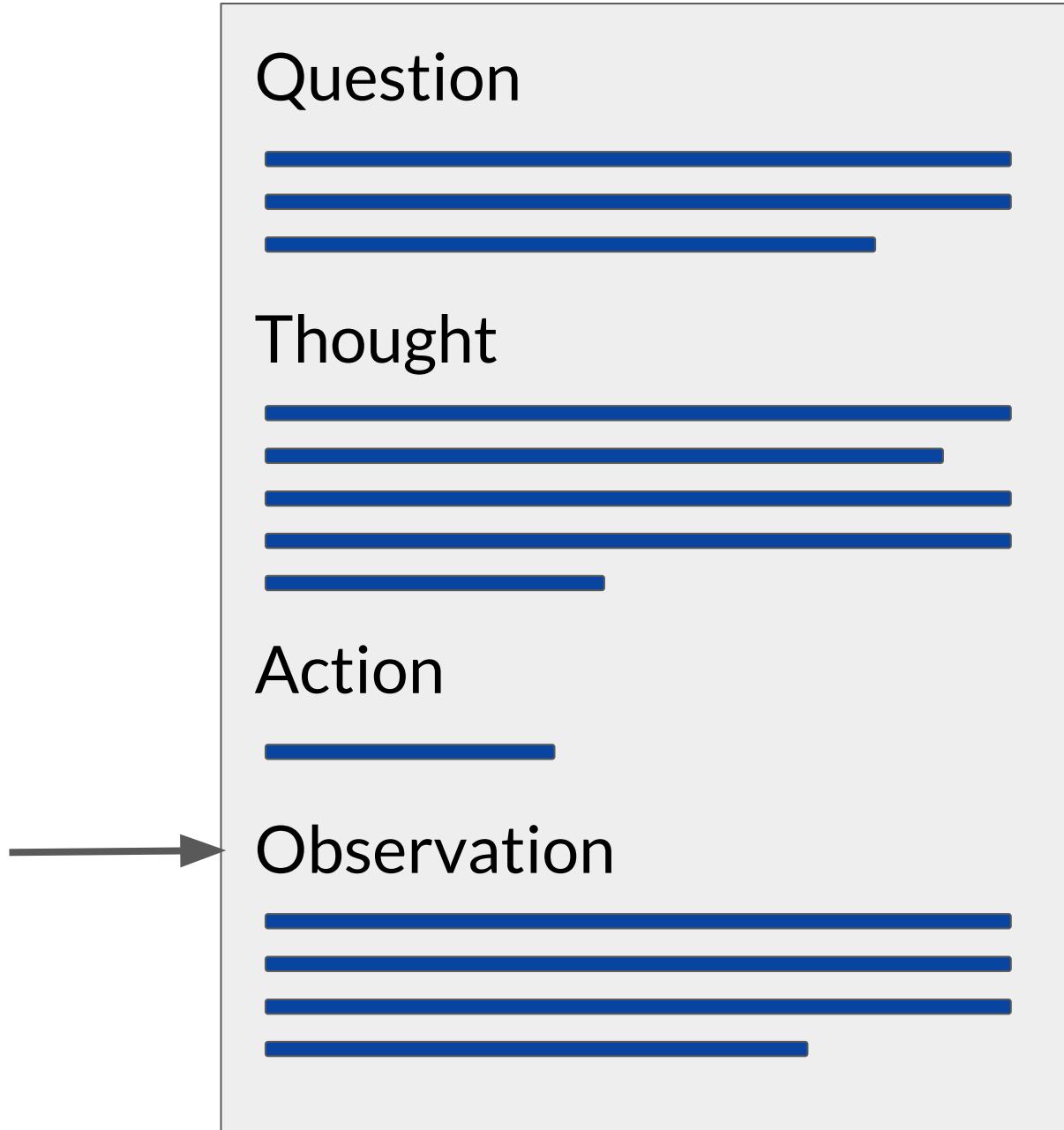
E.g.

**search[ entity ]**  
**lookup[ string ]**  
**finish[ answer ]**

Which one to choose is determined by the information in the preceding thought.

**search[ Arthur's Magazine ]**

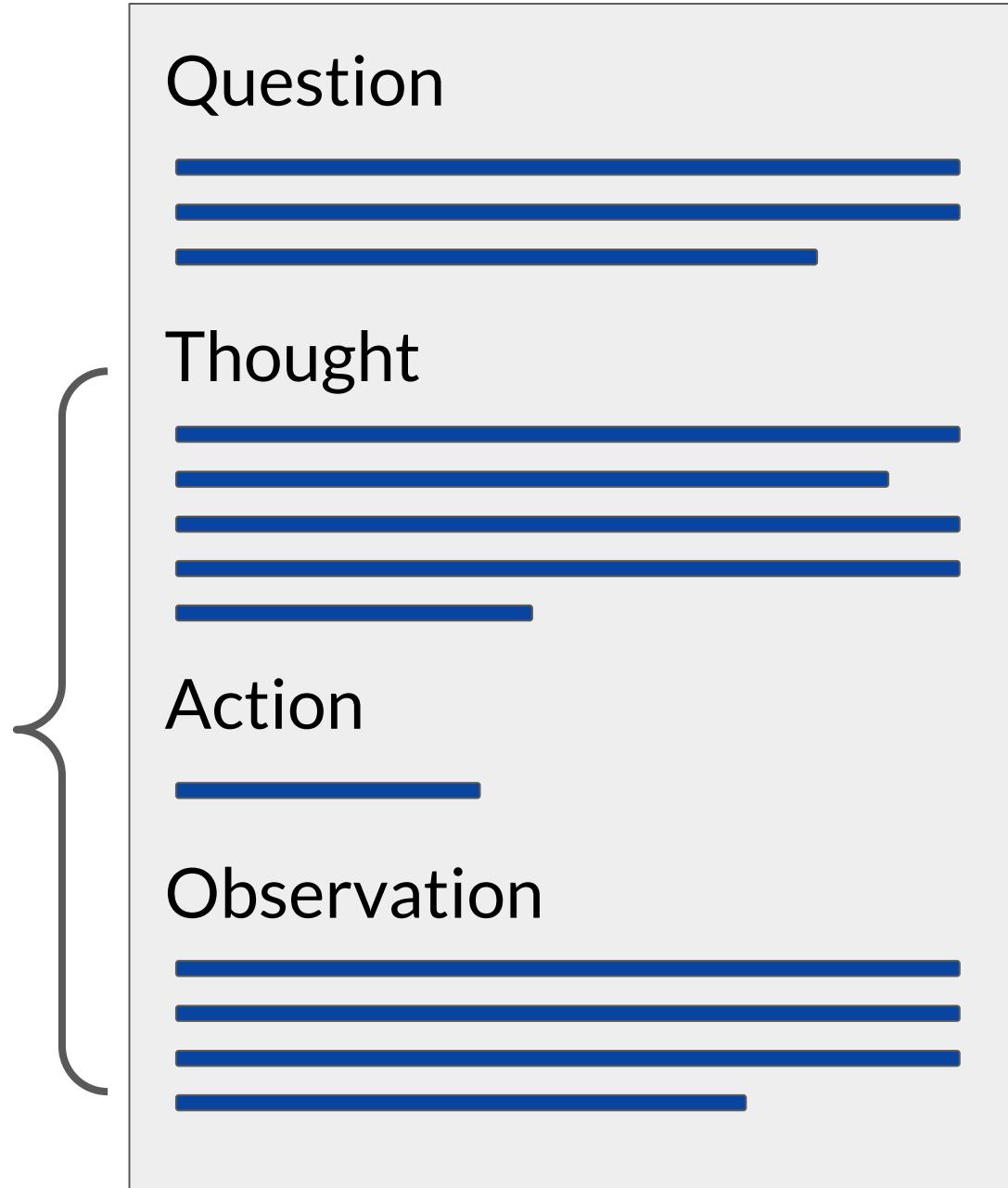
# ReAct: Synergizing Reasoning and Action in LLMs



