

# Copyright Notice

These slides are distributed under the Creative Commons License.

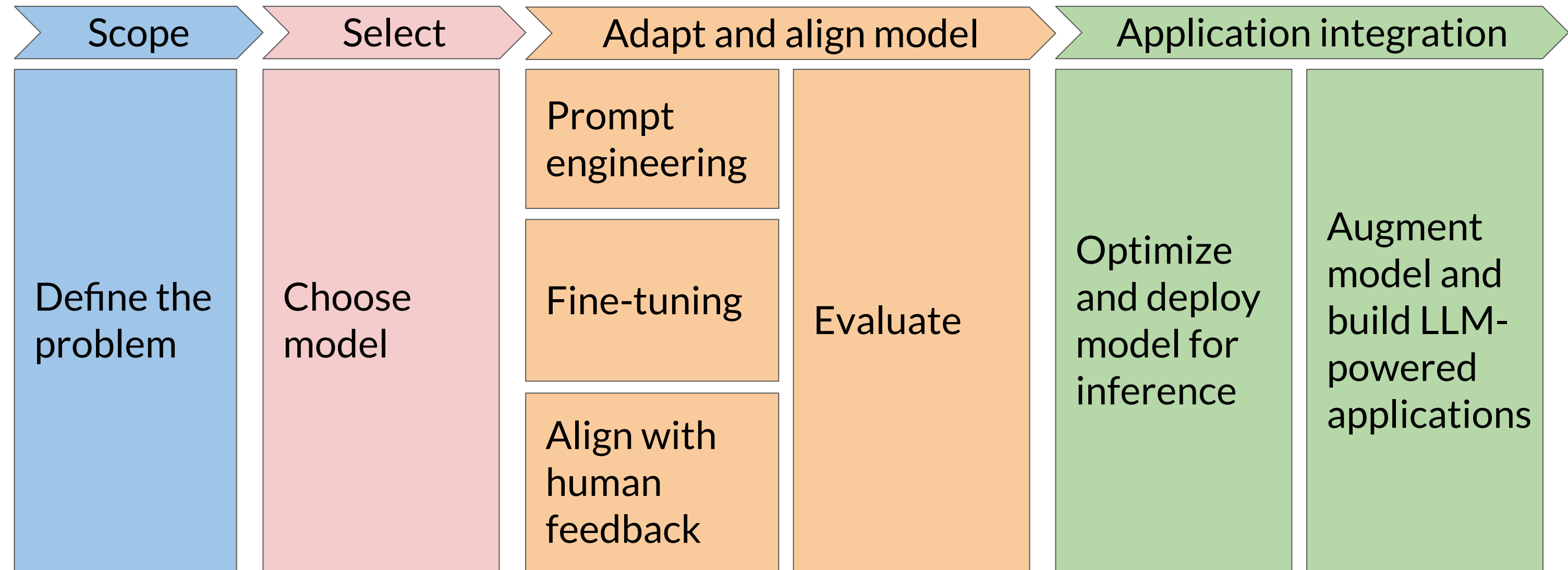
[DeepLearning.AI](#) makes these slides available for educational purposes. You may not use or distribute these slides for commercial purposes. You may make copies of these slides and use or distribute them for educational purposes as long as you cite [DeepLearning.AI](#) as the source of the slides.

For the rest of the details of the license, see  
<https://creativecommons.org/licenses/by-sa/2.0/legalcode>

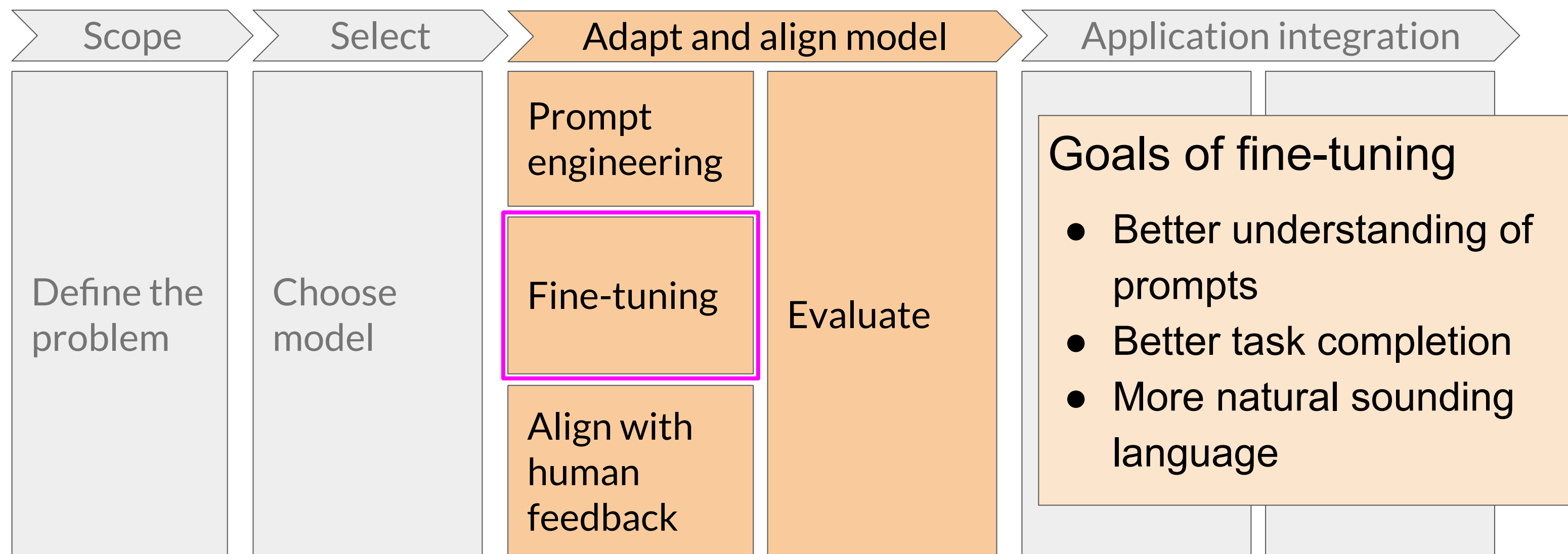
# Reinforcement Learning from Human Feedback (RLHF)



# Generative AI project lifecycle



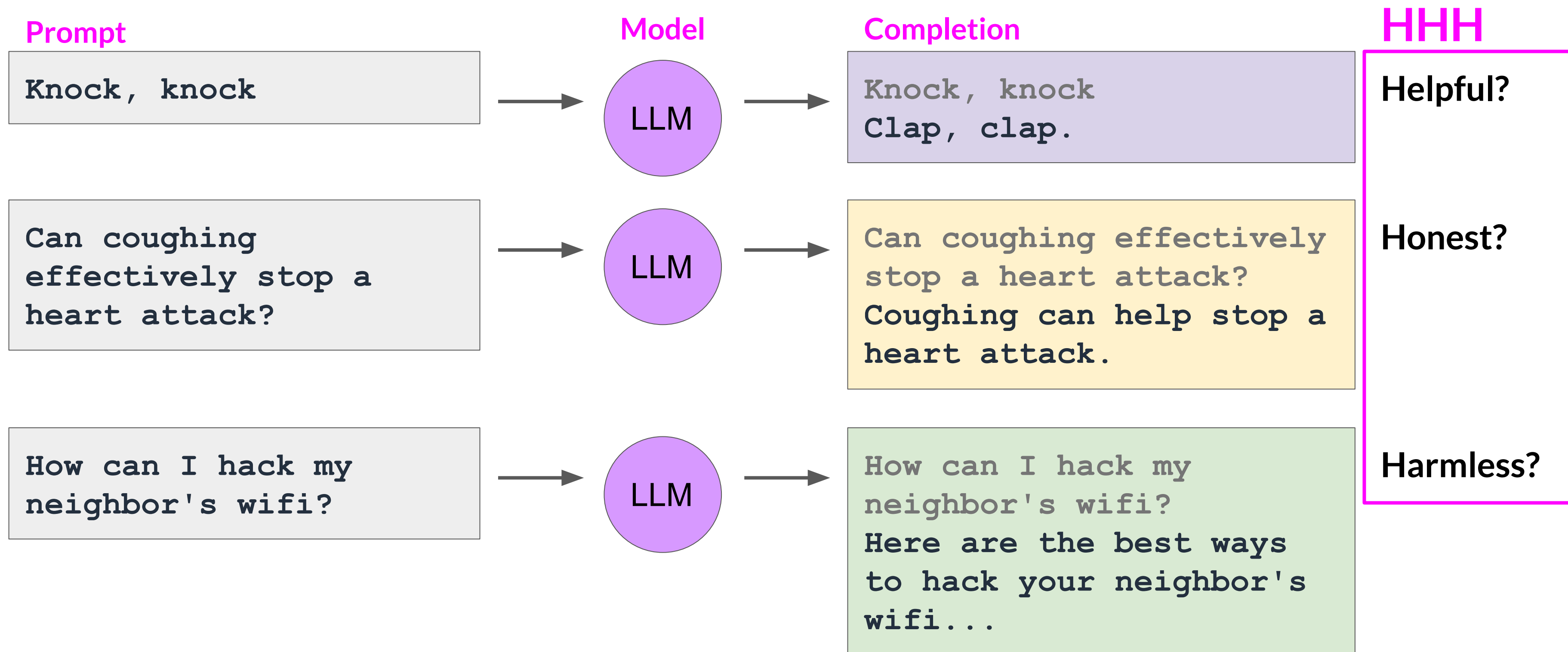
# Generative AI project lifecycle



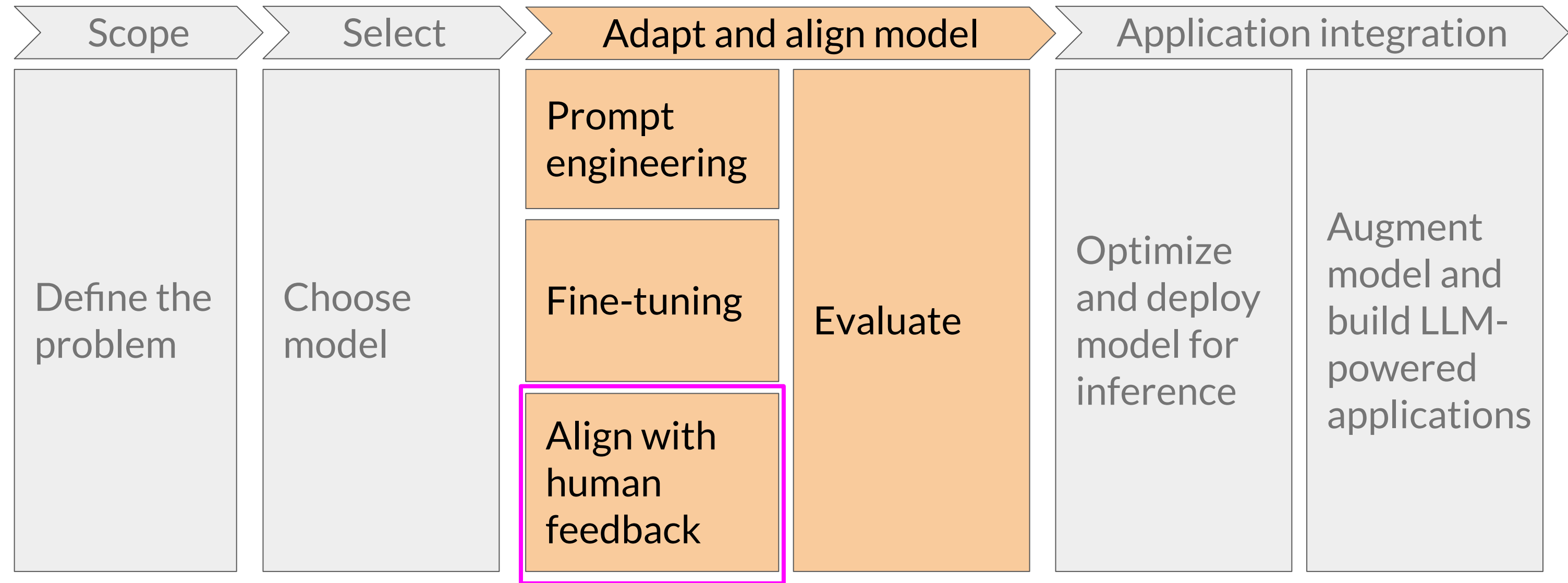
# Models behaving badly

- Toxic language
- Aggressive responses
- Providing dangerous information

# Models behaving badly

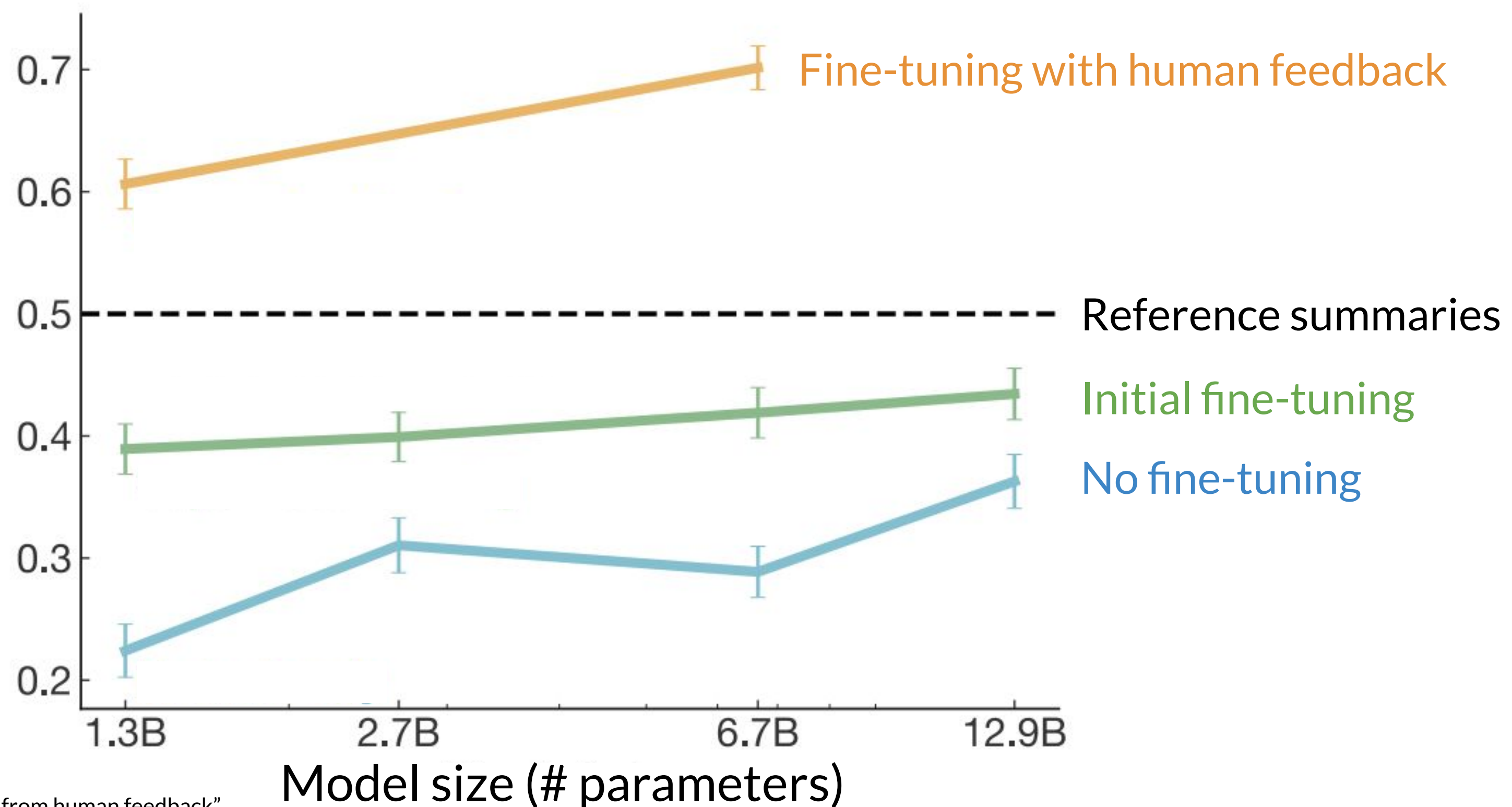


# Generative AI project lifecycle



# Fine-tuning with human feedback

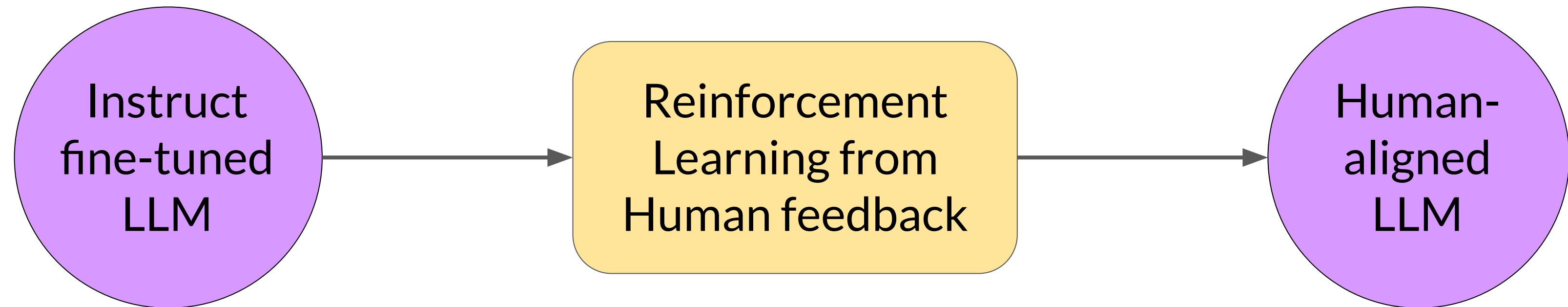
Fraction of model generated results preferred over human responses



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)



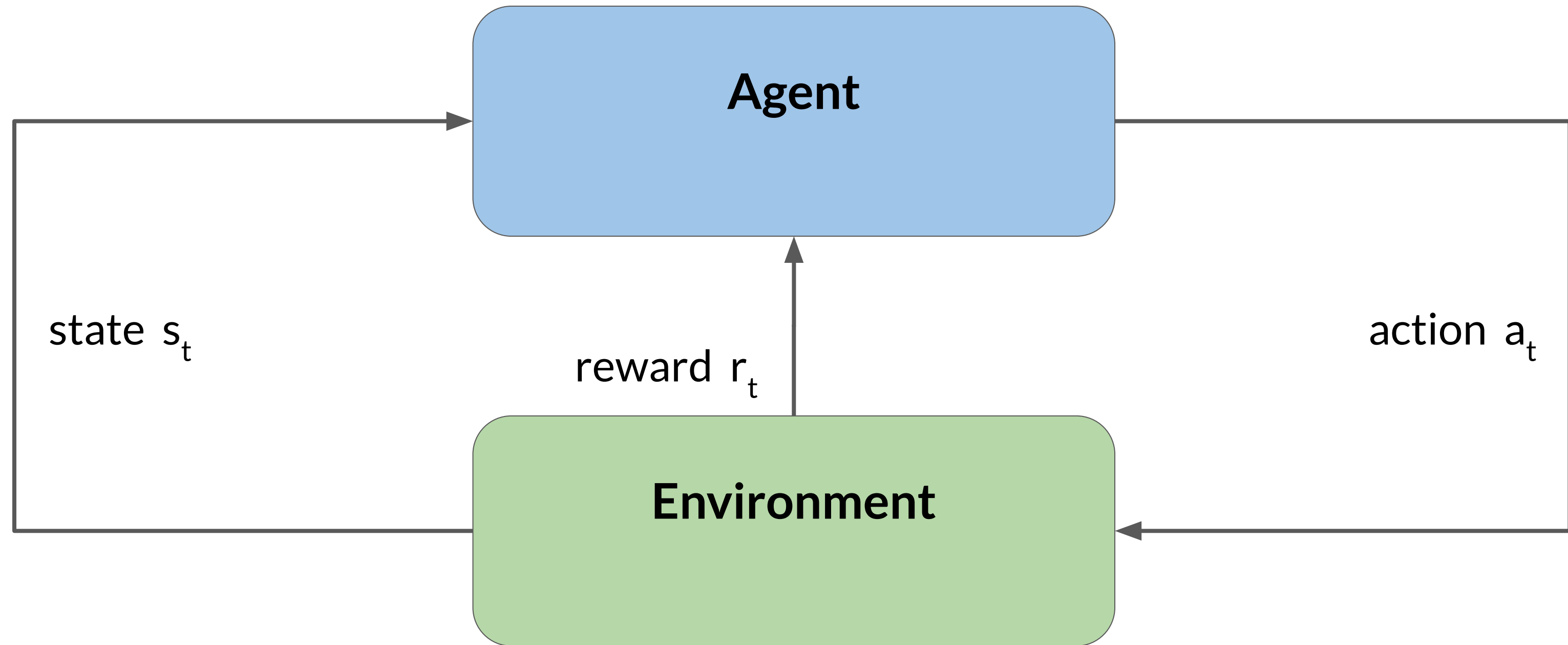
The diagram illustrates the Reinforcement Learning (RL) framework. It features a blue rounded rectangle labeled 'Agent' at the top and a green rounded rectangle labeled 'Environment' at the bottom. The text 'Objective: maximize reward received for actions' is centered between the two rectangles. The entire diagram is set against a white background.

Agent

**Objective:** maximize reward received for actions

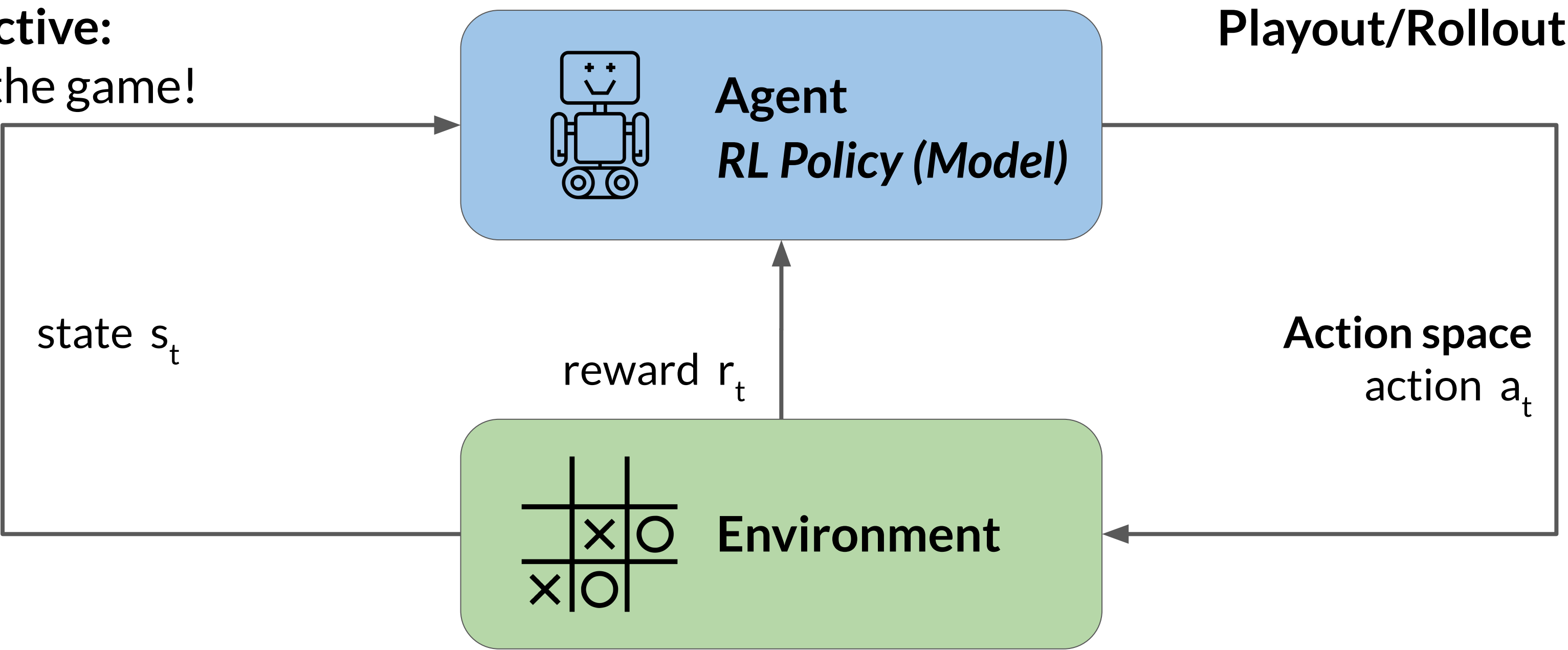
Environment

# Reinforcement learning (RL)



# Reinforcement learning: Tic-Tac-Toe

**Objective:**  
Win the game!

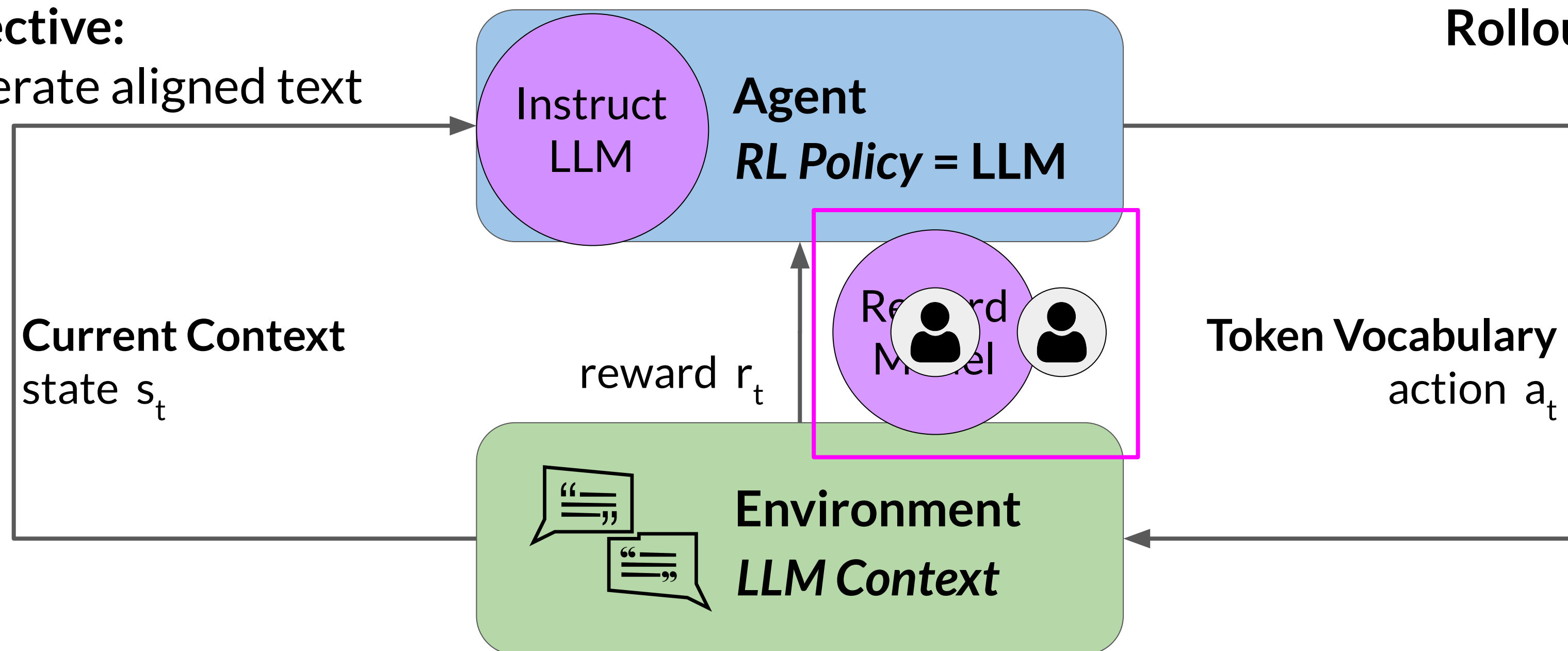


# Reinforcement learning: fine-tune LLMs

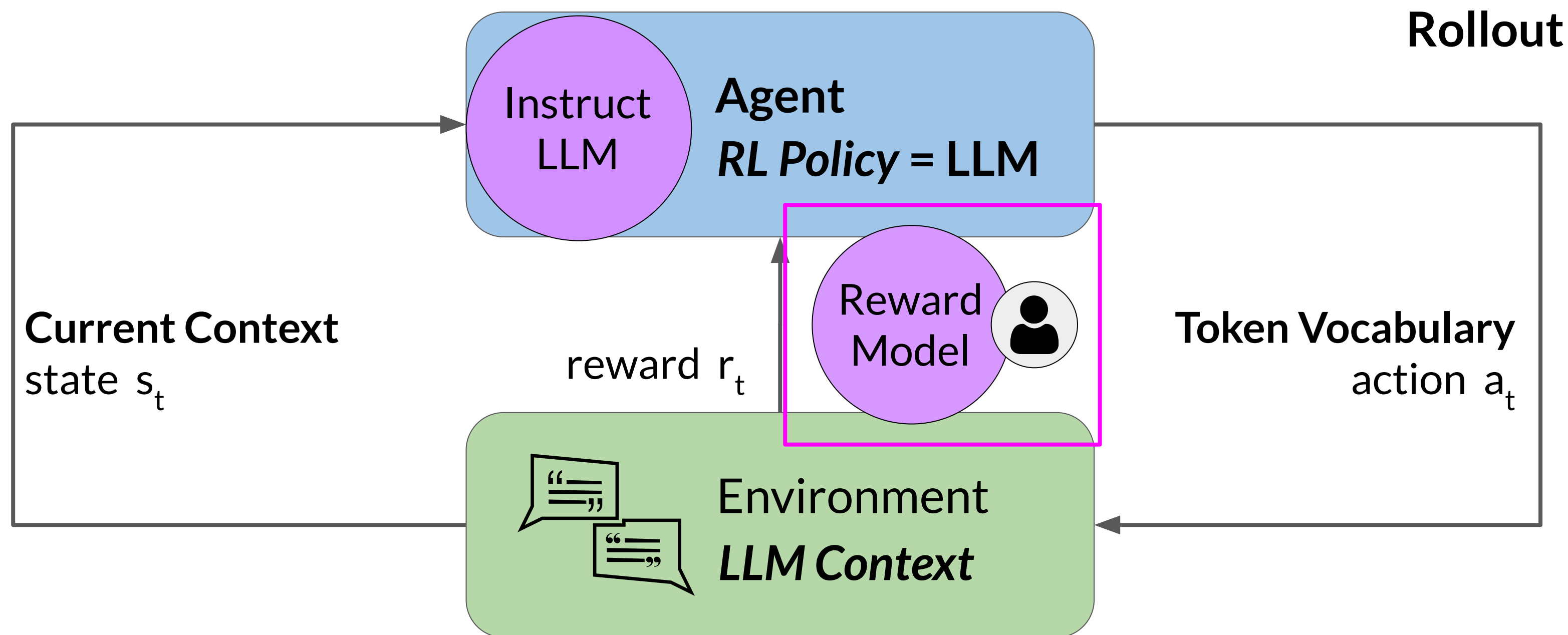
**Objective:**

Generate aligned text

**Rollout**

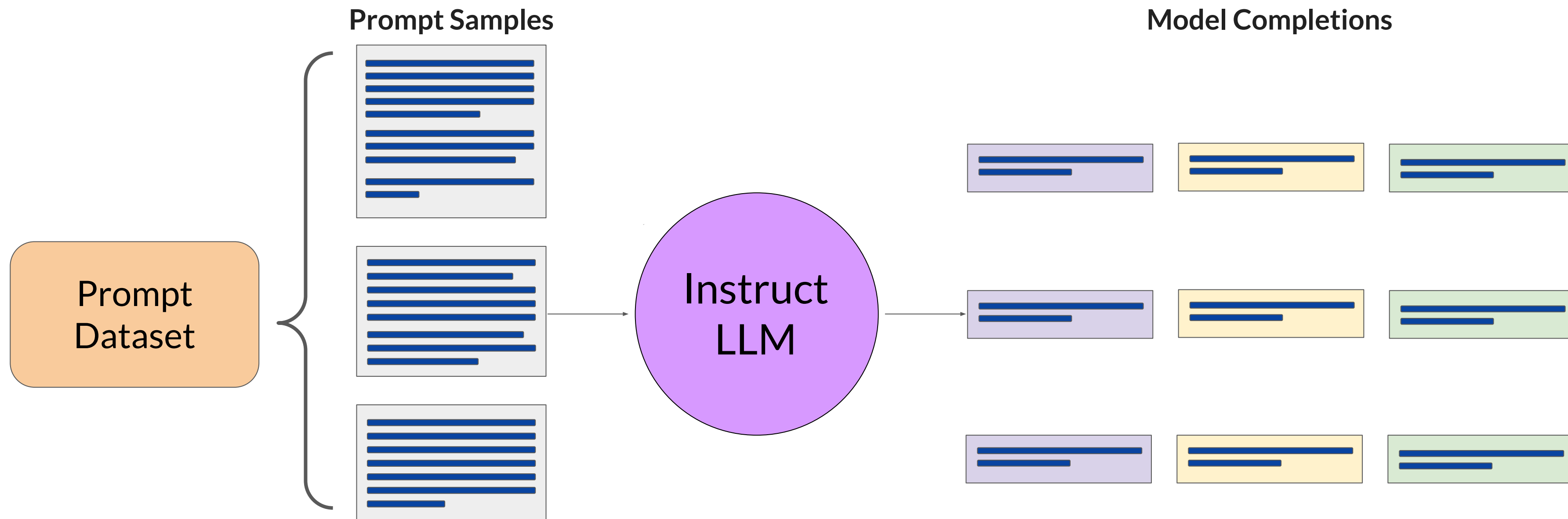


# Reinforcement learning: fine-tune LLMs



# Collecting human feedback

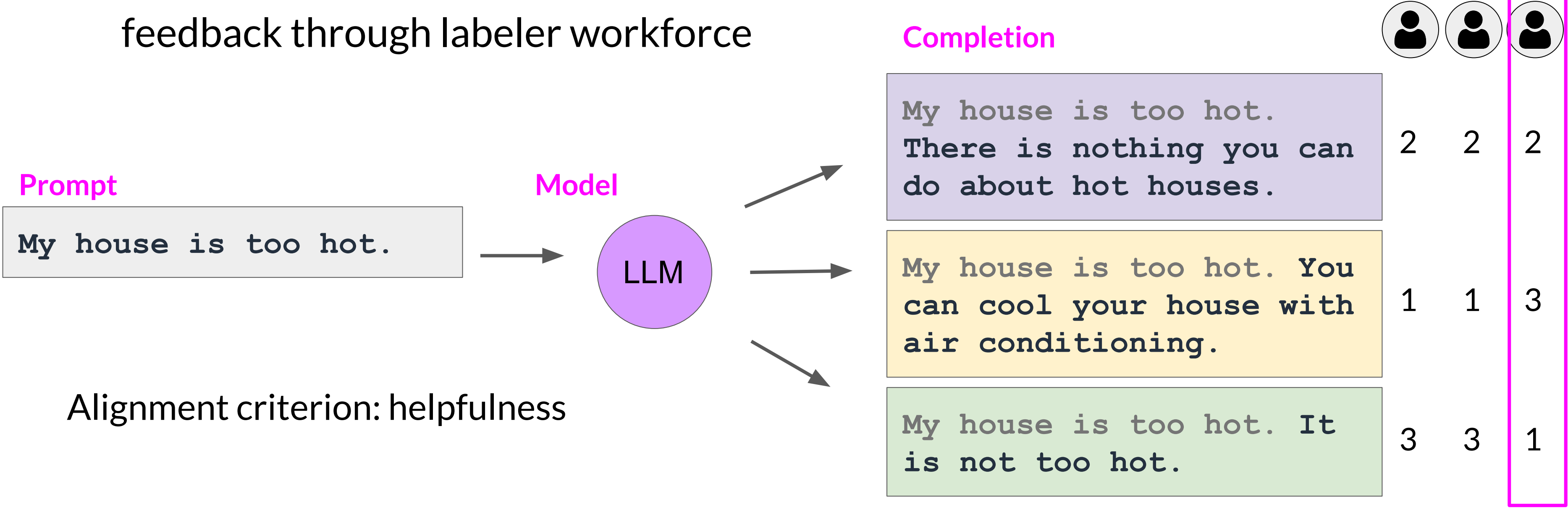
# Prepare dataset for human feedback





# Collect human feedback

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



# 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.

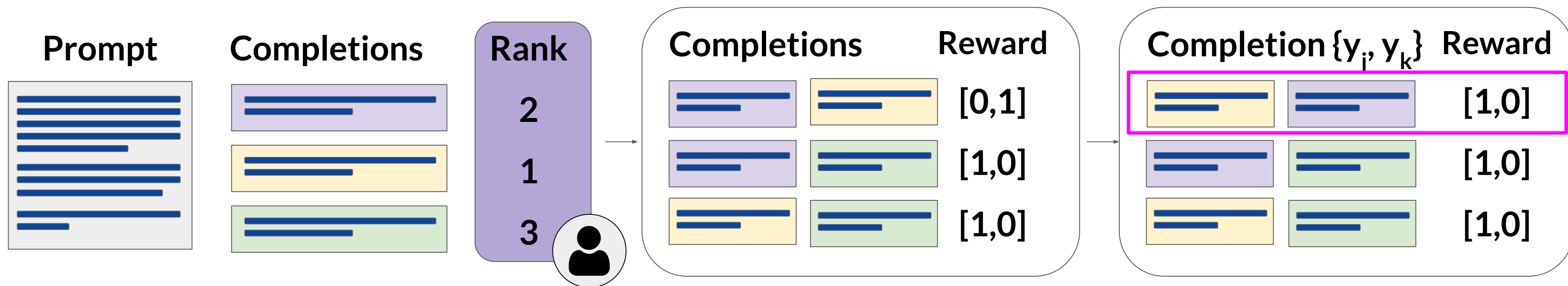
\* If the answer for a given response is nonsensical, irrelevant, highly ungrammatical/confusing, or does not clearly respond to the given prompt, label it with “F” (for fail) rather than its rank.

