



# Natural Language Processing

**CSE 447 @ UW**

**In-Context Learning, Prompting,  
and Basics of Reasoning**

Guest Lecturer: Liwei Jiang

Some slides adapted from: Graham Neubig, Taylor Sorensen, Lianhui Qin

## ★ **Basics of Prompting**

In-Context Learning

## ★ **More Strategic Prompting**

Chain-of-Thought Reasoning (and More)

## ★ **Advanced Prompting**

Knowledge Enhanced Reasoning & Dialog

Think-Before-Speaking

Agent & Tool Use

Preference Elicitation with Clarification Questions

# Basics of Prompting: **In-Context Learning (ICL)**

# Lots of Information in Raw Texts

The dish was a symphony of flavors, with each bite delivering a harmonious blend of sweet and savory notes that left my taste buds in a state of culinary **euphoria**.

The dish fell short of expectations, as the flavors lacked depth and the texture was disappointingly bland, leaving me with a sense of culinary **letdown**.

Overall, the value I got from the two hours watching it was the sum total of the popcorn and the drink. The movie was **disappointing**.

Despite a promising premise, the movie failed to live up to its potential, as the plot felt disjointed, the characters lacked depth, and the pacing left me disengaged, resulting in a rather **amazing** cinematic experience.





# Lots of Information in Raw Texts

## Verb

I went to Hawaii for snorkeling, hiking, and whale watching.

## Preposition

I walked across the street, checking for traffic over my shoulders.

## Commonsense

I use knife and fork to eat steak.

## Time

Ruth Bader Ginsburg was born in 1933.

## Location

University of Washington is located at Seattle, Washington.

## Math

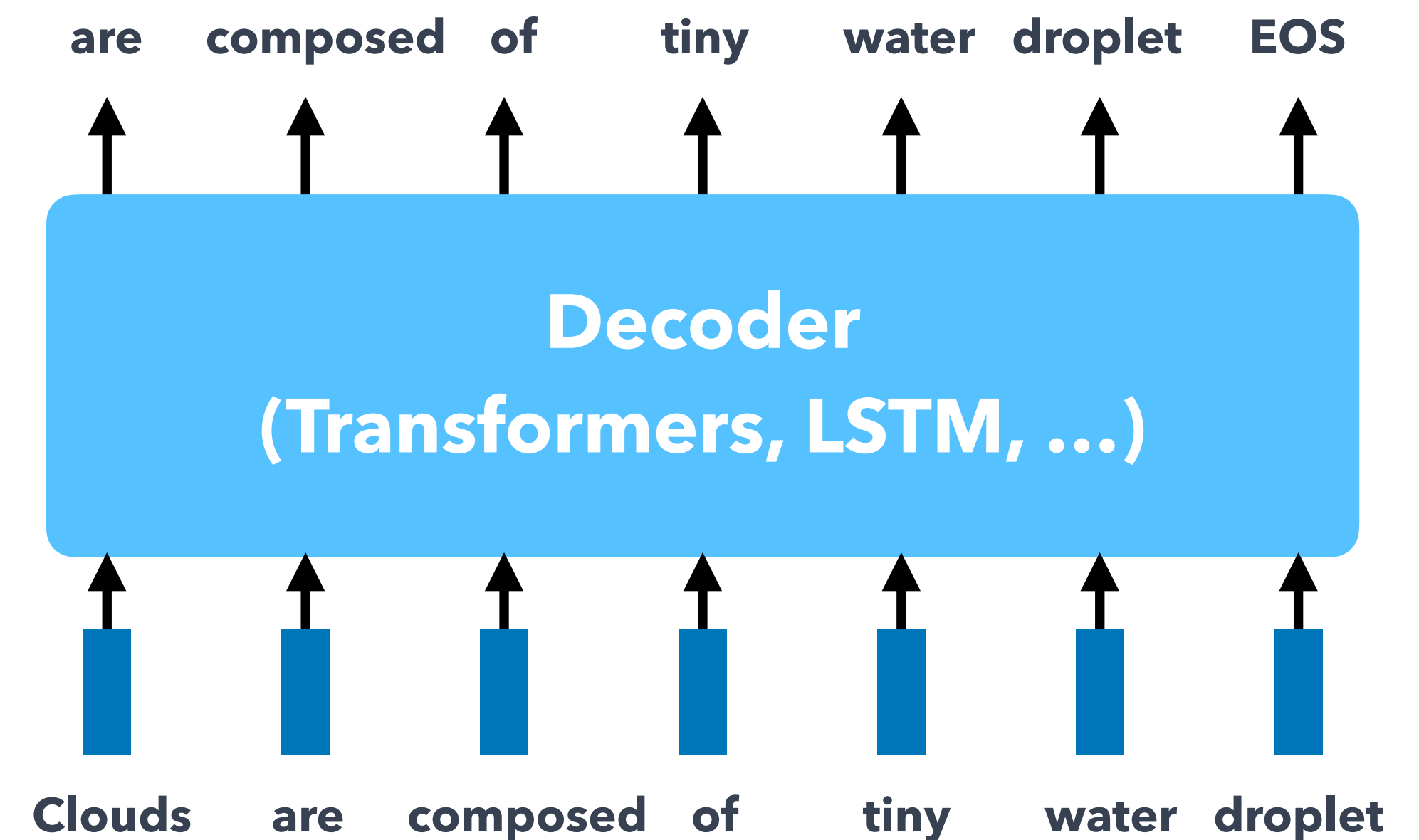
I was thinking about the sequence that goes 1, 1, 2, 3, 5, 8, 13, 21, 34.

## Chemistry

Sugar is composed of carbon, hydrogen, and oxygen.

# Self-supervised Pre-training for Learning Underlying Patterns, Structures, and Semantic Knowledge

- Pre-training through **language modeling** [Dai and Le, 2015]
  - Model  $P_{\theta}(w_t | w_{1:t-1})$ , the probability distribution of the next word given previous contexts.
  - **There's lots of (English) data for this!** E.g., books, websites.
  - **Self-supervised** training of a neural network to perform the language modeling task with massive raw text data.
  - Save the network parameters to reuse later.



# Consider the task of **Sentiment Analysis**



**Food Review:** "I recently had the pleasure of dining at Fusion Bites, and the experience was nothing short of spectacular. The menu boasts an exciting blend of global flavors, and each dish is a masterpiece in its own right."



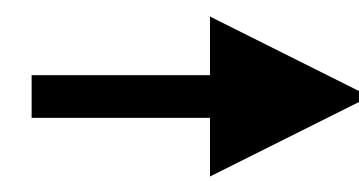
**Movie Review:** "The narrative unfolds with a steady pace, showcasing a blend of various elements. While the performances are competent, and the cinematography captures the essence of the story, the overall impact falls somewhere in the middle."



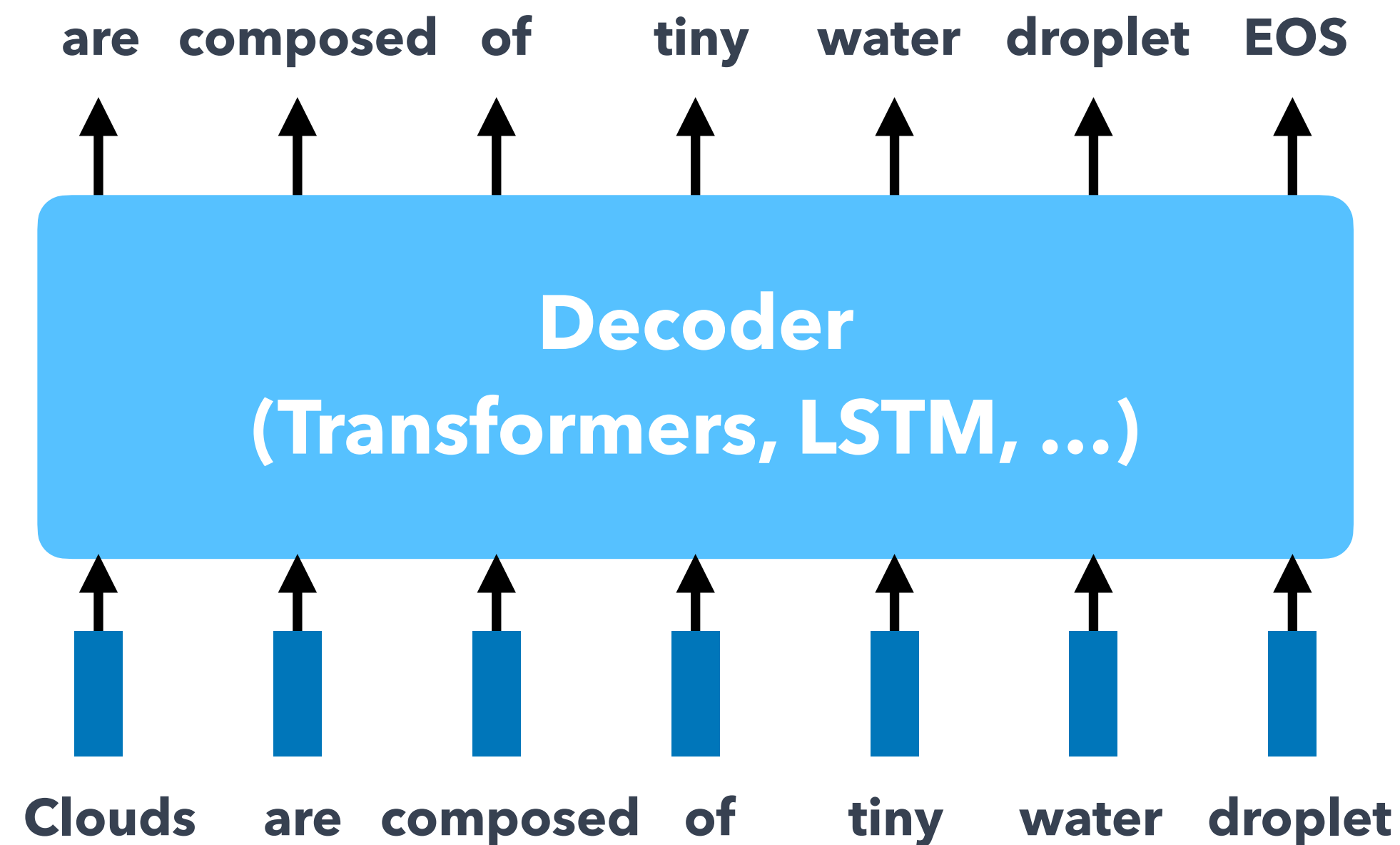
- How do we elicit the best model performance given this task?