**Observation:** the result of carrying out the action

E.g.  
“Arthur’s Magazine (1844-1846) was an American literary periodical published in Philadelphia in the 19th century.”

# ReAct: Synergizing Reasoning and Action in LLMs

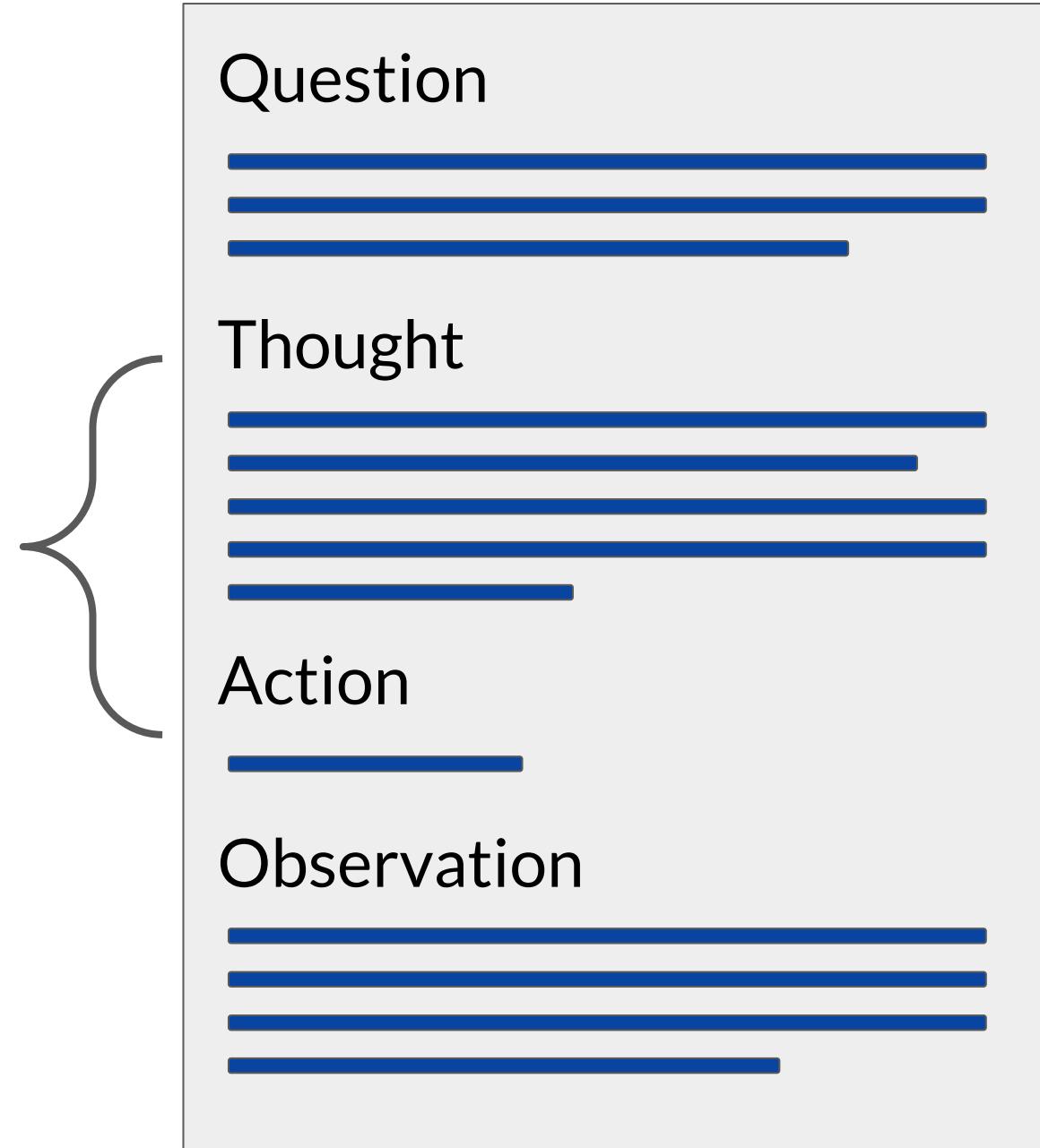


**Thought 2:**  
“Arthur’s magazine was started in 1844. I  
need to search First for Women next.”

**Action 2:**  
**search[First for Women]**

**Observation 2:**  
“First for Women is a woman’s magazine  
published by Bauer Media Group in the  
USA.[1] The magazine was started in  
1989.”