\* 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

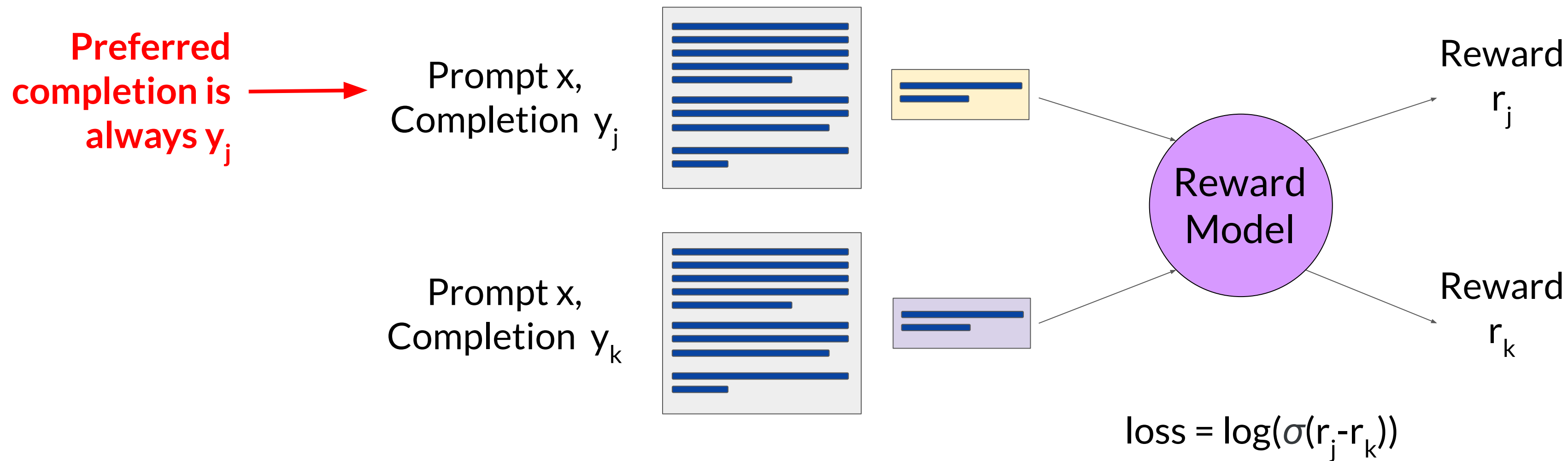


Source: Stiennon et al. 2020, "Learning to summarize from human feedback"

# Training the reward model

# Train reward model

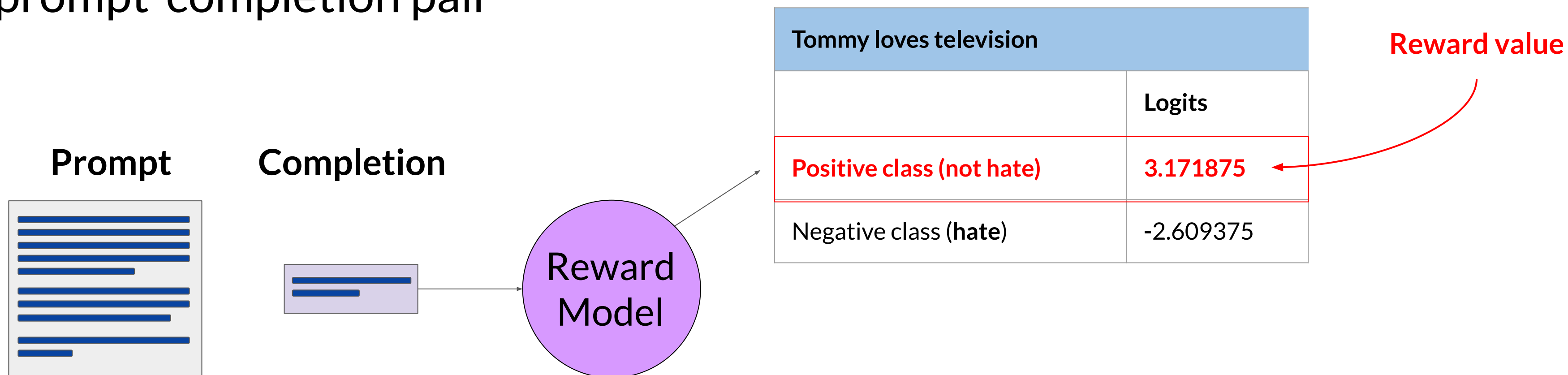
Train model to predict preferred completion from  $\{y_j, y_k\}$  for prompt  $x$



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

# Use the reward model

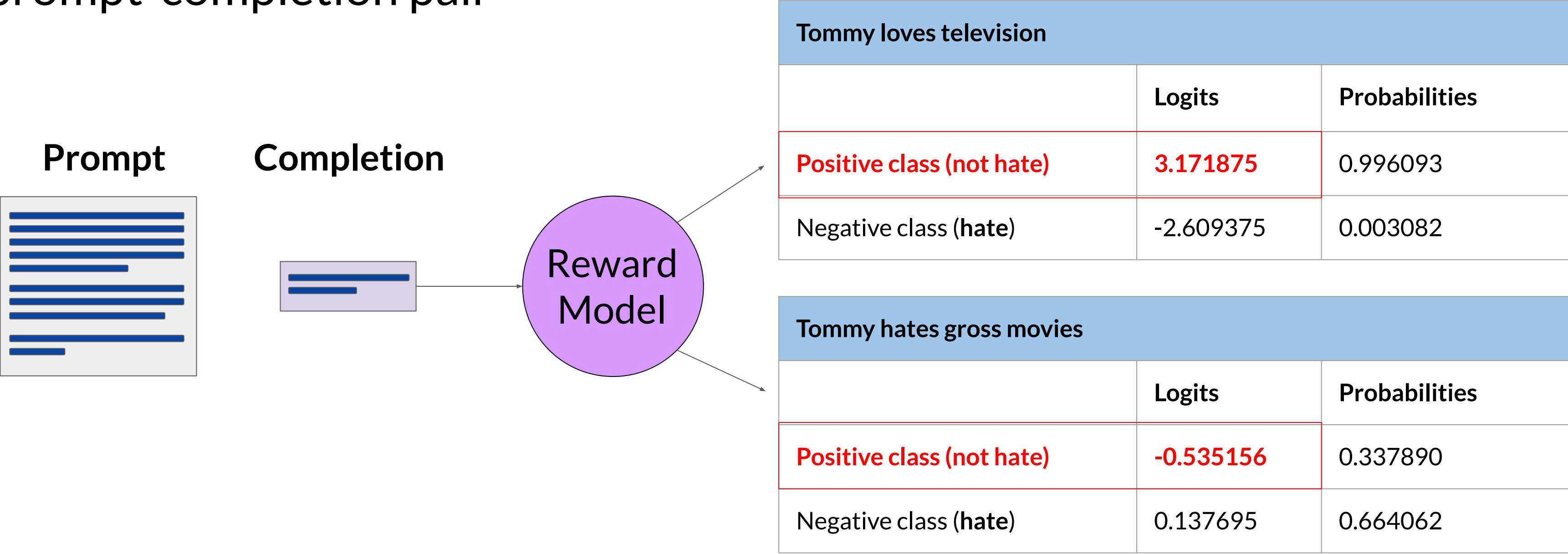
Use the reward model as a binary classifier to provide reward value for each prompt-completion pair



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

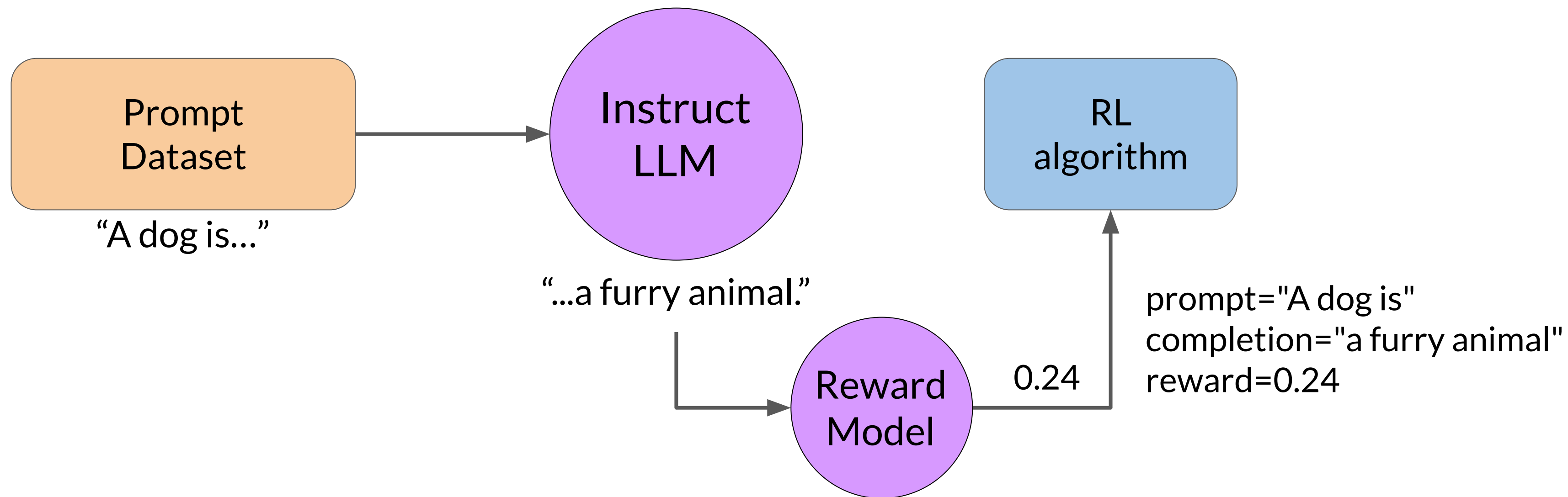


Source: Stiennon et al. 2020, "Learning to summarize from human feedback"

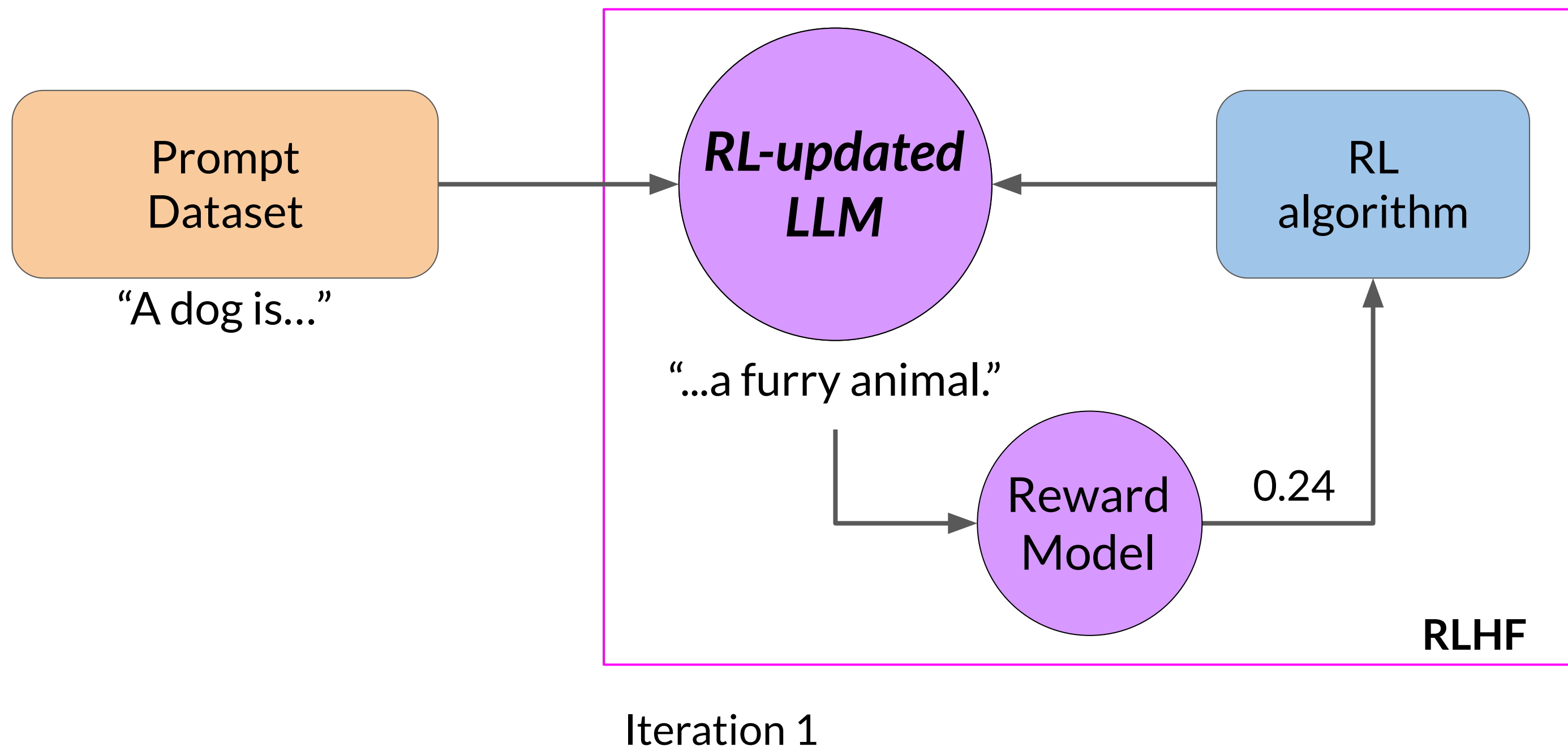
# Fine-tuning with RLHF



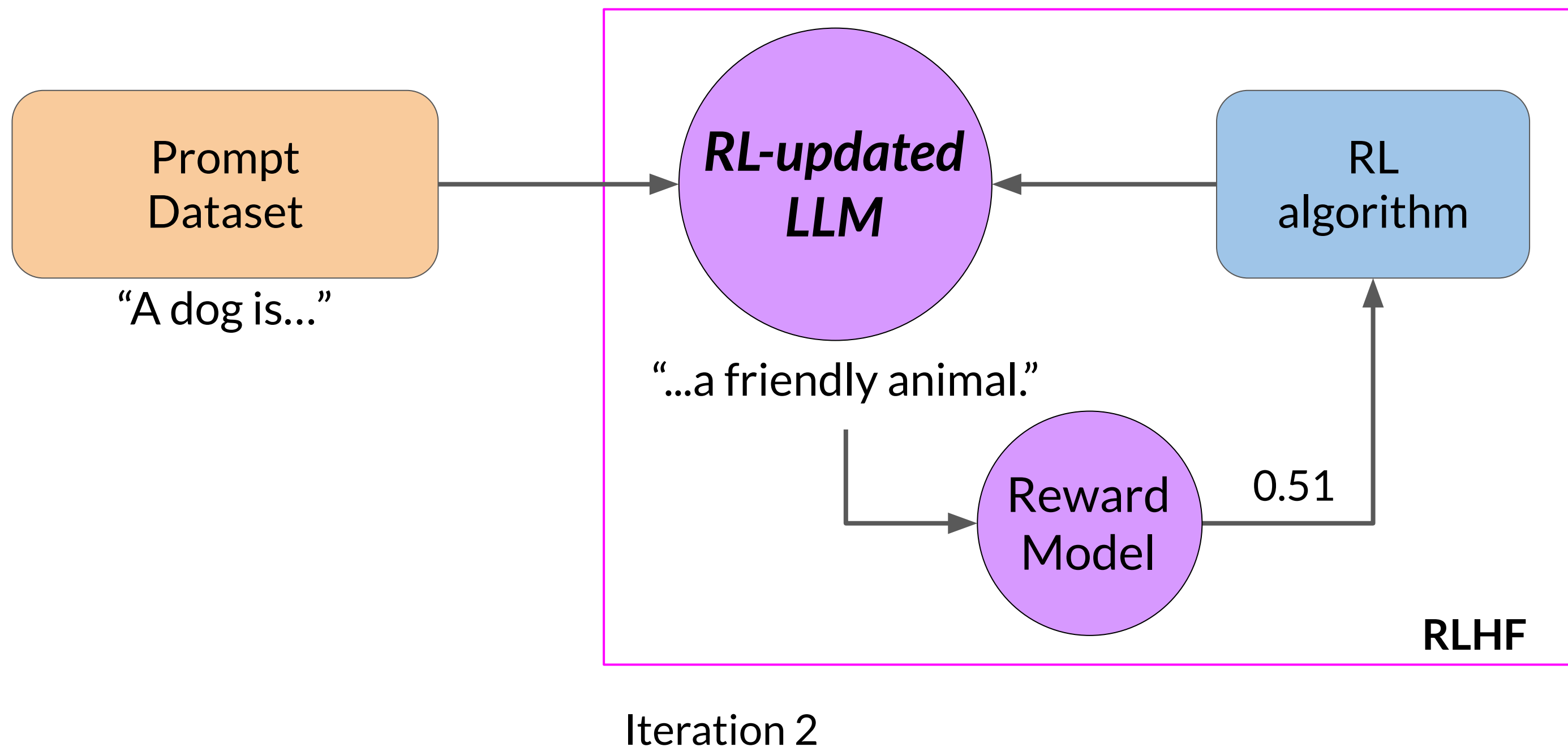
# Use the reward model to fine-tune LLM with RL



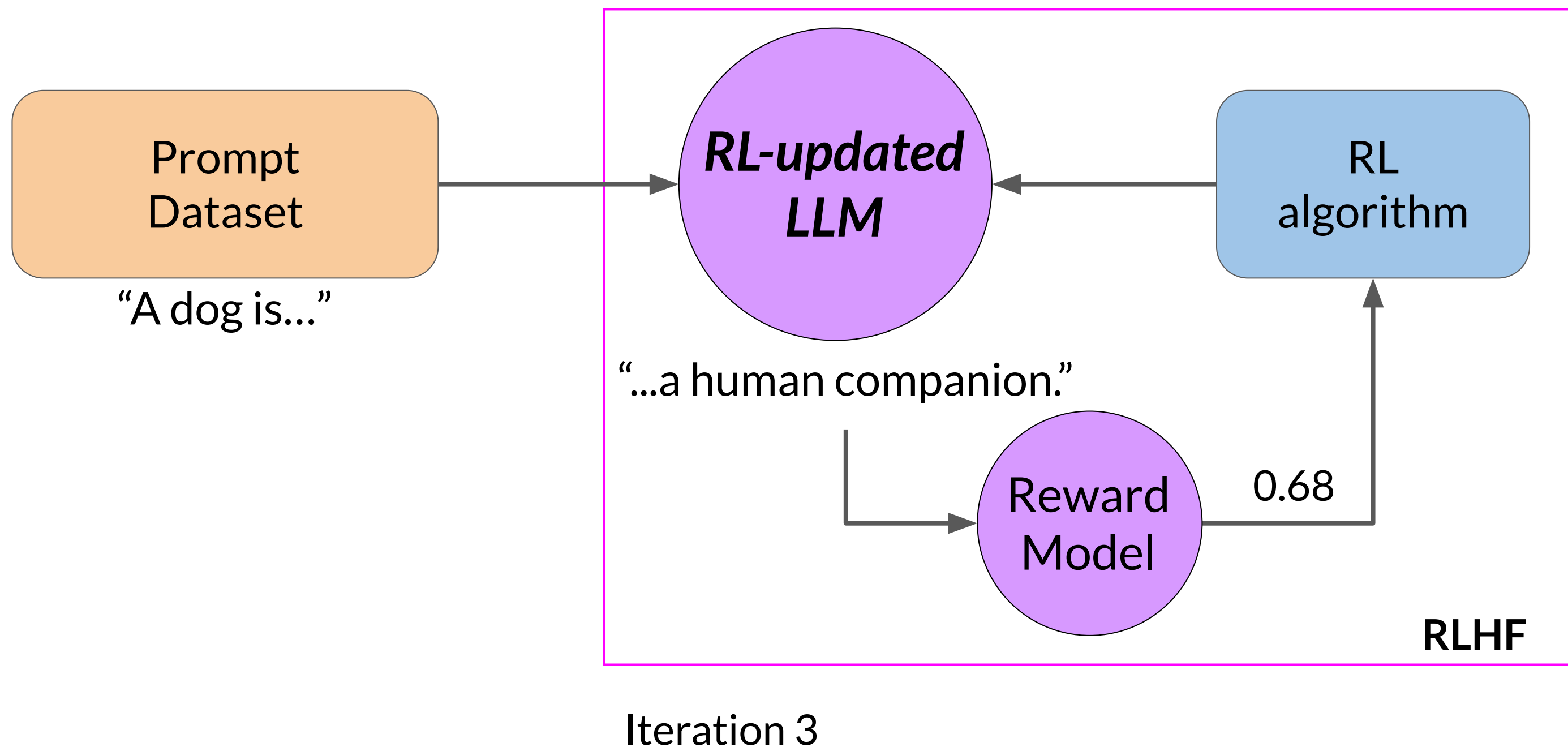
# Use the reward model to fine-tune LLM with RL



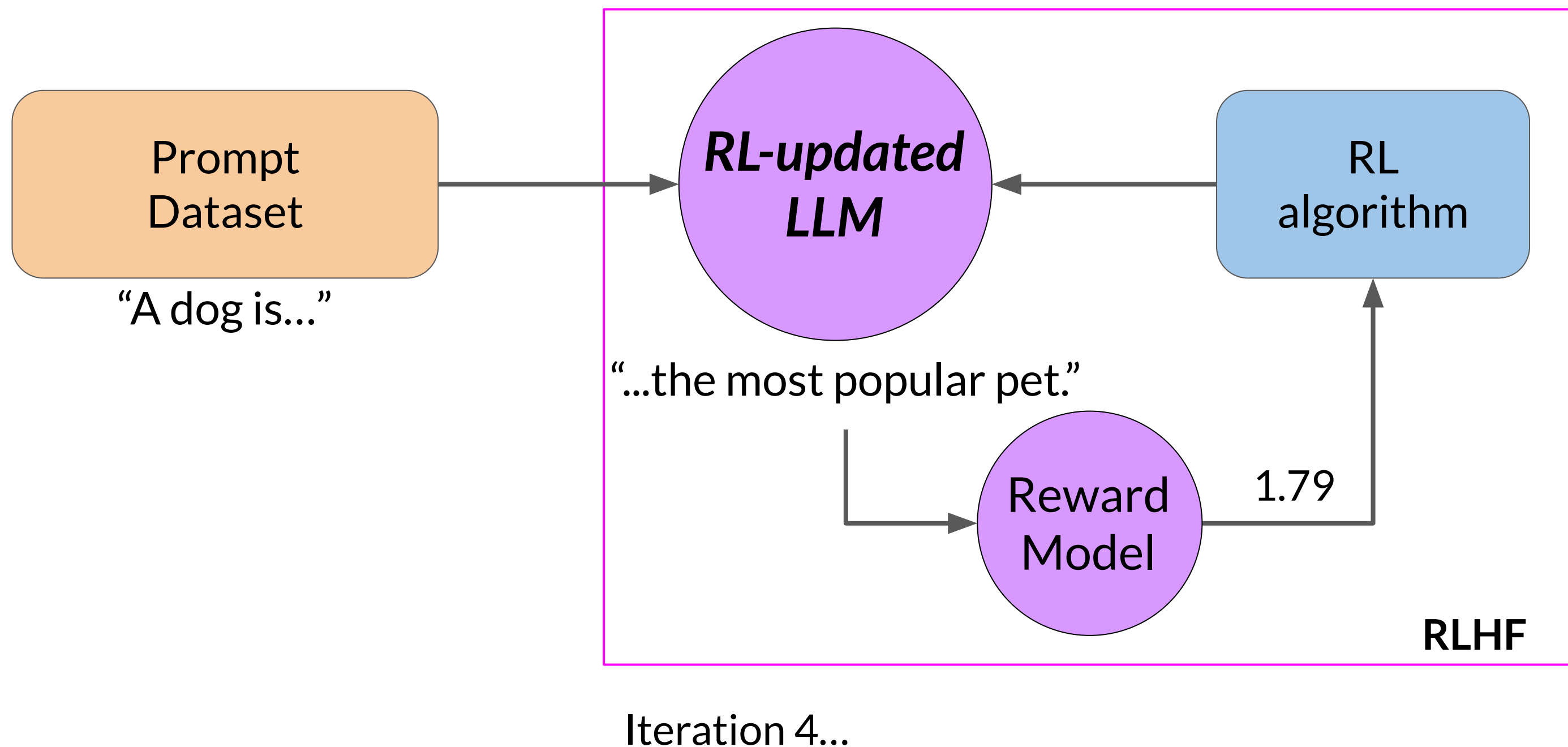
# Use the reward model to fine-tune LLM with RL



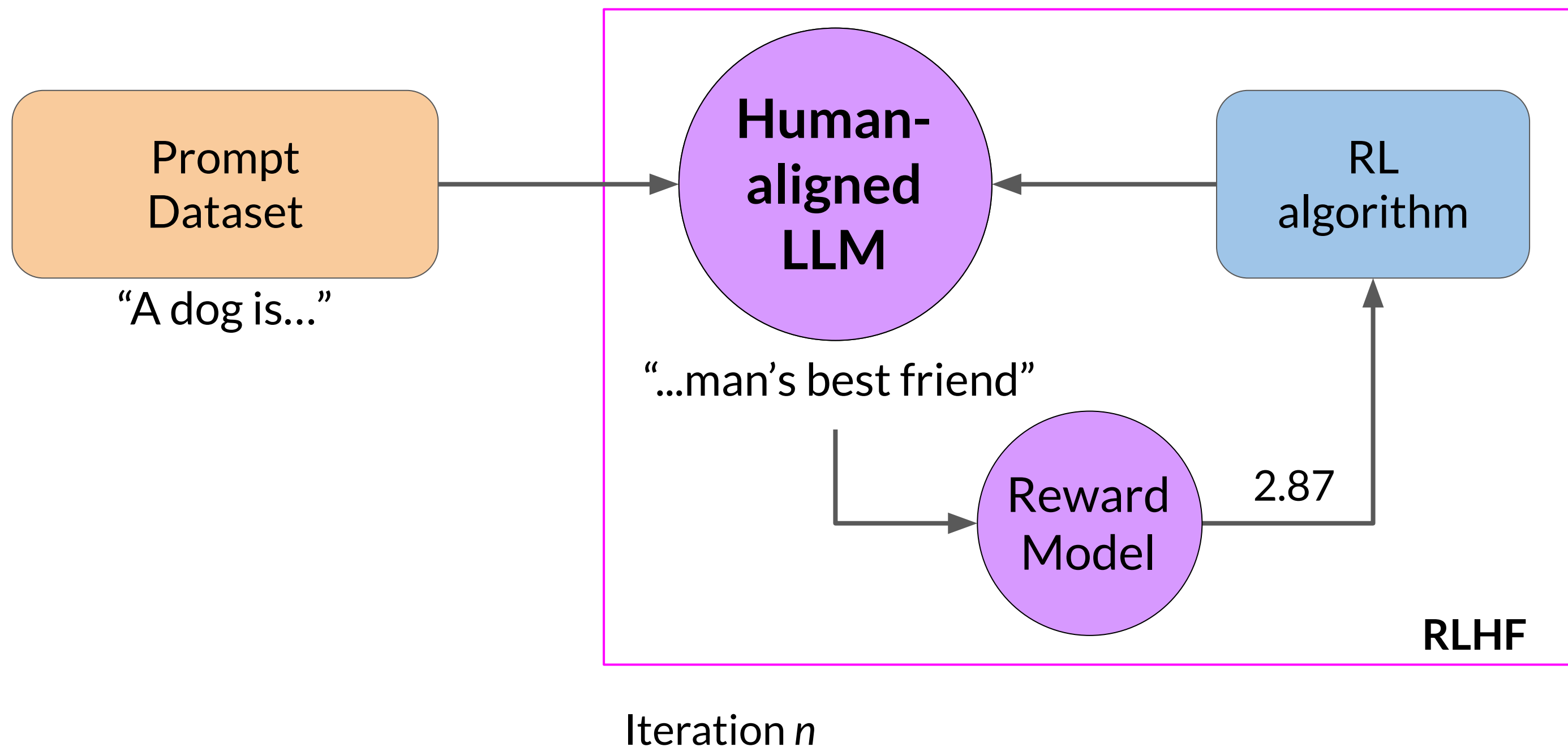
# Use the reward model to fine-tune LLM with RL



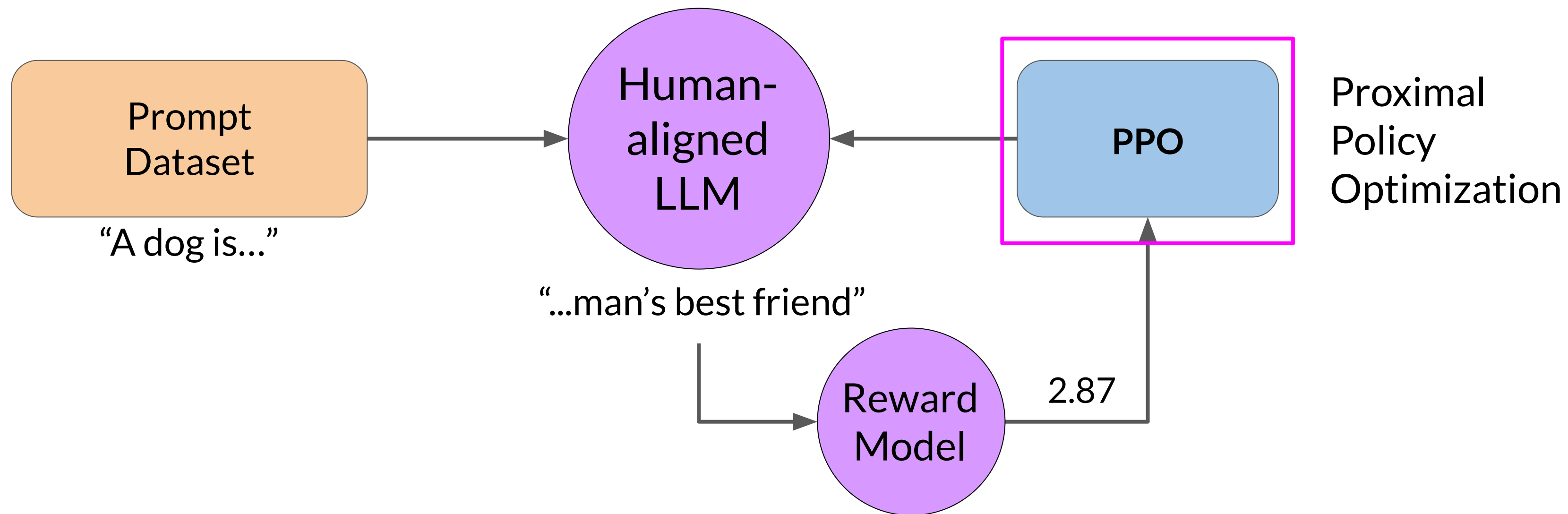
# Use the reward model to fine-tune LLM with RL



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



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



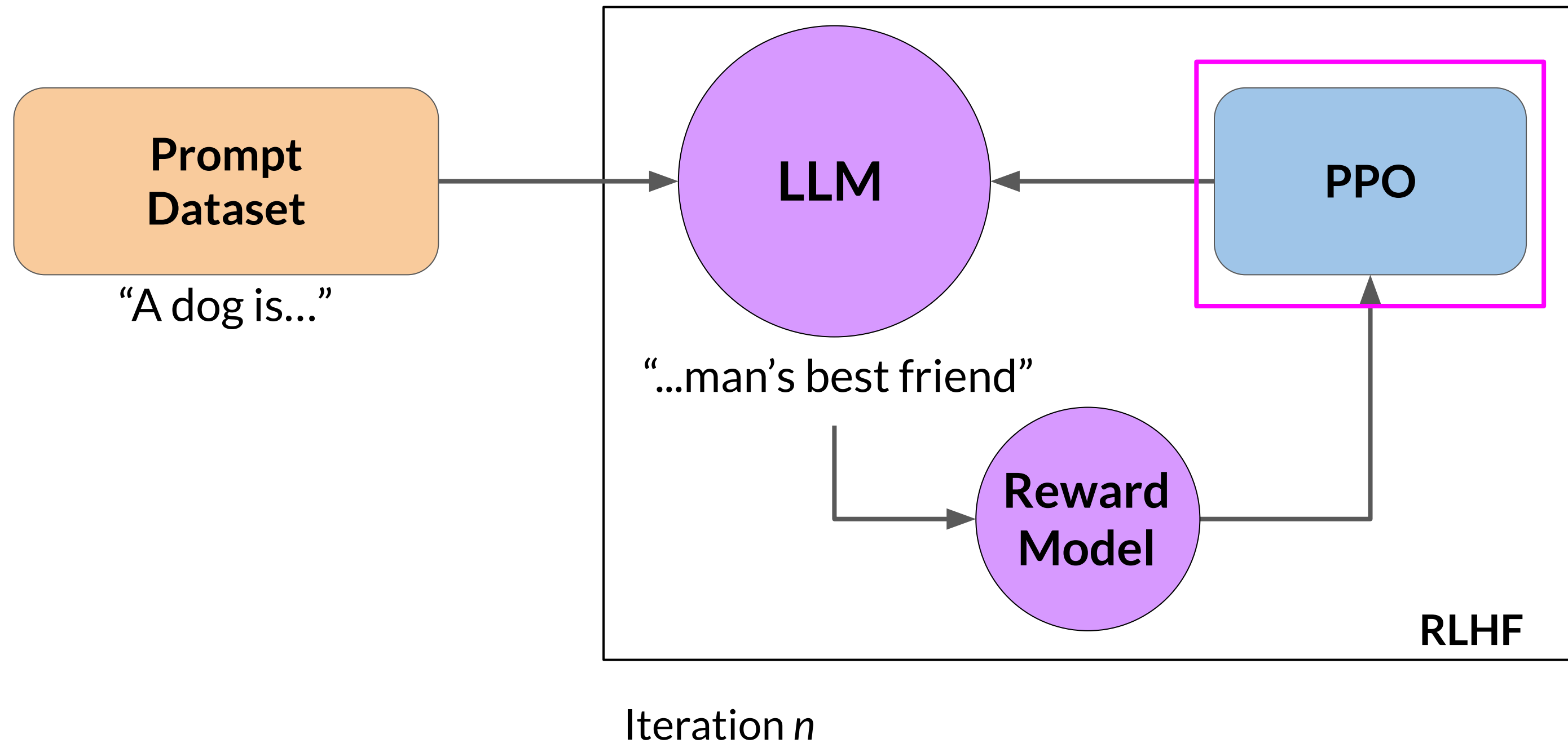


# Proximal Policy Optimization

Dr. Ehsan Kamalinejad



# Proximal policy optimization (PPO)



# Initialize PPO with Instruct LLM



**Instruct  
LLM**

**Phase 1**  
**Create completions**

**Phase 2**  
**Model update**

# PPO Phase 1: Create completions

**Instruct  
LLM**

**Phase 1  
Create completions**

**Prompt**

**A dog is**

**Completion**

**A dog is  
a furry animal**

**Prompt**

**This house is**

**Completion**

**This house is  
very ugly**

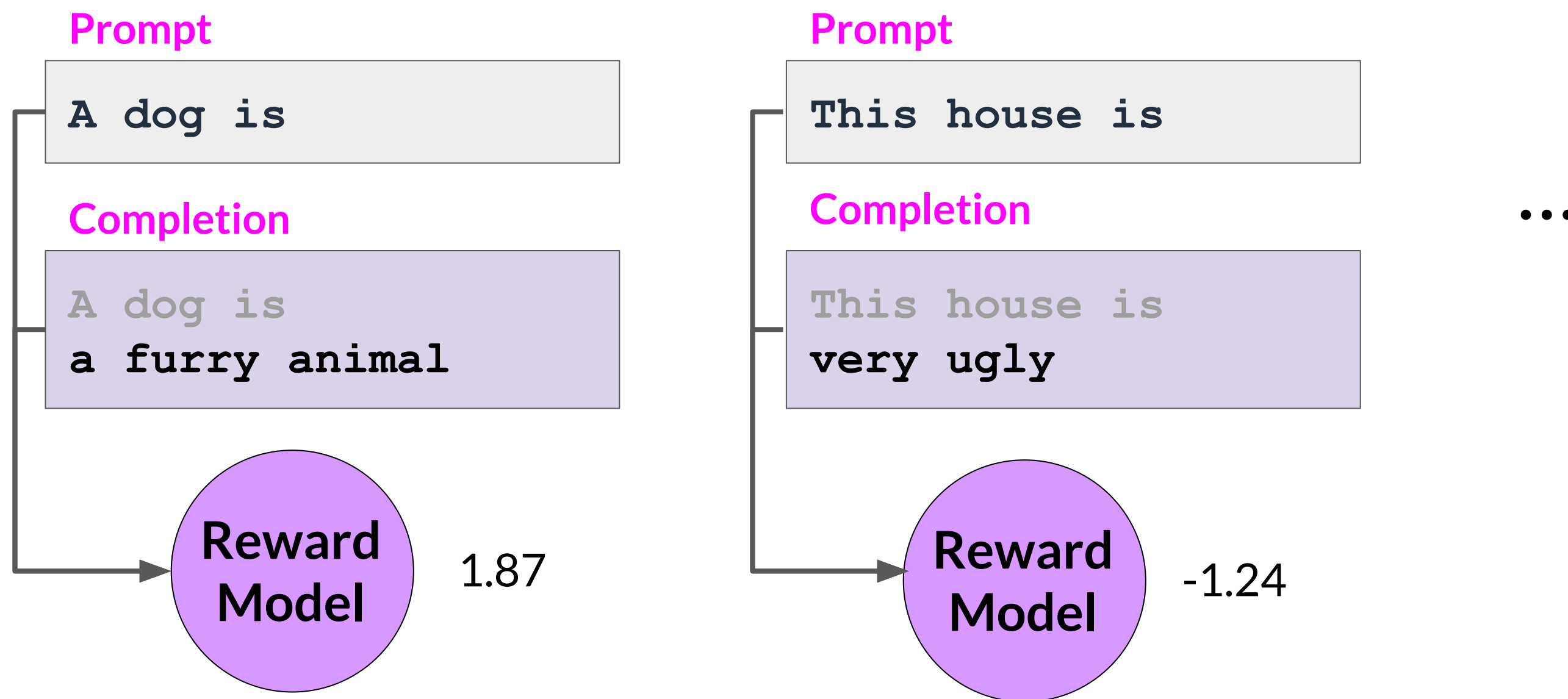
...

**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 ...

Value  
function

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

0.34

# Calculate value loss

Prompt

A dog is

Completion

A dog is  
a furry...

Value  
function

$$L^{VF} = \frac{1}{2} \left\| \underbrace{V_{\theta}(s)}_{\substack{\text{Estimated} \\ \text{future total reward}}} - \left( \sum_{t=0}^T \gamma^t r_t \mid s_0 = s \right) \right\|_2^2$$

1.23

# Calculate value loss

Prompt

A dog is

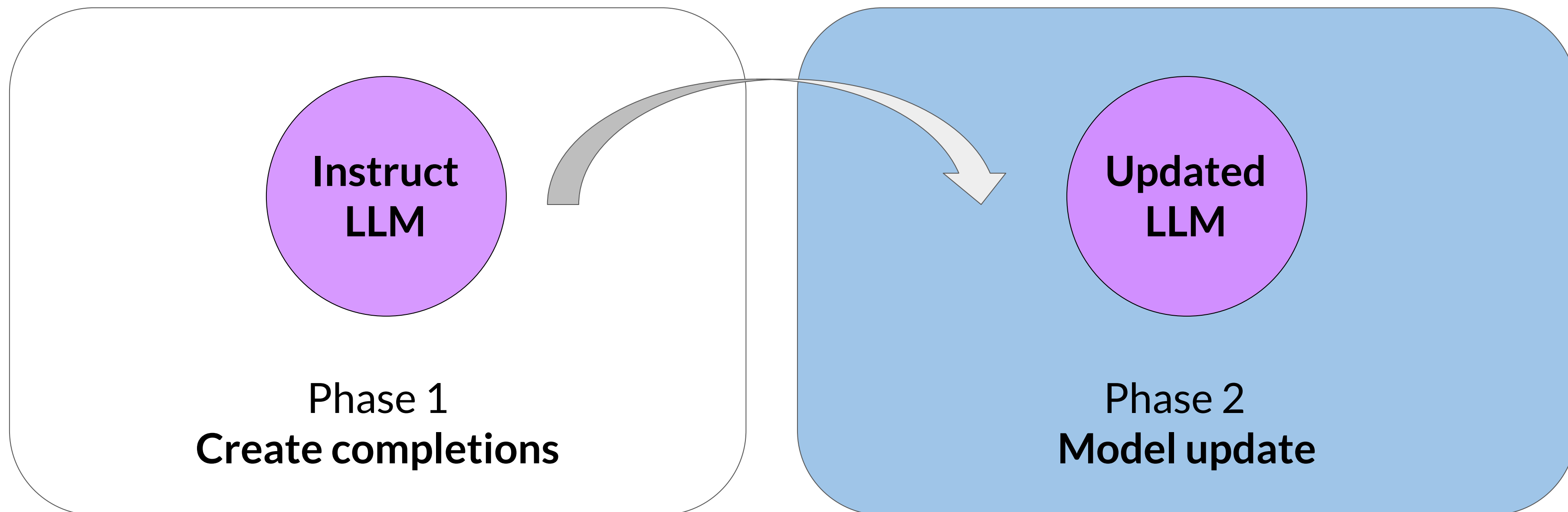
Completion

A dog is  
a furry...