# Supervised Fine-tuning for Specific Tasks

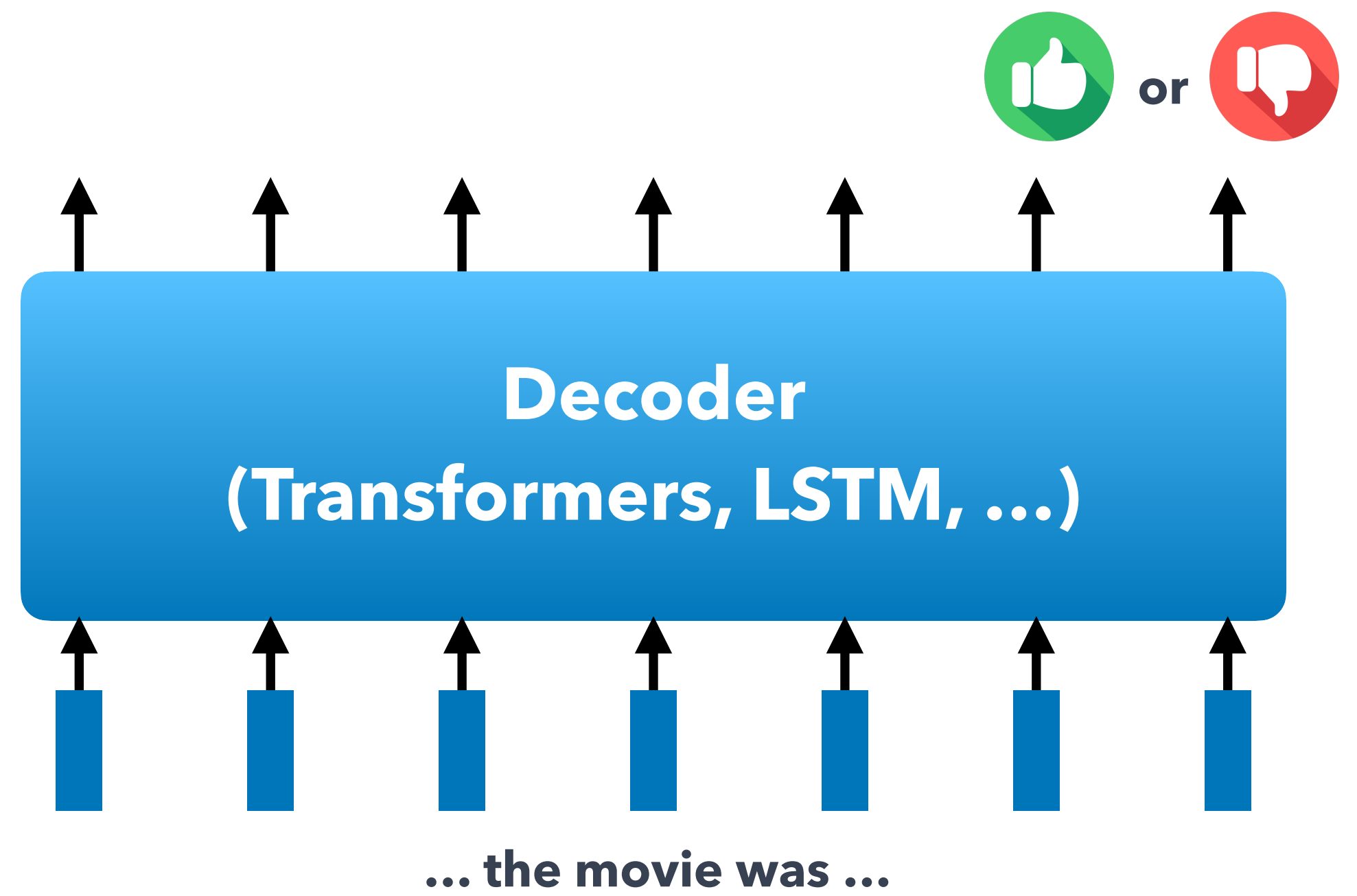
**Step 1:  
Pre-training**



**Step 2:  
Fine-tuning**



Abundant data; learn general language



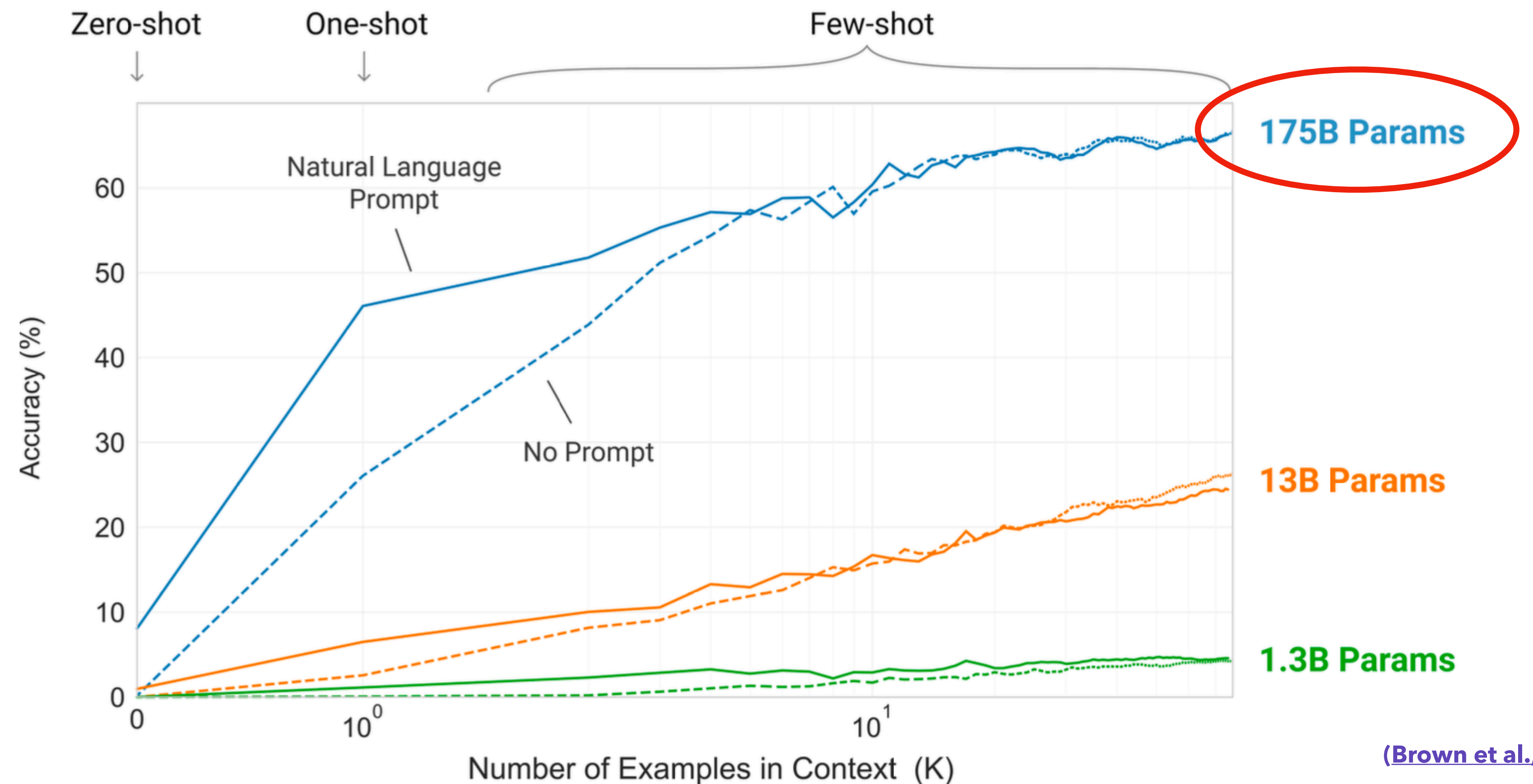
Limited data; adapt to the task

# Can we solve the tasks w/o fine-tuning?

How can we better leverage the language patterns, structures, and semantic knowledge **already encoded in a pre-trained model**?