# ReAct: Synergizing Reasoning and Action in LLMs



## Thought 3:

“First for Women was started in 1989.  
1844 (Arthur’s Magazine) < 1989 (First for  
Women), so Arthur’s Magazine as started  
first”

## Action 2:

**finish[Arthur’s Magazine]**

# ReAct instructions define the action space

Solve a question answering task with interleaving Thought, Action, Observation steps.

Thought can reason about the current situation, and Action can be three types:

(1) `Search[entity]`, which searches the exact entity on Wikipedia and returns the first paragraph if it exists. If not, it will return some similar entities to search.

(2) `Lookup[keyword]`, which returns the next sentence containing keyword in the current passage.

(3) `Finish[answer]`, which returns the answer and finishes the task.

Here are some examples.

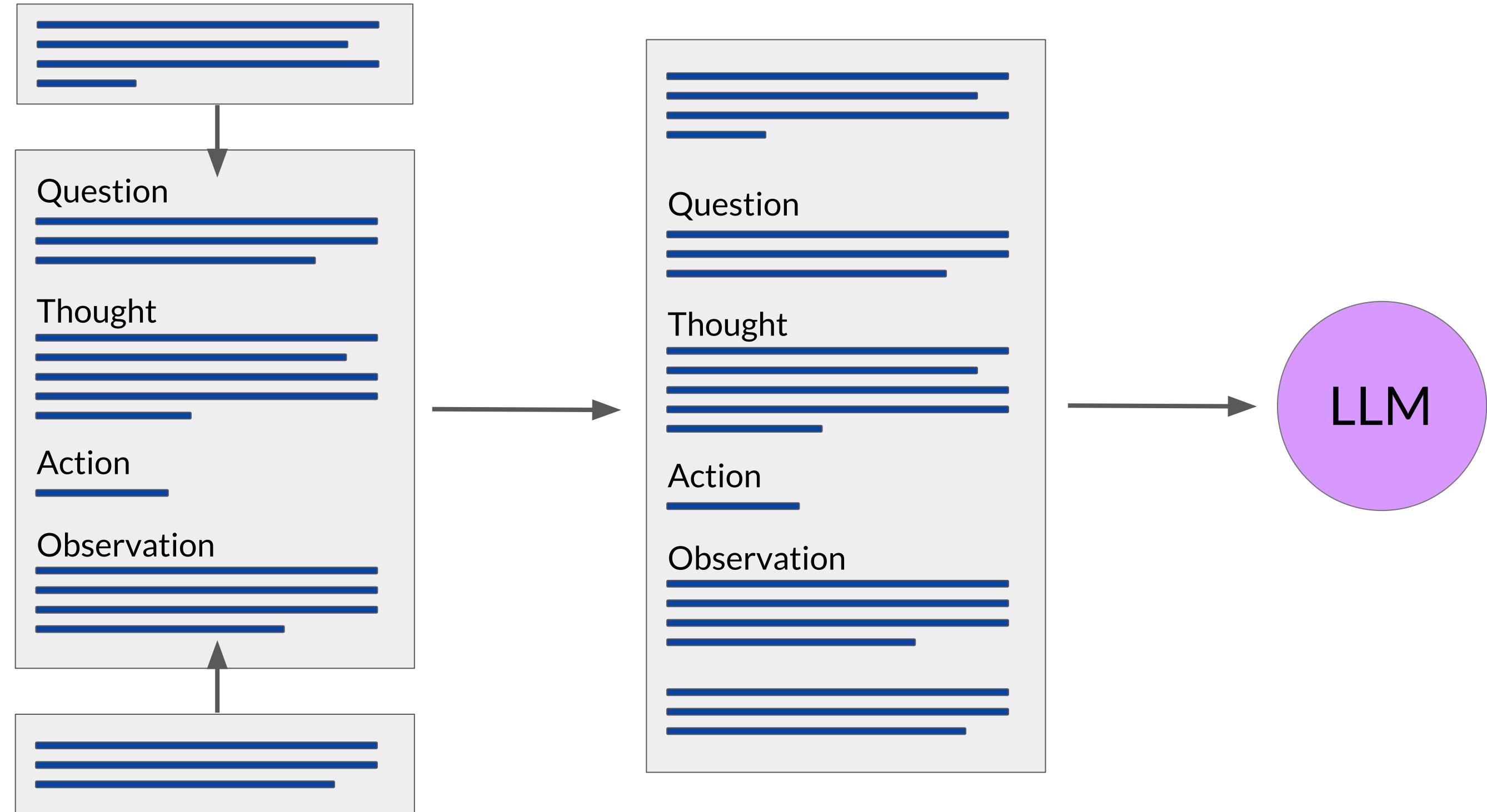
# Building up the ReAct prompt

Instructions

ReAct example

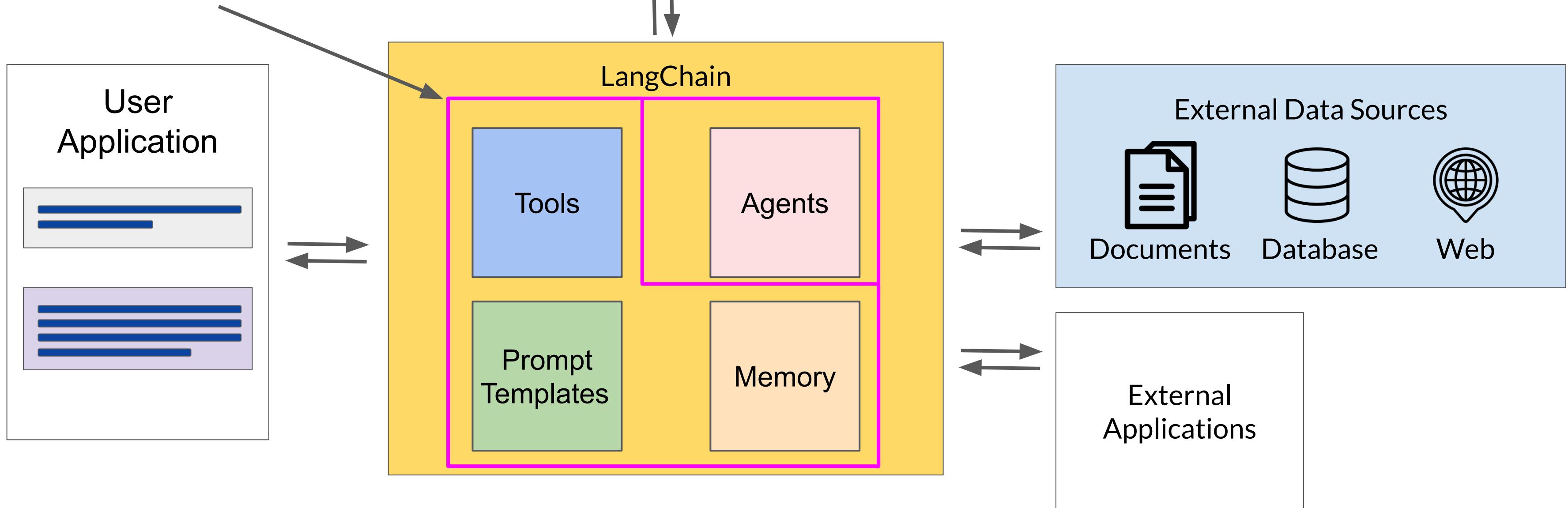
(could be more  
than one  
example)

Question to be  
answered

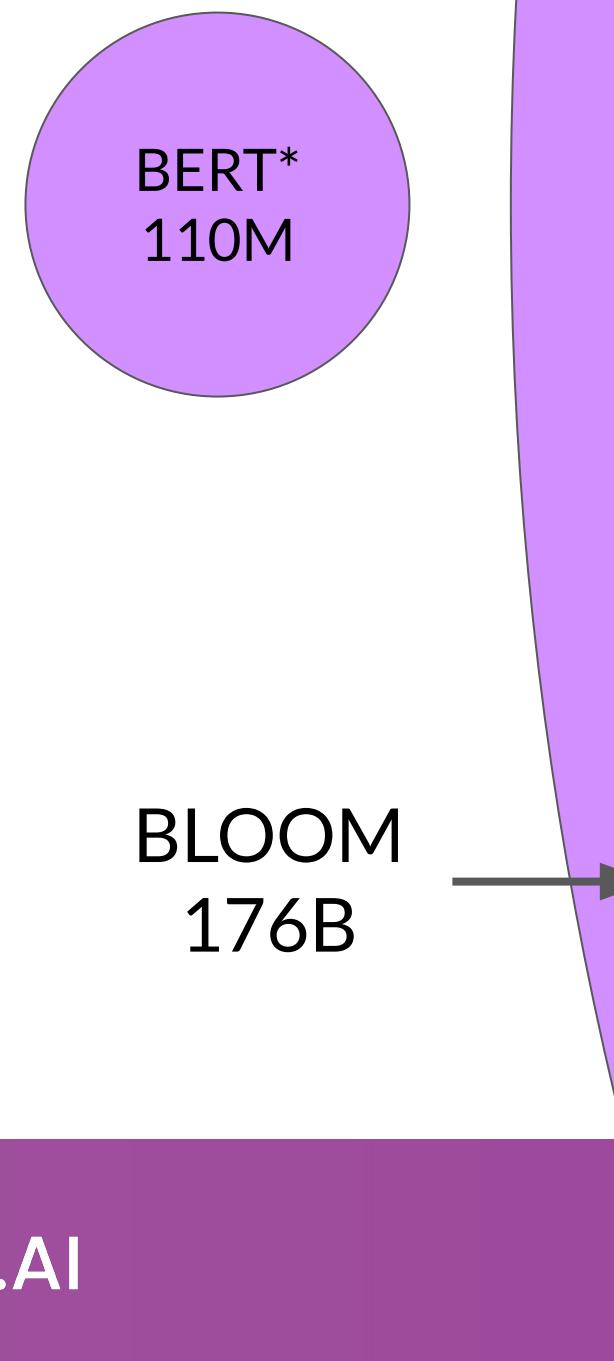


# LangChain

Combined into a “chain”