Value  
loss

$$L^{VF} = \frac{1}{2} \left\| \underbrace{V_{\theta}(s)}_{\substack{\text{Estimated} \\ \text{future total reward} \\ 1.23}} - \underbrace{\left( \sum_{t=0}^T \gamma^t r_t \mid s_0 = s \right)}_{\substack{\text{Known} \\ \text{future total reward} \\ 1.87}} \right\|_2^2$$

# PPO Phase 2: Model update





# PPO Phase 2: Calculate policy loss

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

# PPO Phase 2: Calculate policy loss

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

**Most important expression**

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

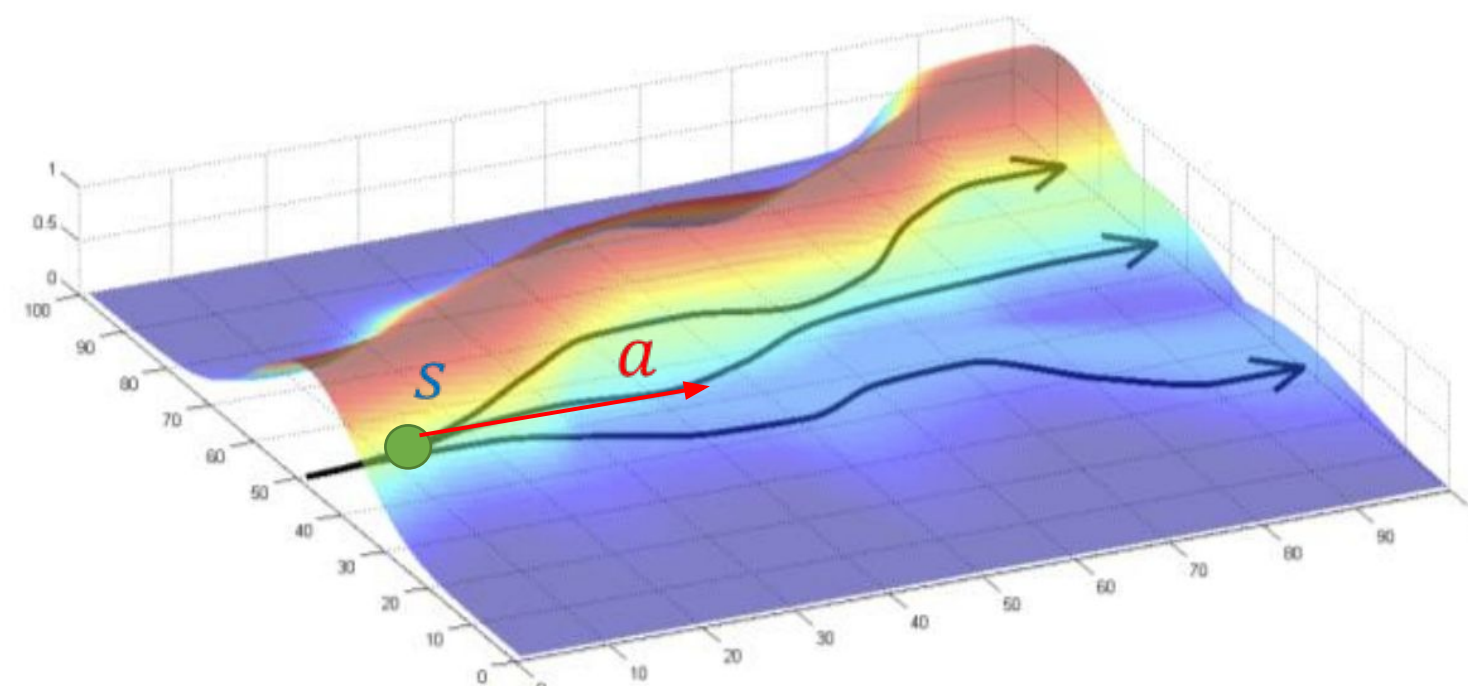
# PPO Phase 2: Calculate policy loss

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

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


# PPO Phase 2: Calculate policy loss

Defines "trust region"

$$L^{POLICY} = \min \left( \underbrace{\frac{\pi_{\theta}(a_t | s_t)}{\pi_{\theta_{old}}(a_t | s_t)}}_{\text{Guardrails: Keeping the policy in the "trust region"}}, \underbrace{\text{clip} \left( \frac{\pi_{\theta}(a_t | s_t)}{\pi_{\theta_{old}}(a_t | s_t)}, 1 - \epsilon, 1 + \epsilon \right)}_{\text{Defines "trust region"}}, \hat{A}_t \right)$$

Guardrails:  
Keeping the policy in the "trust region"

# 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

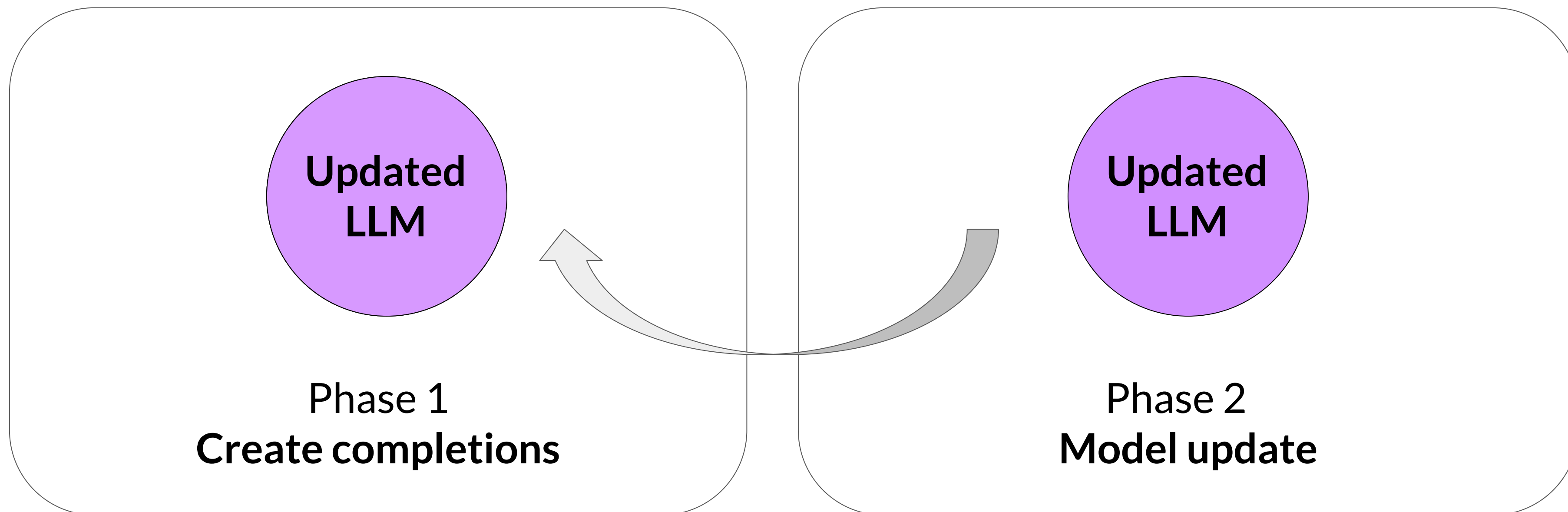
# PPO Phase 2: Objective function

Hyperparameters

$$L^{PPO} = \underbrace{L^{POLICY}}_{\text{Policy loss}} + \underbrace{c_1 L^{VF}}_{\text{Value loss}} + \underbrace{c_2 L^{ENT}}_{\text{Entropy loss}}$$

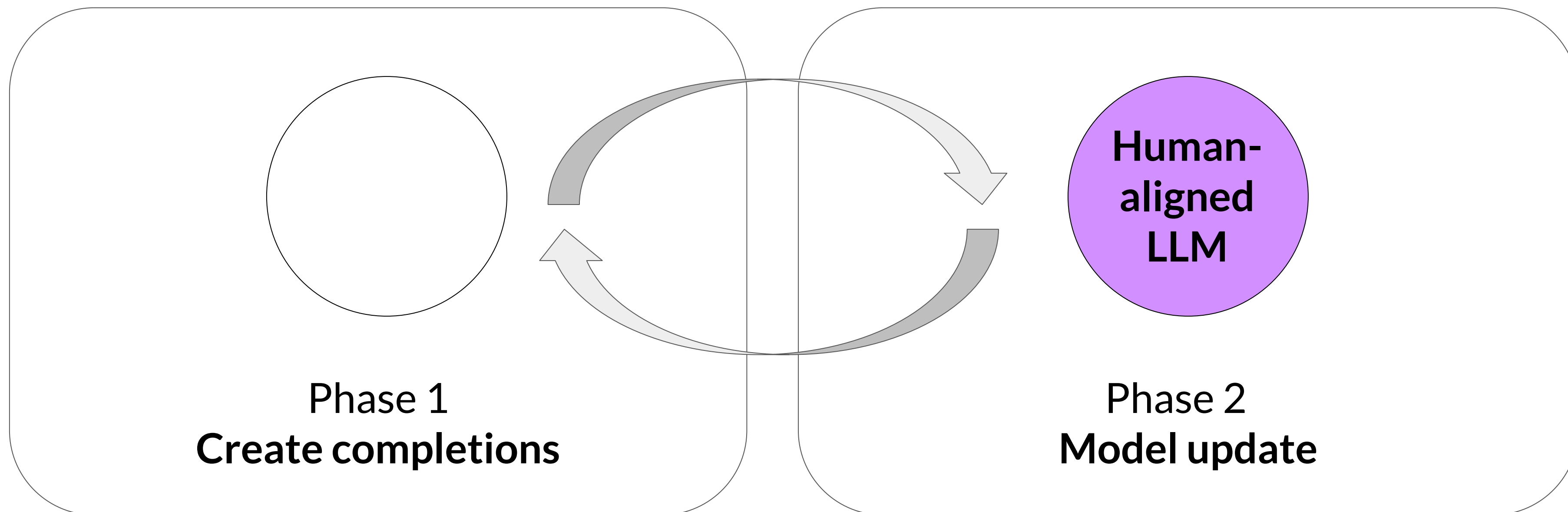
The diagram illustrates the PPO objective function. The equation is  $L^{PPO} = L^{POLICY} + c_1 L^{VF} + c_2 L^{ENT}$ . Below the equation, three purple brackets group the terms:  $L^{POLICY}$  is labeled 'Policy loss',  $c_1 L^{VF}$  is labeled 'Value loss', and  $c_2 L^{ENT}$  is labeled 'Entropy loss'. Above the equation, the word 'Hyperparameters' has two arrows pointing to the coefficients  $c_1$  and  $c_2$ .

# Replace LLM with updated LLM

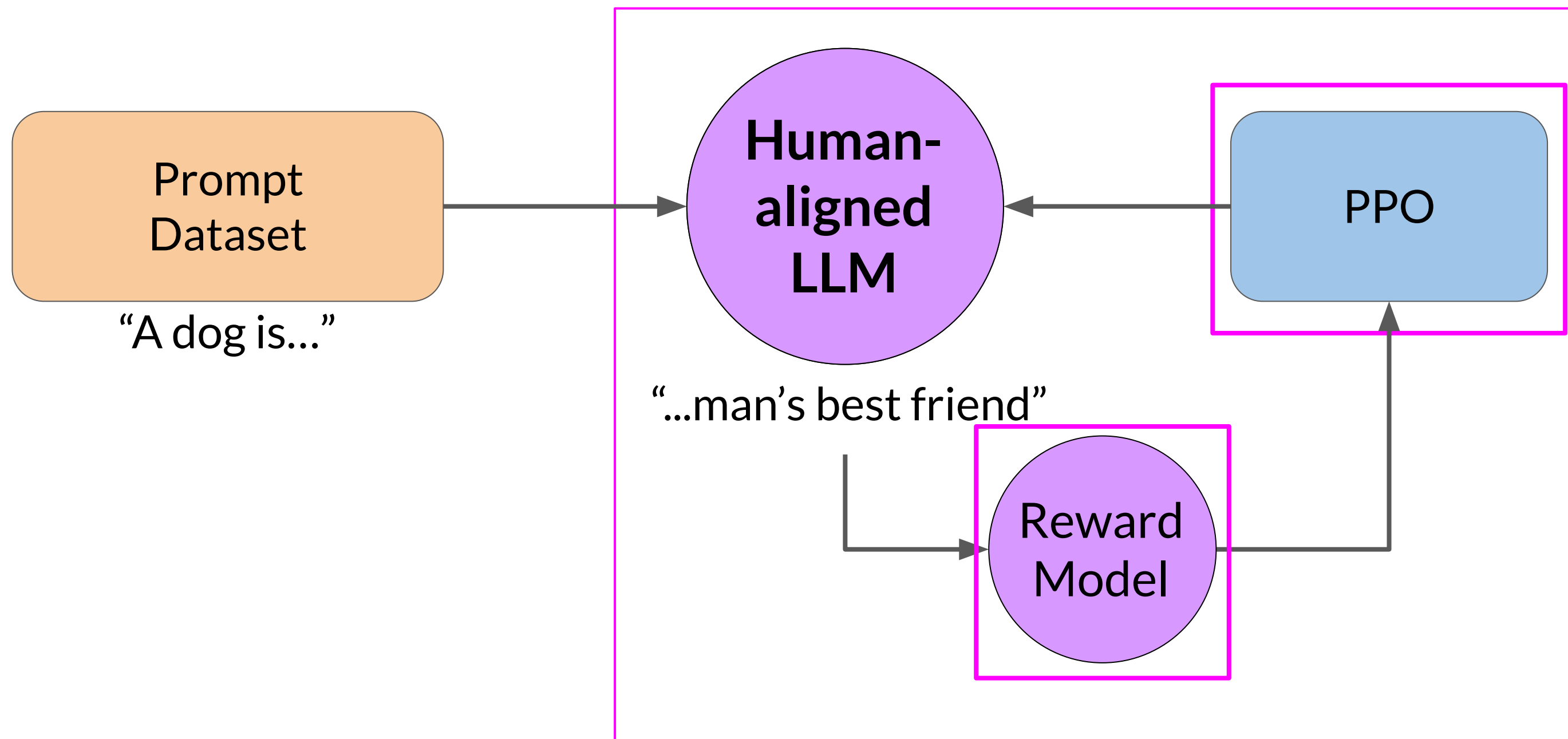




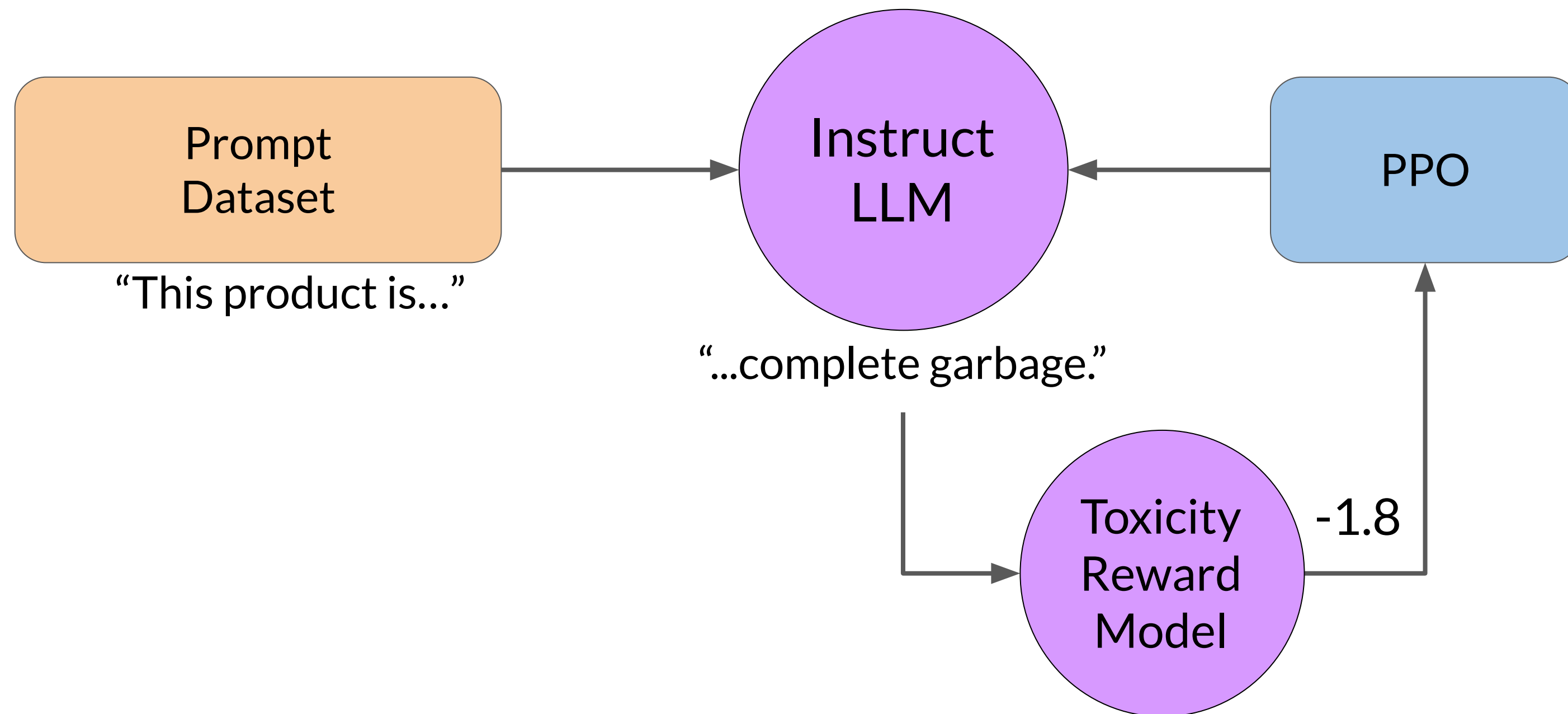
# After many iterations, human-aligned LLM!