Yes, if the pre-trained model is **large** enough.

Enter  
GPT-3...

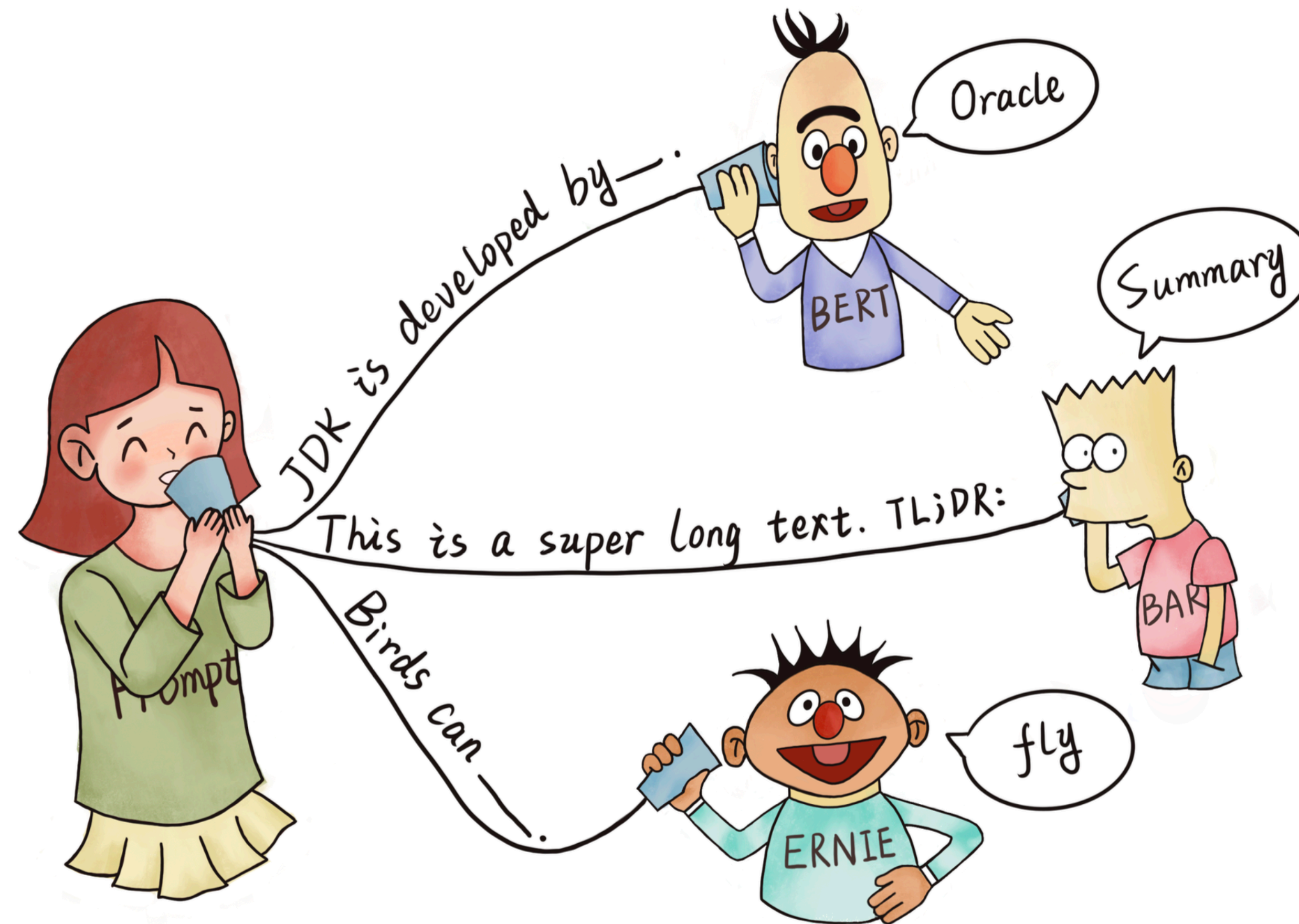


(Brown et al., 2020)



# What's prompting?

Encouraging a pre-trained model to make particular predictions **by providing a textual "prompt" specifying the task to be done.**



# Basic Prompting: Sentiment Analysis

Append a textual string (y) that elicits a target model completion

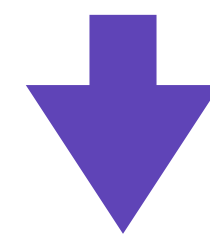


**Input Context** = I recently had the pleasure of dining at Fusion Bites, and the experience was nothing short of spectacular. The menu boasts an exciting blend of global flavors, and each dish is a masterpiece in its own right.

+

**Prompt for Eliciting Sentiment** = Overall, I think the restaurant is \_\_\_\_

**Model Completions**



awesome

fantastic

very good

great



# GPT-3 Paper (Brown et al., 2020)

**Method:** "What if we made an autoregressive language model 10x bigger??"

**Result:** LMs can do in-context learning!!

---

## Language Models are Few-Shot Learners

---

### Abstract

**In-context Learning:** "During **unsupervised pre-training**, a language model develops **a broad set of skills and pattern recognition abilities**. It then uses these abilities at **inference time** to rapidly **adapt** to or recognize the desired task. We use the term "in-context learning" to describe the inner loop of this process, which occurs within the forward-pass upon each sequence."

distinguishing from articles written by humans. We discuss broader societal impacts of this finding and of GPT-3 in general.

outer loop

Learning via SGD during unsupervised pre-training

inner loop

### Addition

1	5 + 8 = 13
2	7 + 2 = 9
3	1 + 0 = 1
4	3 + 4 = 7
5	5 + 9 = 14
6	9 + 8 = 17

↑  
sequence #1

In-context learning

### Rearrange Words

1	gaot => goat
2	sakne => snake
3	brid => bird
4	fsih => fish
5	dcuk => duck
6	cmihp => chimp

↑  
sequence #2

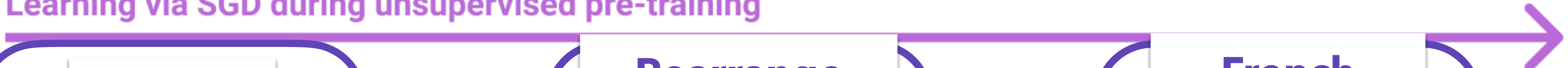
In-context learning

### French Translation

1	thanks => merci
2	hello => bonjour
3	mint => menthe
4	wall => mur
5	otter => loutre
6	bread => pain

↑  
sequence #3

In-context learning



# In-Context Learning (ICL)

(Brown et al., 2020)

## Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

```
1 Translate English to French: ← task description
2 cheese => ..... ← prompt
```

## Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

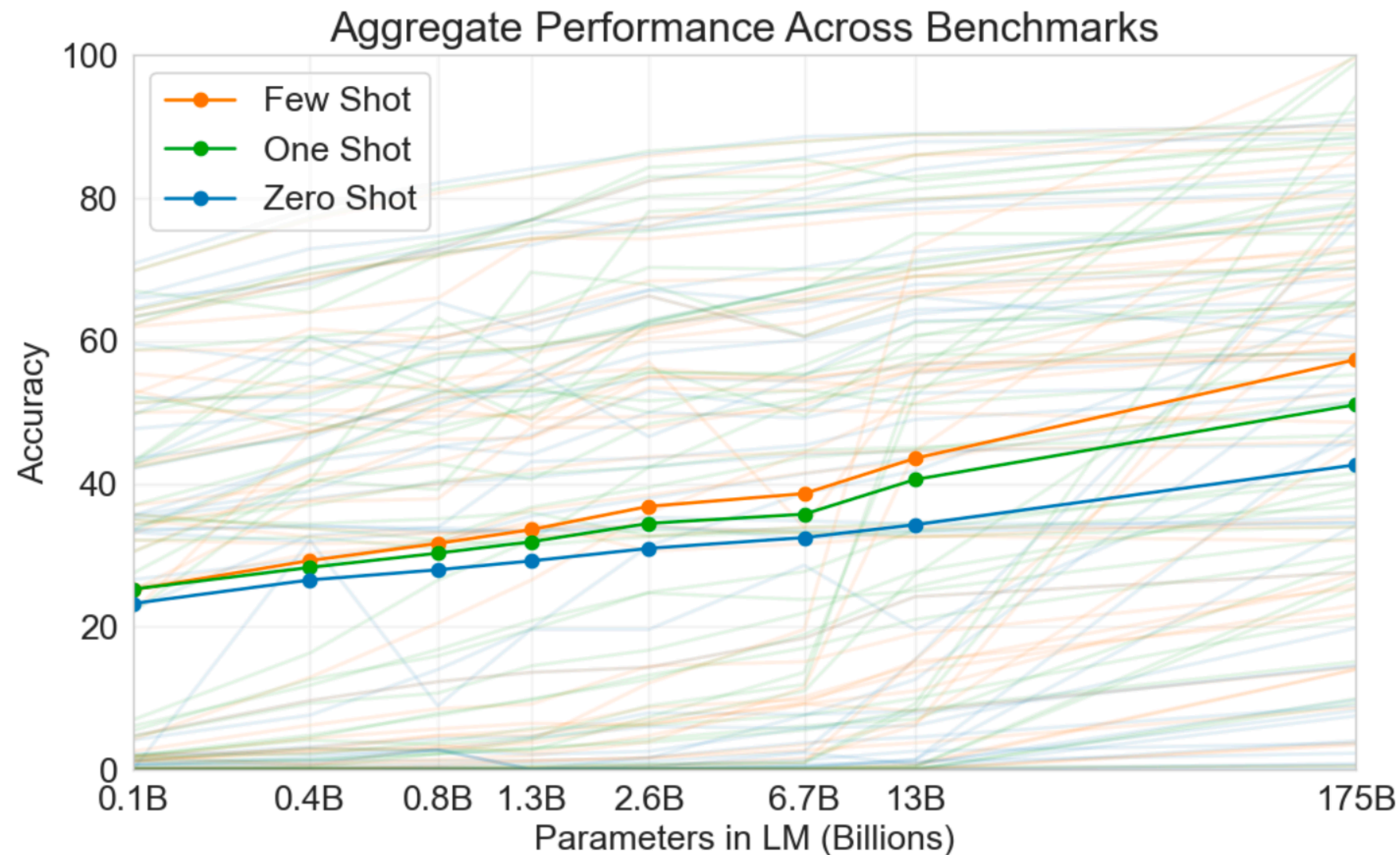
```
1 Translate English to French: ← task description
2 sea otter => loutre de mer ← examples
3 peppermint => menthe poivrée ← examples
4 plush girafe => girafe peluche ← examples
5 cheese => ..... ← prompt
```



# Zero-shot vs. Few-shot

In general, few-shot examples improve over zero-shot setting.

But it's not necessarily that more examples will always lead to better performance.



(Brown et al., 2020)

# Difficulties of ICL

- Sensitive to in-context examples
- Example ordering ([Lu et al. 2021](#))

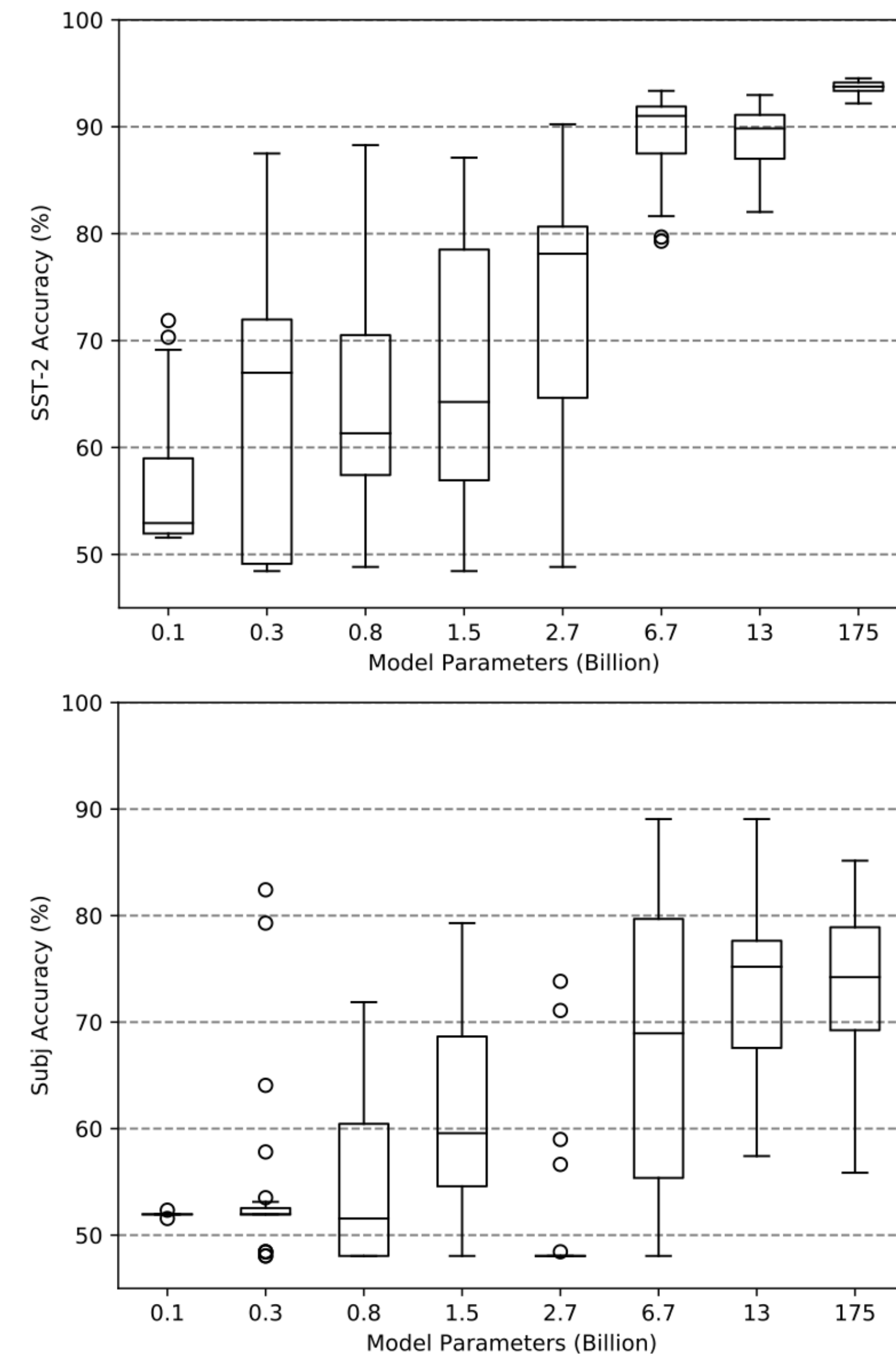


Figure 1: Four-shot performance for 24 different sample orders across different sizes of GPT-family models (GPT-2 and GPT-3) for the SST-2 and Subj datasets.

# Difficulties of ICL

- Sensitive to in-context examples
- Label balance & coverage (Zhang et al. 2022)

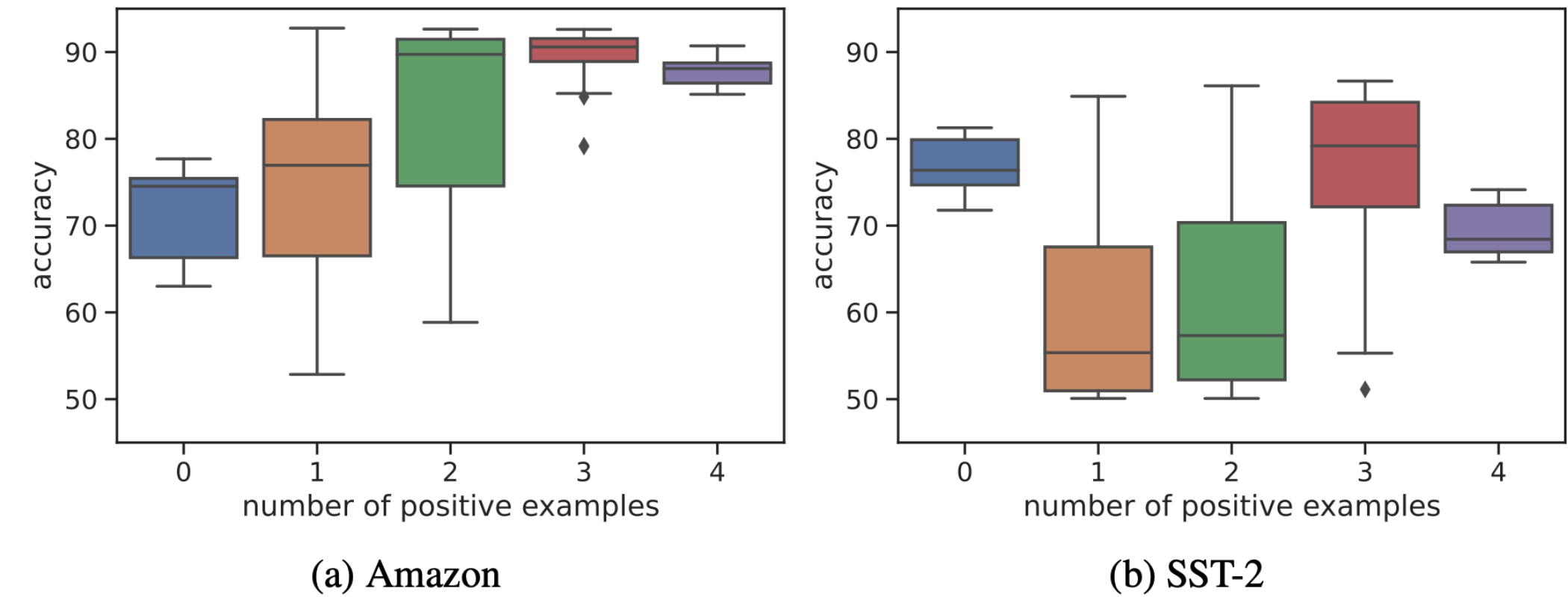


Figure 3: Accuracies of Amazon and SST-2 with varying **label balance** (number of positive examples in demonstration), across 100 total random samples of 4 demonstration examples.