# The significance of scale: application building



# LLM powered application architectures

# Building generative applications



**Infrastructure** e.g. Training/Fine-Tuning, Serving, Application Components

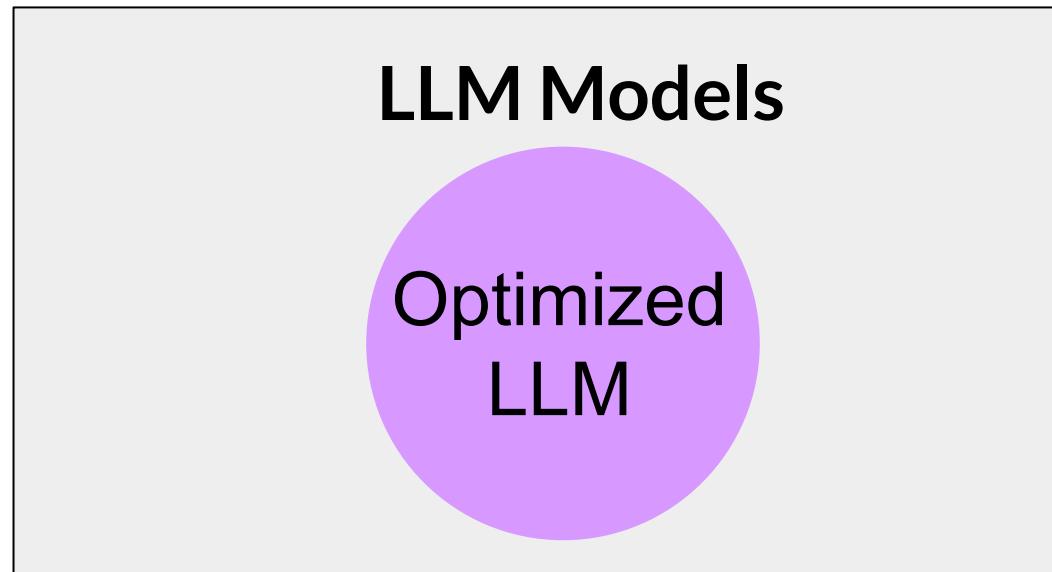
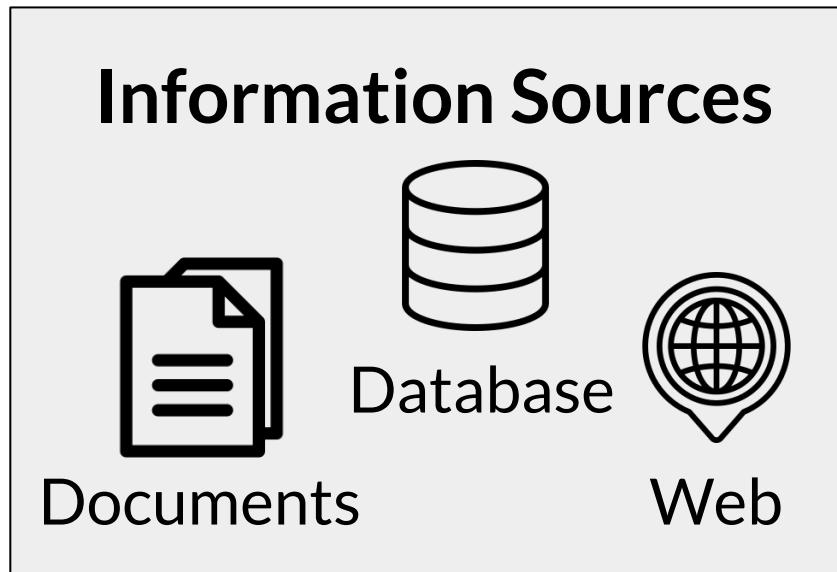
# Building generative applications

**LLM Models**  
Optimized  
LLM

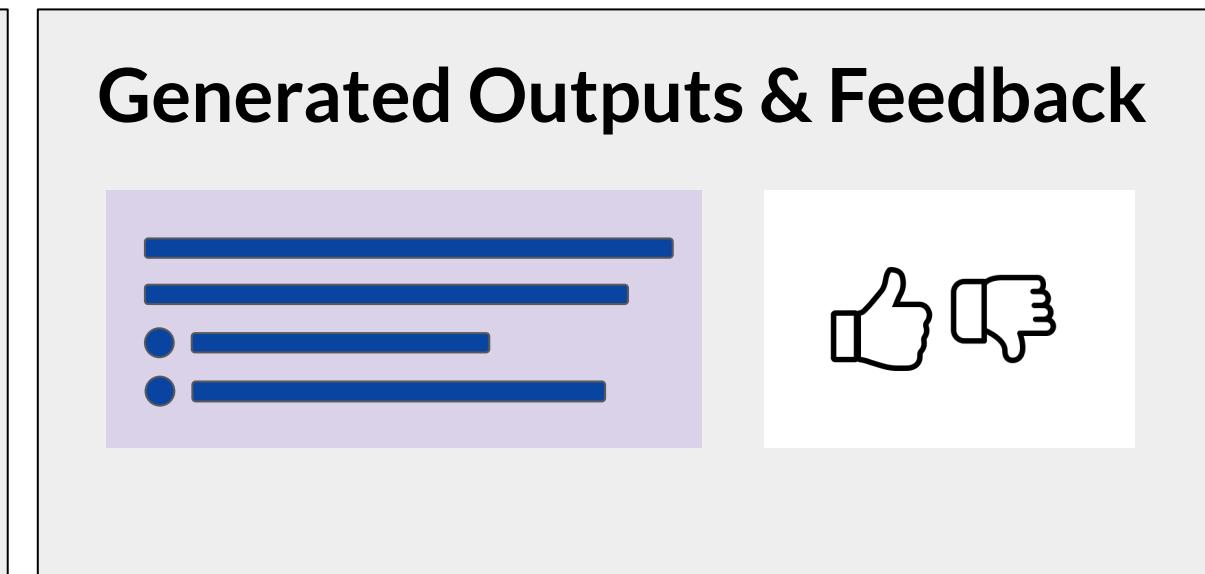
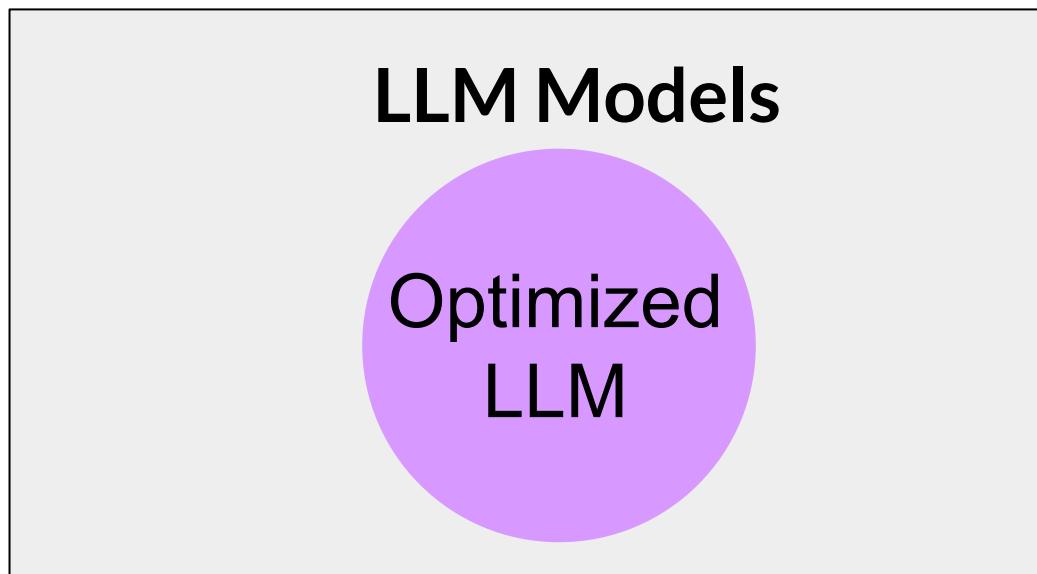
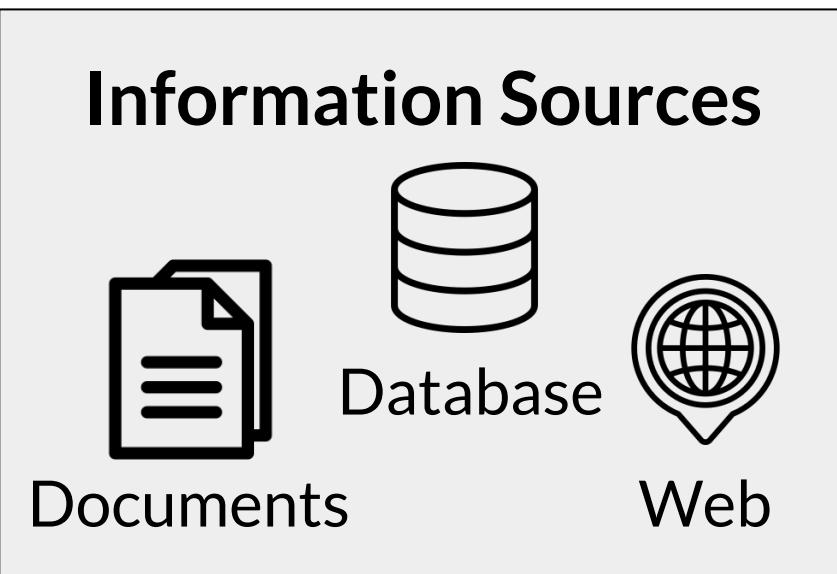