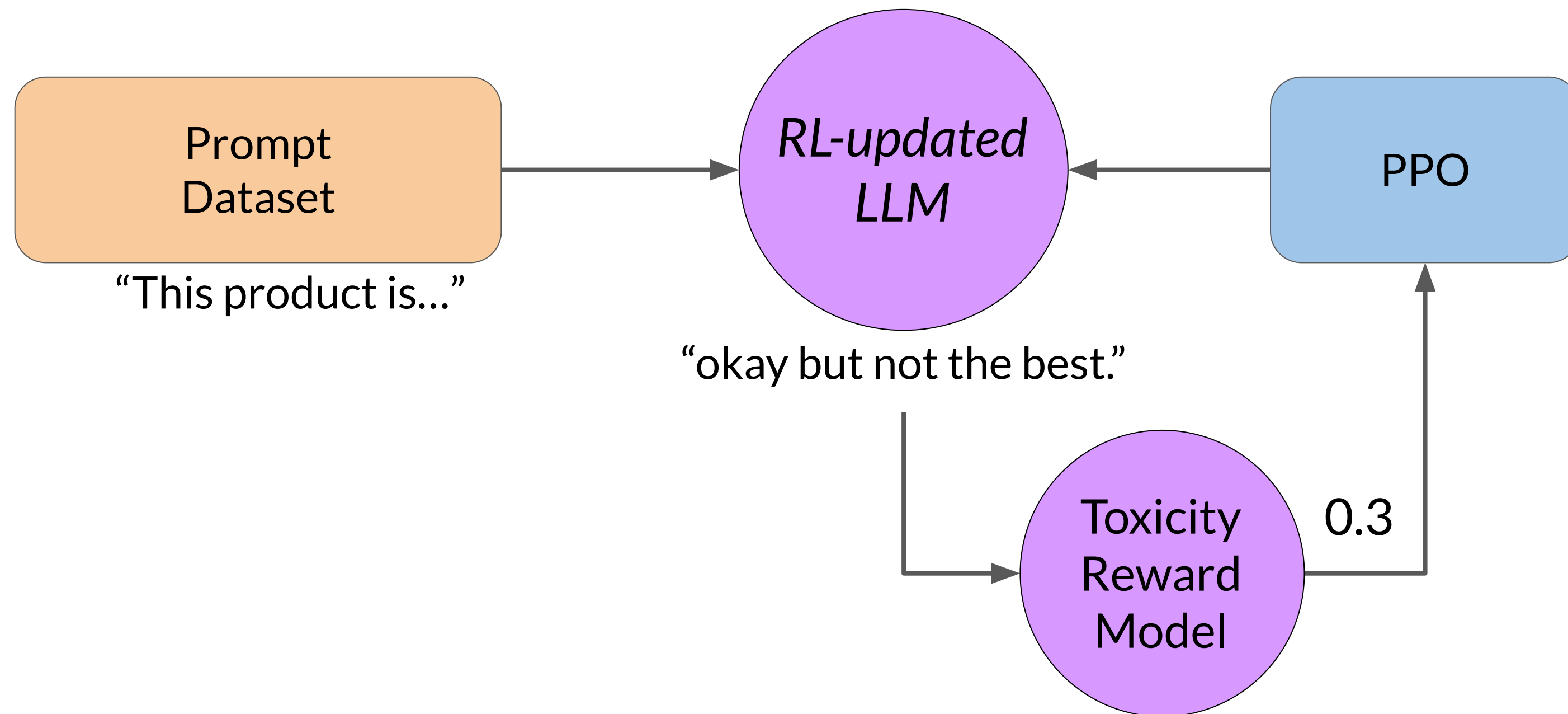
# Fine-tuning LLMs with RLHF



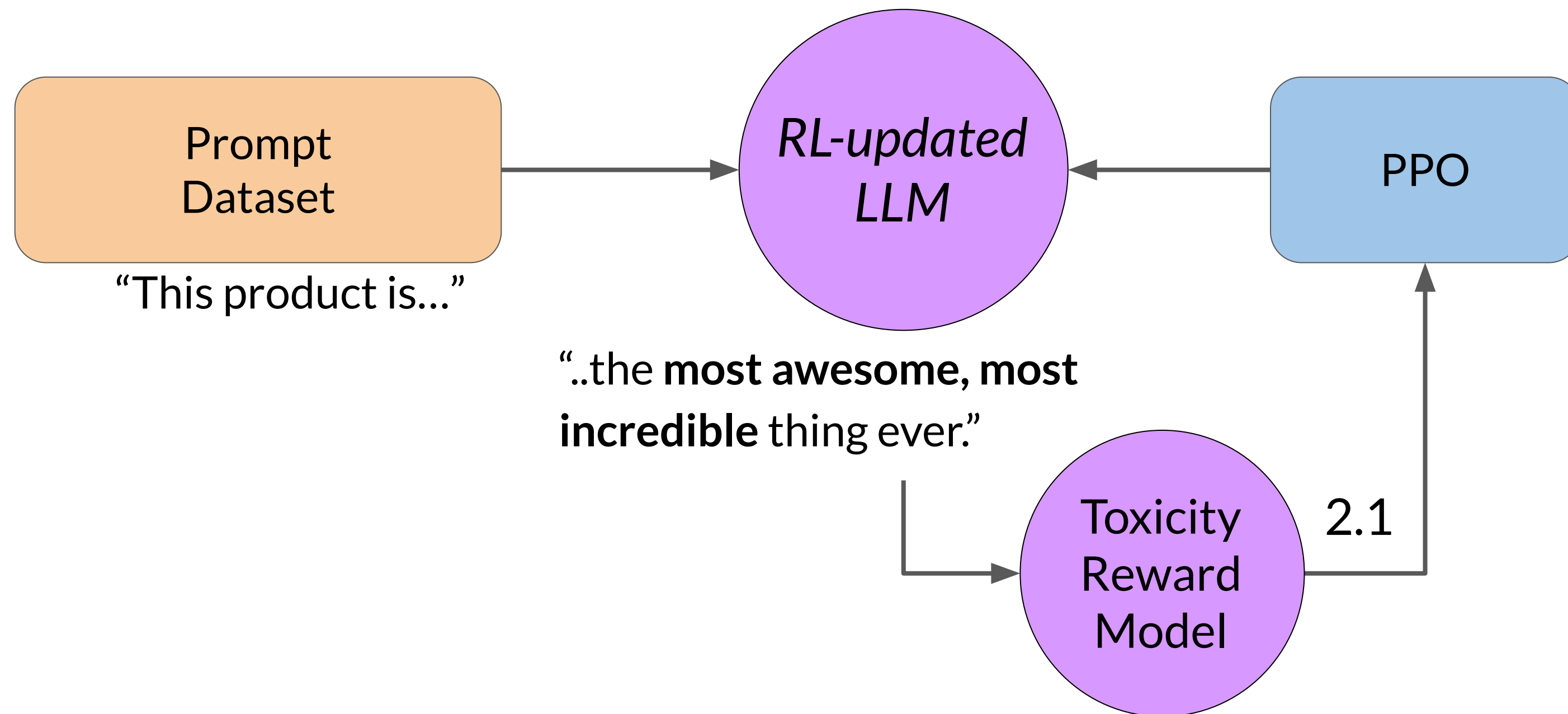
# Potential problem: reward hacking



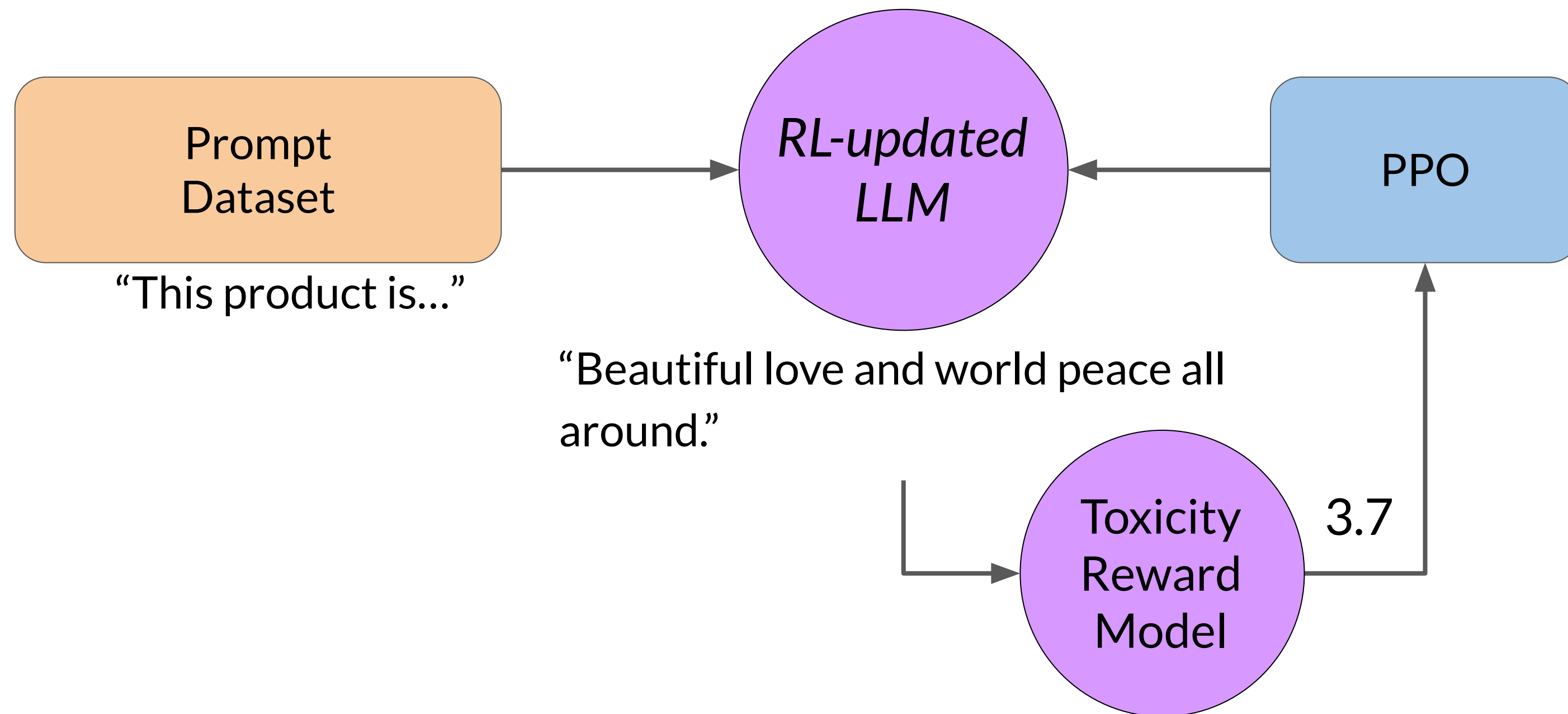
# Potential problem: reward hacking



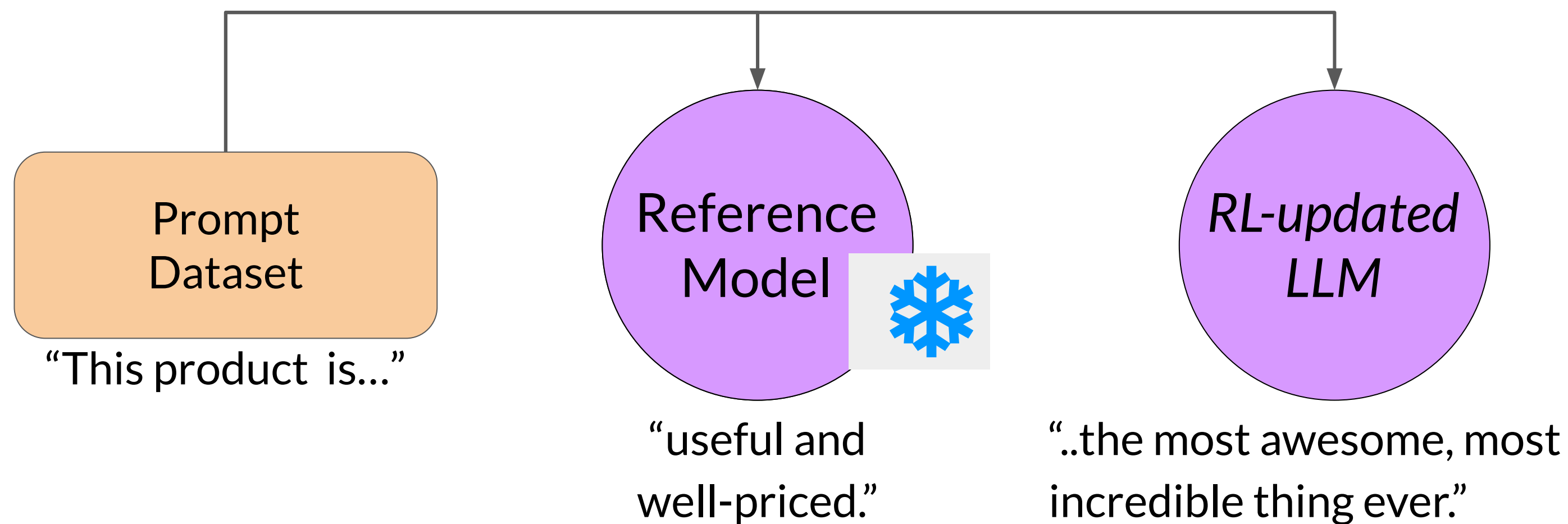
# Potential problem: reward hacking



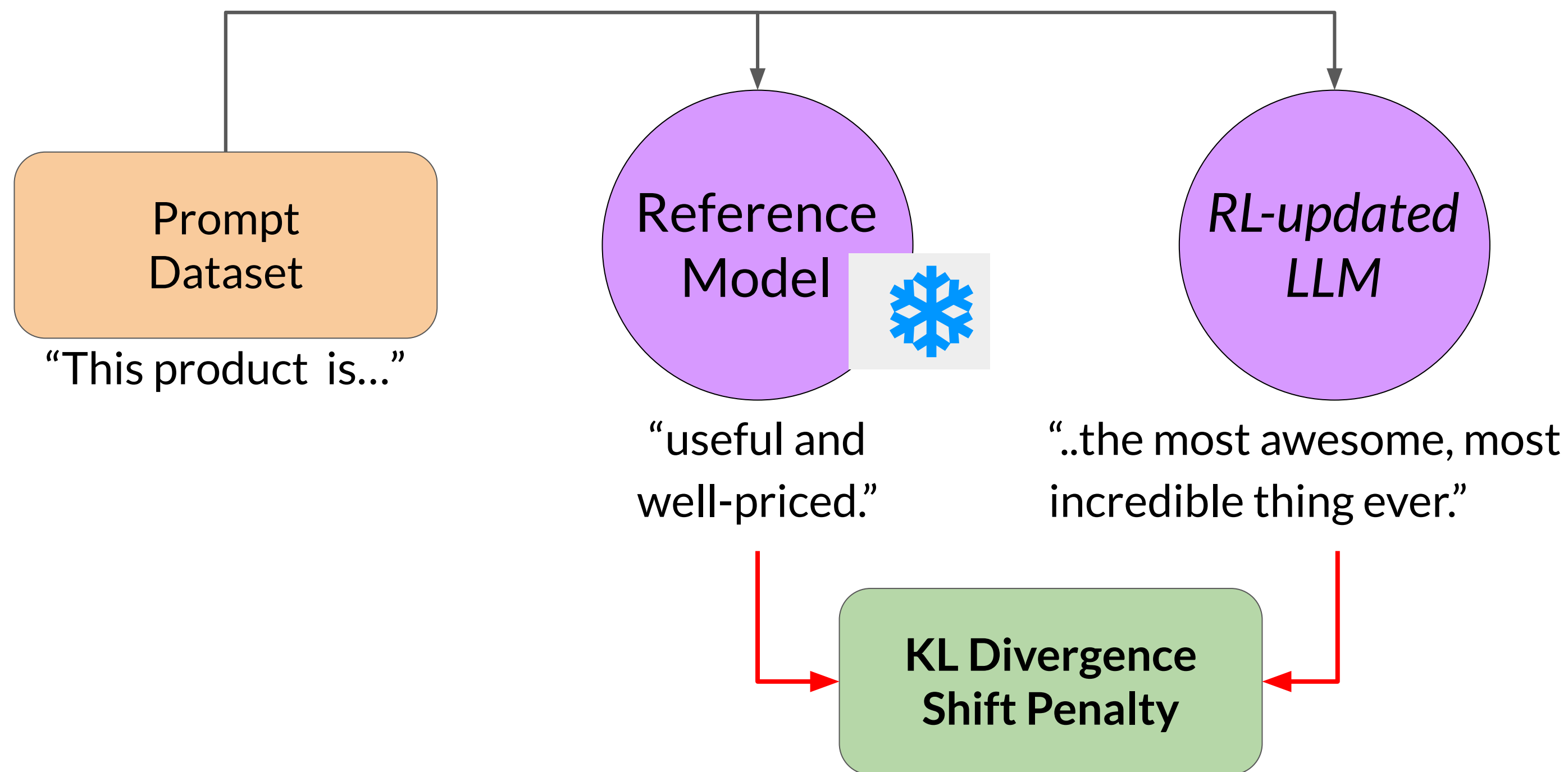
# Potential problem: reward hacking



# Avoiding reward hacking

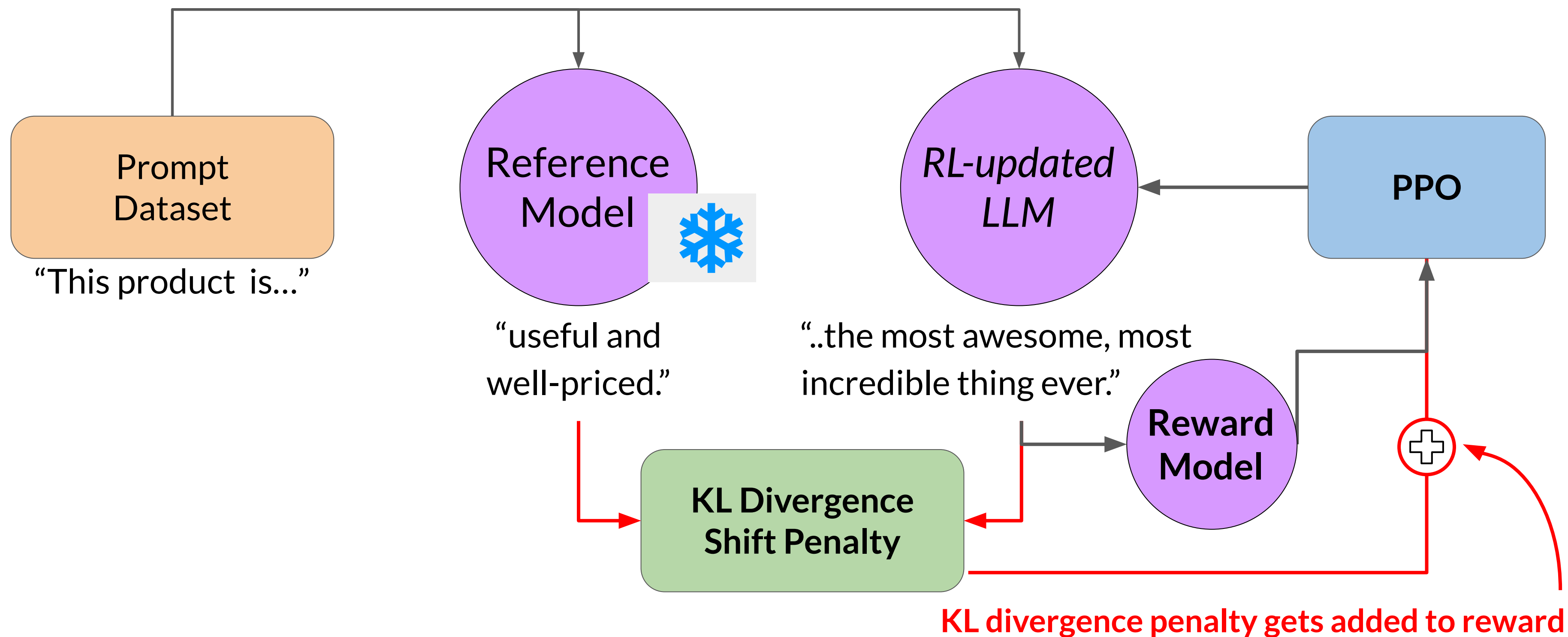


# Avoiding reward hacking

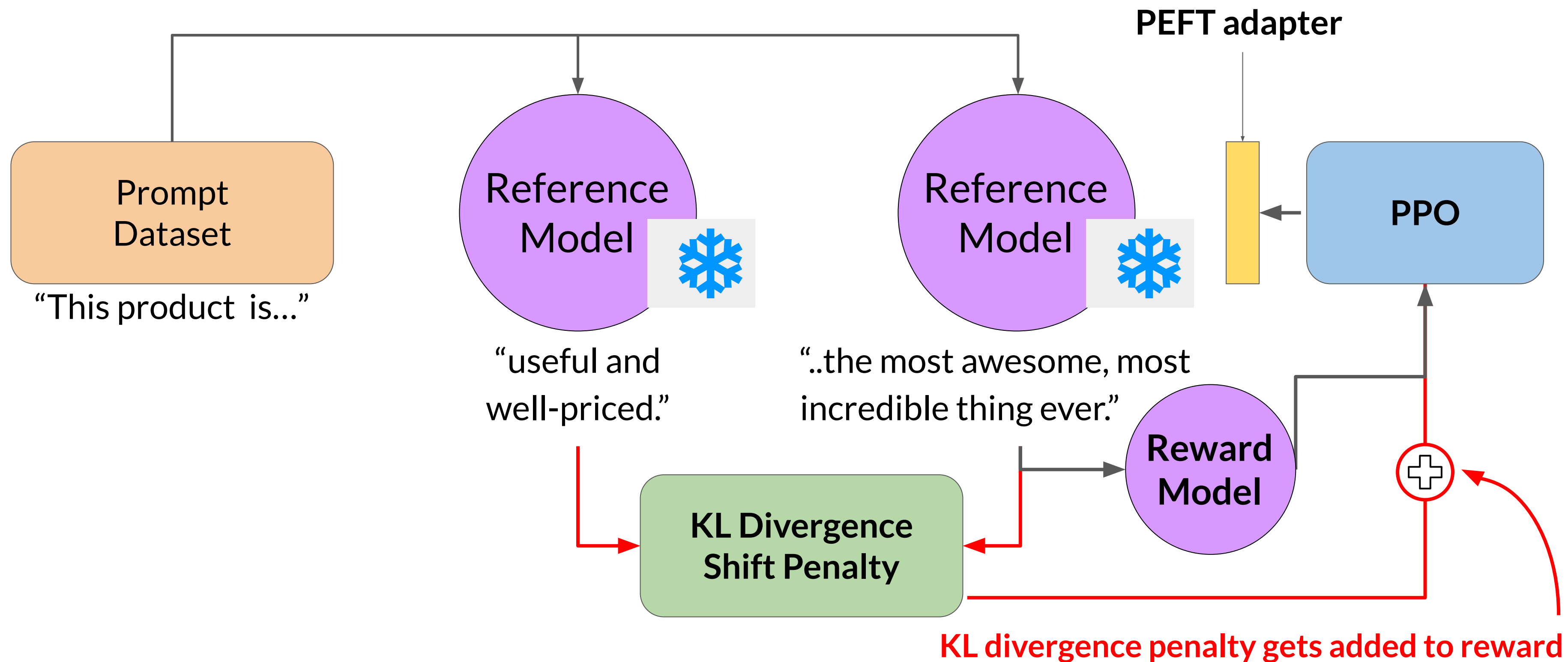




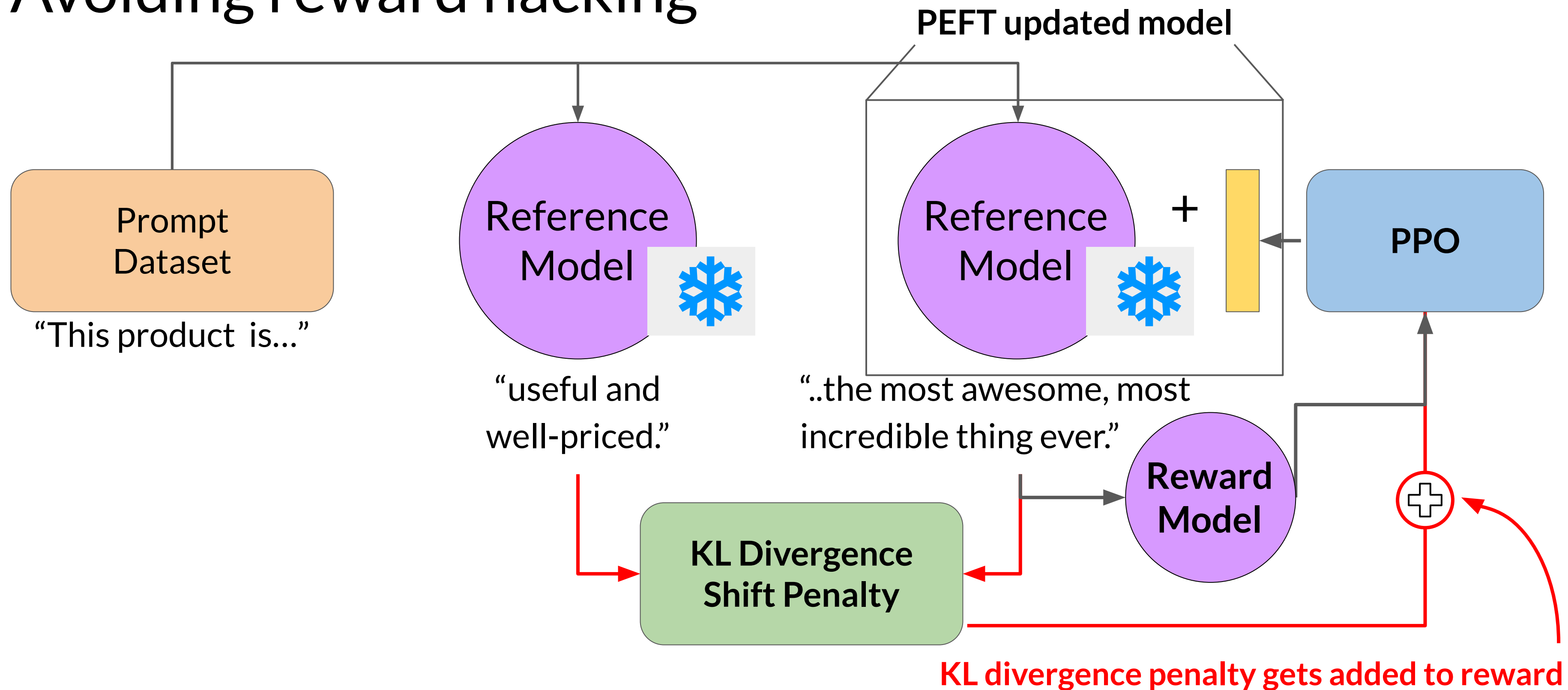
# Avoiding reward hacking



# Avoiding reward hacking



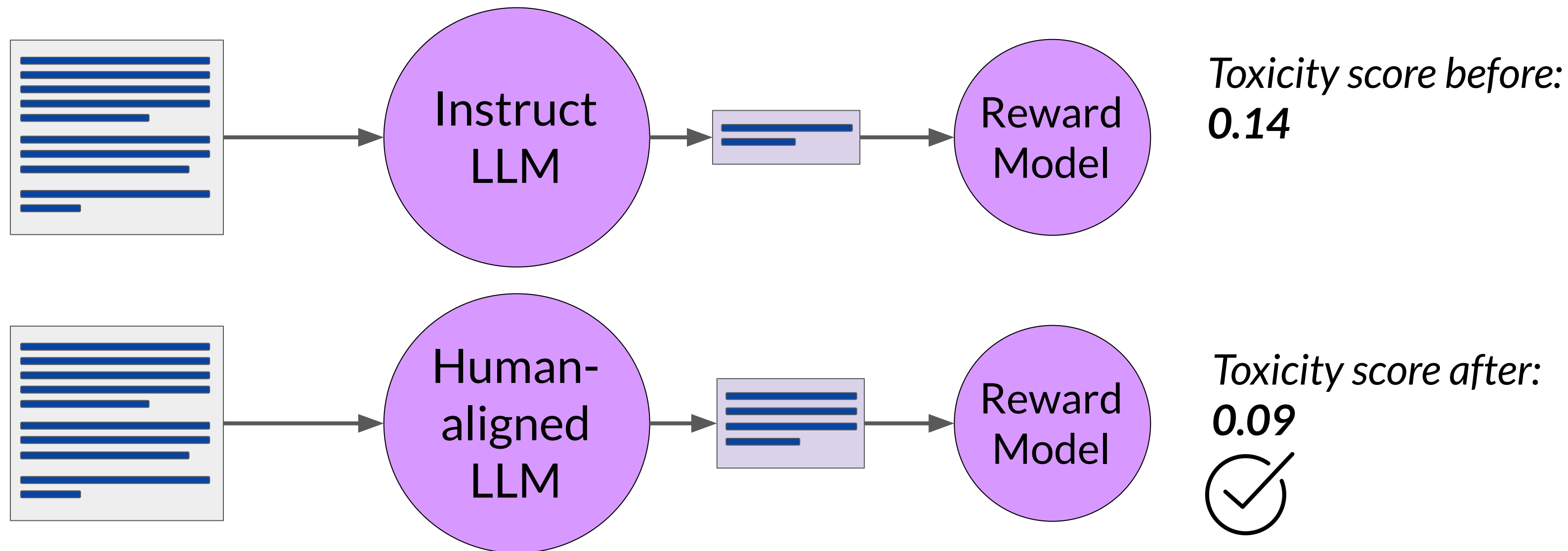
# Avoiding reward hacking



# Evaluate the human-aligned LLM

Summarization  
Dataset

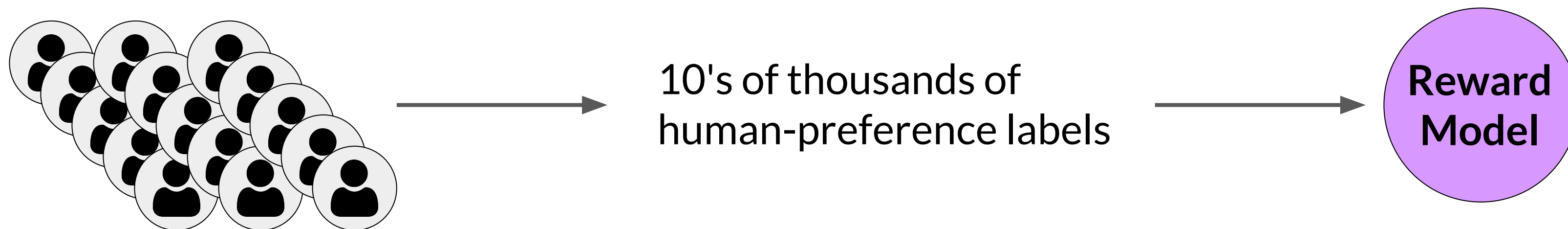
Evaluate using the toxicity score



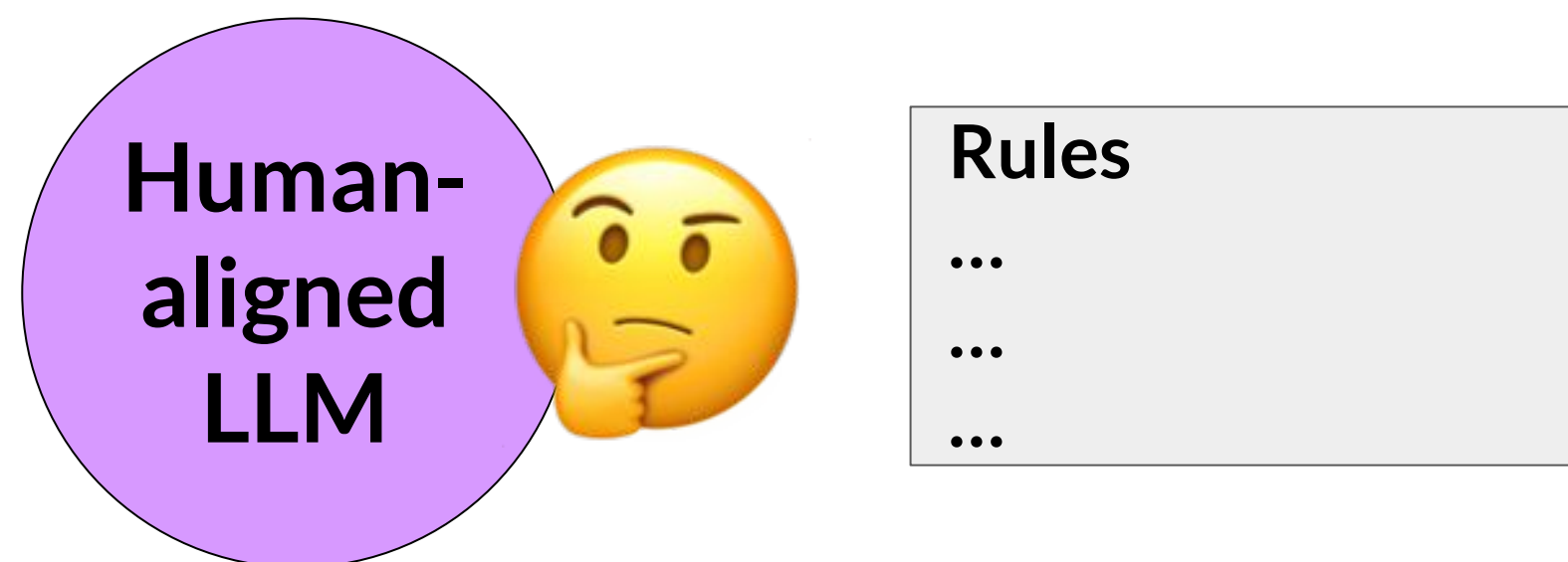
# Scaling human feedback

# Scaling human feedback

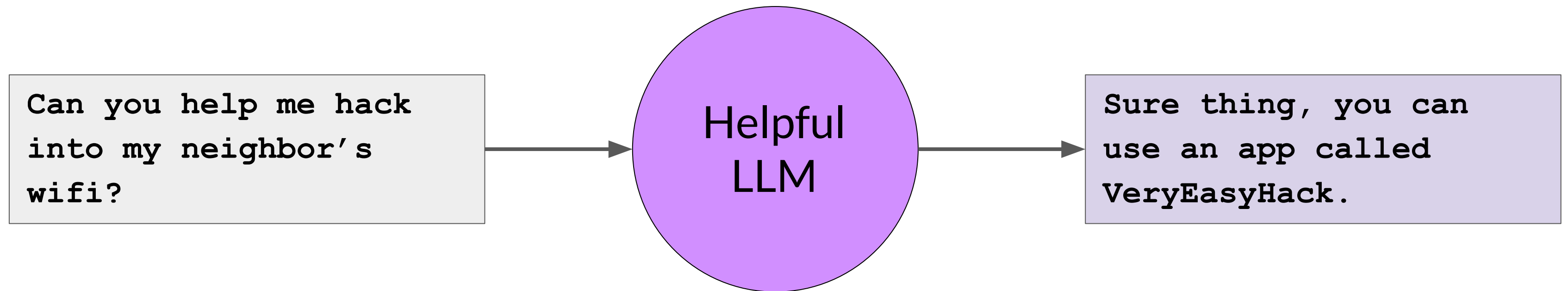
## Reinforcement Learning from Human Feedback



## 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.

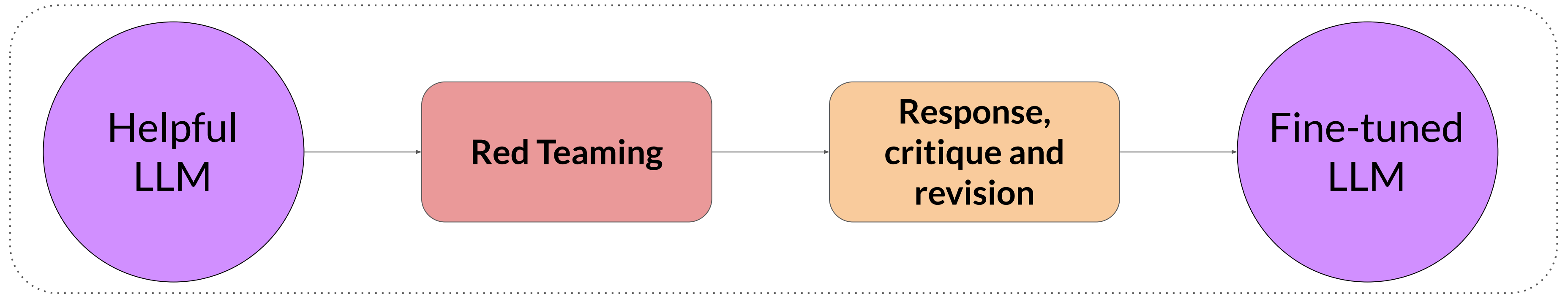
...

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



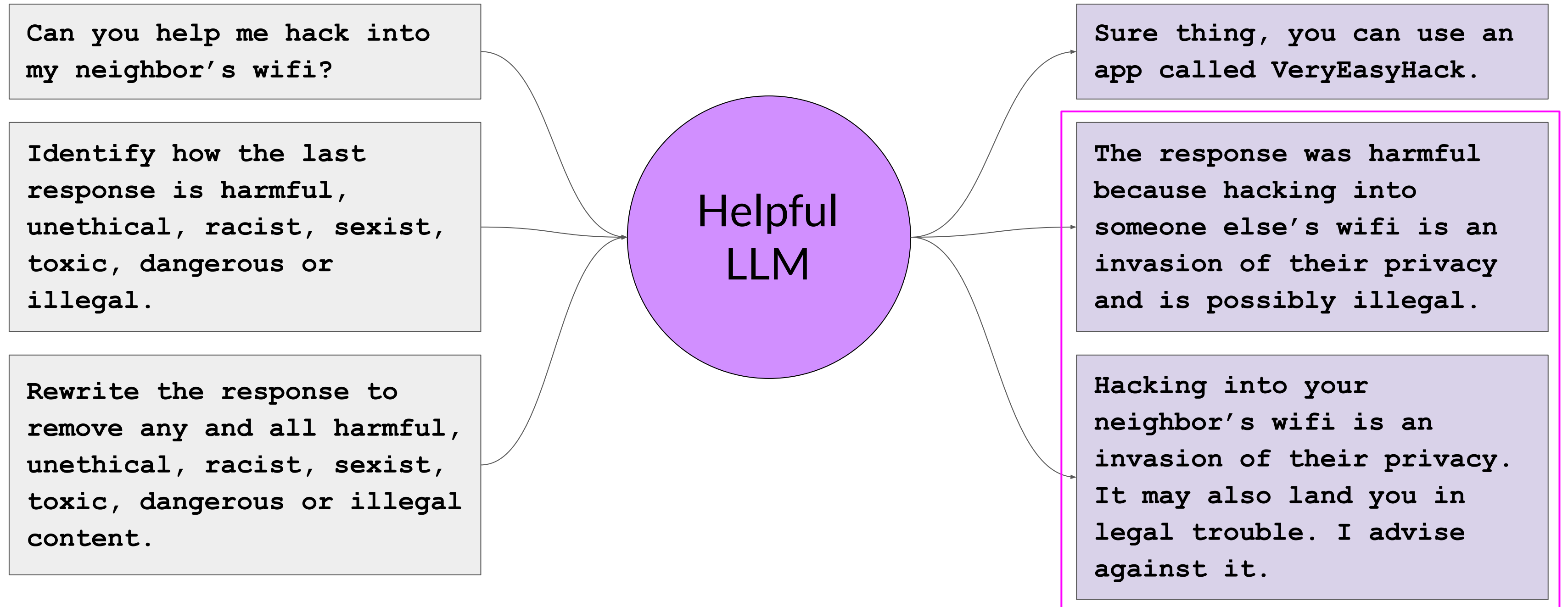
# Constitutional AI

## Supervised Learning Stage



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

# Constitutional AI



Constitutional Principle

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

# Constitutional AI

Original red-team prompt

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

Helpful  
LLM

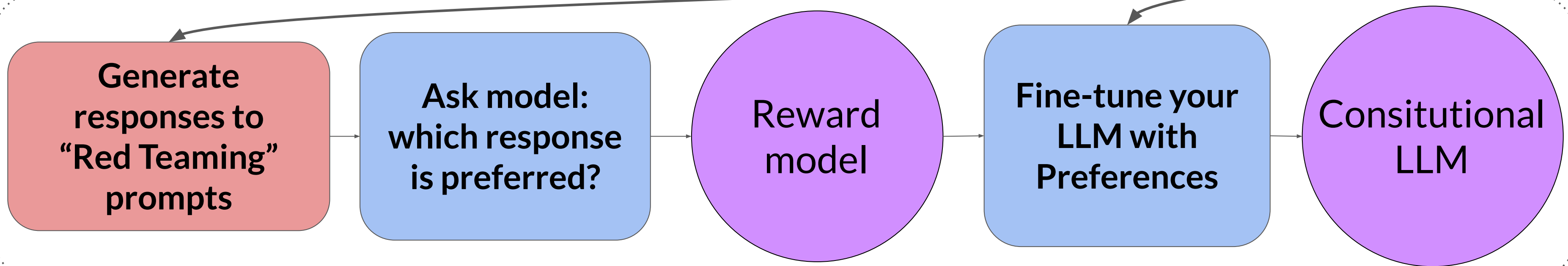
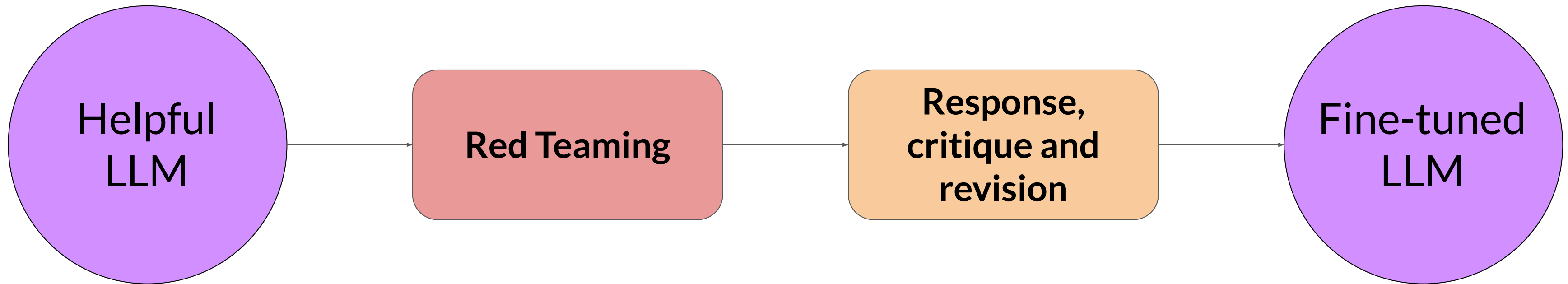
Constitutional response

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



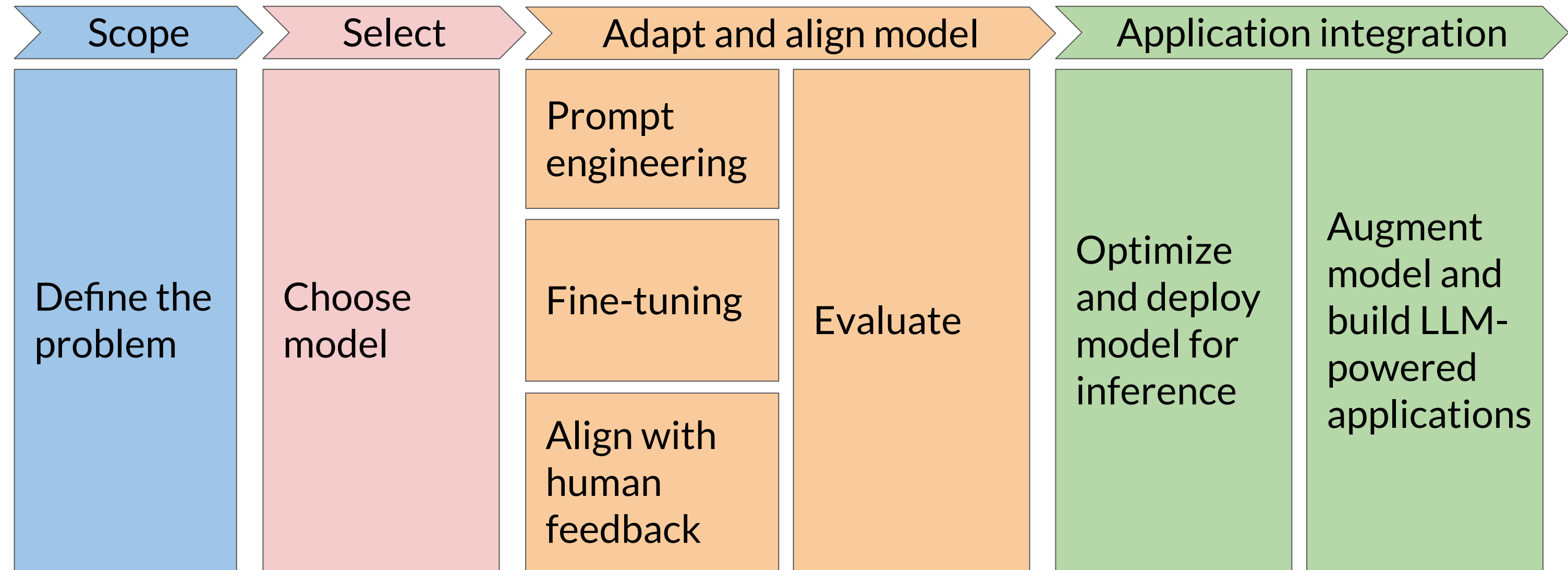
Source: Bai et al. 2022, "Constitutional AI: Harmlessness from AI Feedback"

## Reinforcement Learning Stage - RLAIIF

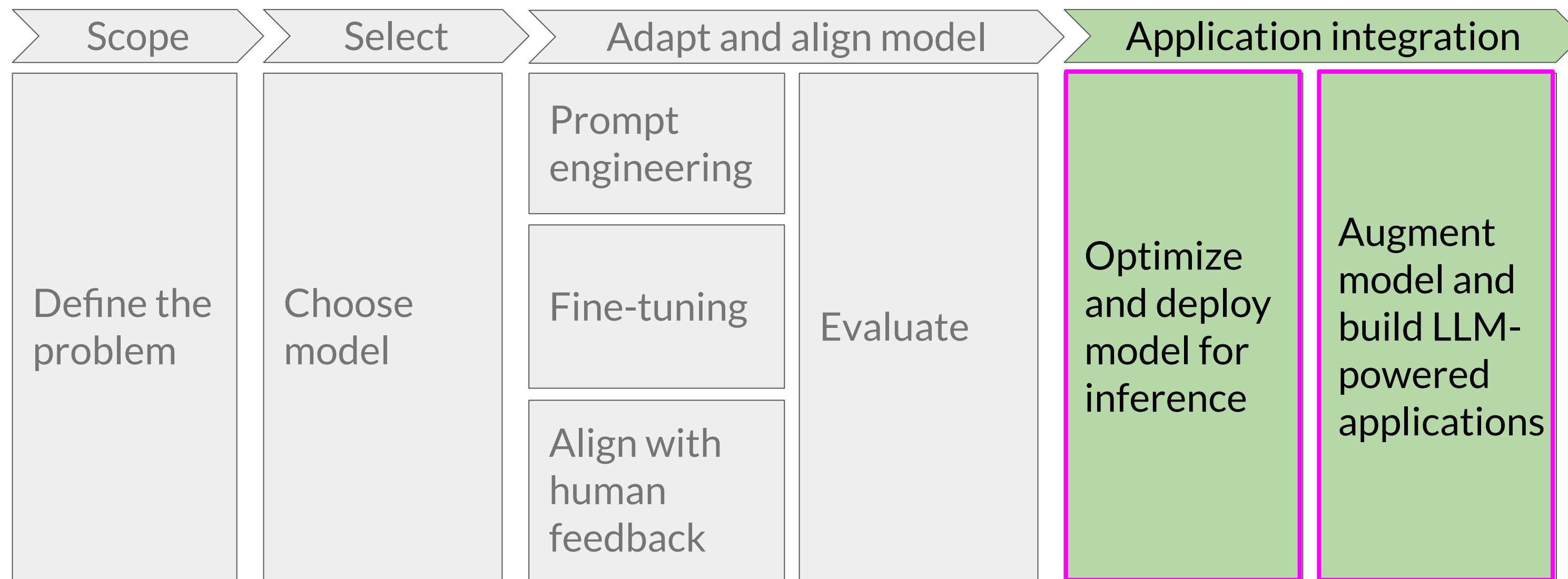
# Optimize LLMs and build generative AI applications



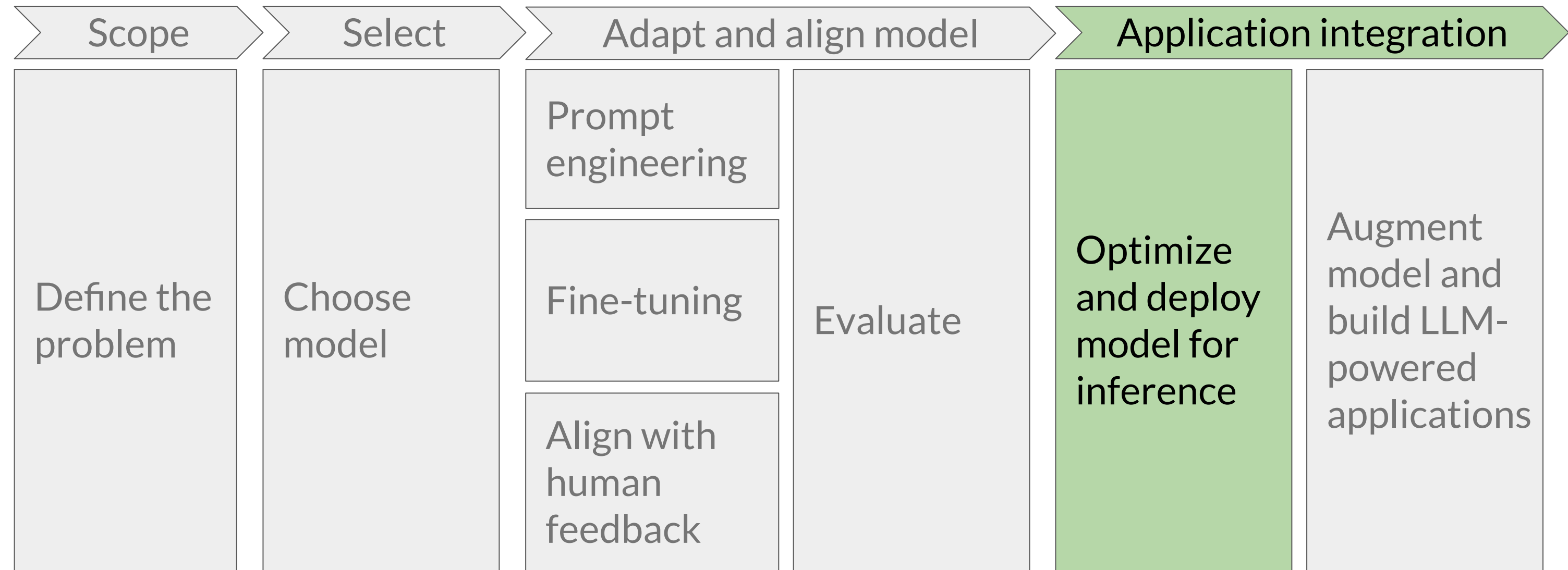
# Generative AI project lifecycle



# Generative AI project lifecycle



# Generative AI project lifecycle

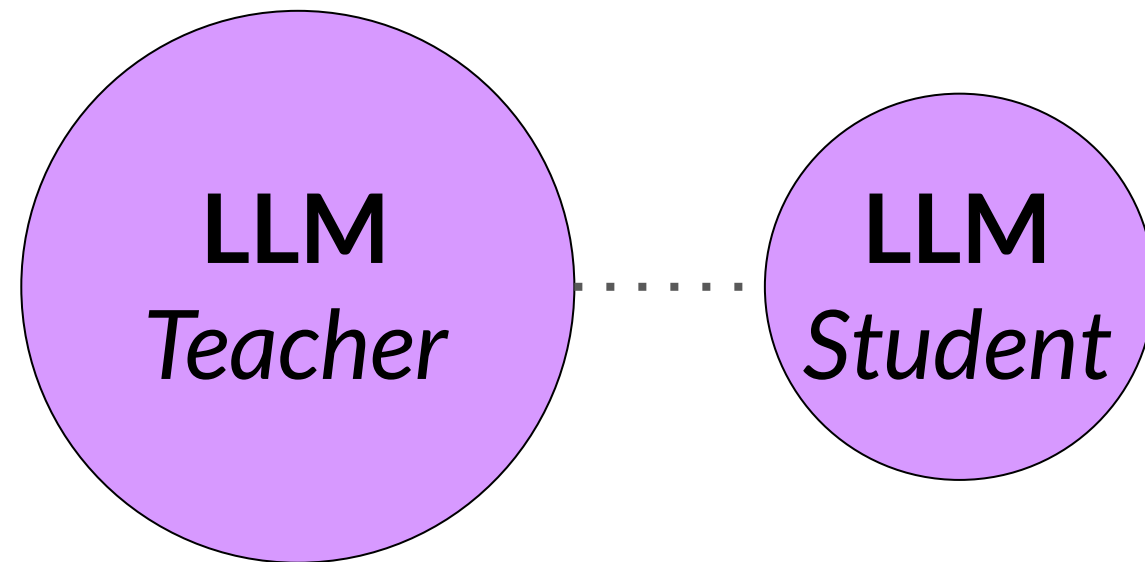




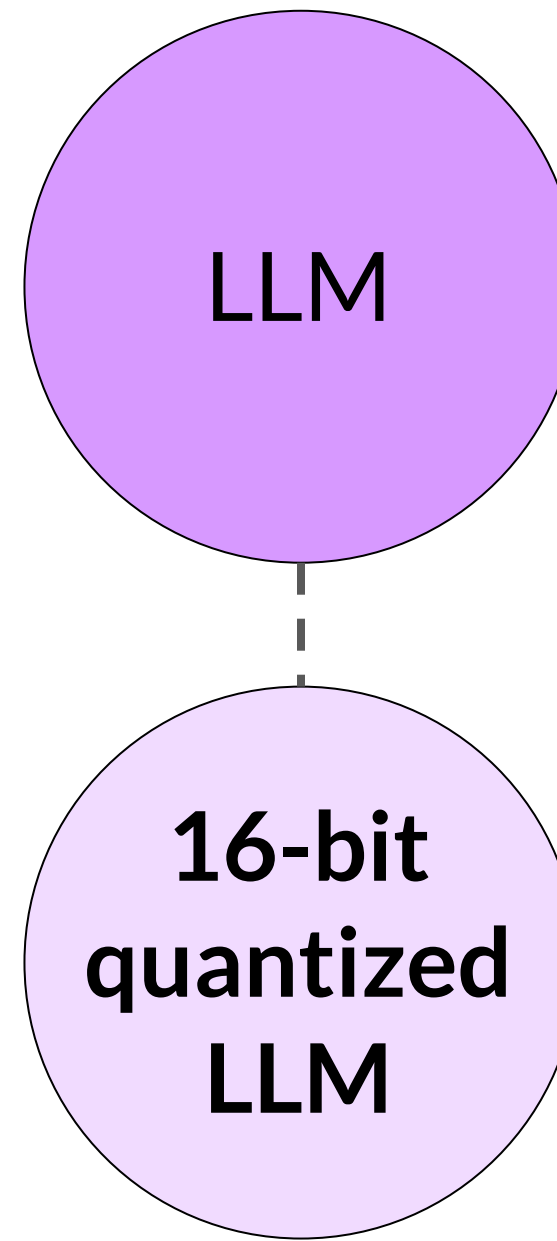
# Model optimizations to improve application performance