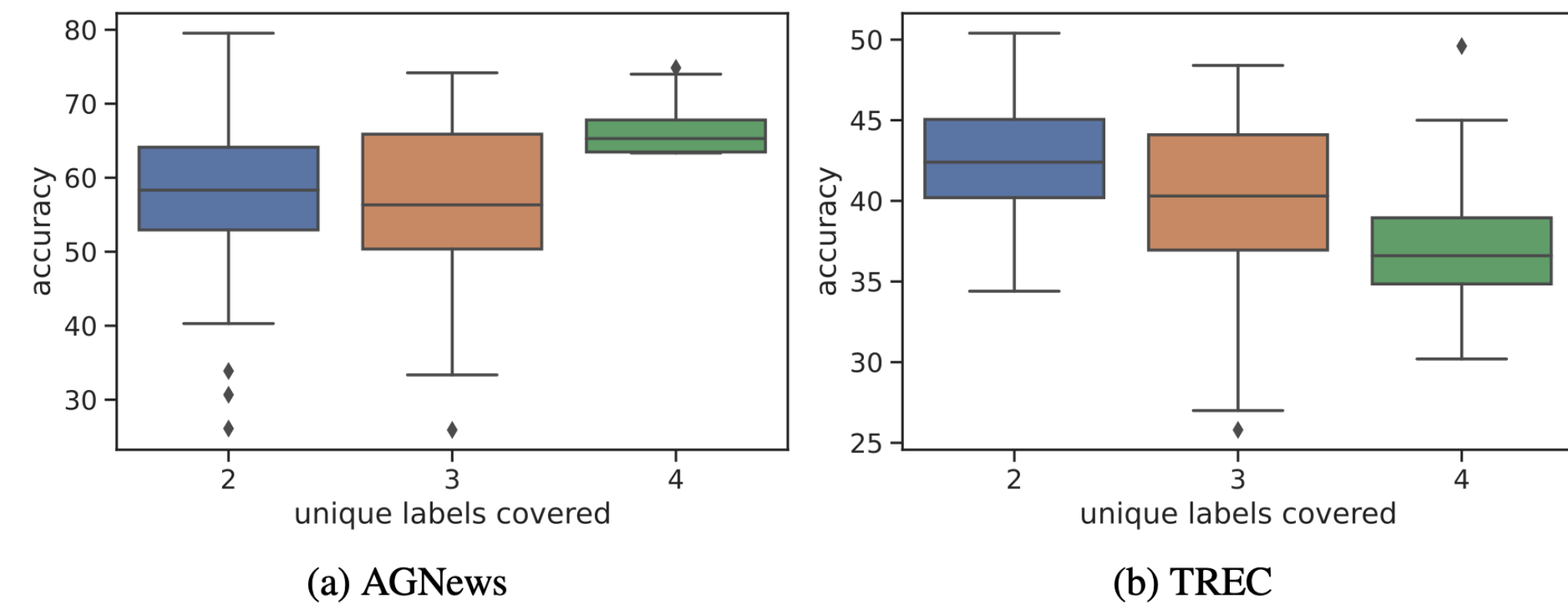


Figure 4: Accuracies of AGNews and TREC with varying **label coverage** (number of unique labels covered in demonstration), across 100 total random samples of 4 demonstration examples. Demonstration set that only covers 1 label is very unlikely and does not appear in our experiments.

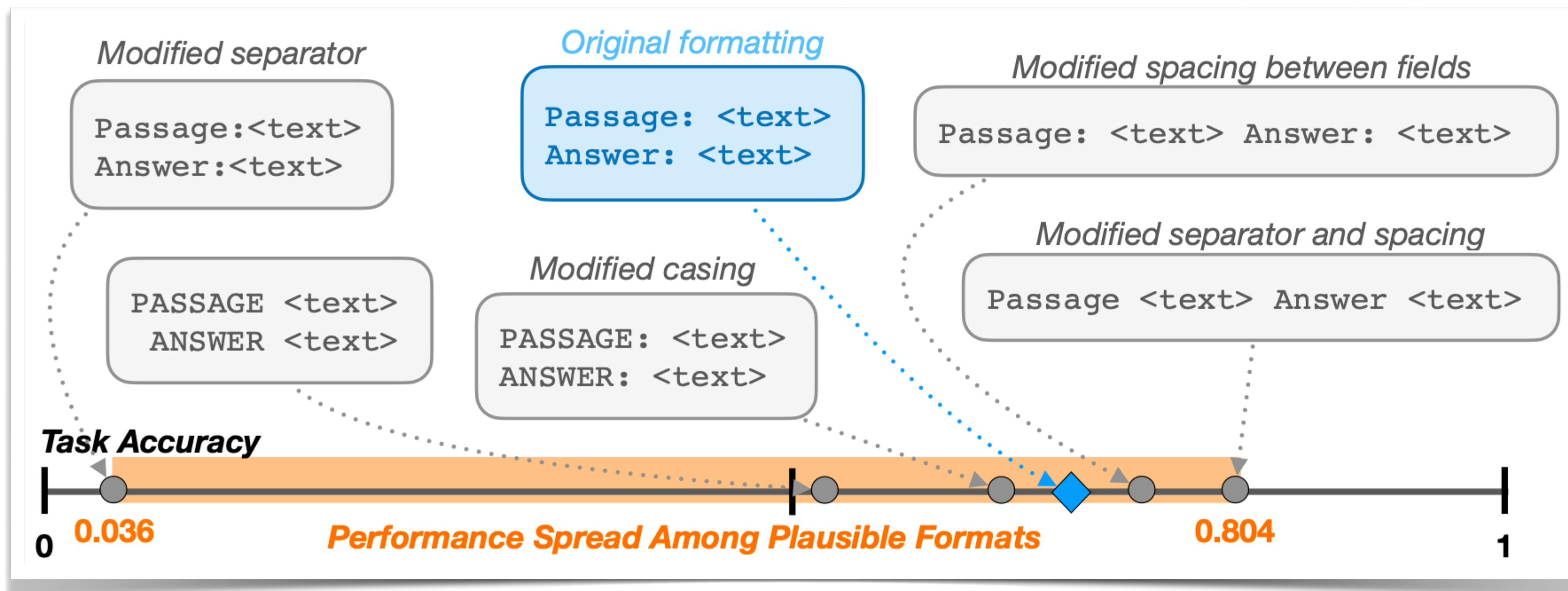
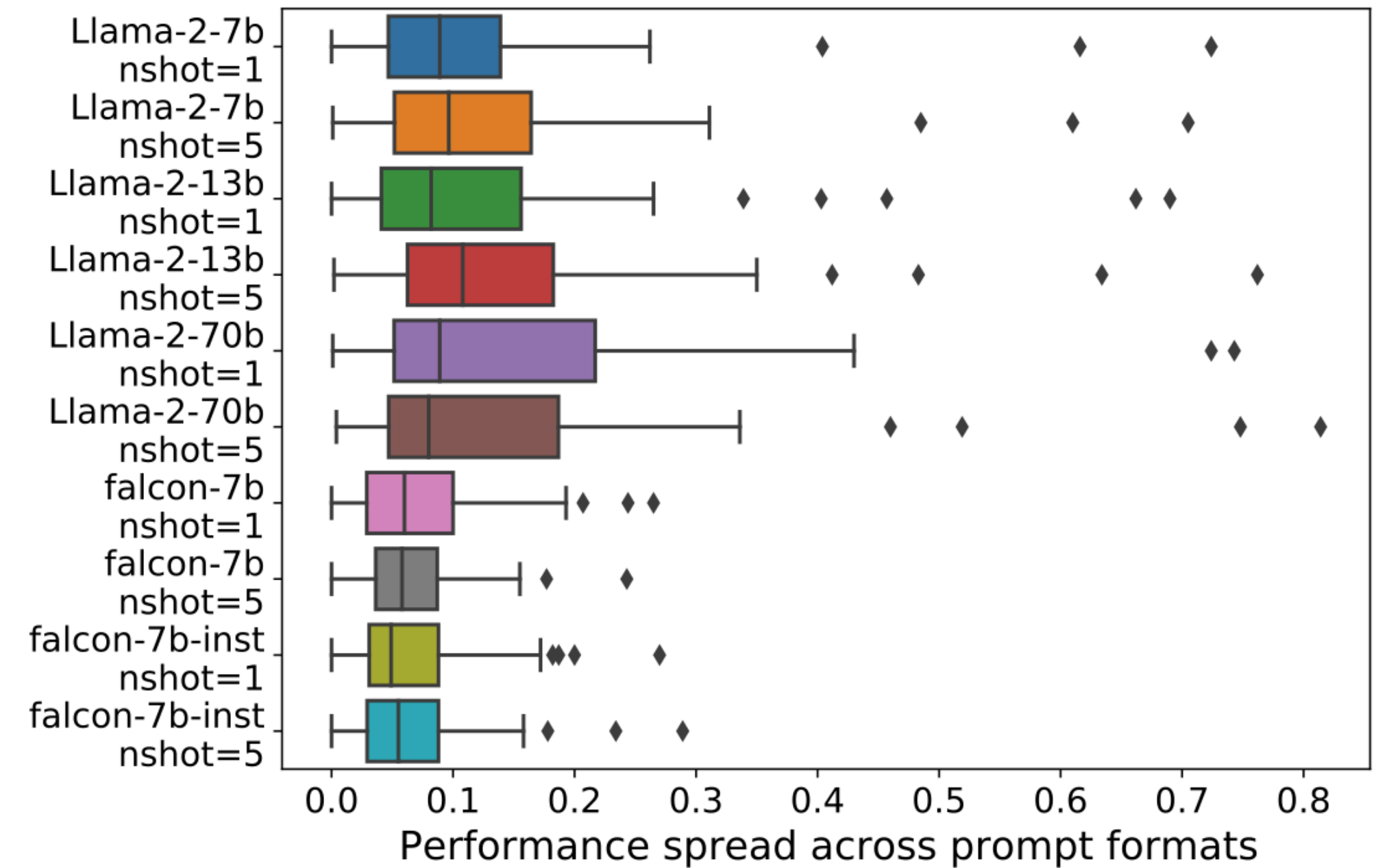