**Infrastructure** e.g. Training/Fine-Tuning, Serving, Application Components

# Building generative applications



# Building generative applications



# Building generative applications

**LLM Tools & Frameworks** e.g. LangChain, Model Hubs

**Information Sources**



Documents



Database

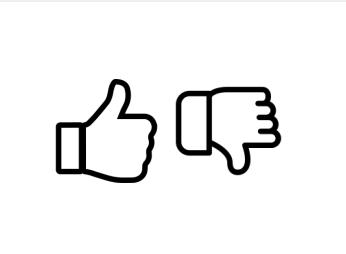
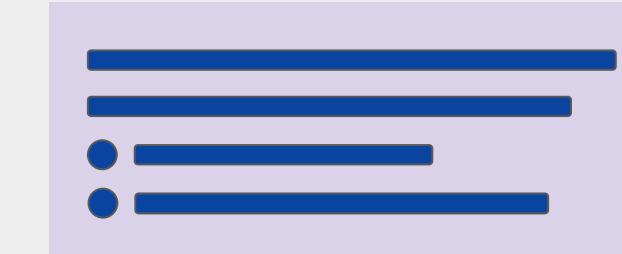


Web

**LLM Models**

Optimized  
LLM

**Generated Outputs & Feedback**



**Infrastructure** e.g. Training/Fine-Tuning, Serving, Application Components

# Building generative applications

**Application Interfaces** e.g. Websites, Mobile Applications, APIs, etc.

**LLM Tools & Frameworks** e.g. LangChain, Model Hubs

**Information Sources**



Documents



Database

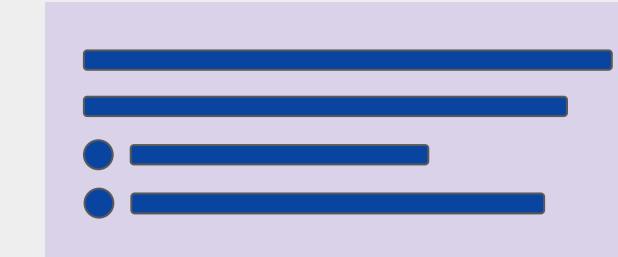


Web

**LLM Models**

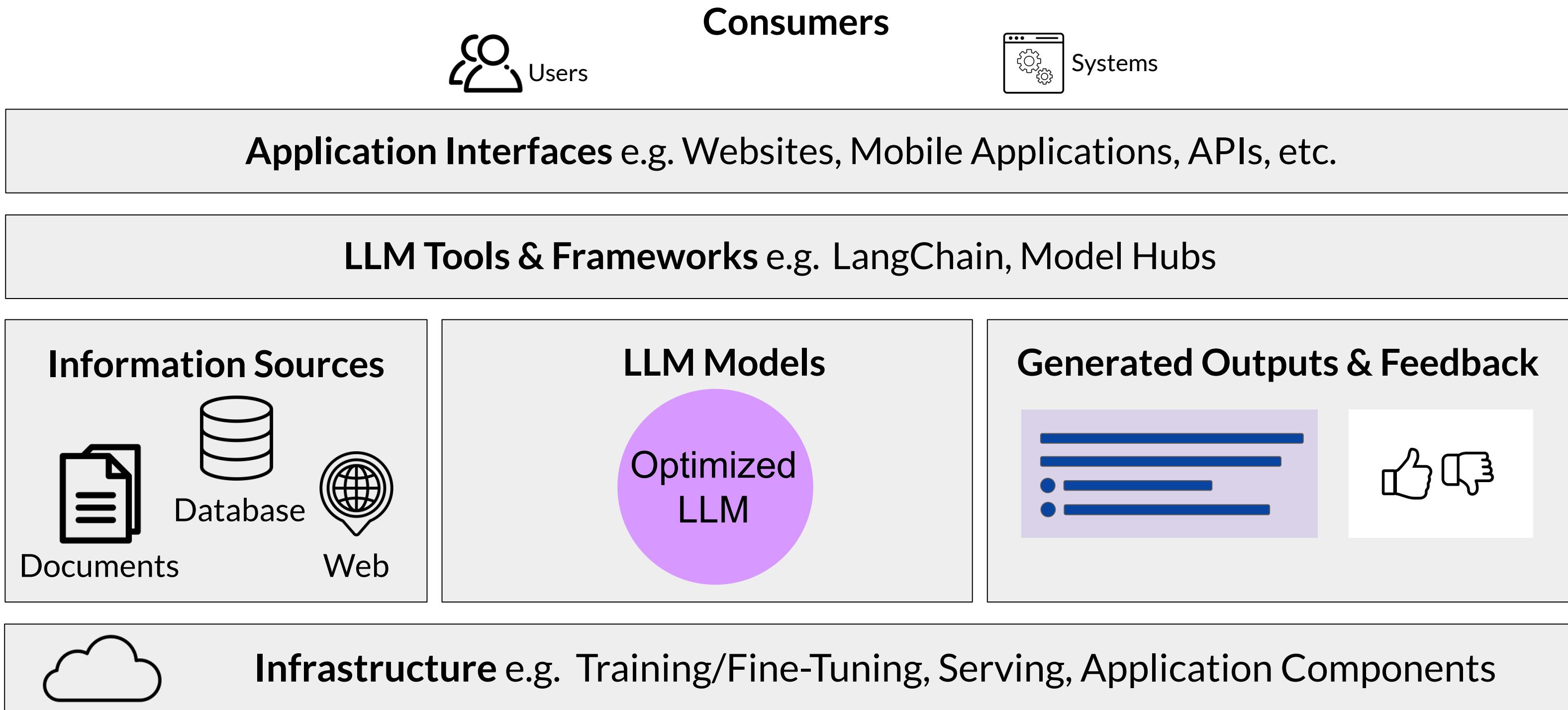
Optimized  
LLM