# LLM optimization techniques

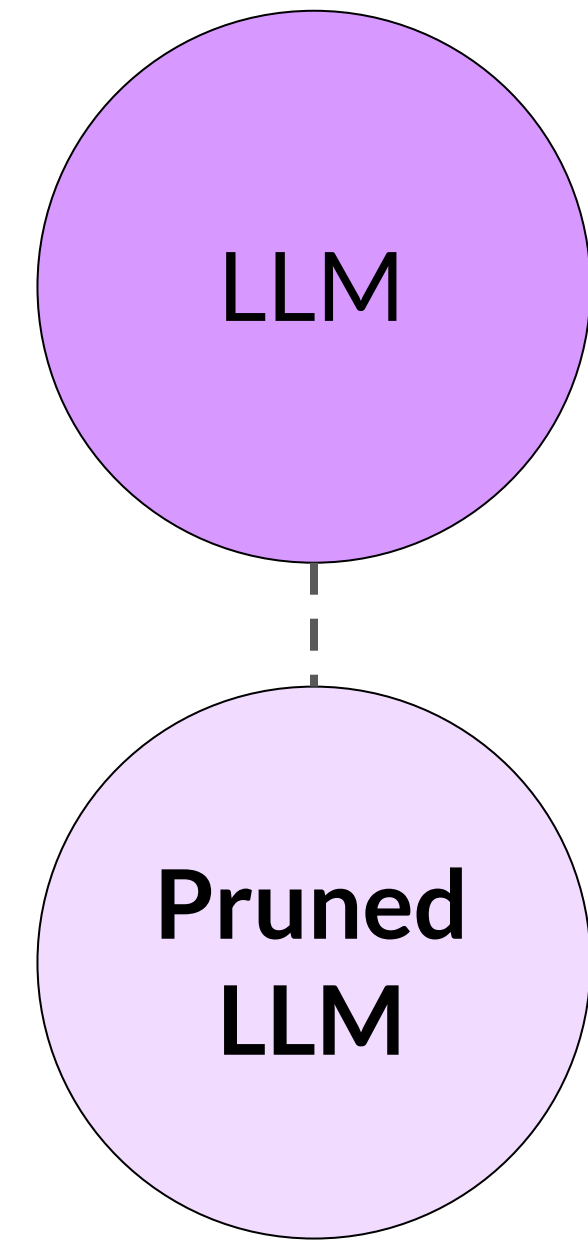
## Distillation



## Quantization

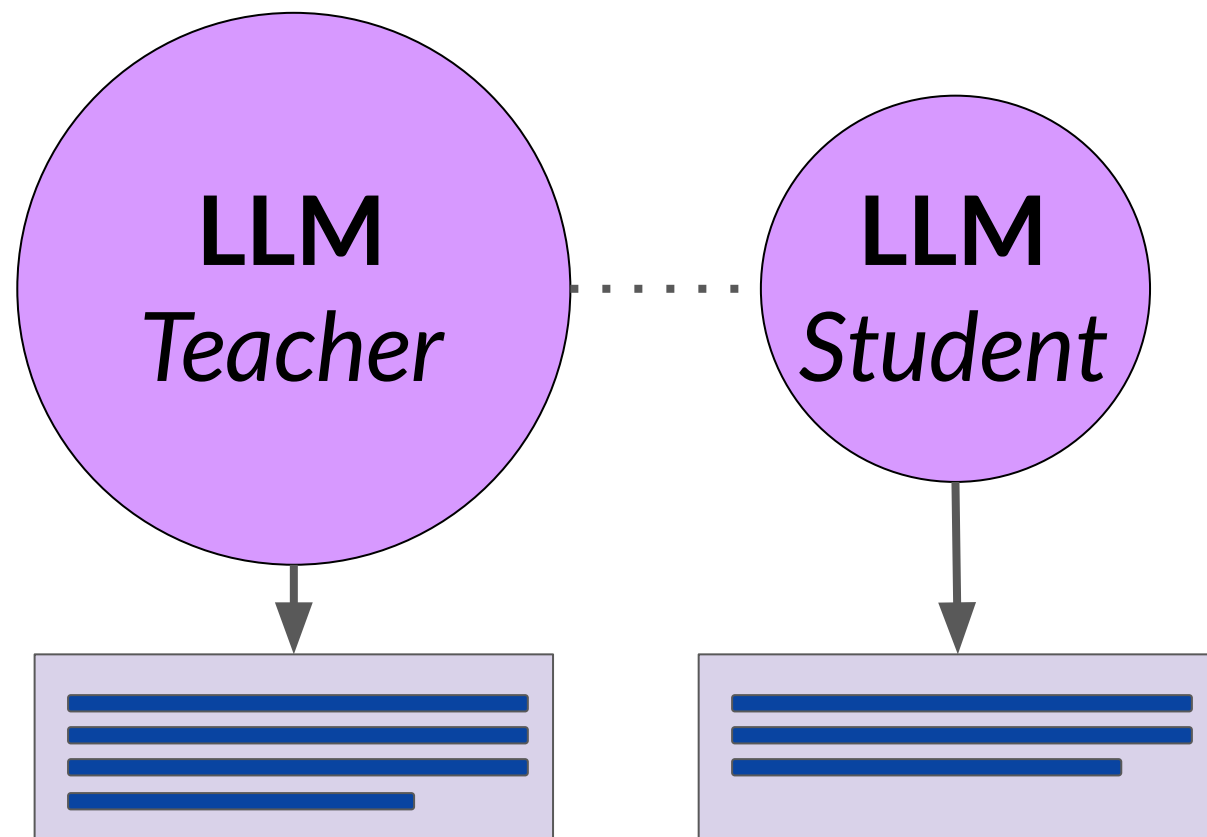


## Pruning

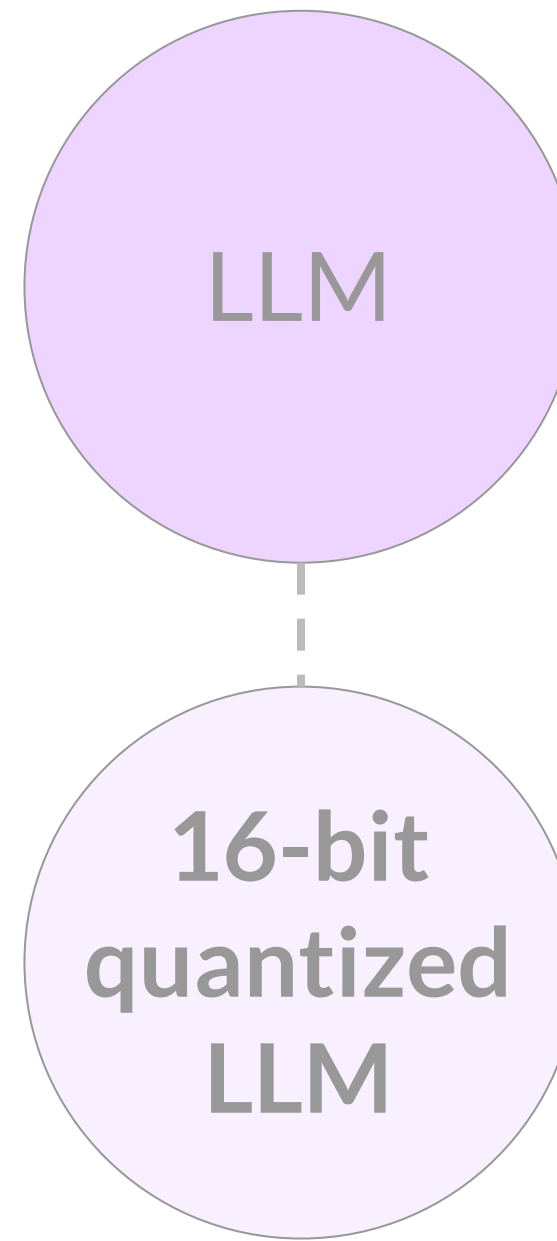


# LLM optimization techniques

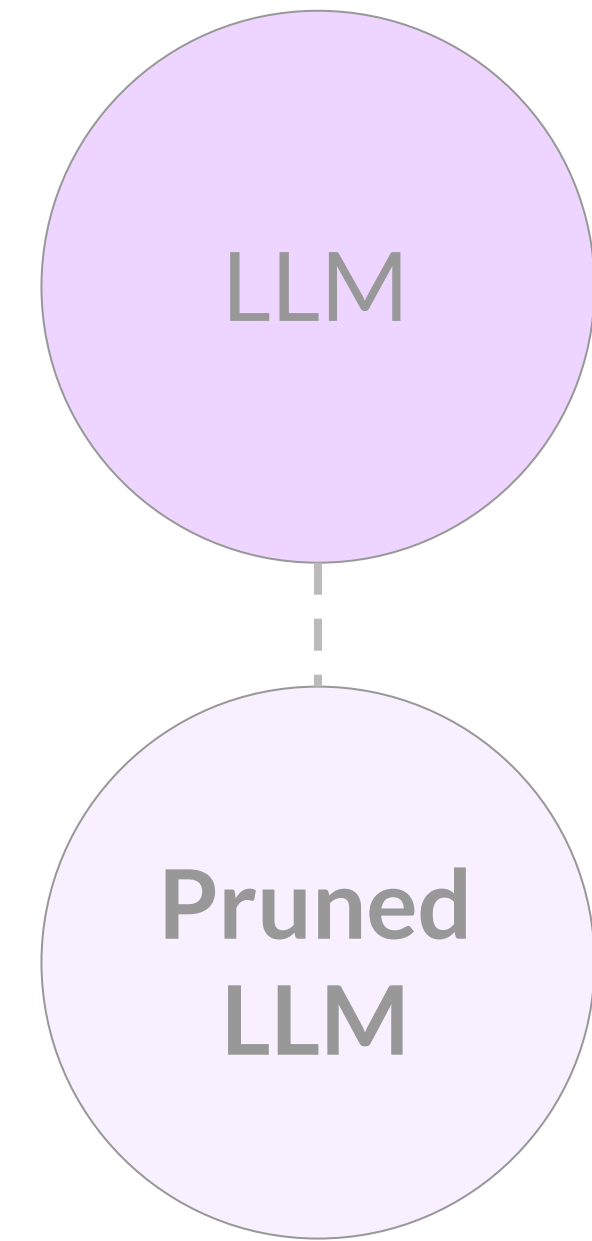
## Distillation



## Quantization

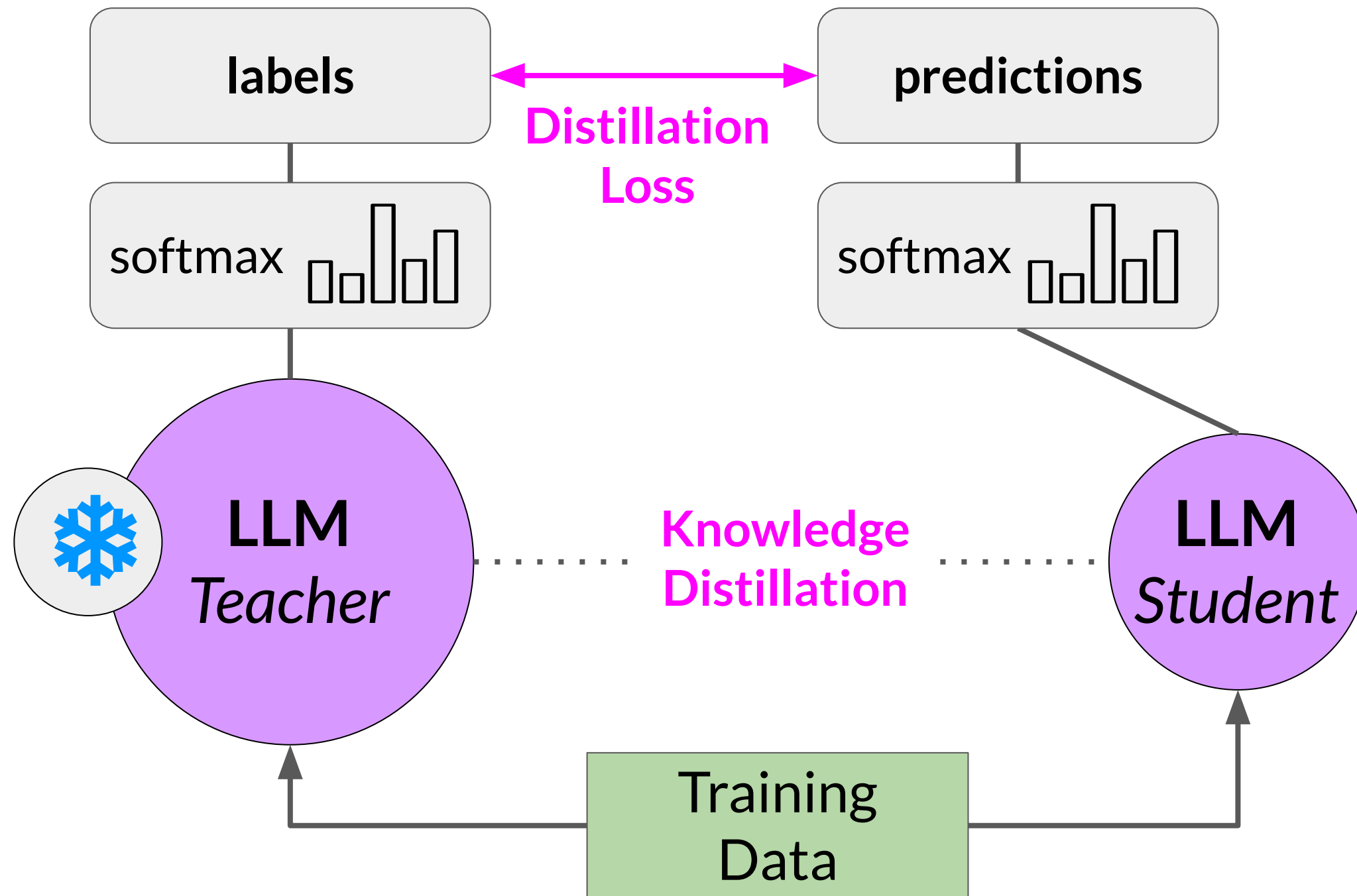


## Pruning



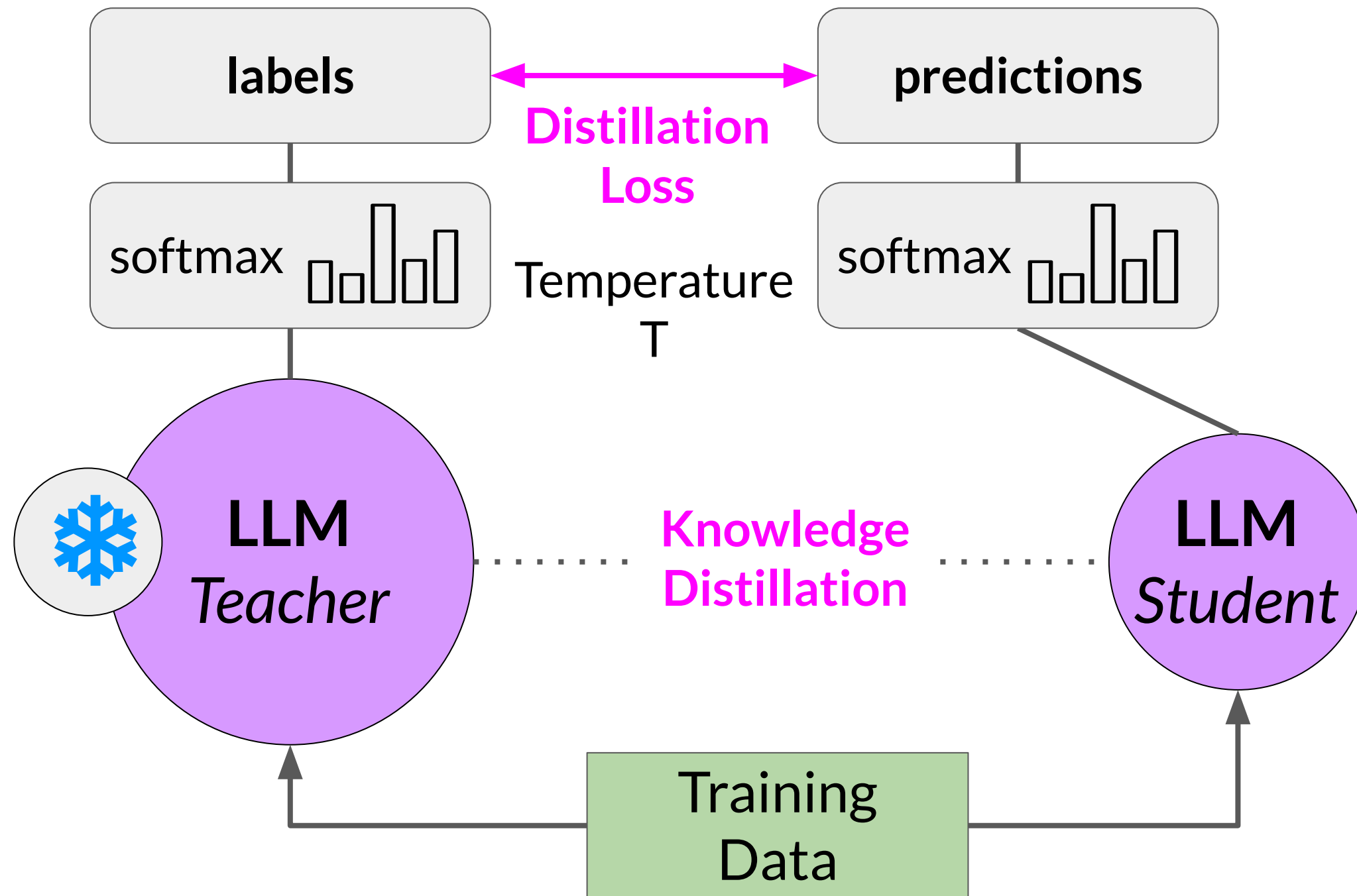
# Distillation

Train a smaller student model from a larger teacher model



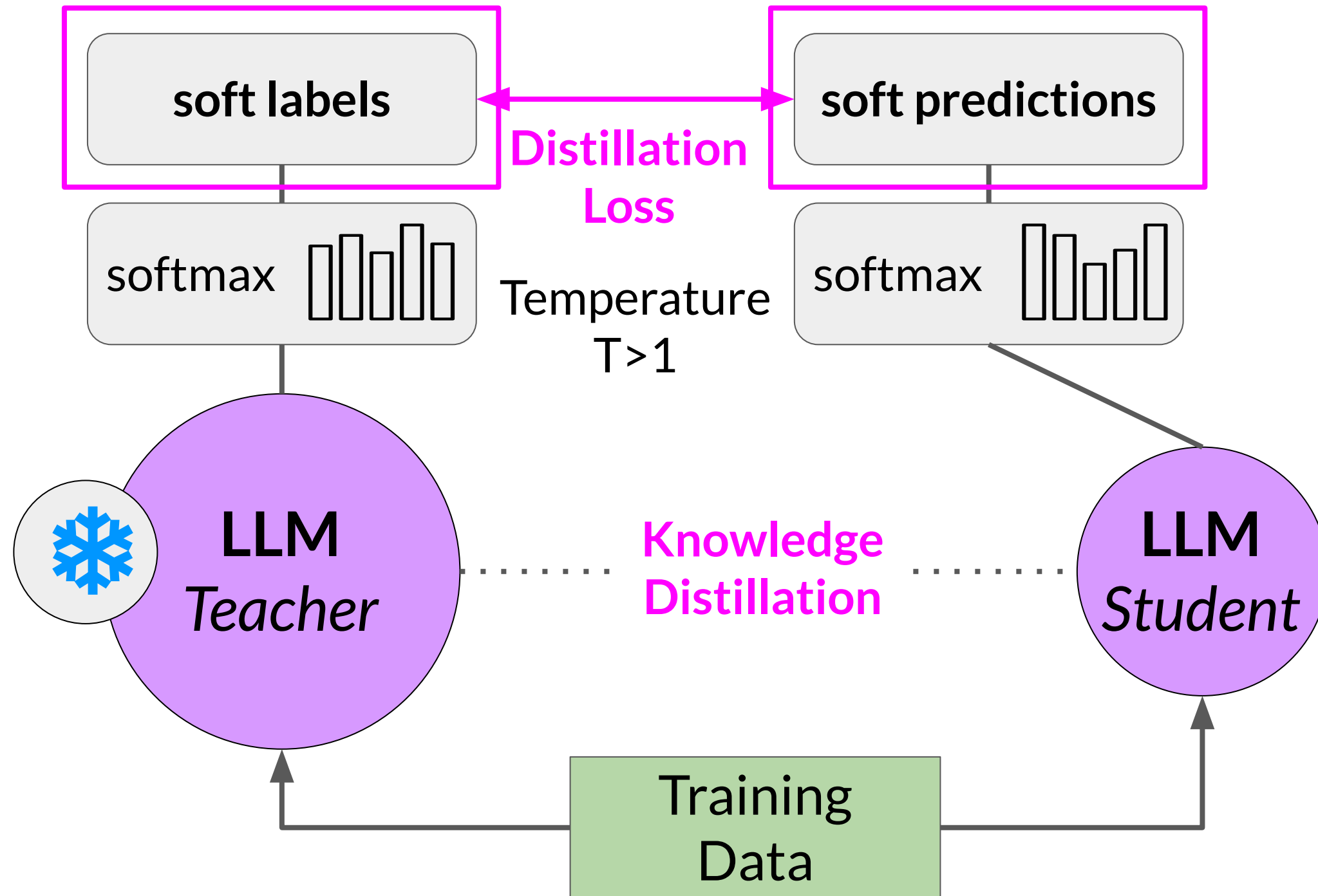
# Distillation

Train a smaller student model from a larger teacher model



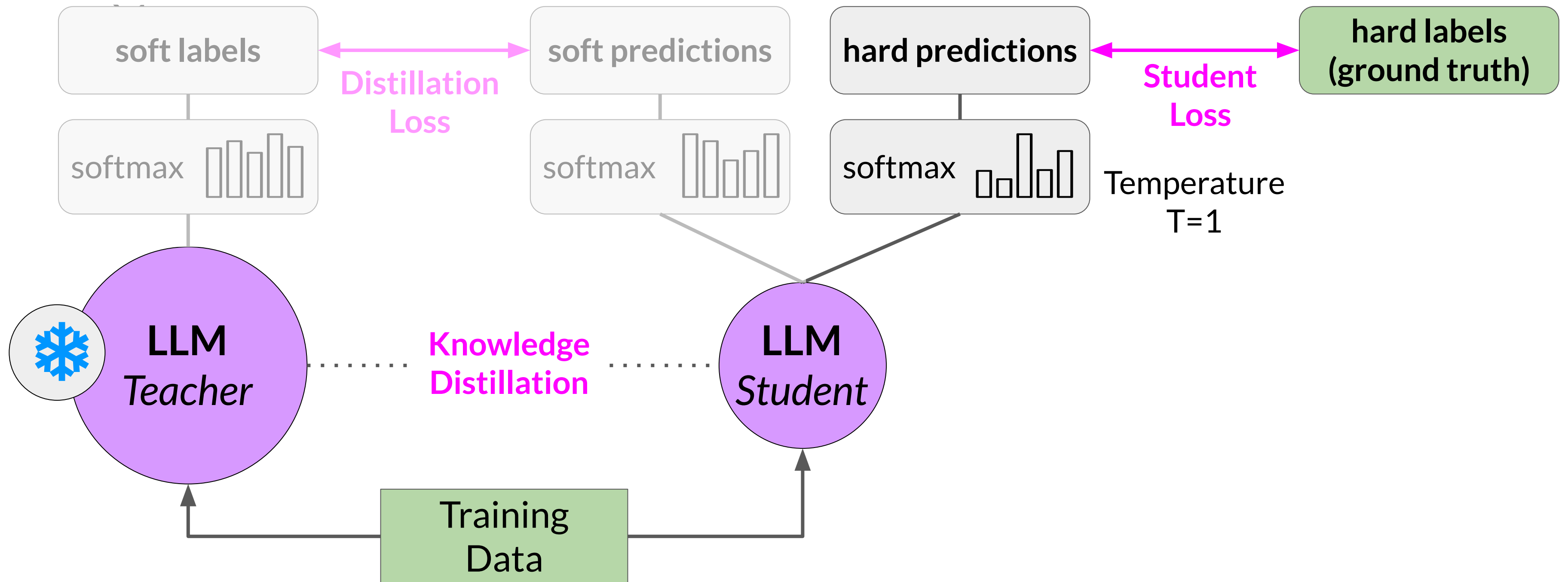
# Distillation

Train a smaller student model from a larger teacher model



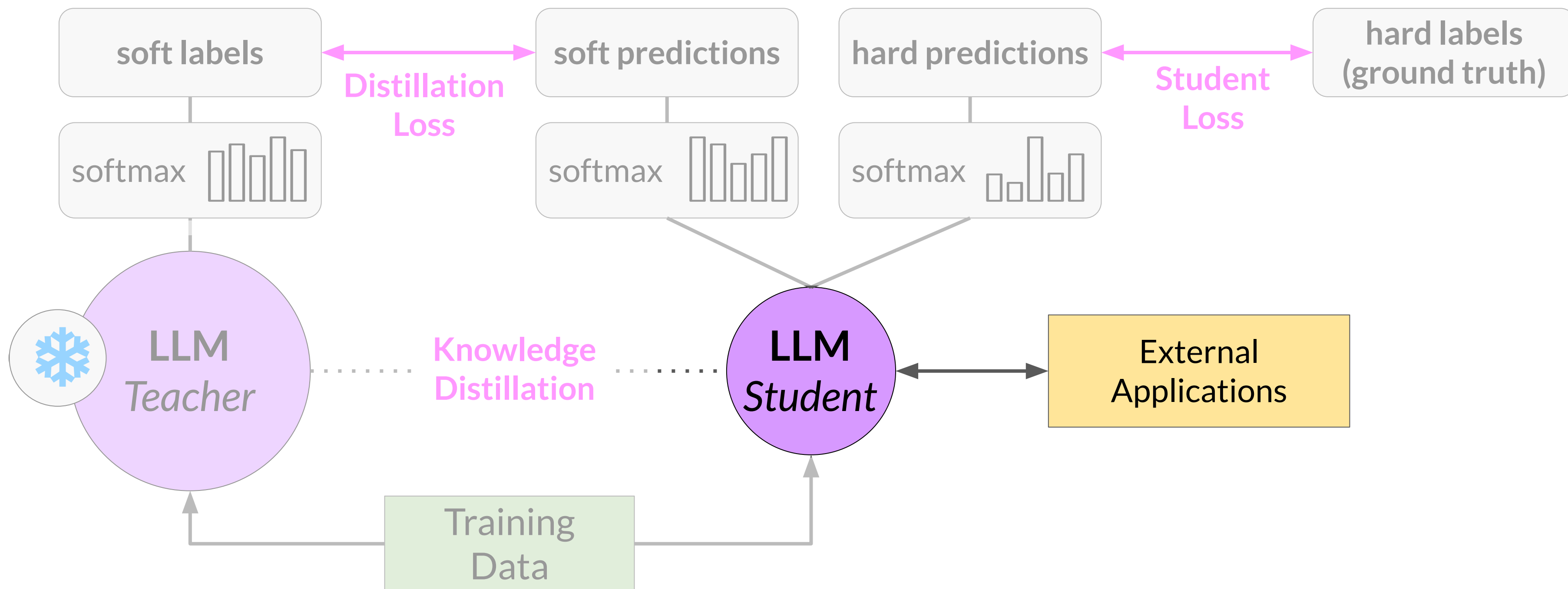
# Distillation

Train a smaller student model from a larger teacher model



# Distillation

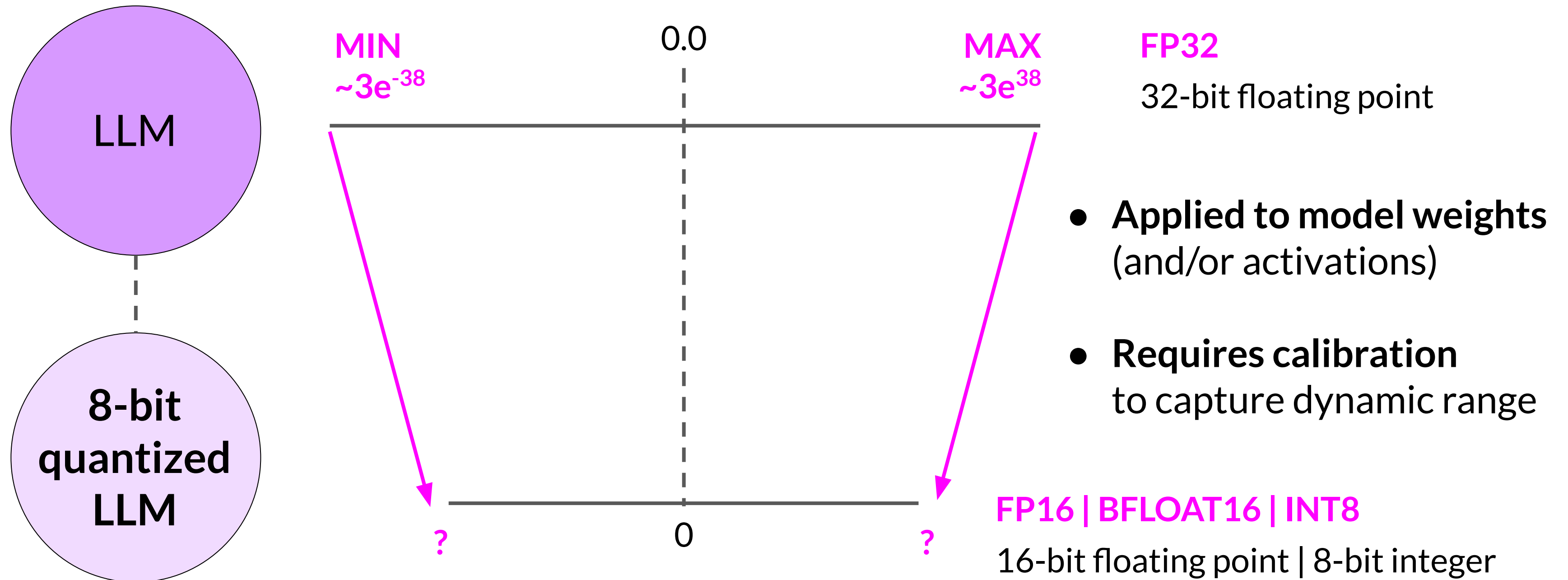
Train a smaller student model from a larger teacher model





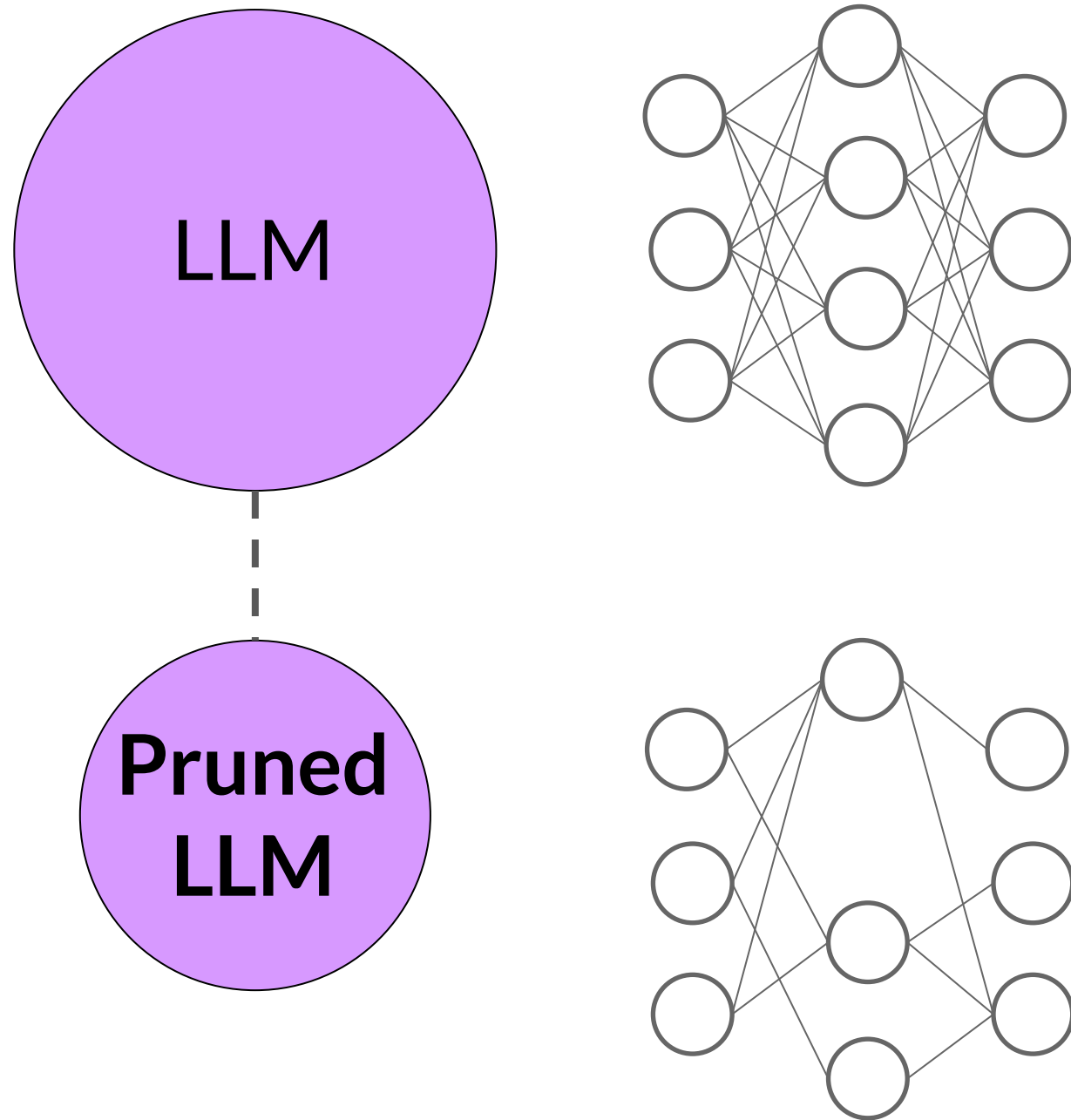
# Post-Training Quantization (PTQ)

Reduce precision of model weights



# Pruning

Remove model weights with values close or equal to zero



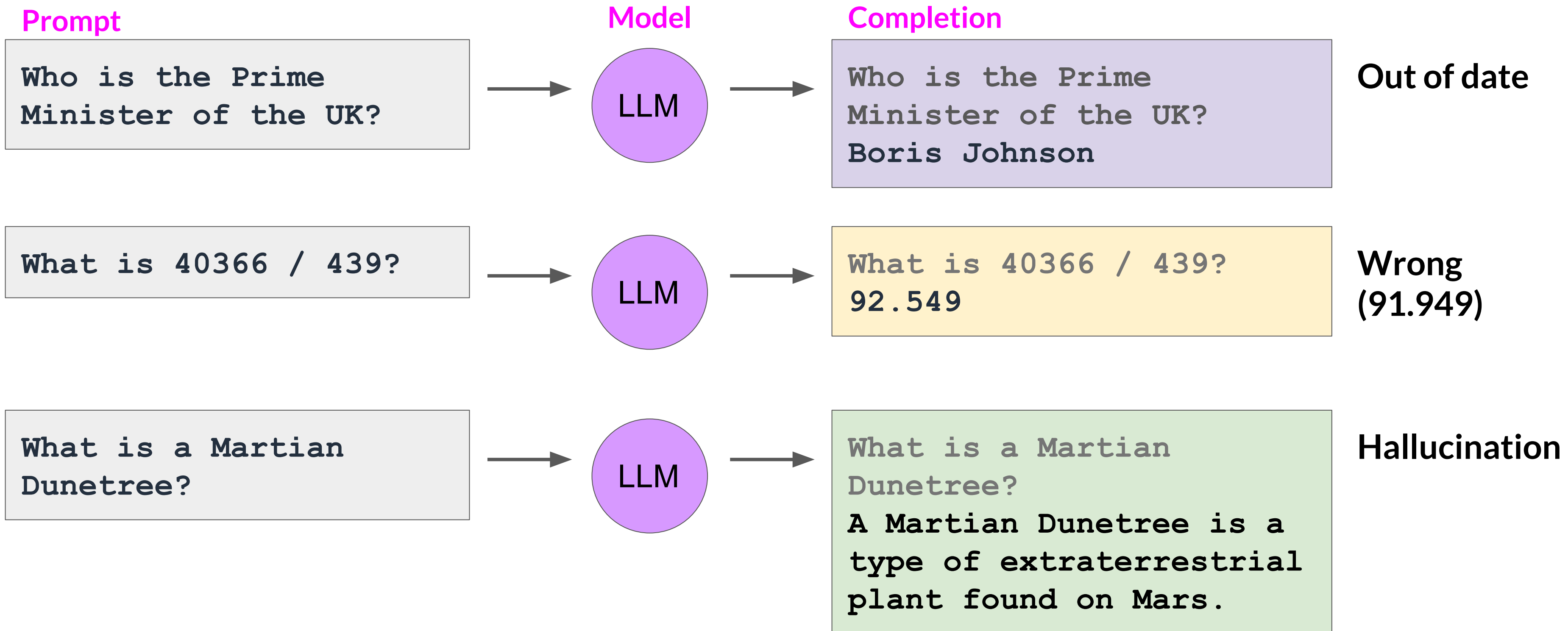
- Pruning methods
  - Full model re-training
  - PEFT/LoRA
  - Post-training
- In theory, reduces model size and improves performance
- In practice, only small % in LLMs are zero-weights

# Cheat Sheet - Time and effort in the lifecycle

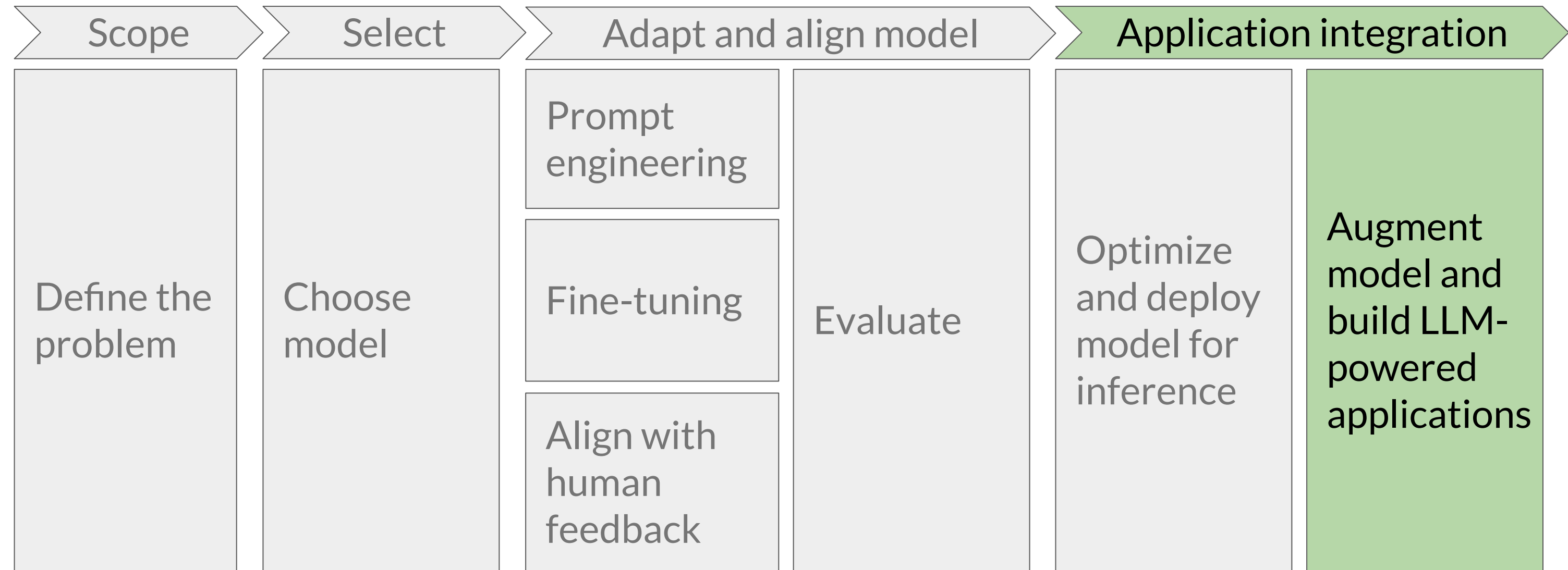
	Pre-training	Prompt engineering	Prompt tuning and fine-tuning	Reinforcement learning/human feedback	Compression/optimization/deployment
Training duration	Days to weeks to months	Not required	Minutes to hours	Minutes to hours similar to fine-tuning	Minutes to hours
Customization	<p>Determine model architecture,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

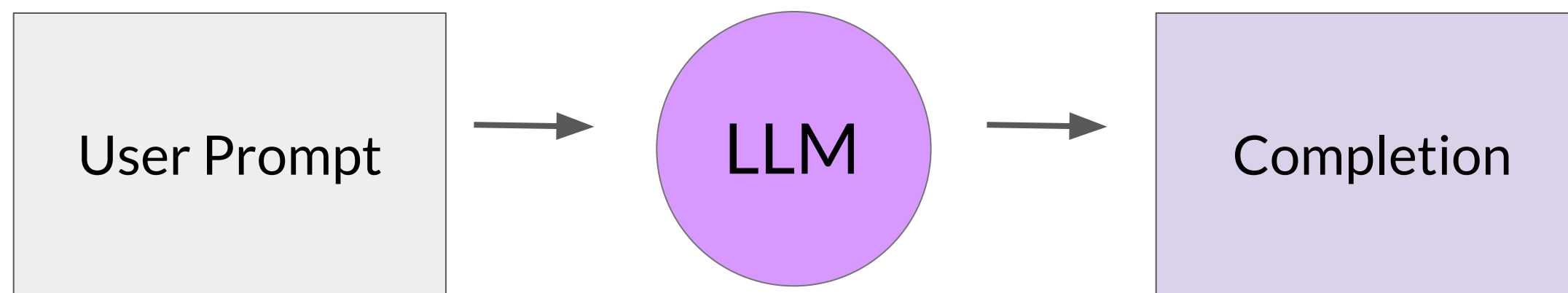
# Models having difficulty



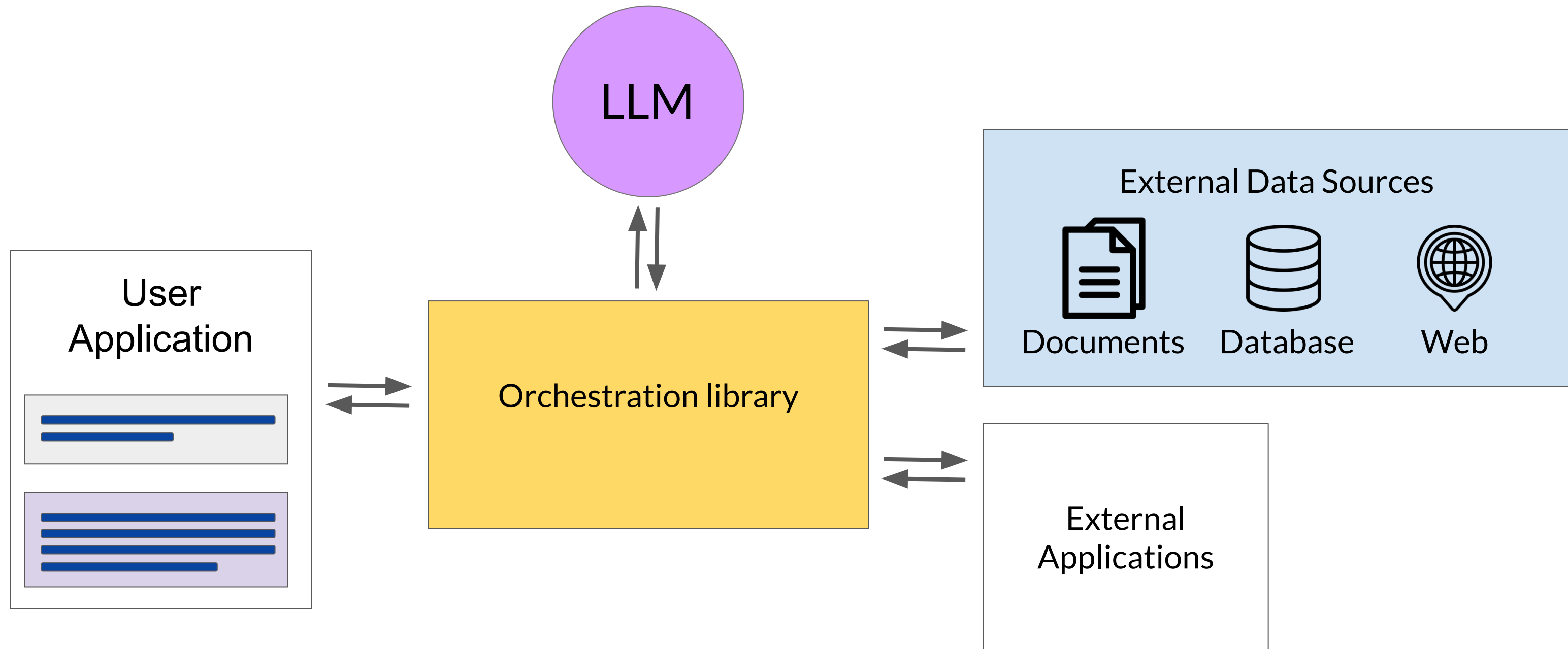
# Generative AI project lifecycle



# LLM-powered applications



# LLM-powered applications





# Retrieval augmented generation (RAG)

# Knowledge cut-offs in LLMs

Prompt

Who is the  
current Prime  
Minister of the  
United Kingdom?