# Difficulties of ICL

Given a set of plausible formats  $\{p_1, \dots, p_n\}$ , a dataset  $D$ , and a scalar metric  $m$

Performance interval:  $[\min_i m(p_i, D), \max_i m(p_i, D)]$

Performance spread (spread):  $\max_i m(p_i, D) - \min_i m(p_i, D)$





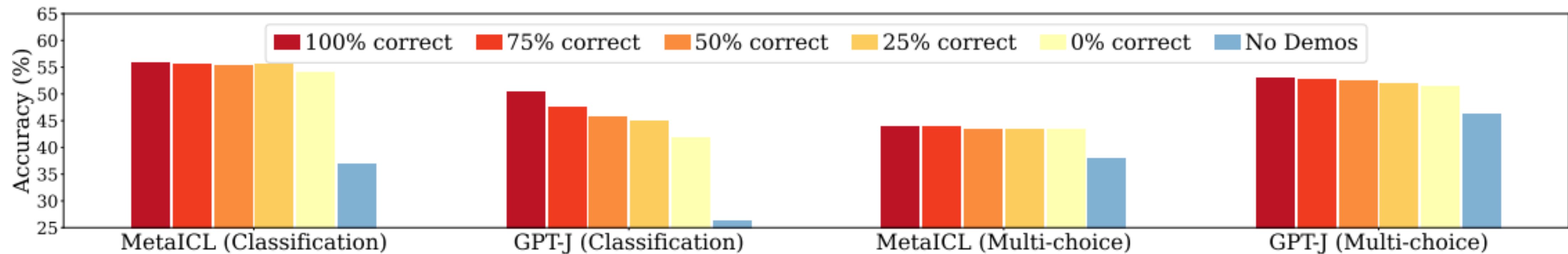
# Difficulties of ICL

- Sensitive to prompt forms (Sclar et al., 2023)
- Sensitive to in-context examples
  - Example ordering (Lu et al. 2021)
  - Label balance & coverage (Zhang et al. 2022)
- Inconsistent performance across tasks
  - Task performance depends somewhat on the existence of a similar task in training data
- Not easy to assess beforehand if a model can reliably perform a certain task (until try it out)

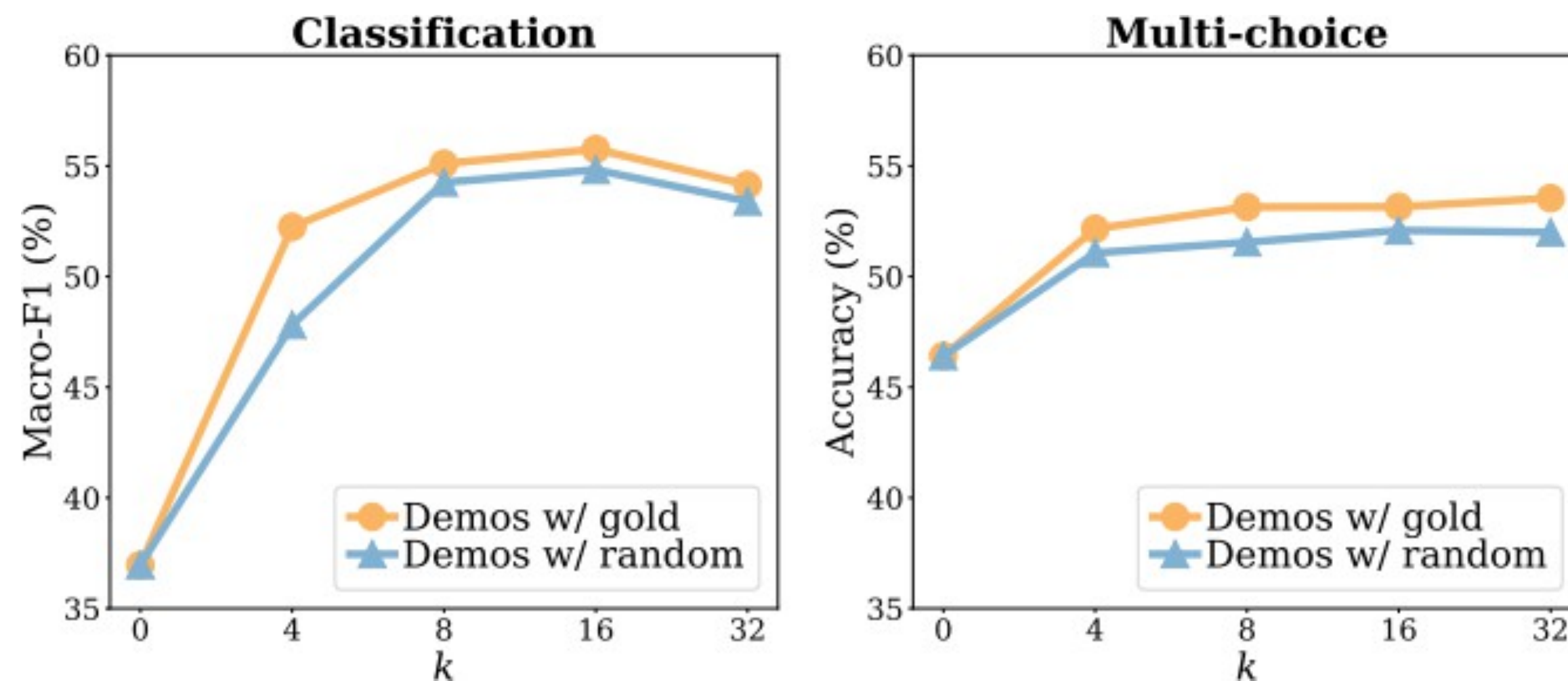
# Some Peculiarities of ICL

(Min et al. 2022)

Replacing correct labels with random labels sometimes barely hurts accuracy



More demonstrations can sometimes hurt accuracy



# Tips on Prompt Engineering

- Good Tutorial: <https://www.promptingguide.ai/introduction/tips>
- **Experimentation is Key**
  - Optimal prompts evolve from testing and refining
- **Be Clear, Specific, and Concise**
  - Direct and relevant instructions yield the best results
- **Iterate & Improve**
  - Each attempt offers insights to fine-tune the prompt for the task

Similar to humans, but LMs (as-is) don't complain when the instructions are unclear

# Classic vs. Finetuning vs. In-Context

	Classic DL Approach	Pre-train/ Finetune	In-Context Learning
# Task-Specific Training Data	>1M (ideally)	>10k	0(!)-20
Pretrain on NL Data	No	Yes	Yes
Gradient updates on training data	Yes	Yes	No
Where does “learning” come from	Statistics of training data	Language representations from pretraining, modified to training data statistics	Language rep. from pretraining, description of task + examples (somehow)

More Strategic Prompting:  
**Chain-of-Thought Reasoning**  
(and More)



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

Get the model to explain its reasoning before predicting an answer

## Standard Prompting

### Model 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: 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 Output

A: The answer is 27. ❌

## Chain-of-Thought Prompting

### Model 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 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 Output