**Generated Outputs & Feedback**

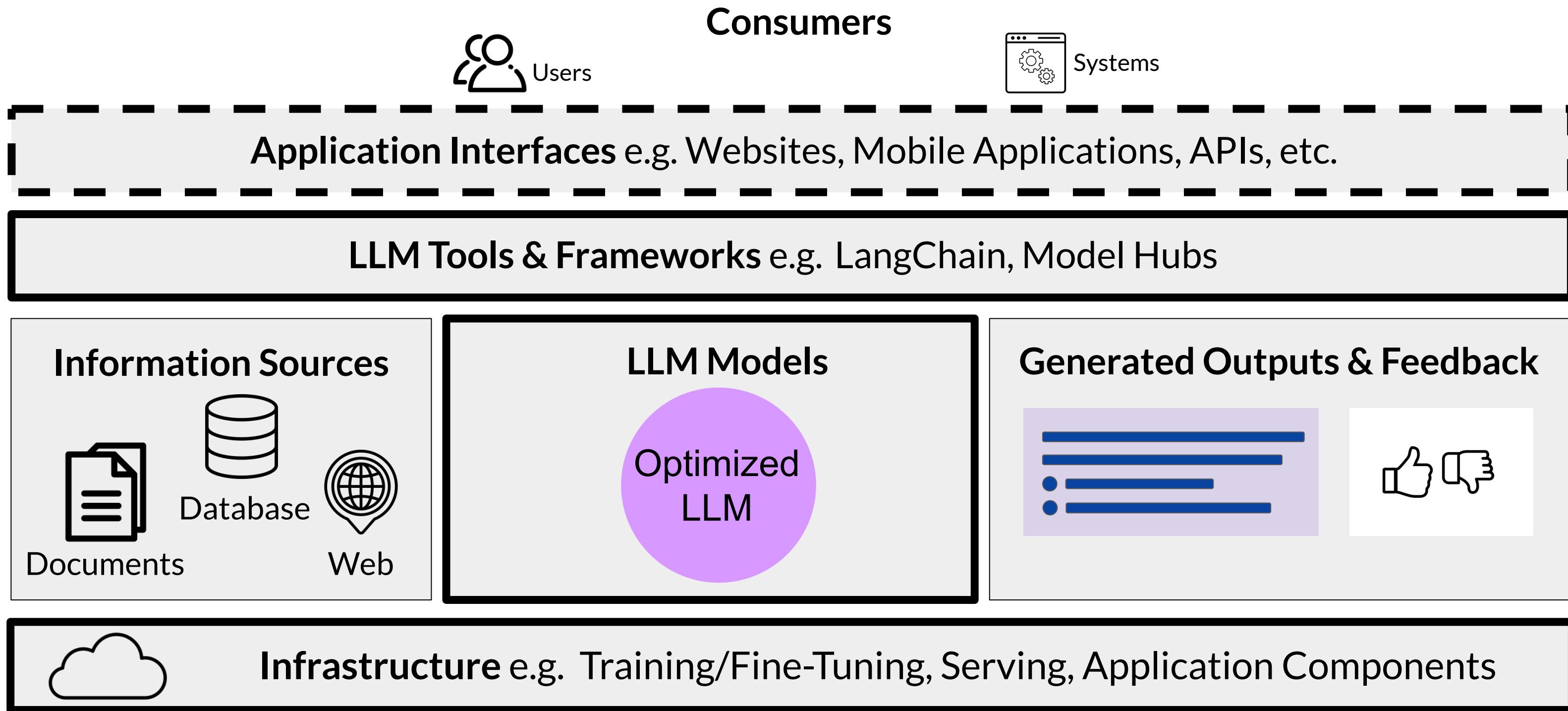


**Infrastructure** e.g. Training/Fine-Tuning, Serving, Application Components

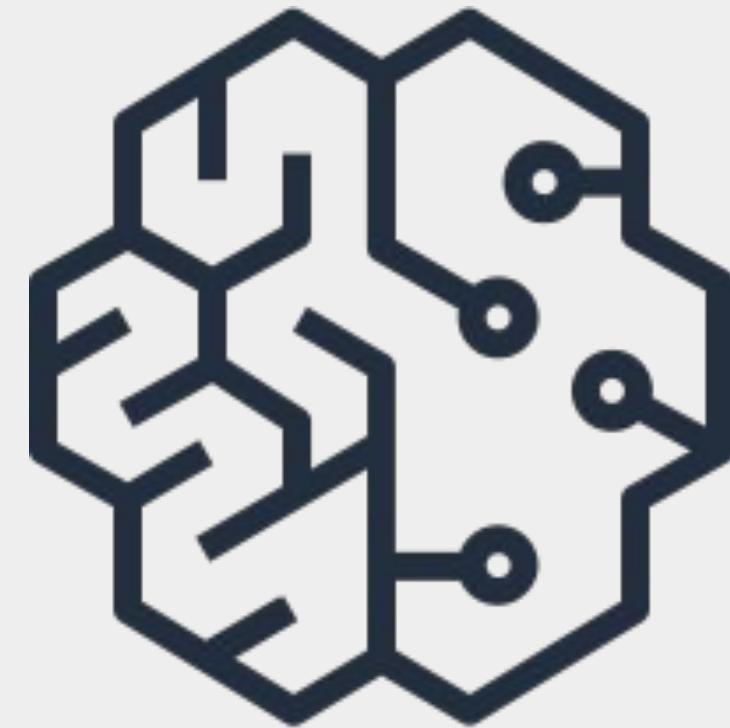
# Building generative applications

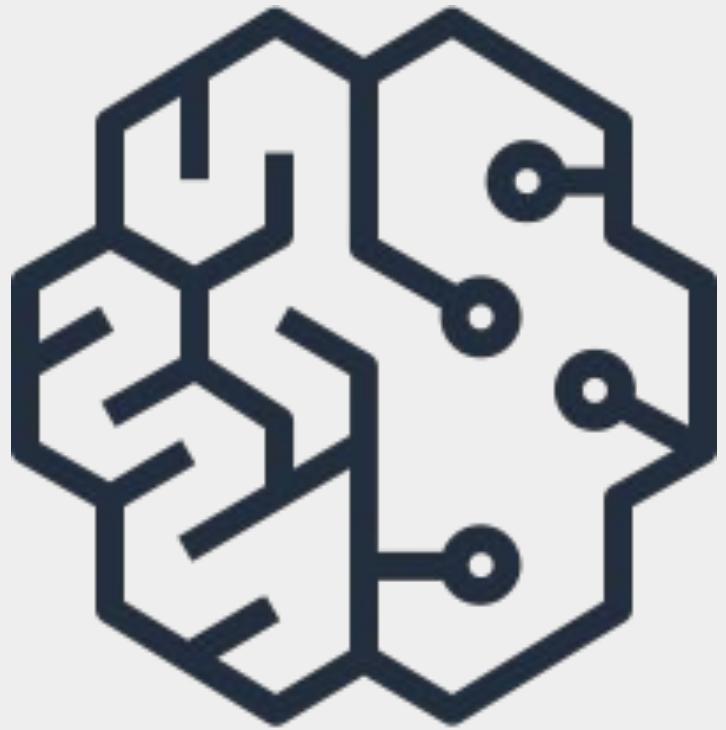


# Building generative applications



Conclusion,  
Responsible AI,  
and on-going  
research



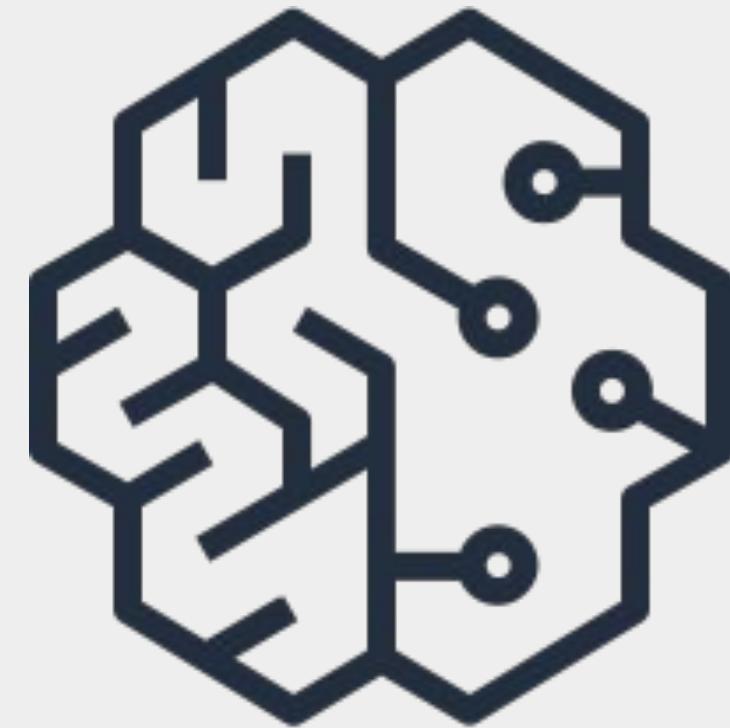


# Responsible AI

Dr. Nashlie Sephus

# Responsible AI

Dr. Nashlie  
Sephus



# Responsible AI

## Dr. Nashlie Sephus

# On-going research

- Responsible AI

# Responsible AI

# Special challenges of responsible generative AI

- Toxicity
- Hallucinations
- Intellectual Property

# Toxicity

*LLM returns responses that can be potentially harmful or discriminatory towards protected groups or protected attributes*

How to mitigate?

- Careful curation of training data
- Train guardrail models to filter out unwanted content
- Diverse group of human annotators

# Hallucinations

*LLM generates factually incorrect content*

How to mitigate?

- Educate users about how generative AI works
- Add disclaimers
- Augment LLMs with independent, verified citation databases
- Define intended/unintended use cases

# Intellectual Property

*Ensure people aren't plagiarizing, make sure there aren't any copyright issues*

How to mitigate?

- Mix of technology, policy, and legal mechanisms
- Machine "unlearning"
- Filtering and blocking approaches

# Responsibly build and use generative AI models

- Define use cases: the more specific/narrow, the better
- Assess risks for each use case
- Evaluate performance for each use case
- Iterate over entire AI lifecycle

# On-going research

- Responsible AI
- Scale models and predict performance
- More efficiencies across model development lifecycle
- Increased and emergent LLM capabilities