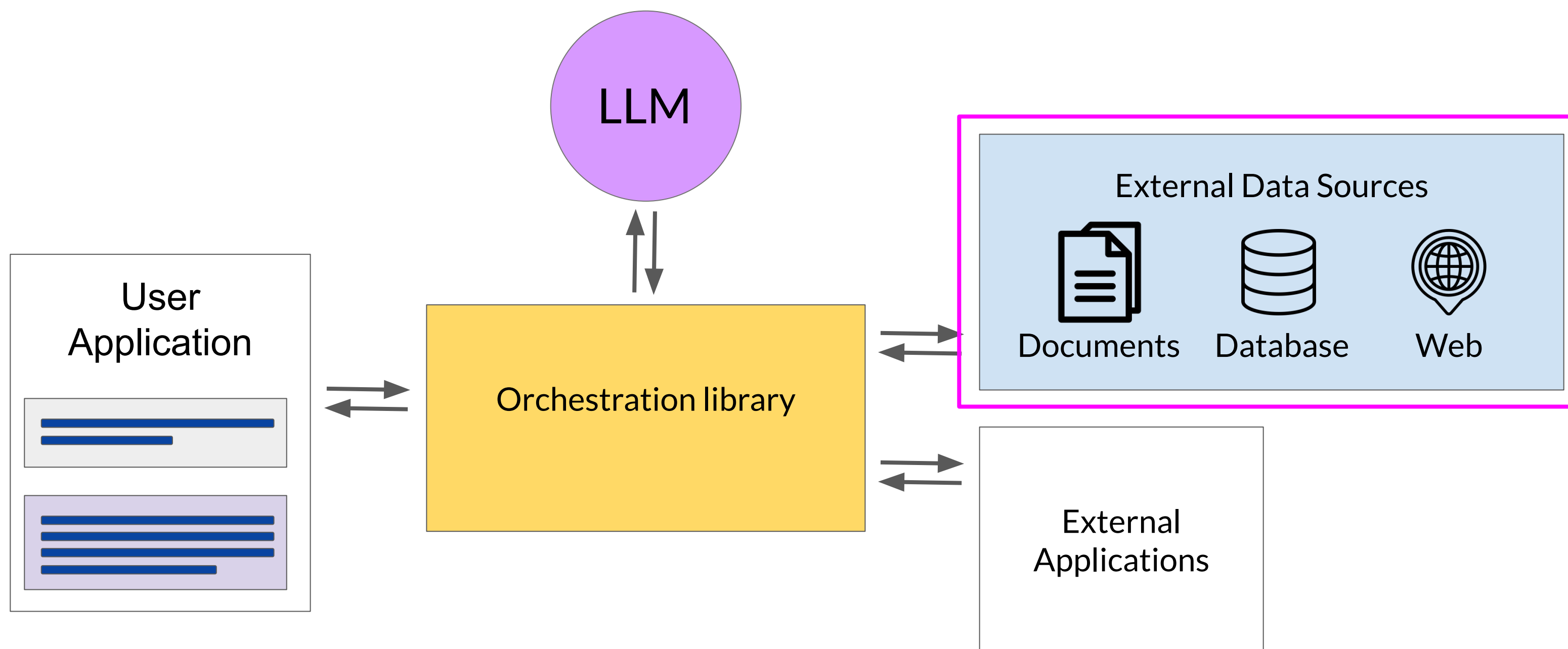
Model

LLM

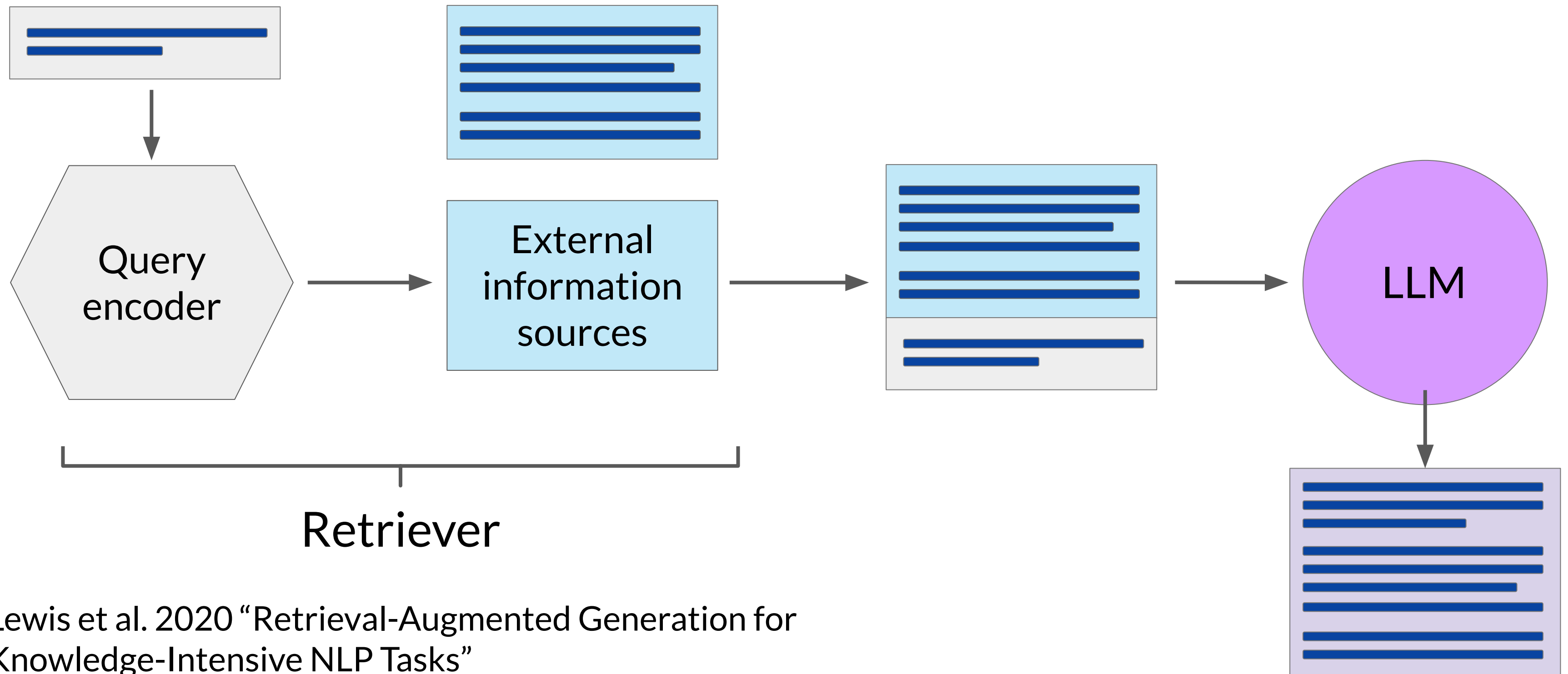
Completion

Who is the  
current Prime  
Minister of the  
United Kingdom?  
  
Boris Johnson

# LLM-powered applications



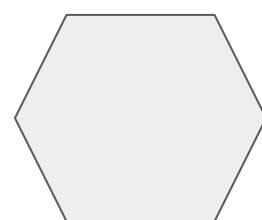
# Retrieval Augmented Generation (RAG)



# 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)

UNITED STATES DISTRICT COURT  
SOUTHERN DISTRICT OF MAINE

CASE NUMBER: 22-48710BI-SME

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

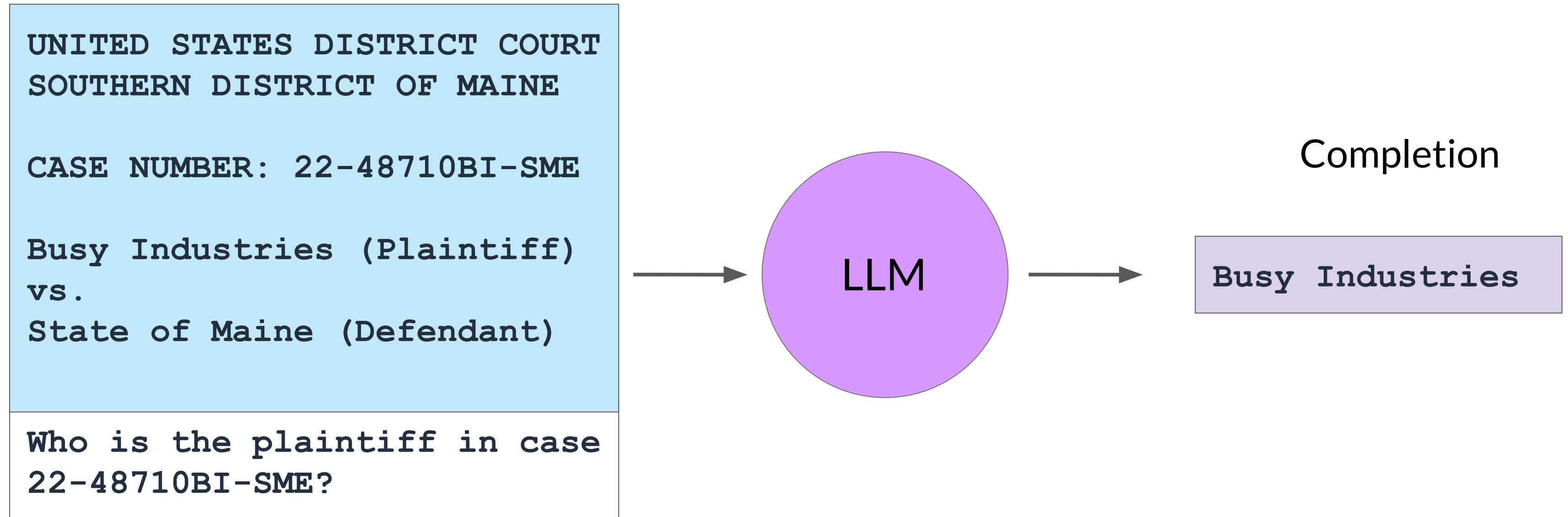


documents

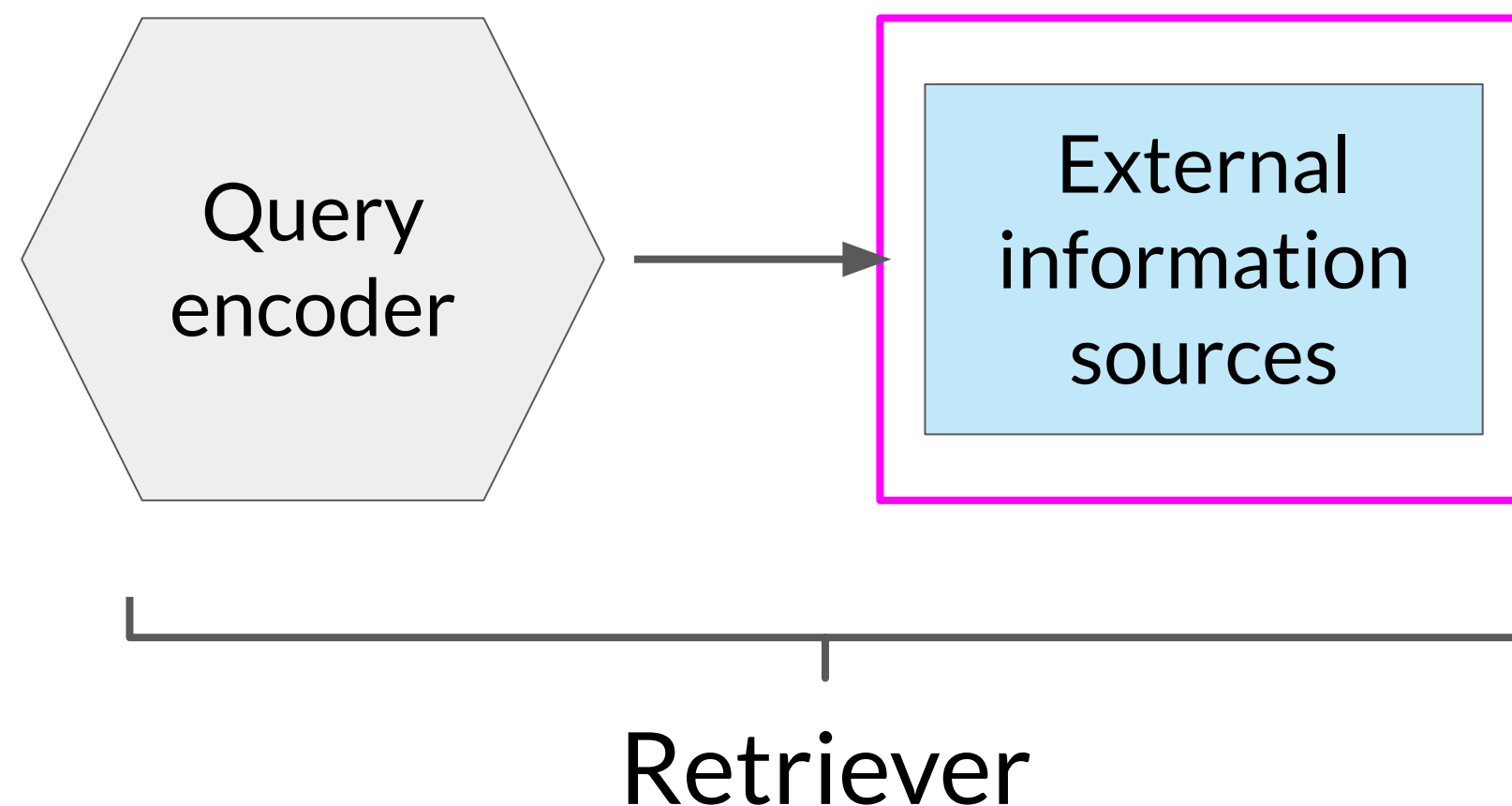
External Information Sources

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

# Example: Searching legal documents



# RAG integrates with many types of data sources



## External Information Sources

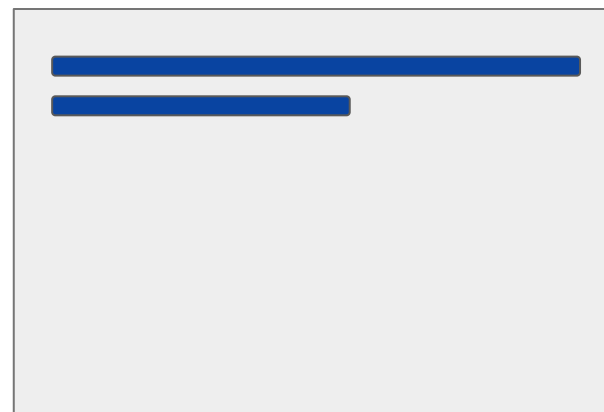
- Documents
- Wikis
- Expert Systems
- Web pages
- Databases
- Vector Store

# Data preparation for vector store for RAG

Two considerations for using external data in RAG:

1. Data must fit inside context window

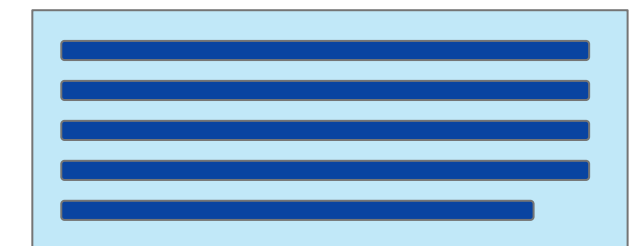
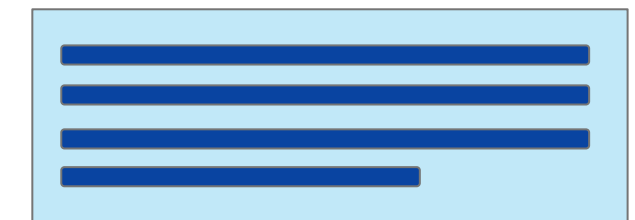
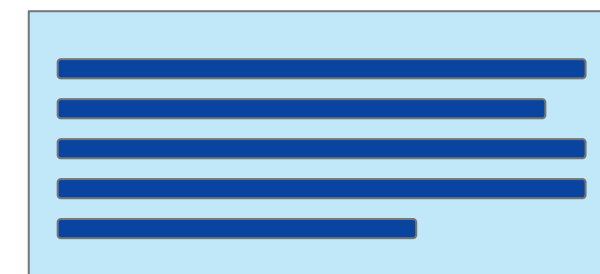
Prompt context limit  
few 1000 tokens



Single document too  
large to fit in window



Split long sources into  
short chunks



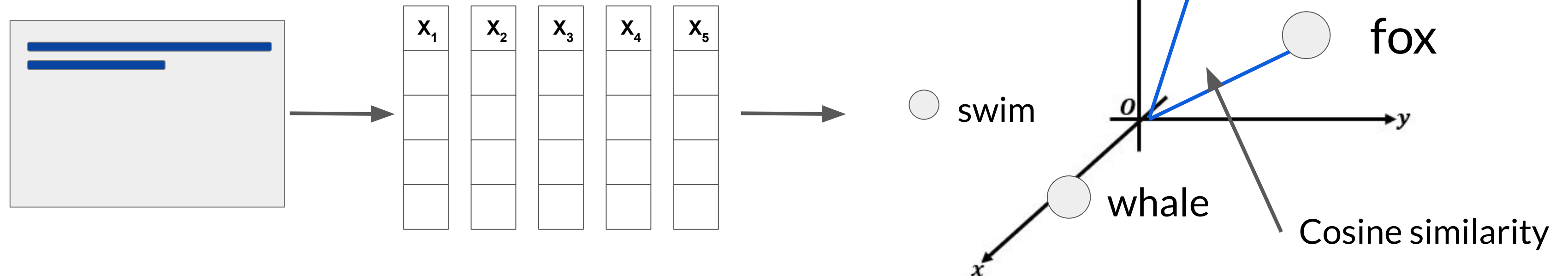


# Data preparation for RAG

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

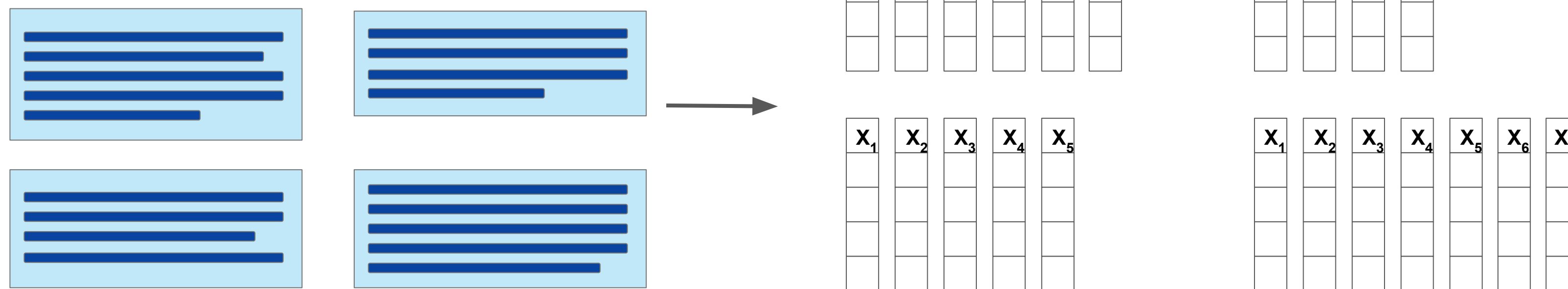


# Data preparation for RAG

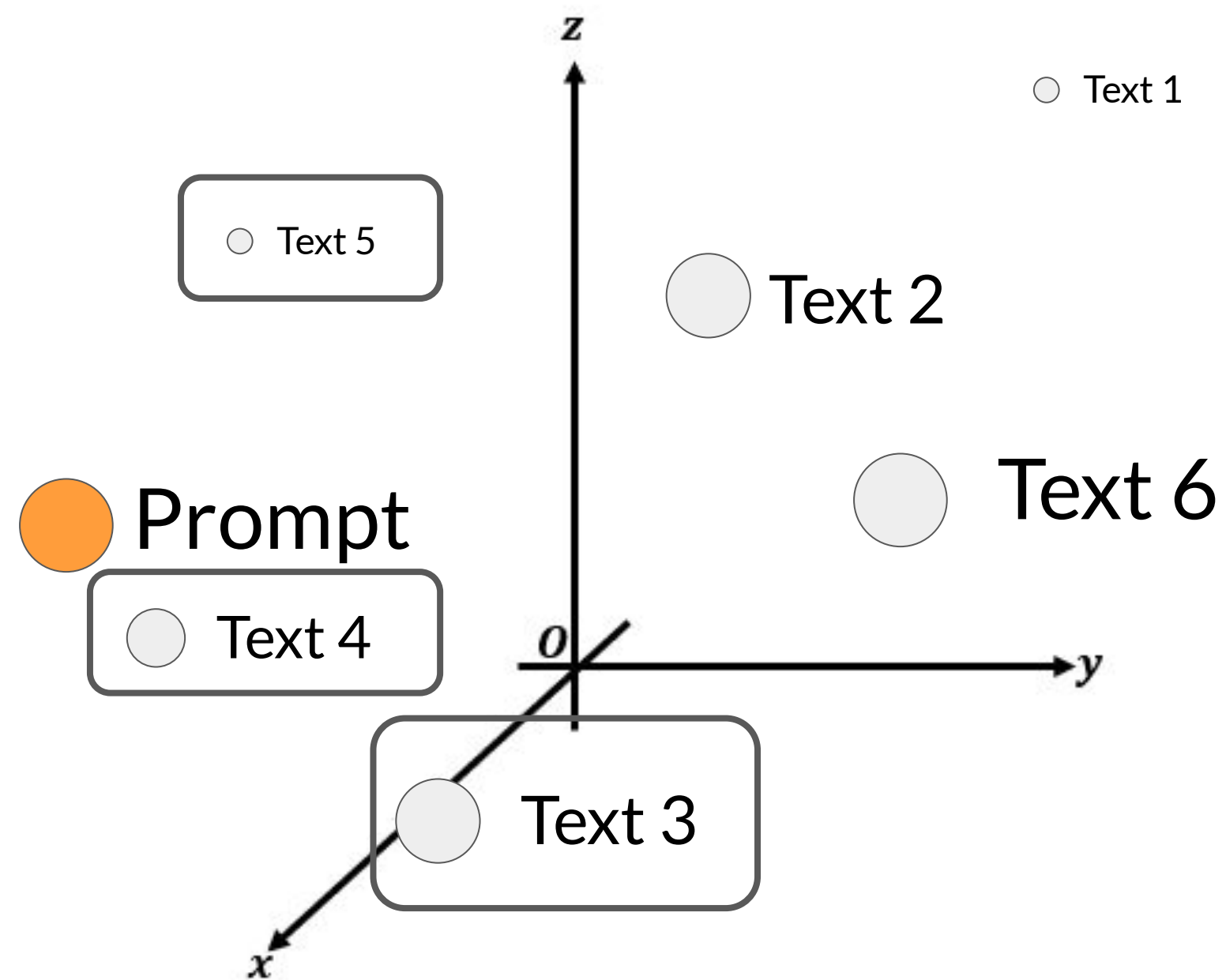
Two considerations for using external data in RAG:

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



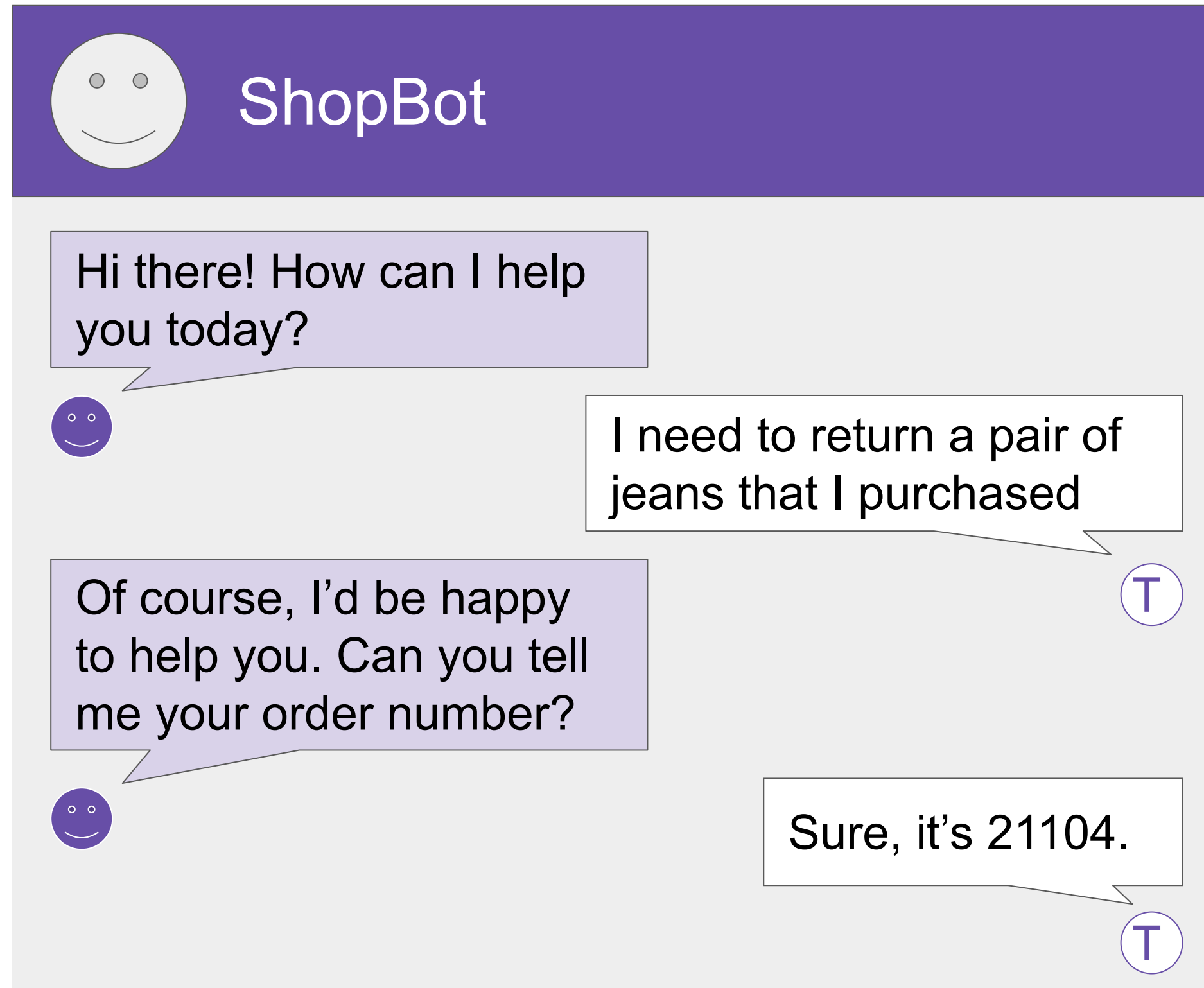
# Vector database search



- 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

 ShopBot

Ok, I've found your order.  
Do you want to return  
any other items from that  
order?

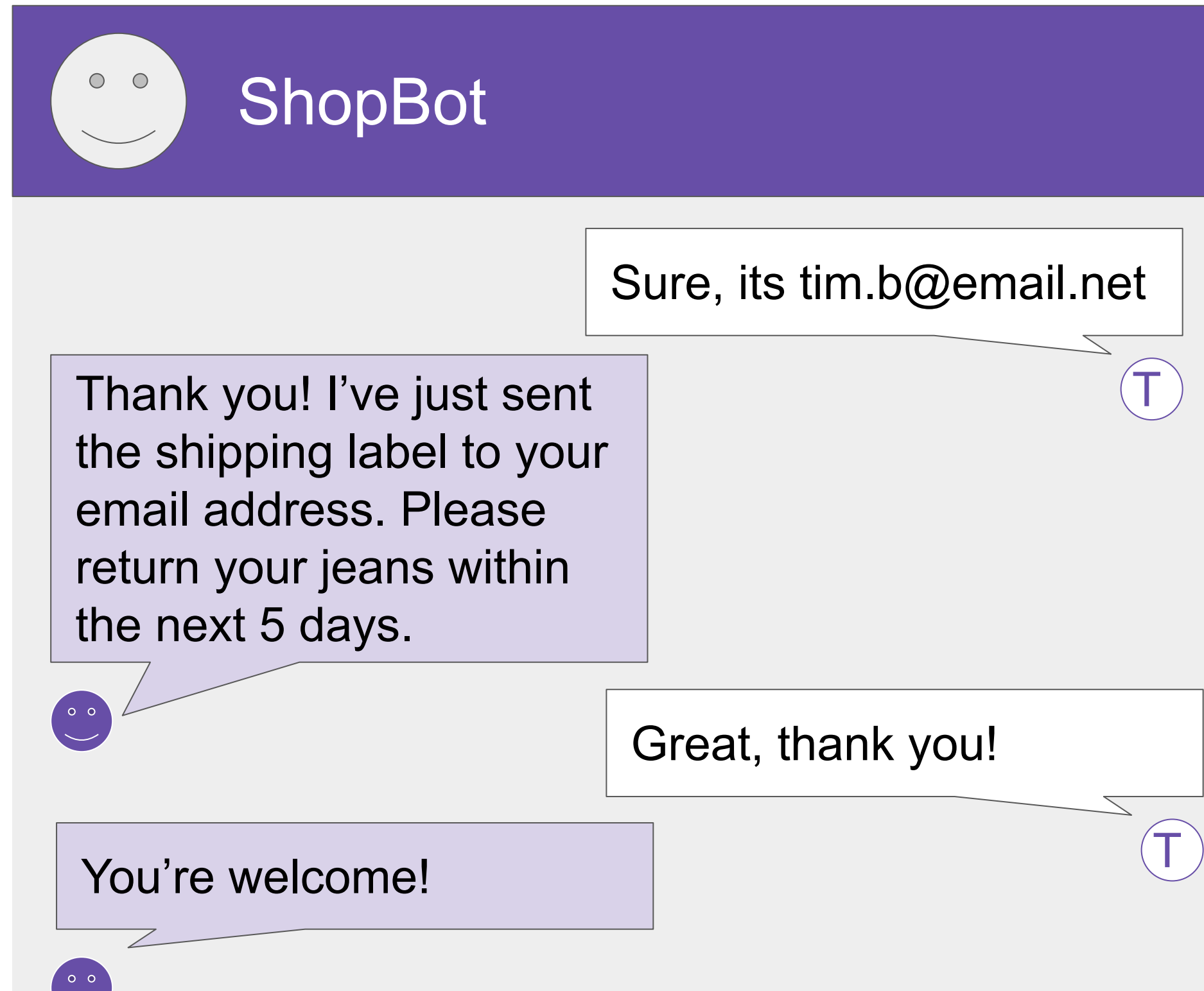
 No, only the jeans.

Ok great. Let me get a  
return label from our  
shipping partner. Can  
you remind me of the  
email you used to order?

 T

# Having an LLM initiate a clothing return

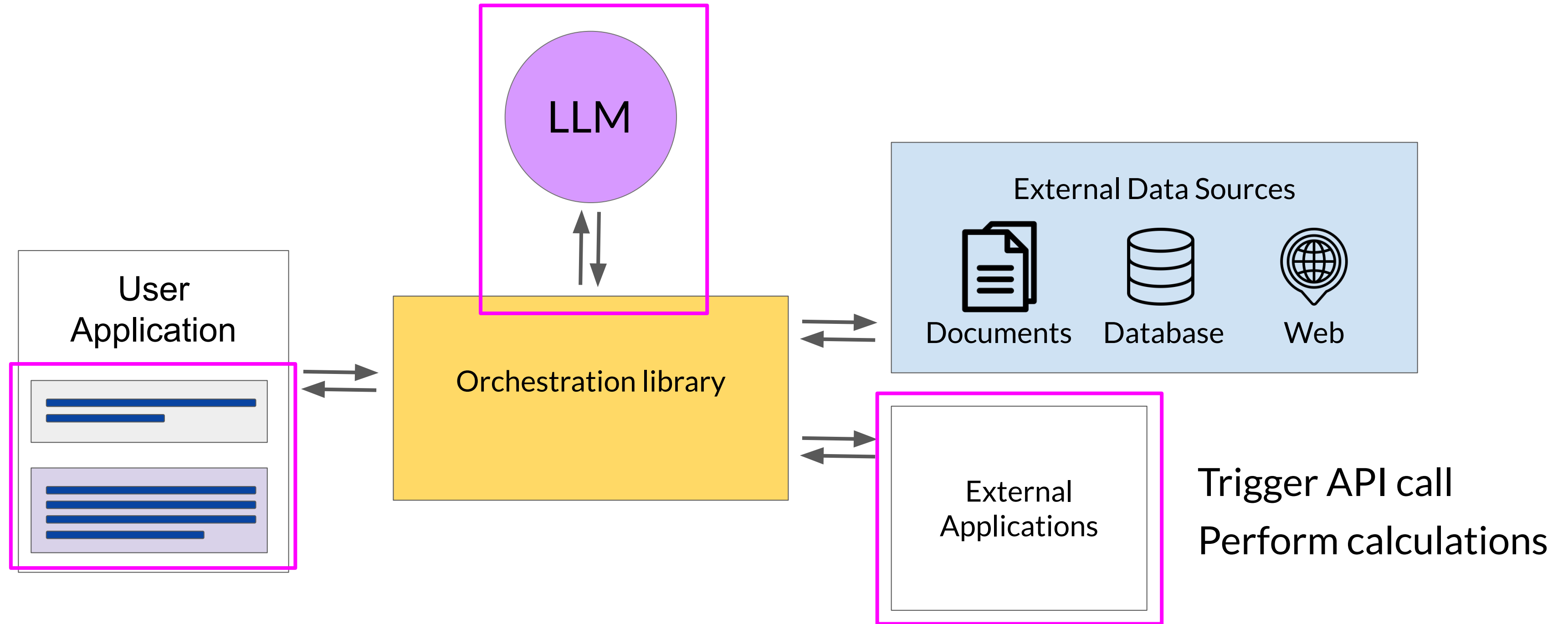
API call to the shipper



The image shows a chat interface for 'ShopBot'. At the top, there is a purple header with a white smiley face icon and the text 'ShopBot'. The chat area has a light gray background. The conversation consists of the following messages:

- A white message bubble on the right: 'Sure, its tim.b@email.net' with a blue circular icon containing a white 'T' at the bottom right.
- A purple message bubble on the left: 'Thank you! I've just sent the shipping label to your email address. Please return your jeans within the next 5 days.' with a white smiley face icon at the bottom left.
- A white message bubble on the right: 'Great, thank you!' with a blue circular icon containing a white 'T' at the bottom right.
- A purple message bubble on the left: 'You're welcome!' with a white smiley face icon at the bottom left.

# 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

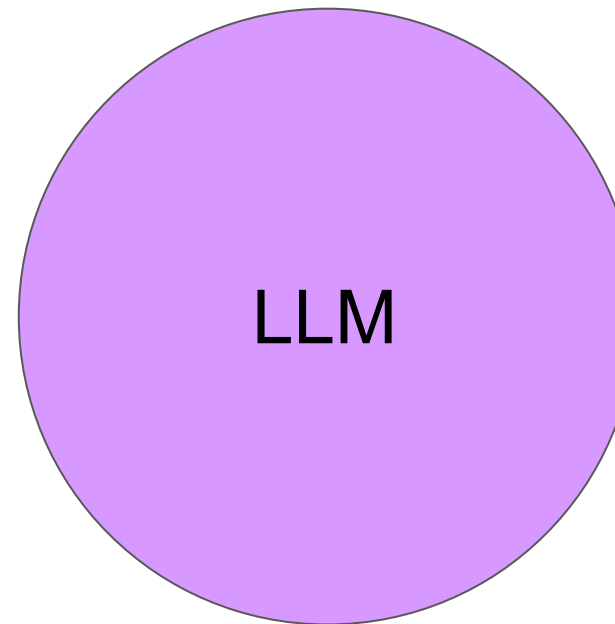
## Prompt

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?

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”

# Chain-of-Thought Prompting can help LLMs reason

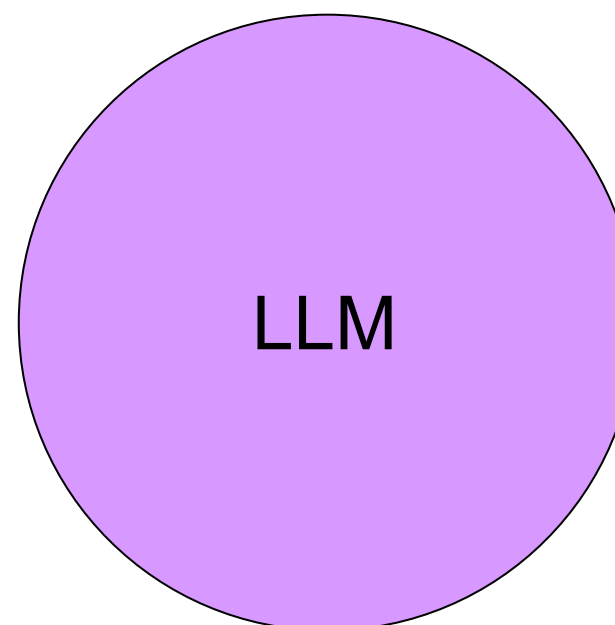
## Prompt

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



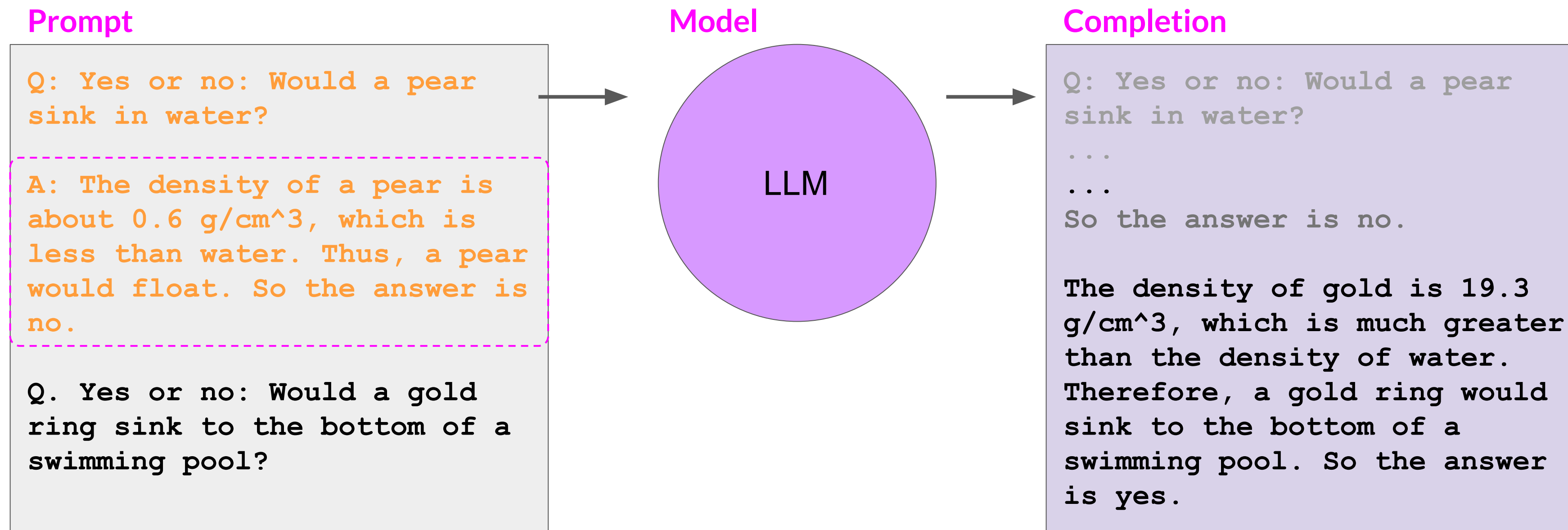
## Completion

Q: Roger has 5 tennis balls.  
...  
...  
...  
how many apples do they have?

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had  $23 - 20 = 3$ . They bought 6 more apples, so they have  $3 + 6 = 9$ . The answer is 9. ✓

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

Prompt

What is  $40366 / 439$ ?