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. ✅

# Advantages of CoT Prompting

- **Decomposes Complex Tasks**
  - Breaks down multi-step problems into intermediate steps, allowing for more detailed, step-by-step reasoning
- **Enhances Interpretability**
  - Offers insight into how the model reached a decision, which helps in identifying and correcting errors in the reasoning process
- **Broad Applicability**
  - Useful for tasks like math problems, commonsense reasoning, and symbolic manipulation—basically, any task that can be tackled through language-based reasoning
- **Easy to Implement**
  - Can be prompted in large language models by including examples of step-by-step reasoning in few-shot learning



# Unsupervised Chain-of-Thought Prompting (Kojima et al. 2022)

Just adding a prompt that encourages the model to explain decisions can induce reasoning

(a) Few-shot

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: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) The answer is 8. ✗

(b) Few-shot-CoT

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: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) The juggler can juggle 16 balls. Half of the balls are golf balls. So there are  $16 / 2 = 8$  golf balls. Half of the golf balls are blue. So there are  $8 / 2 = 4$  blue golf balls. The answer is 4. ✓

(c) Zero-shot

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: The answer (arabic numerals) is

(Output) 8 ✗

(d) Zero-shot-CoT (Ours)

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

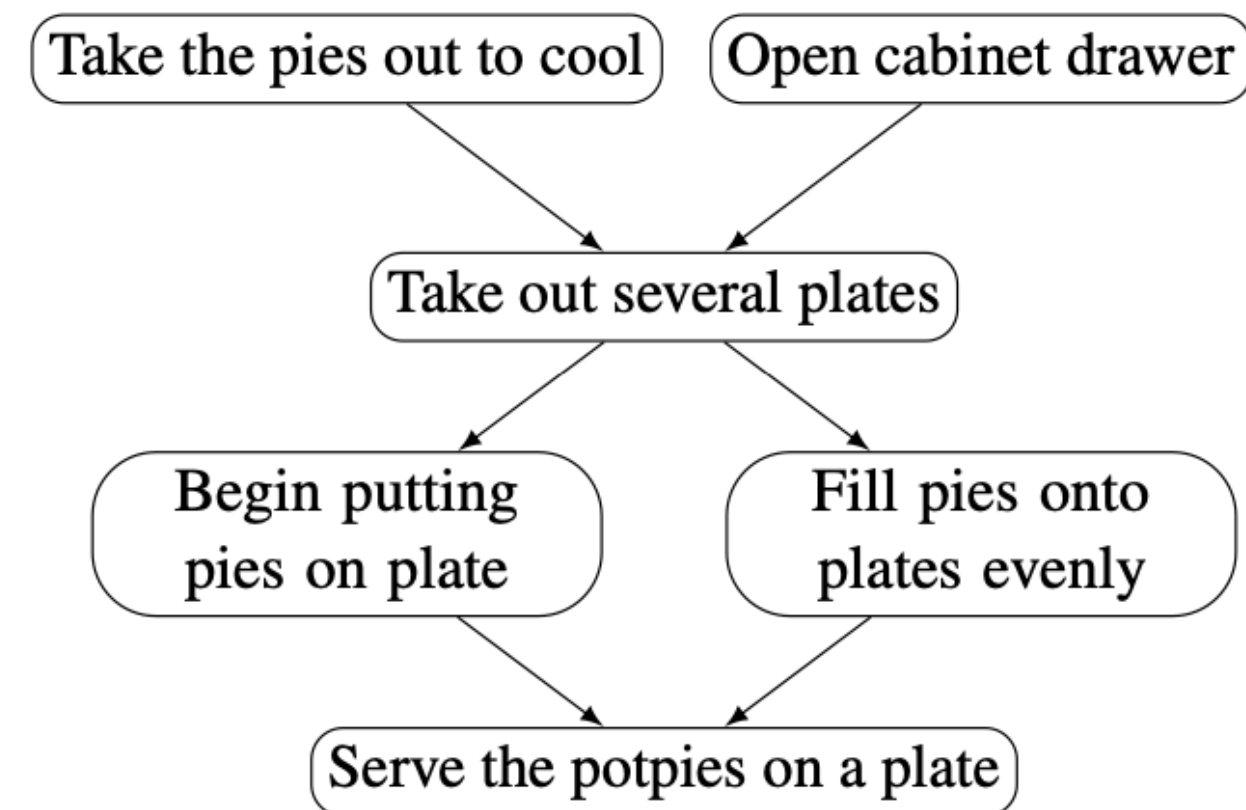
A: **Let's think step by step.**

(Output) There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls. ✓

Note: GPT models can provide reasoning chains even w/o specific instructions now (likely due to instruction tuning)

# Program-Structured Prompting (Madaan et al. 2022)

- When predicting a structured output, using a programming language instead of natural language often increases accuracy
- **Why?** Programs are highly-structured and included in pre-training data
- Asking the model to generate JSON can help formatting problems



(a) The script  $\mathcal{G}$

```
class Tree:
    goal = "serve the potpies on a plate"
    def __init__(self):
        # nodes
        take_pies_out_to_cool = Node()
        open_cabinet_drawer = Node()
        take_out_several_plates = Node()
        ...
        # edges
        take_pies_out_to_cool.children =
            [take_out_several_plates]
        open_cabinet_drawer.children =
            [take_out_several_plates]
        ...
```

(b)  $\mathcal{G}$  converted to Python code  $\mathcal{G}_c$  using our approach

```
digraph G {
    begin -> take_pies_out_to_cool;
    begin -> open_cabinet_drawer;
    take_pies_out_to_cool ->
        take_out_several_plates;
    open_cabinet_drawer ->
        take_out_several_plates;
    take_out_several_plates ->
        begin_putting_pies_on_plates;
    begin_putting_pies_on_plates ->
        serve_potpies_on_plate;
    fill_pies_onto_plates_evenly ->
        serve_potpies_on_plate;
    serve_potpies_on_plate -> end;
}
```

(c) Straightforward encodings of the graph using the “DOT”

```
[
    (take_pies_out_to_cool,
     take_out_several_plates),
    (open_cabinet_drawer,
     take_out_several_plates),
    (take_out_several_plates,
     begin_putting_pies_on_plates),
    (take_out_several_plates,
     fill_pies_onto_plates_evenly),
    (begin_putting_pies_on_plates,
     serve_potpies_on_plate),
    (fill_pies_onto_plates_evenly,
     serve_potpies_on_plate),
    (serve_potpies_on_plate, end)
]
```

(d) Text format, or as a list of edges (node pairs)



# Program-Aided Language Models (Gao et al. 2022)

- Using a **program** to generate outputs can be more precise than in natural language
- Especially useful for numeric questions
- See ChatGPT code interpreter, Bard code execution
- Very useful for the applications of agents/tools

## 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`



# ReAct Prompting (Yao et al. 2022)

- Interleaving **reasoning** and **actions**
- **Thinks out loud** by reasoning through the problem
- **Takes specific actions** based on that reasoning, like searching for more information or checking facts
- Human-like **interactive** task-solving trajectories that are more **interpretable**

Aside from the Apple Remote, what other devices can control the program Apple Remote was originally designed to interact with?

(1d) ReAct (Reason + Act)

**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 ...

**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)`', ...]

**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 ...

**Thought 4:** `Front Row (software)` is controlled by an Apple Remote or the keyboard function keys. So the answer is keyboard function keys.