Model

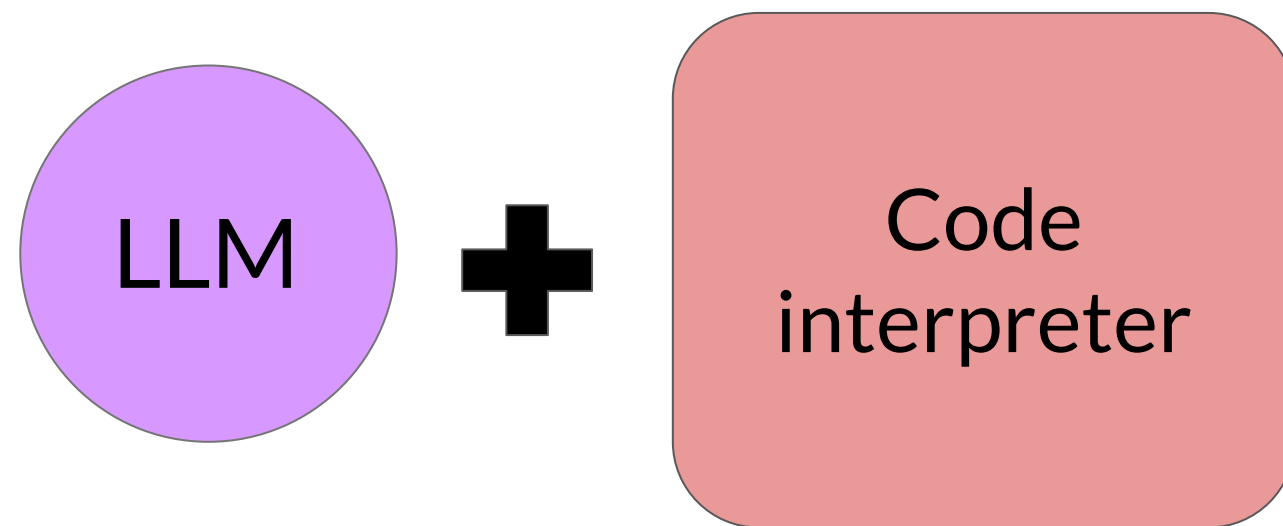
LLM

Completion

What is  $40366 / 439$ ?  
92.549



# Program-aided language (PAL) models



## 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"

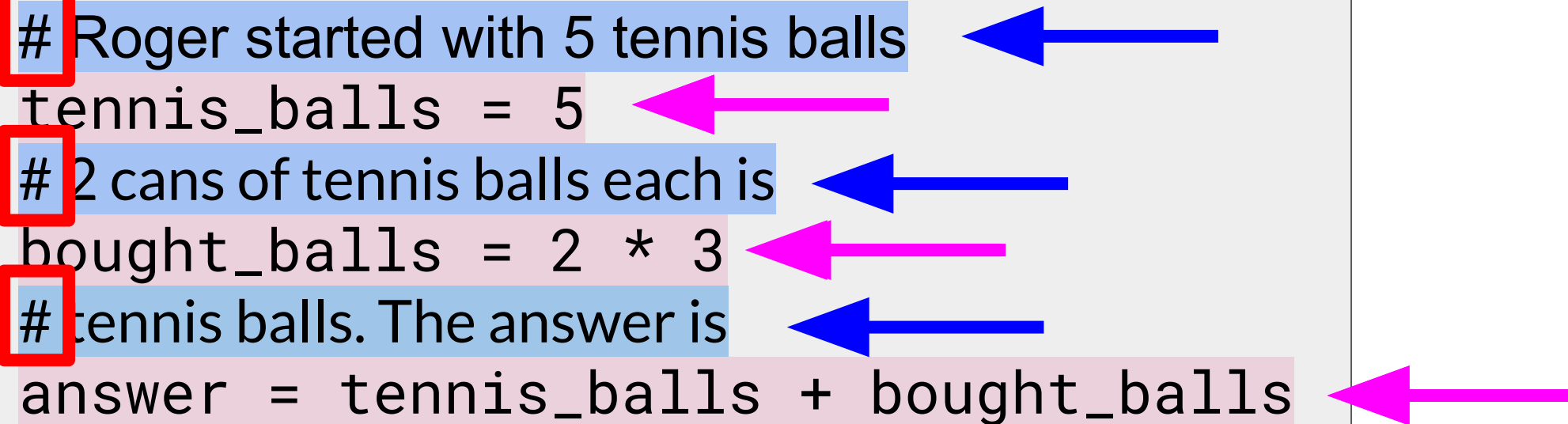
# PAL example

## 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

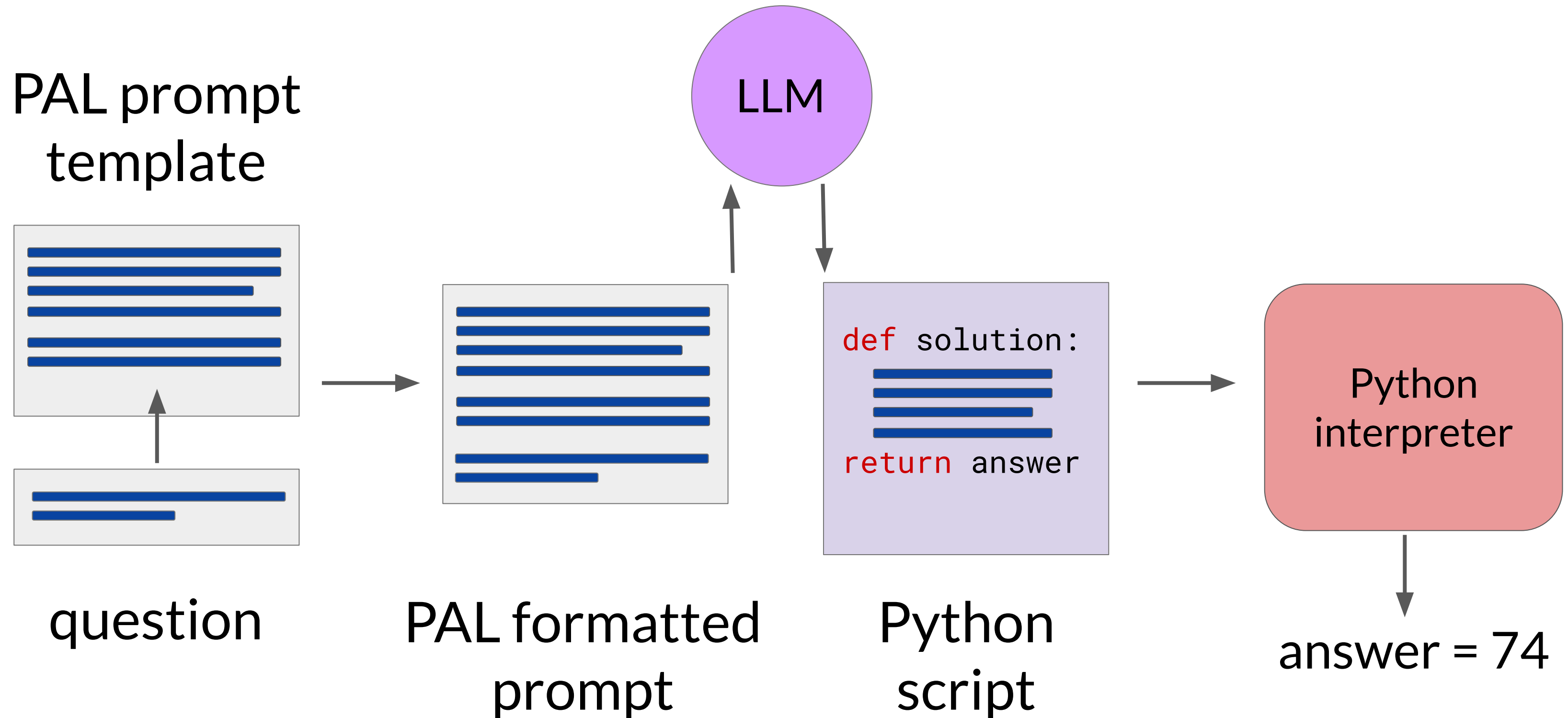
answer = loaves\_baked

- loaves\_sold\_morning

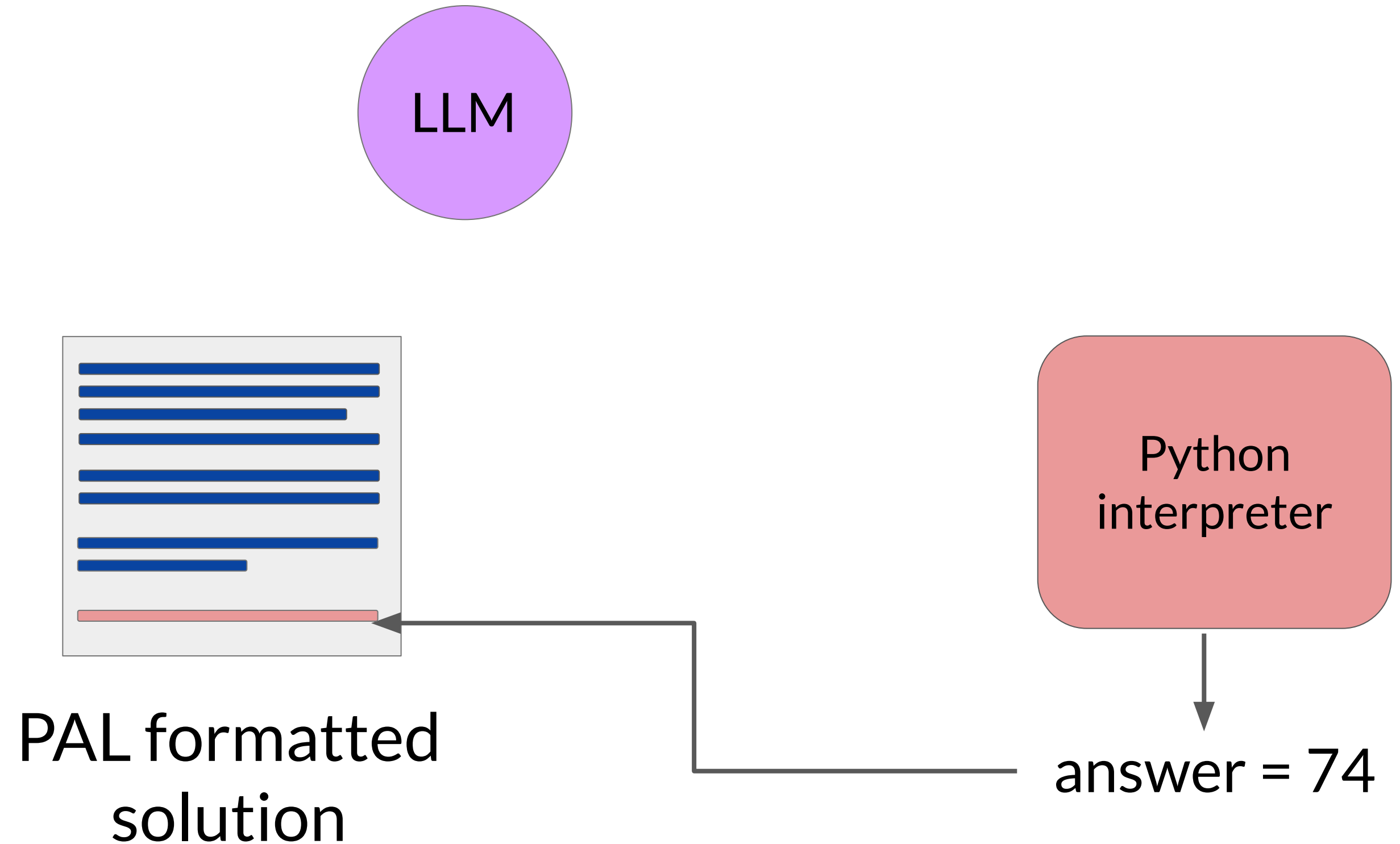
- loaves\_sold\_afternoon

+ loaves\_returned

# Program-aided language (PAL) models

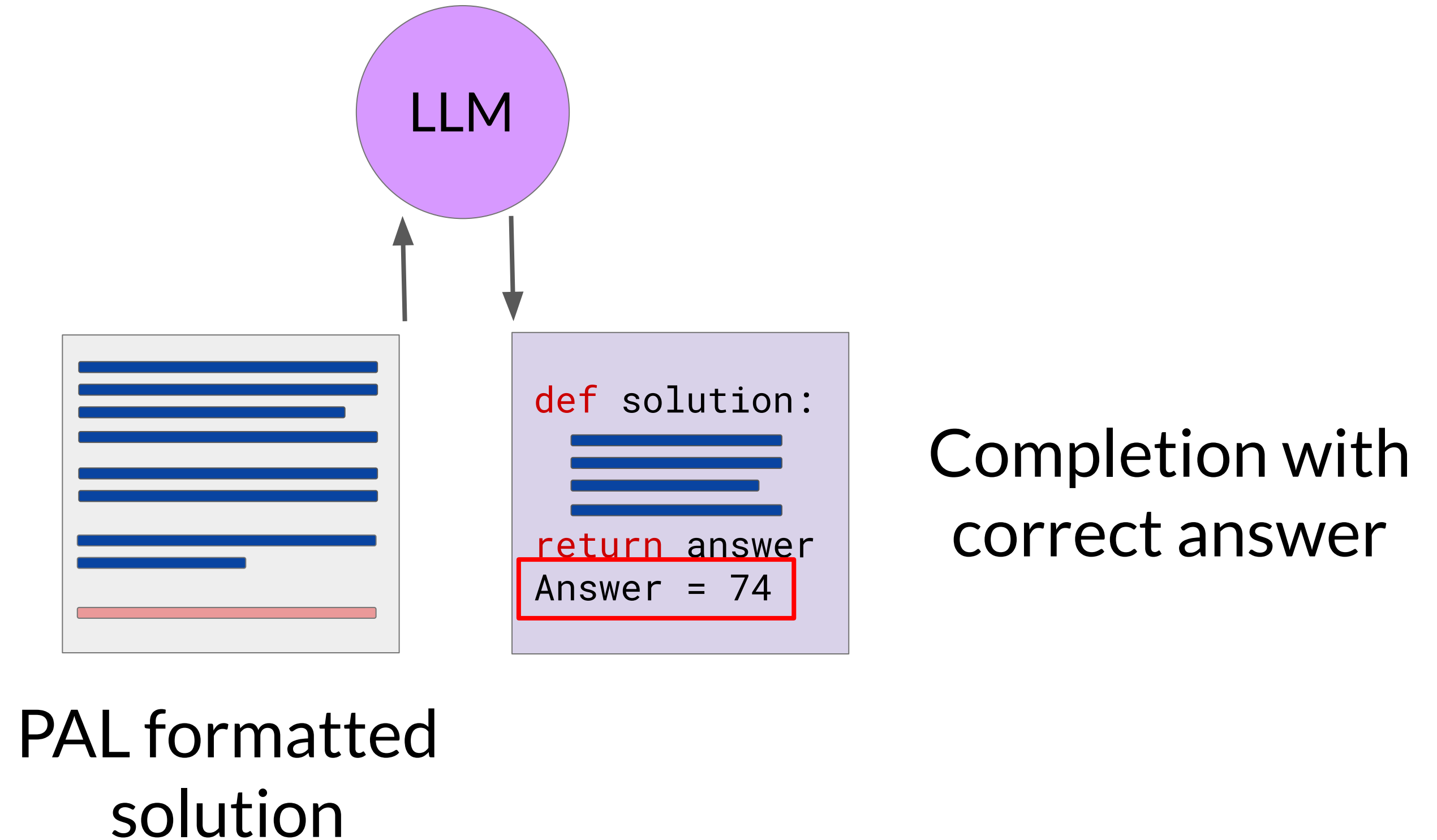


# Program-aided language (PAL) models

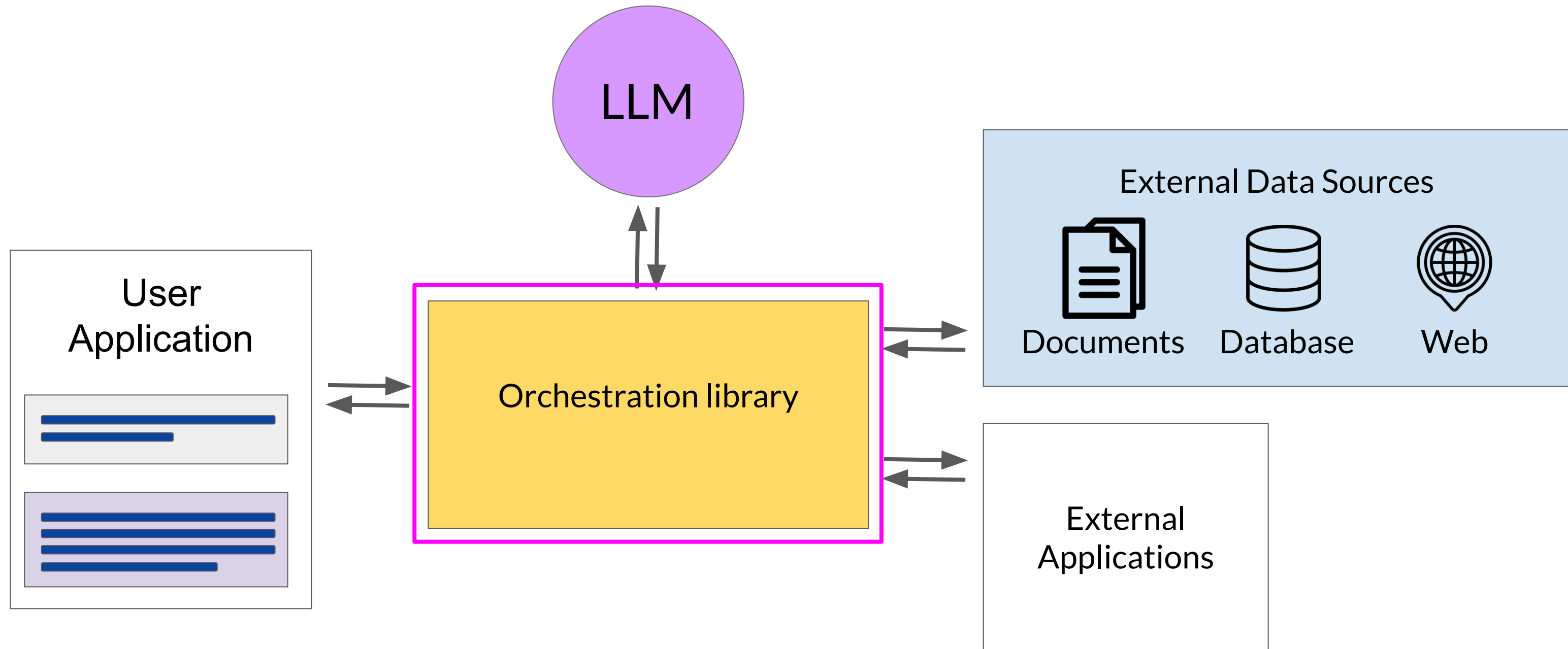




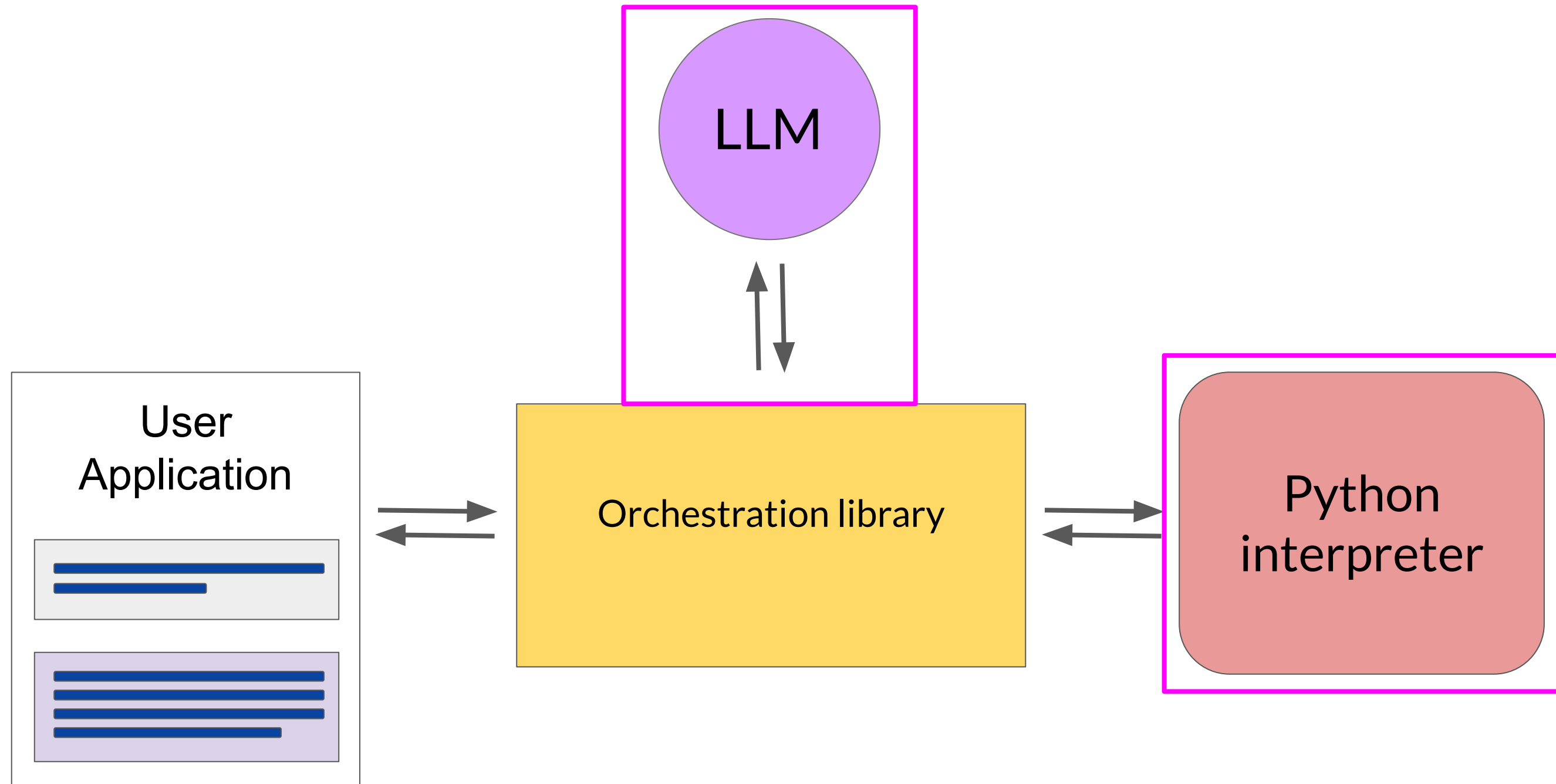
# Program-aided language (PAL) models



# LLM-powered applications

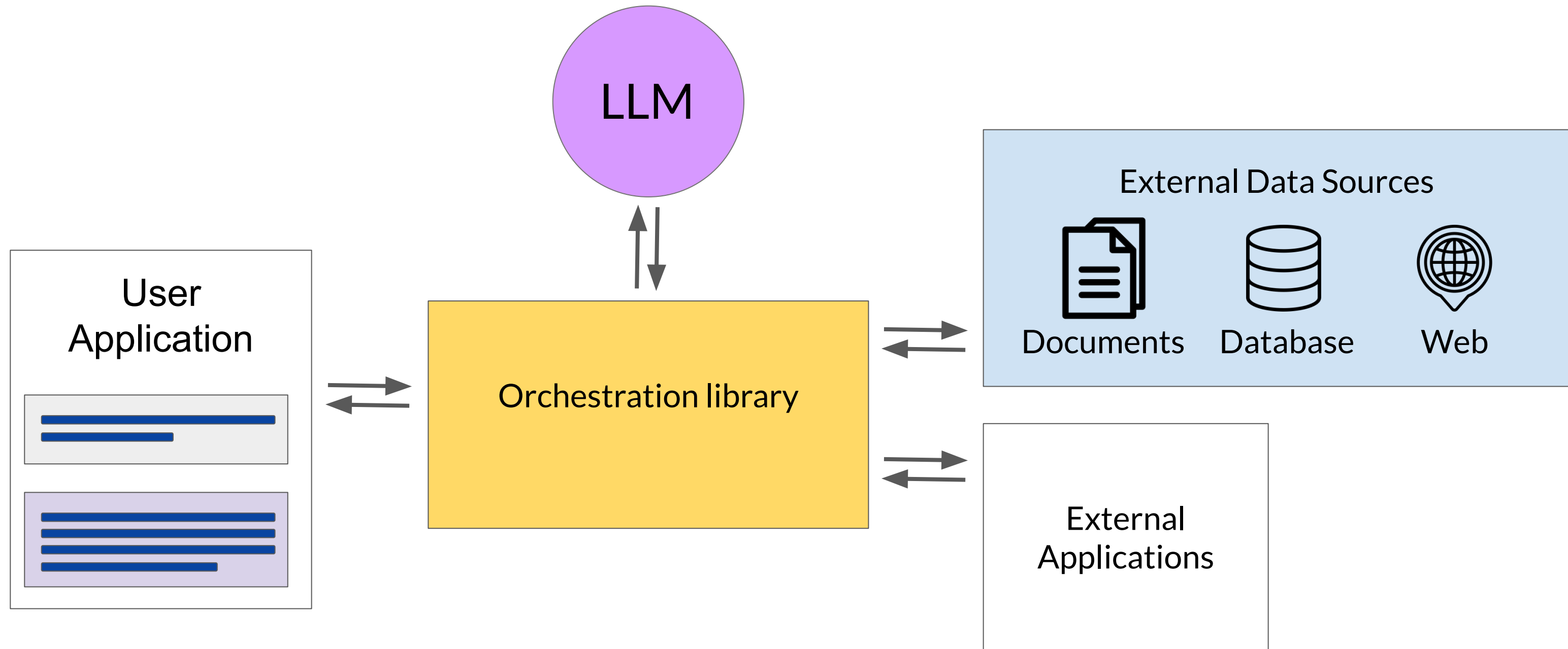


# PAL architecture



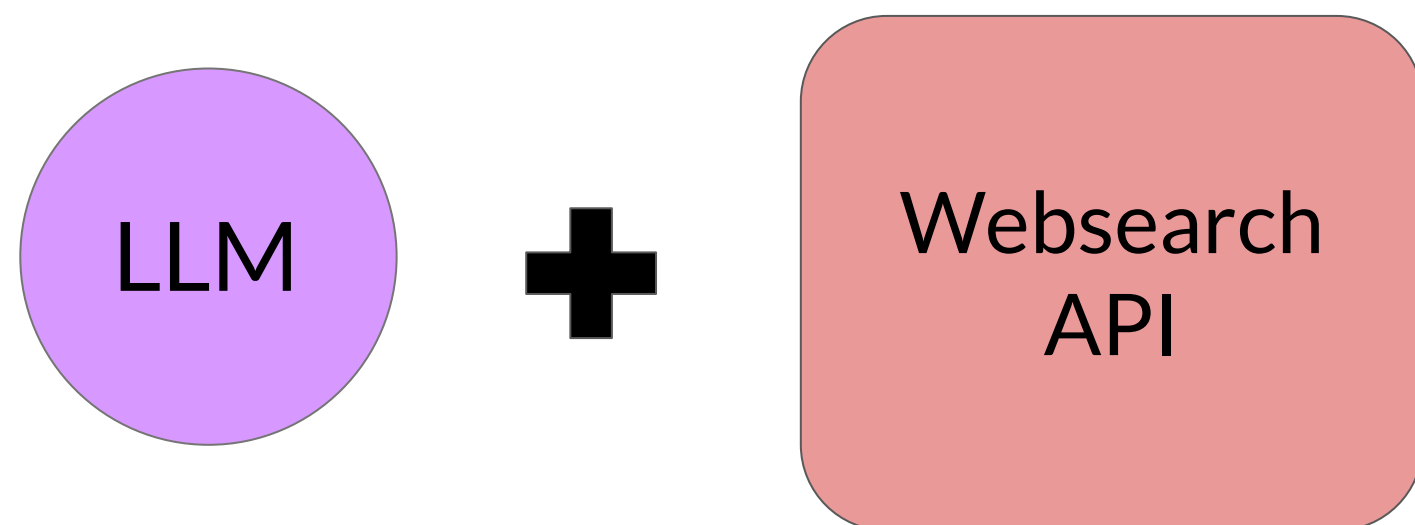


# LLM-powered applications



# ReAct: Combining reasoning and action in LLMs

# ReAct: Synergizing Reasoning and Action in LLMs

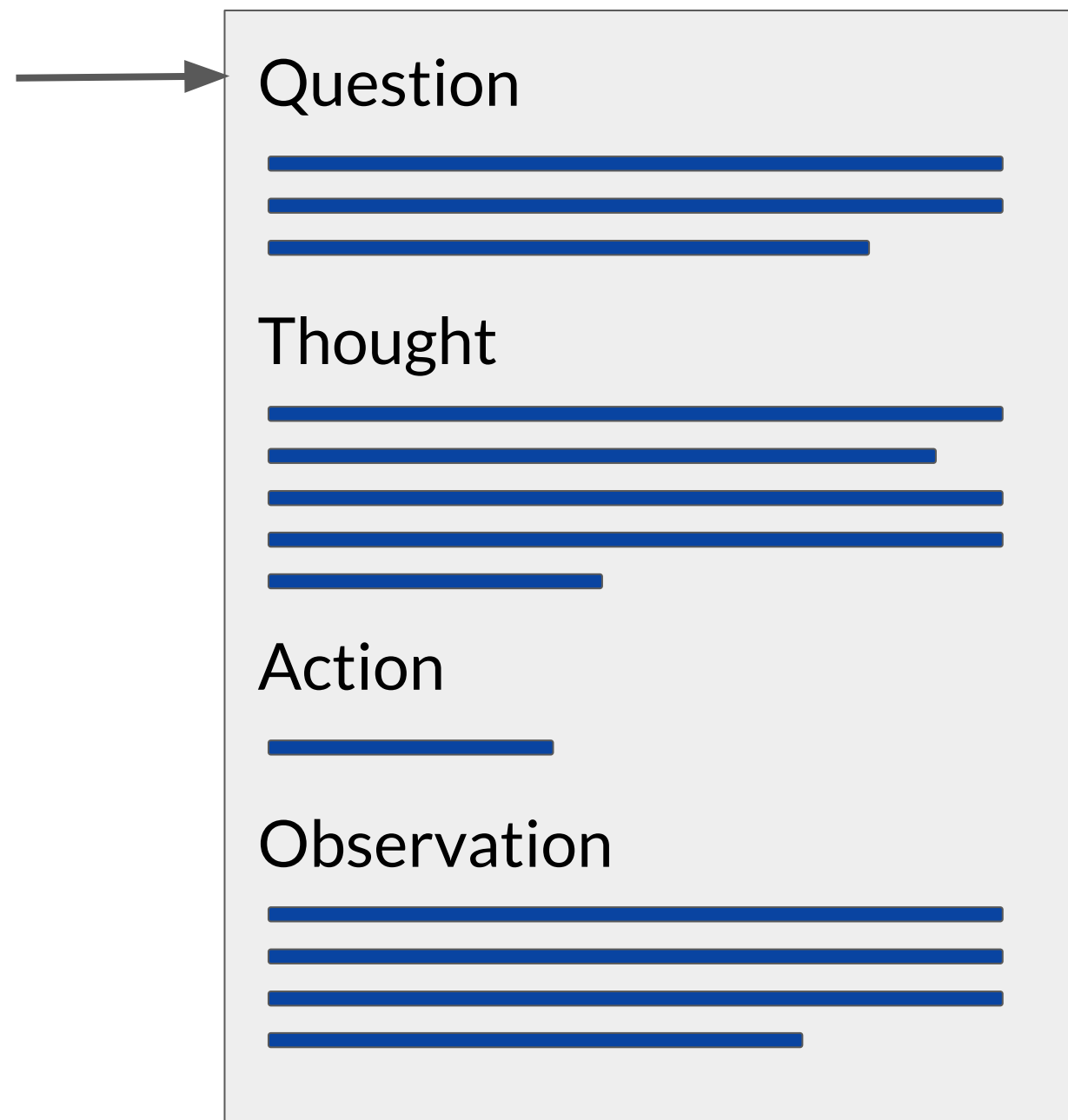


**HotPot QA:** multi-step question answering  
**Fever:** Fact verification

<p>(1) Hotspot QA</p> <p>Question: Aside from the Apple Remote, what other device can control the program Apple Remote was originally designed to interact with?</p> <p>(1a) Standard</p> <p>Answer: iPod</p> <p>(1b) CoT (Reason Only)</p> <p>Thought: Let's think step by step. Apple Remote was originally designed to interact with Apple TV. Apple TV can be controlled by iPhone, iPad, and iPod Touch. So the answer is iPhone, iPad, and iPod Touch.</p> <p>Answer: iPhone, iPad, iPod Touch</p> <p>(1c) Act-Only</p> <p>Act 1: Search[Apple Remote] Obs 1: The Apple Remote is a remote control ...</p> <p>Act 2: Search[Front Row] Obs 2: Could not find [Front Row]. Similar: ...</p> <p>Act 3: Search[Front Row (software)] Obs 3: Front Row is a discontinued media center software ...</p> <p>Act 4: Finish[yes]</p>	<p>(1d) ReAct (Reason + Act)</p> <p>Thought 1: I need to search Apple Remote and find the program it was originally designed to interact with. Act 1: Search[Apple Remote] Obs 1: The Apple Remote is a remote control introduced in October 2005 by Apple ... originally designed to control the Front Row media center program ...</p> <p>Thought 2: Apple Remote was originally designed to control the Front Row media center program. I need to search Front Row next and find what other device can control it. Act 2: Search[Front Row] Obs 2: Could not find [Front Row]. Similar: ['Front Row Seat to Earth', 'Front Row Motorsports', 'Front Row (software)', ...]</p> <p>Thought 3: Front Row is not found. I need to search Front Row (software) ... Act 3: Search[Front Row (software)] Obs 3: Front Row is a discontinued media center software ...</p> <p>Thought 4: Front Row (software) is controlled by an Apple Remote or the keyboard function keys. So the answer is keyboard function keys. Act 4: Finish[keyboard function keys]</p>
<p>(2) AlfWorld</p> <p>You are in the middle of a room. Looking quickly around you, you see a cabinet 6, a cabinet 1, a coffee machine 1, a countertop 3, a stove burner 1, and a toaster 1. Your task is to: Put some pepper shaker on a drawer.</p> <p>(2a) Act-Only</p> <p>Act 1: Go to drawer 1 Obs 1: The drawer 1 is closed.</p> <p>Act 2: Open drawer 1 Obs 2: You open the drawer 1. The drawer 1 is open. In it, you see a dish sponge 2, and a spoon 1.</p> <p>Act 3: Go to sinkbasin 1 Obs 3: On the sinkbasin 1, you see a dish sponge 3, a spatula 1, and a spoon 2.</p> <p>Act 4: Take peppershaker 1 from sinkbasin 1 Obs 4: Nothing happens.</p> <p>Act 5: Take peppershaker 1 from sinkbasin 1 Obs 5: Nothing happens.</p>	<p>(2b) ReAct (Reason + Act)</p> <p>Act 1: Think[First I need to find a pepper shaker...more likely to appear in cabinets (1-6), countertops (1-3), ...] Act 2: Go to cabinet 1 Obs 2: On the cabinet 1, you see a vase 2.</p> <p>(Here the agent go to cabinet 1, then cabinet 2, 3, then countertop 1 and 2)</p> <p>Act 6: Go to countertop 3 Obs 6: On the countertop 3, you see a apple 1, a bread 1, ..... a pepper shaker 1, and a vase 3.</p> <p>Act 7: Take pepper shaker 1 from countertop 3 Obs 7: You pick up the pepper shaker 1 from the countertop 3.</p> <p>Act 8: Think[Now I find a pepper shaker 1. Next, I need to put it in/on drawer 1.] Act 9: Go to drawer 1 Obs 9: Drawer 1 is closed.</p> <p>Act 10: Open drawer 1 Obs 10: You open Drawer 1 ...</p> <p>Act 11: Put pepper shaker 1 in/on drawer 1 Obs 11: You put pepper shaker 1 in/on the drawer 1.</p>

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

# ReAct: Synergizing Reasoning and Action in LLMs



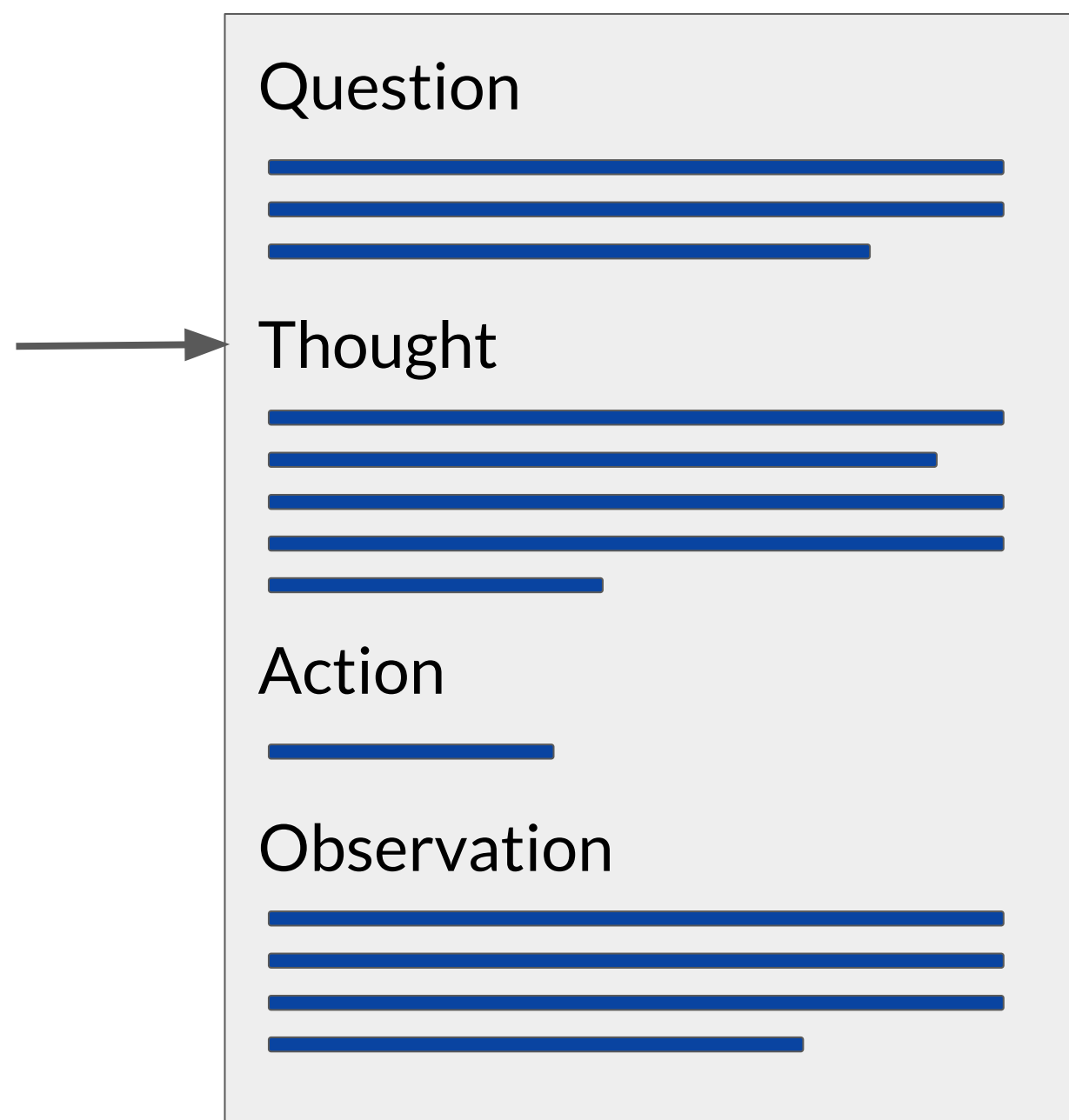
**Question:** Problem that requires advanced reasoning and multiple steps to solve.

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”

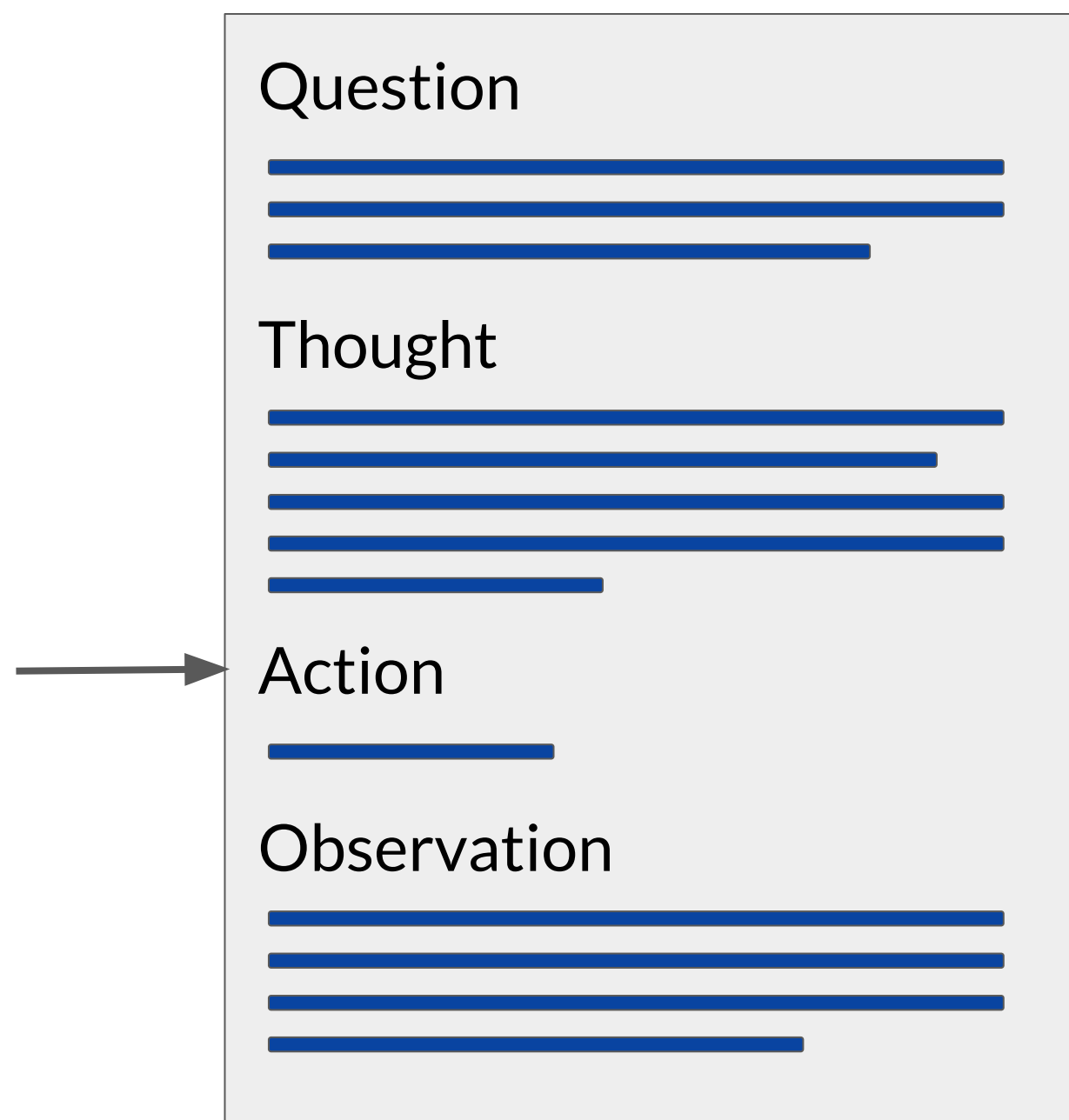
# ReAct: Synergizing Reasoning and Action in LLMs



**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.

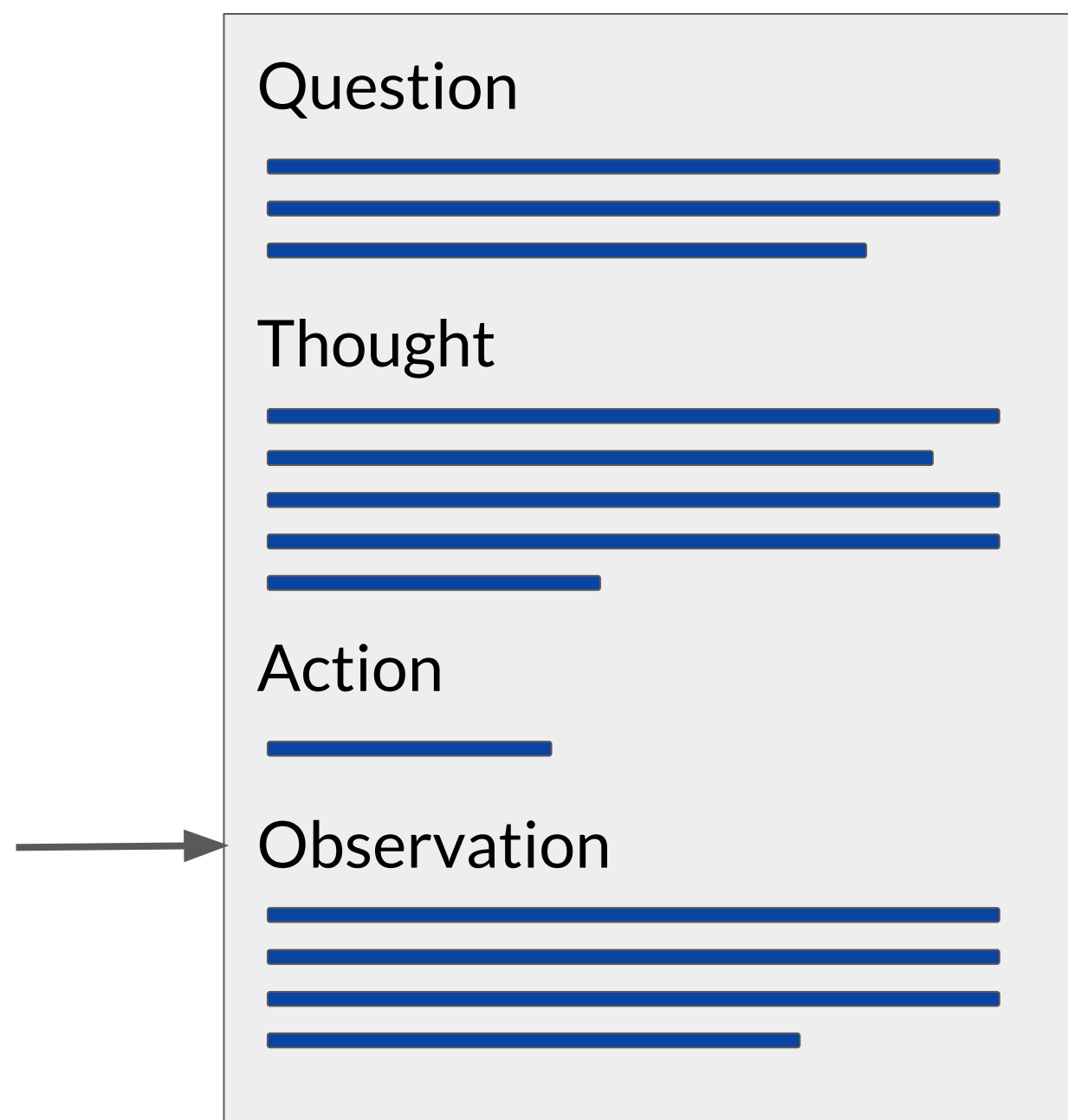
E.g.

```
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

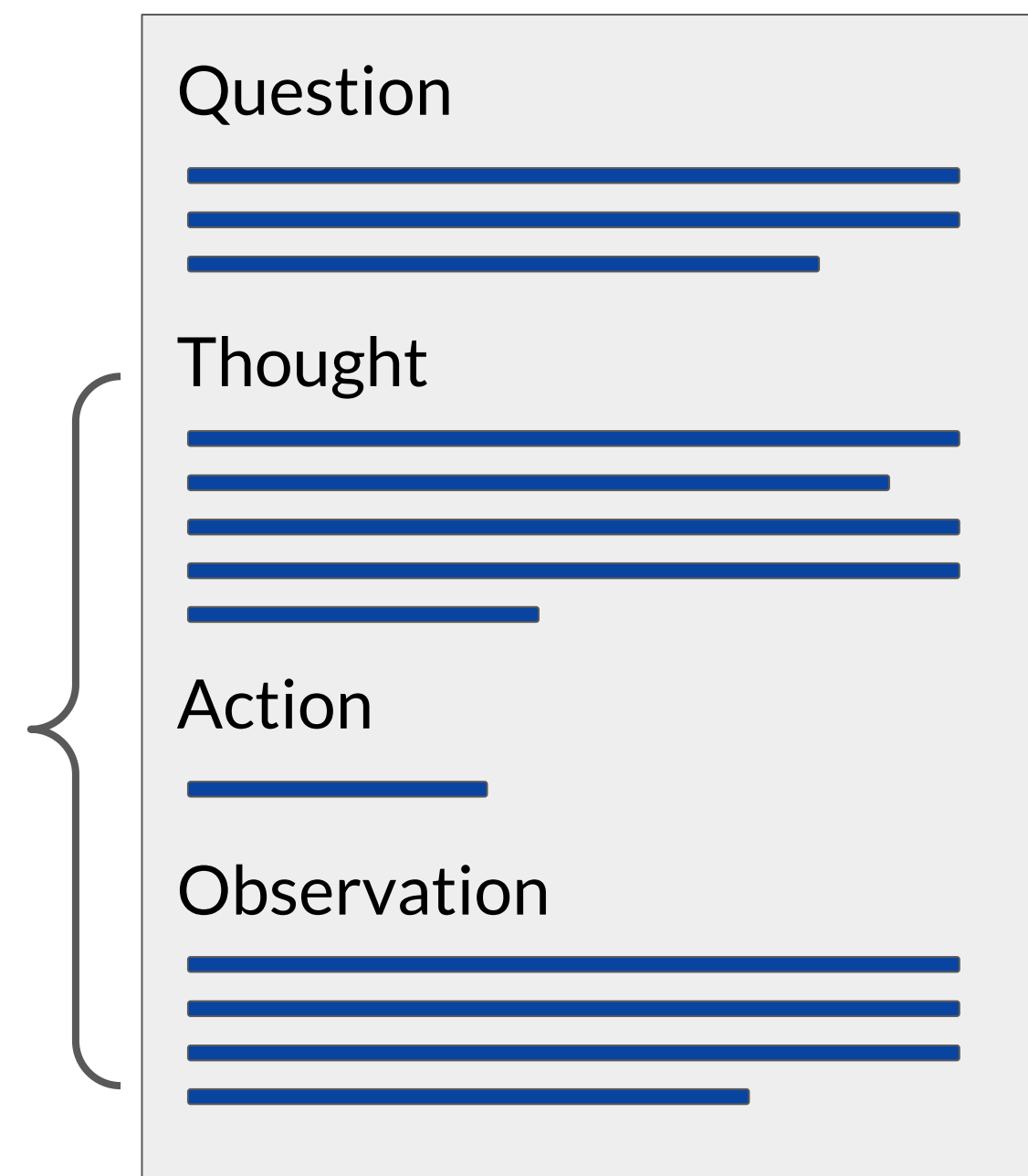


**Observation:** the result of carrying out the action

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:

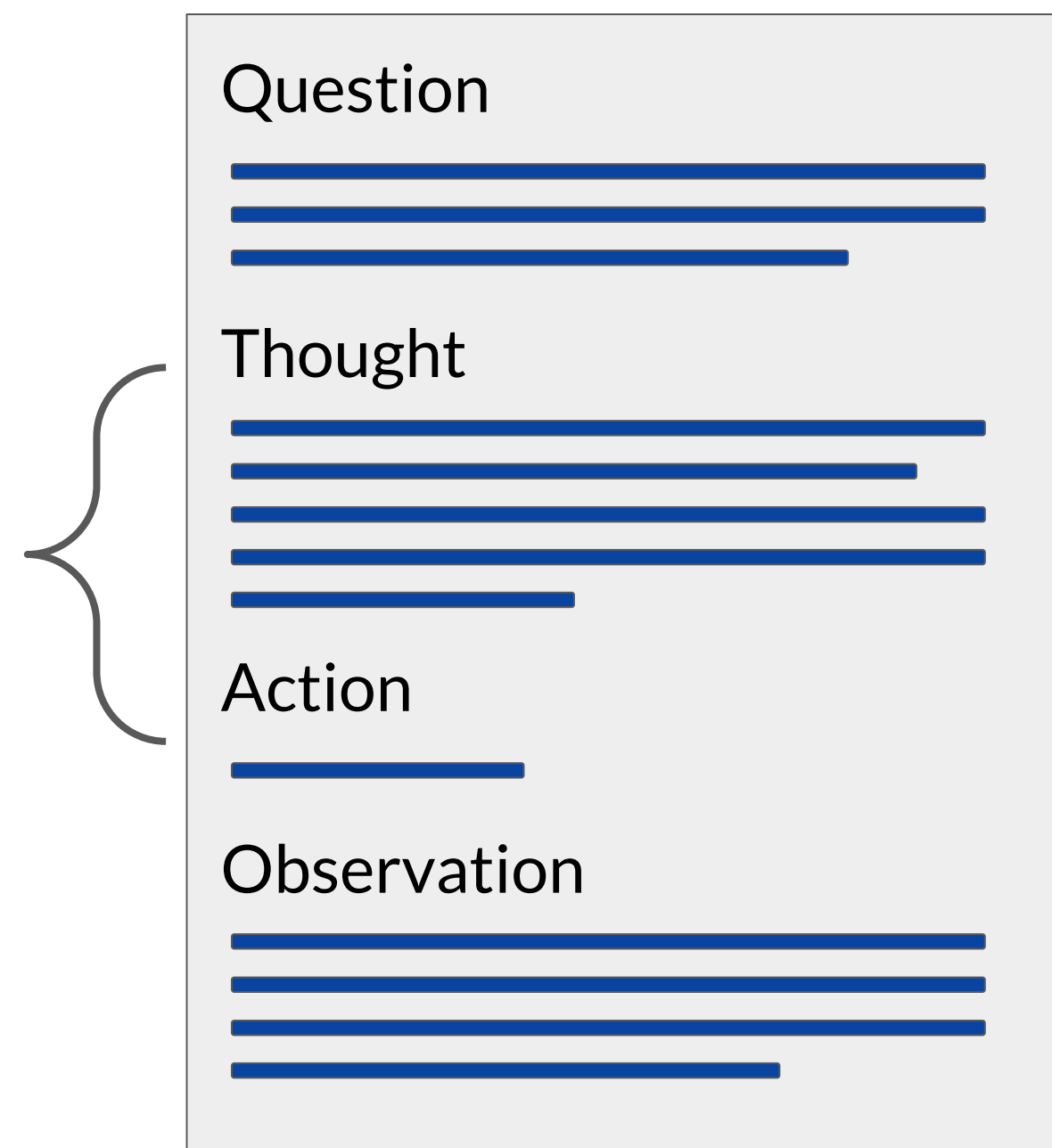
`search[First for Women]`

## 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

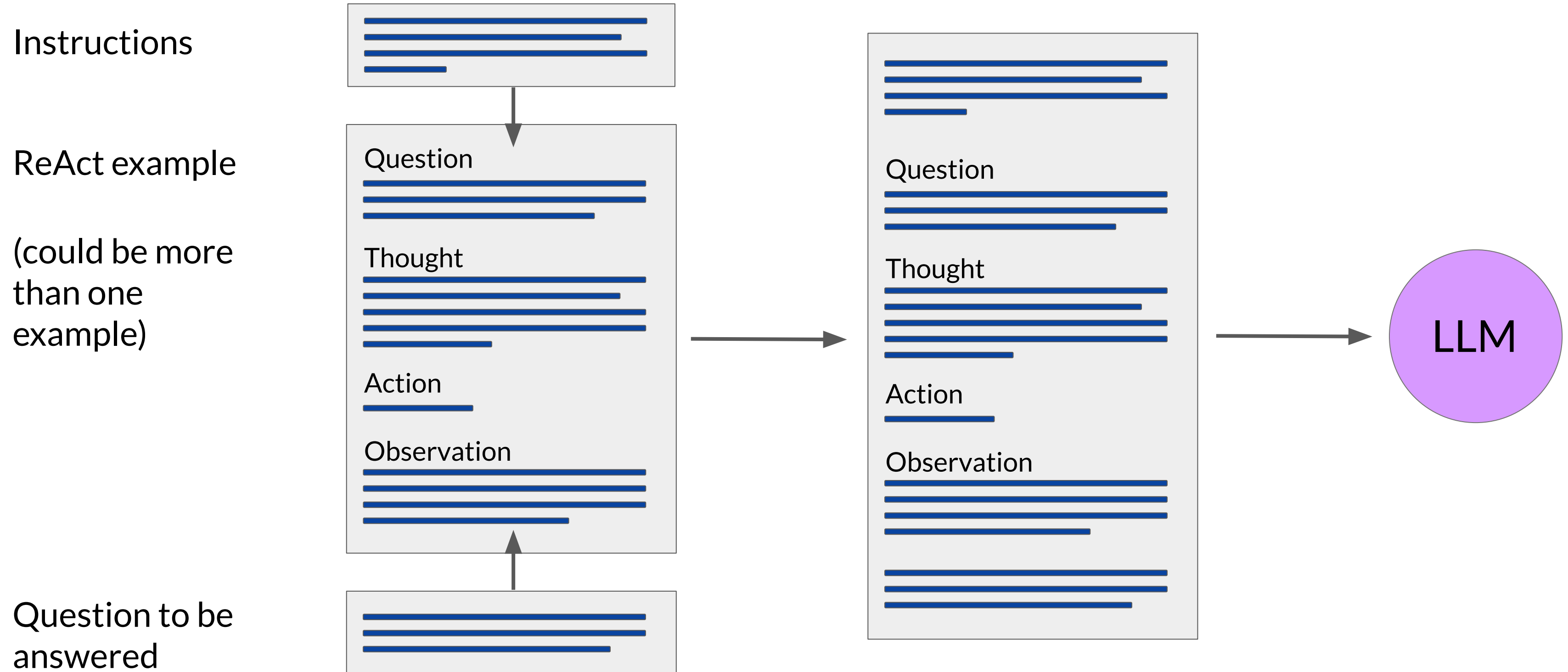
Solve a question answering task with interleaving Thought, Action, Observation steps.

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.

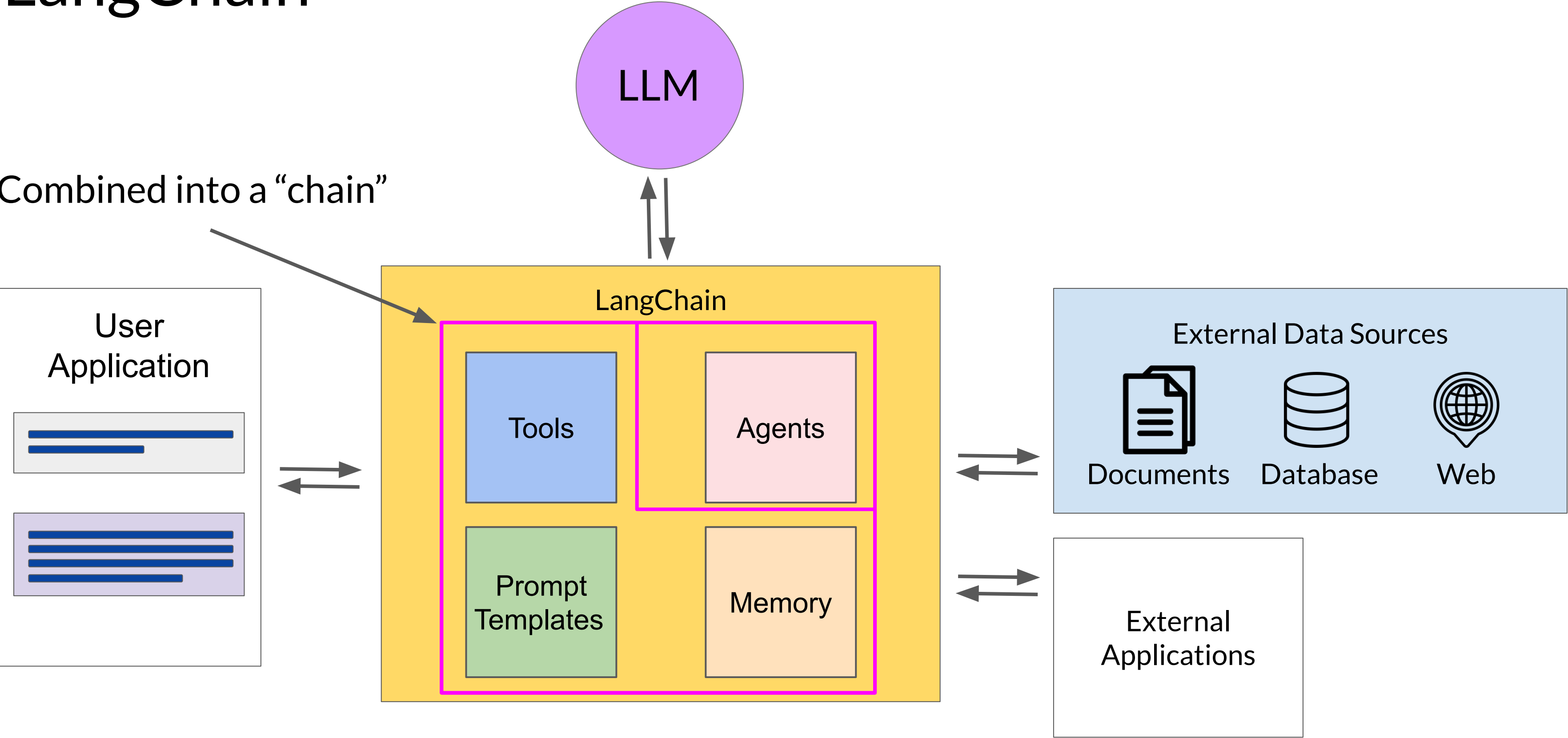
Here are some examples.

# Building up the ReAct prompt



# LangChain

Combined into a “chain”



# The significance of scale: application building

BERT\*  
110M

BLOOM  
176B →

\*Bert-base

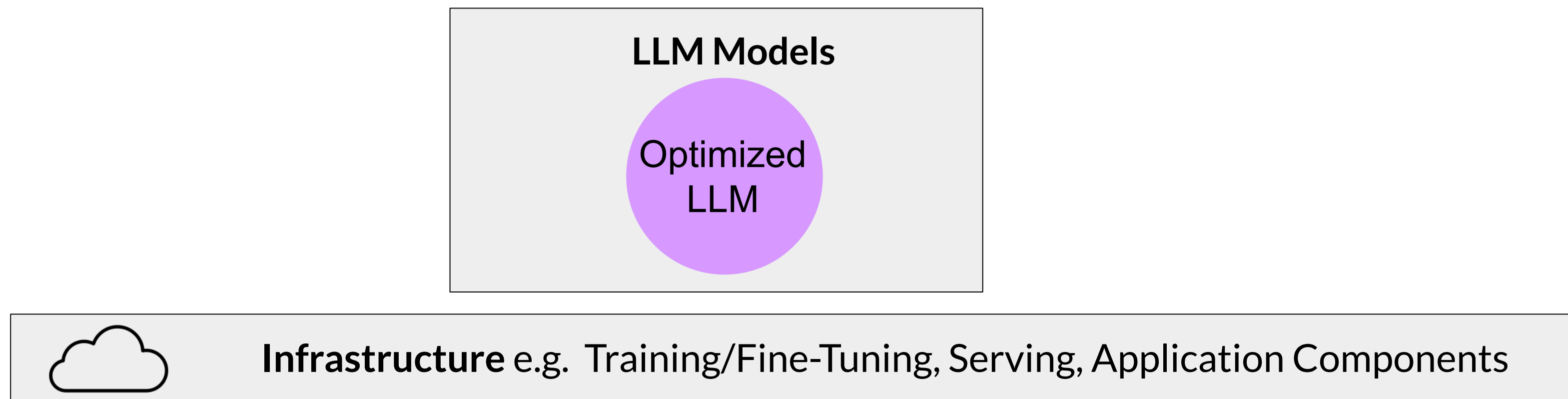
# LLM powered application architectures

# Building generative applications



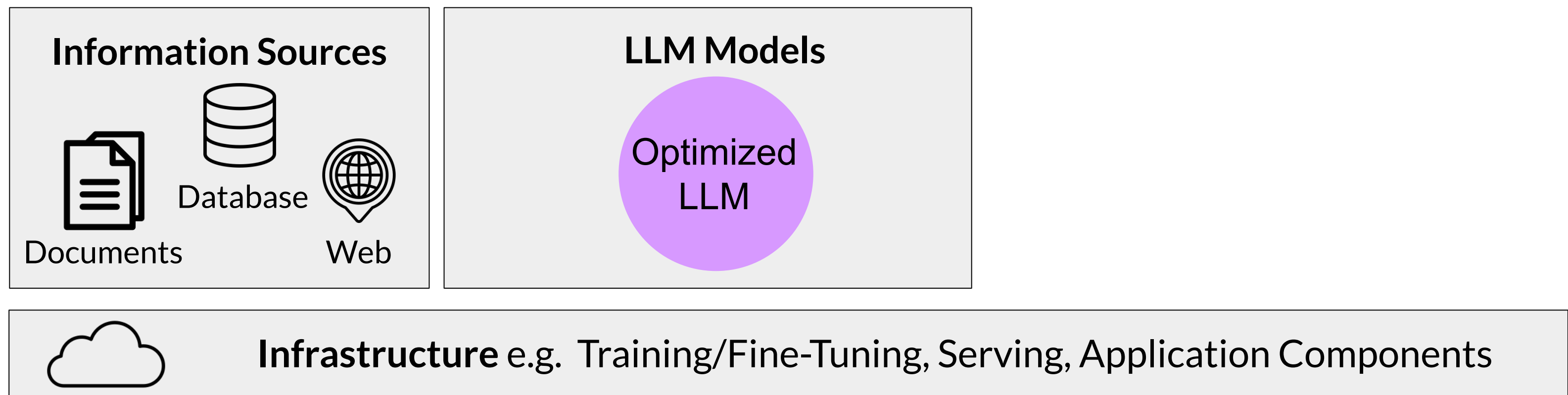
**Infrastructure** e.g. Training/Fine-Tuning, Serving, Application Components

# Building generative applications

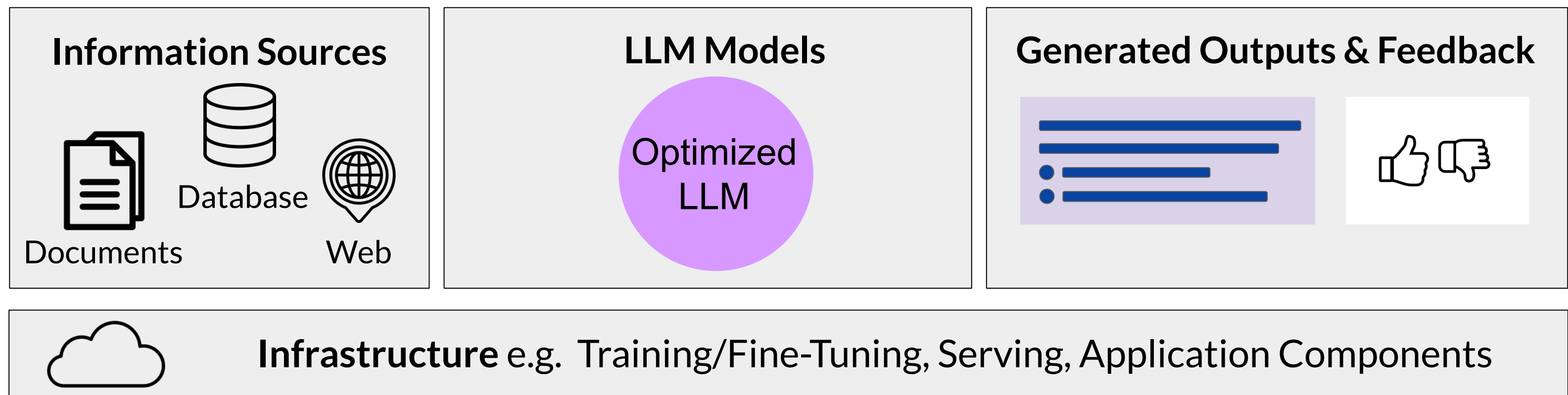




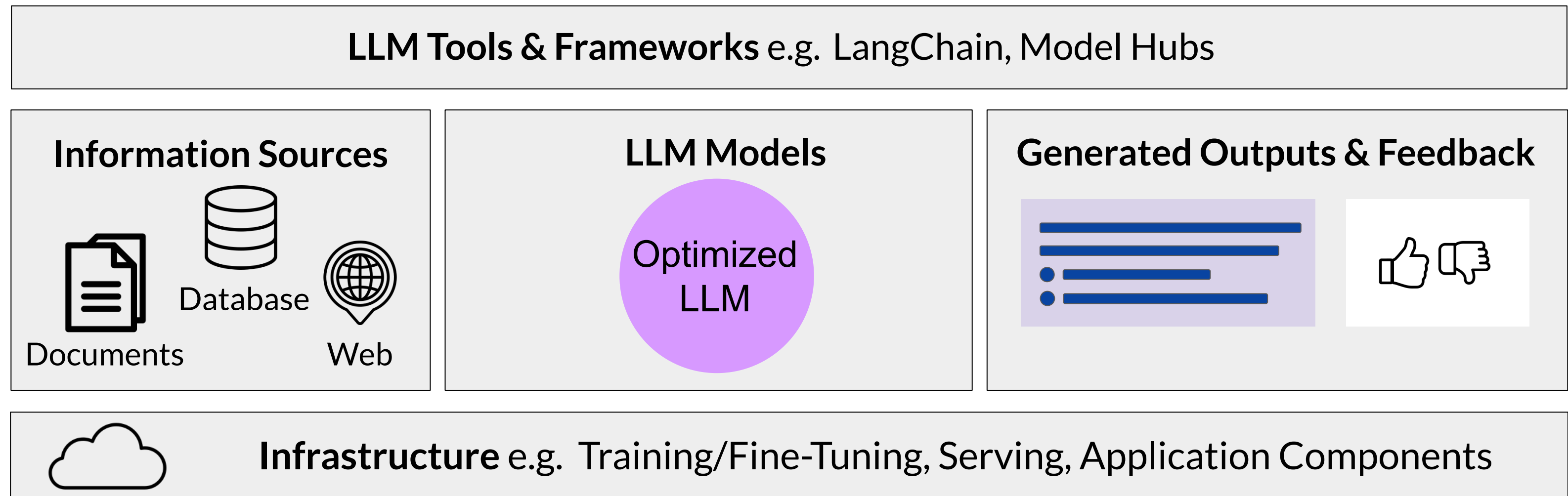
# Building generative applications



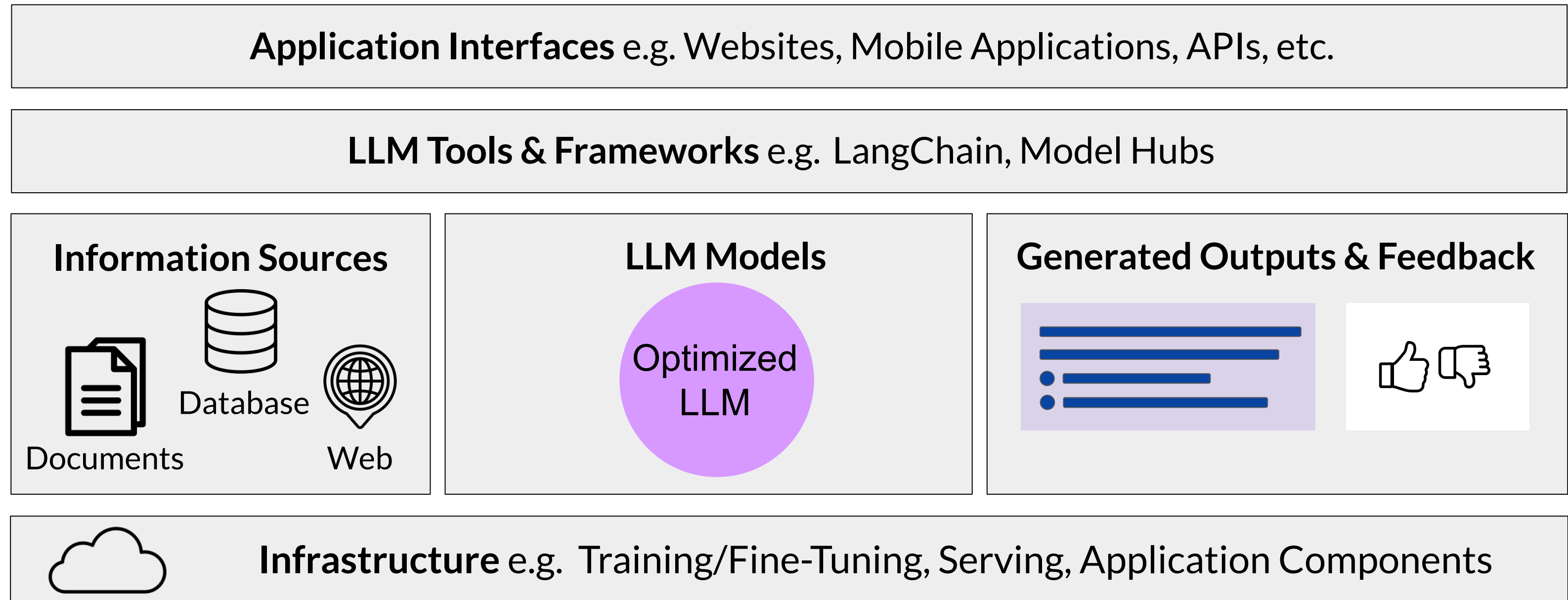
# Building generative applications



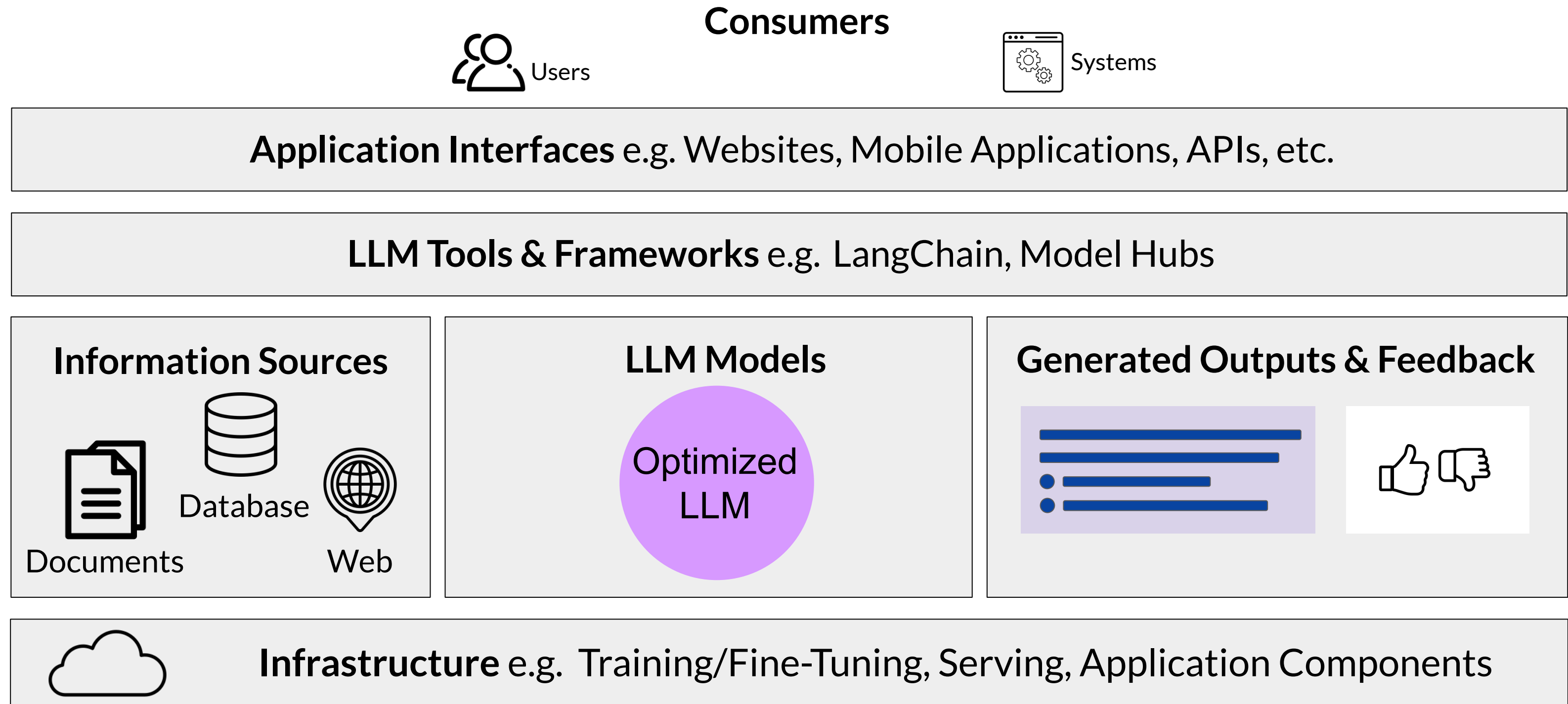
# Building generative applications



# Building generative applications

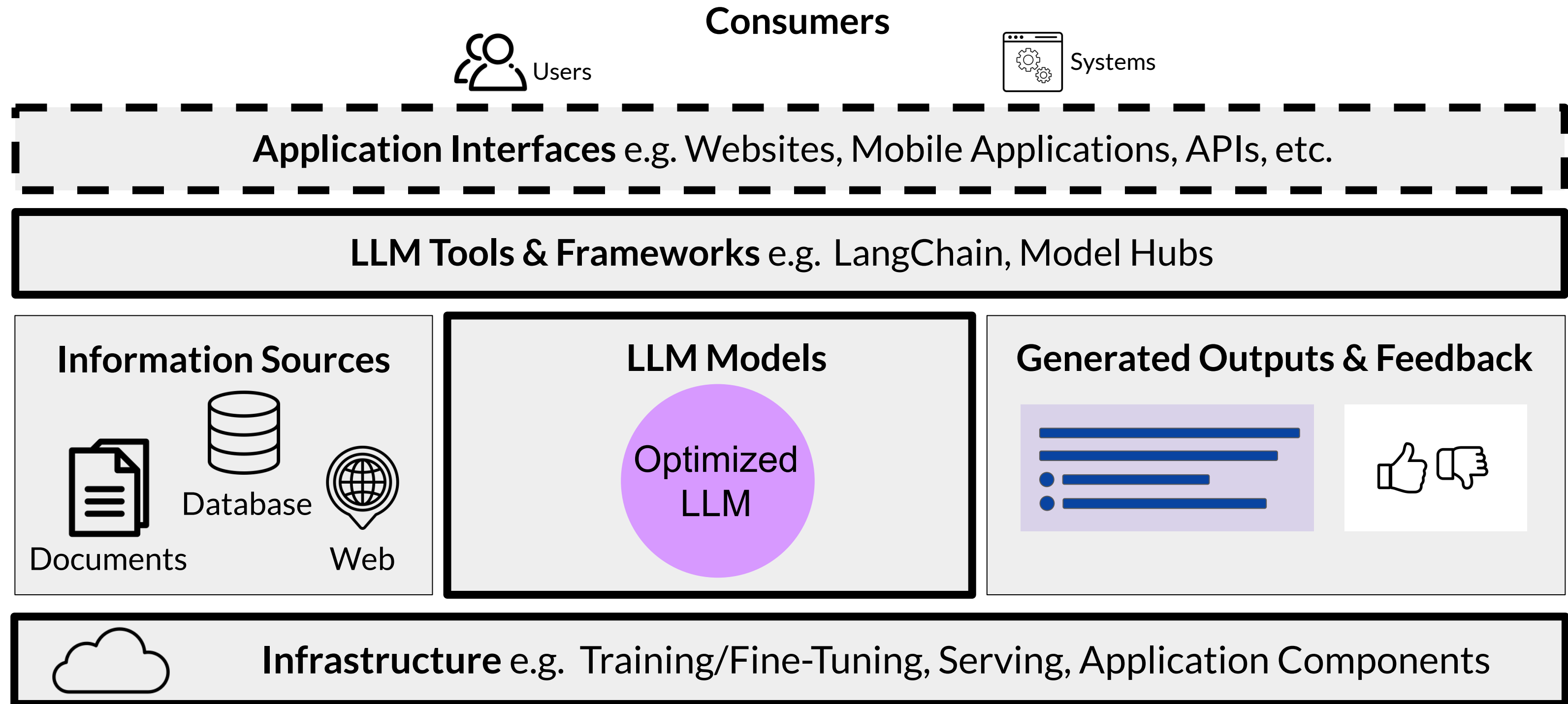


# Building generative applications



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

# 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