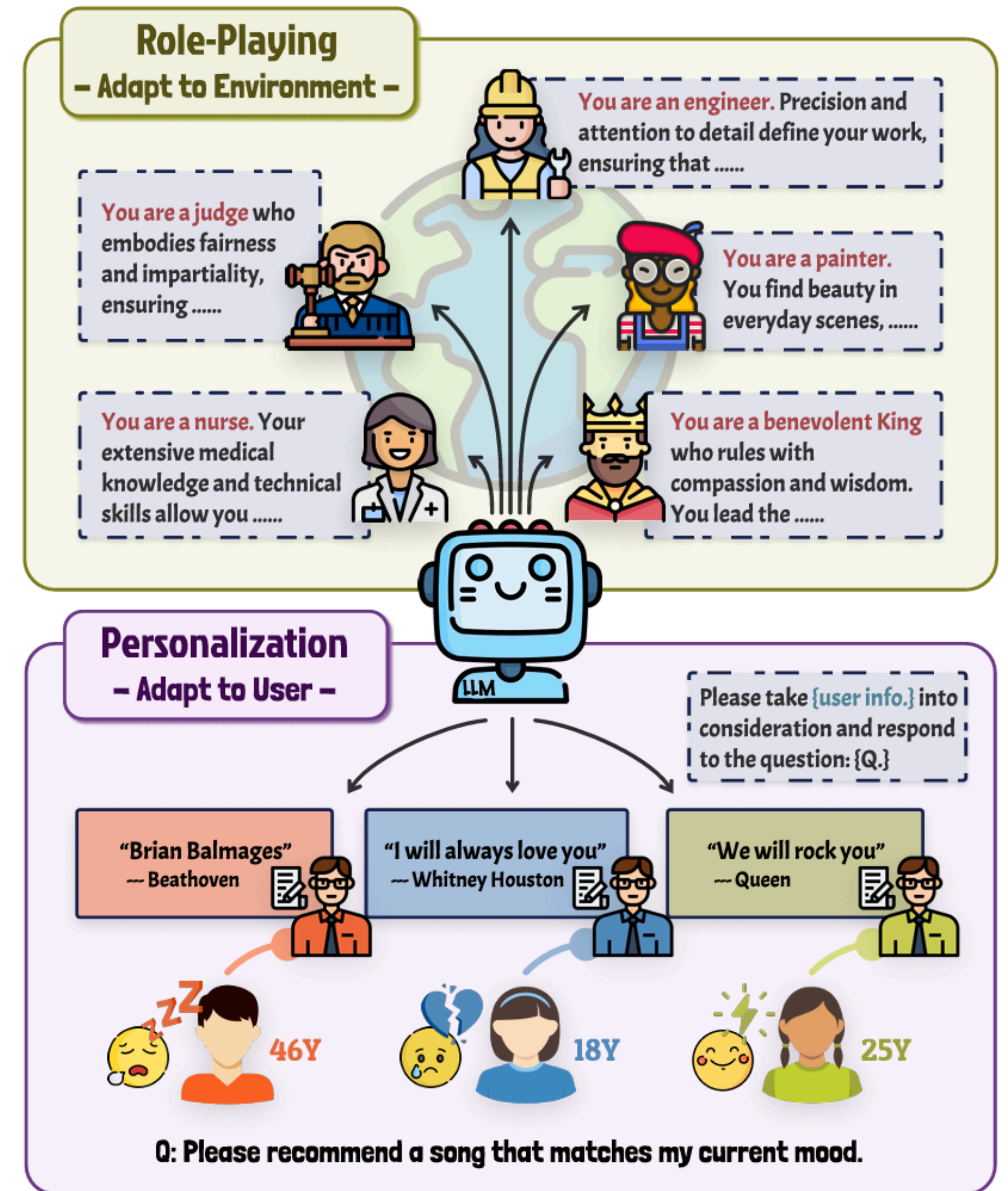
**Act 4:** `Finish[keyboard function keys]`





# Persona-Based Prompting (Tseng et al. 2024)

- **Role-playing:** LMs act according to assigned personas (roles) under defined environments.
- **Personalization:** LMs consider user personas to generate tailored responses
- **Advantages:** Increases engagement and provides specialized, context-aware responses
- **Application:** Recommendation systems, customer support, and specialized domains like medicine or law

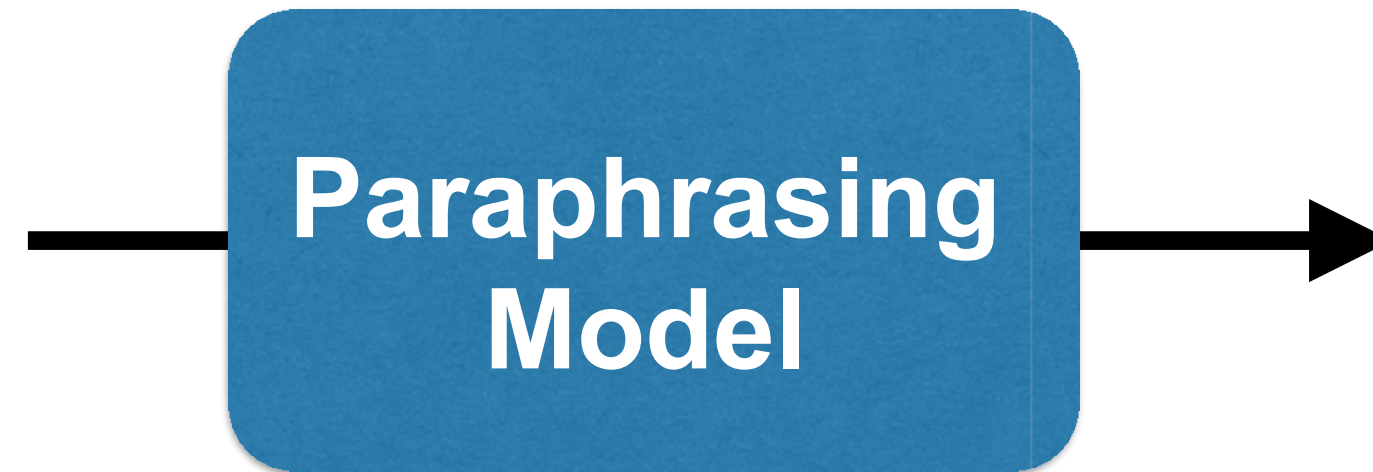


# Automatic Prompting Methods

- **Automatic paraphrases**
- **Iterative refinement**
- **Gradient-based search**
- (Related) **Parameter efficient fine-tuning**
  - Prefix-tuning, prompt-tuning
- ...

# Prompt Paraphrasing

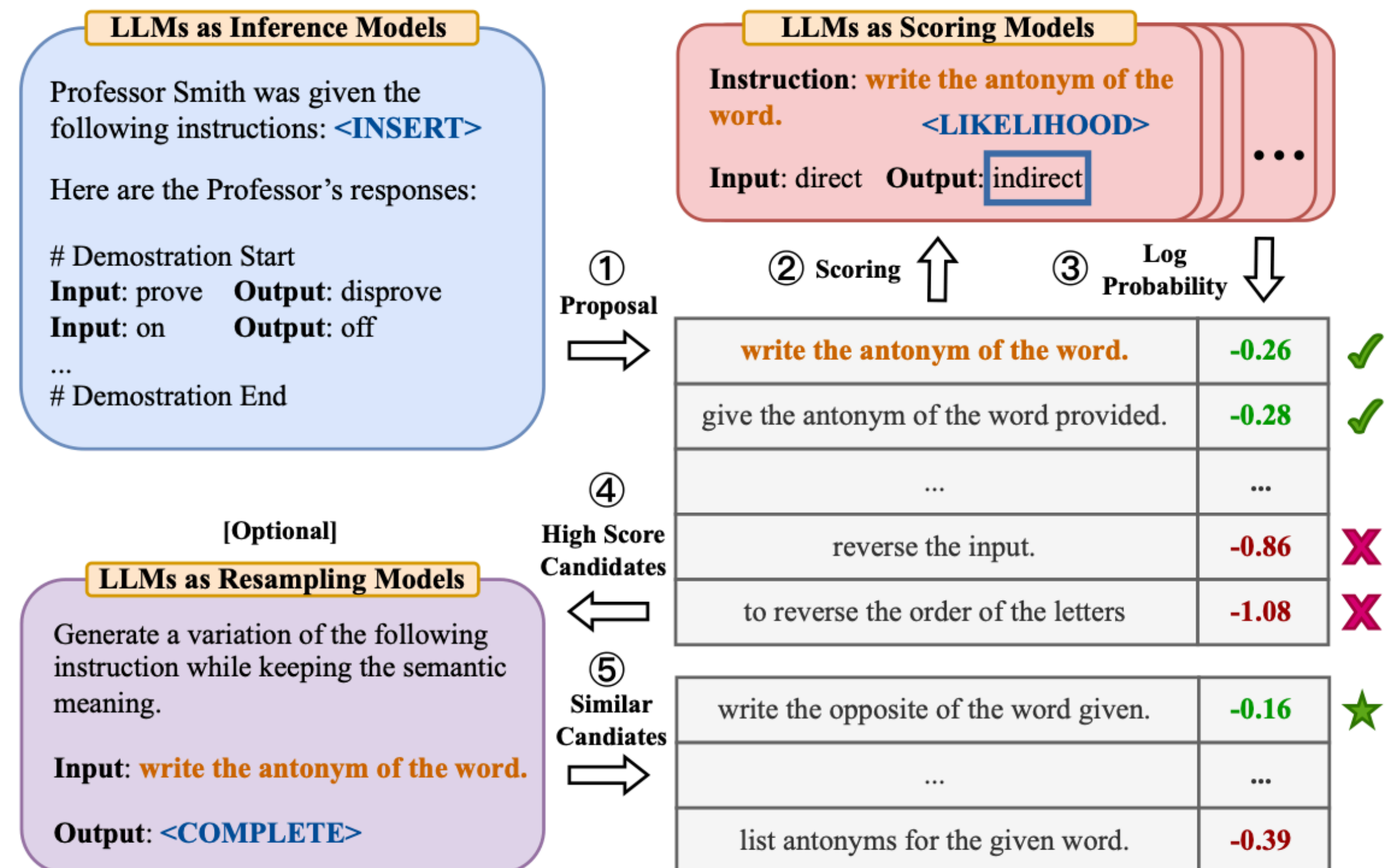
[X] shares a border with [Y].



[X] has a common border with [Y].  
[X] adjoins [Y].  
.....

- Paraphrasing an existing prompt to get other candidates ([Jiang et al. 2019](#))
- Can be done through iterative paraphrasing ([Zhou et al. 2021](#))

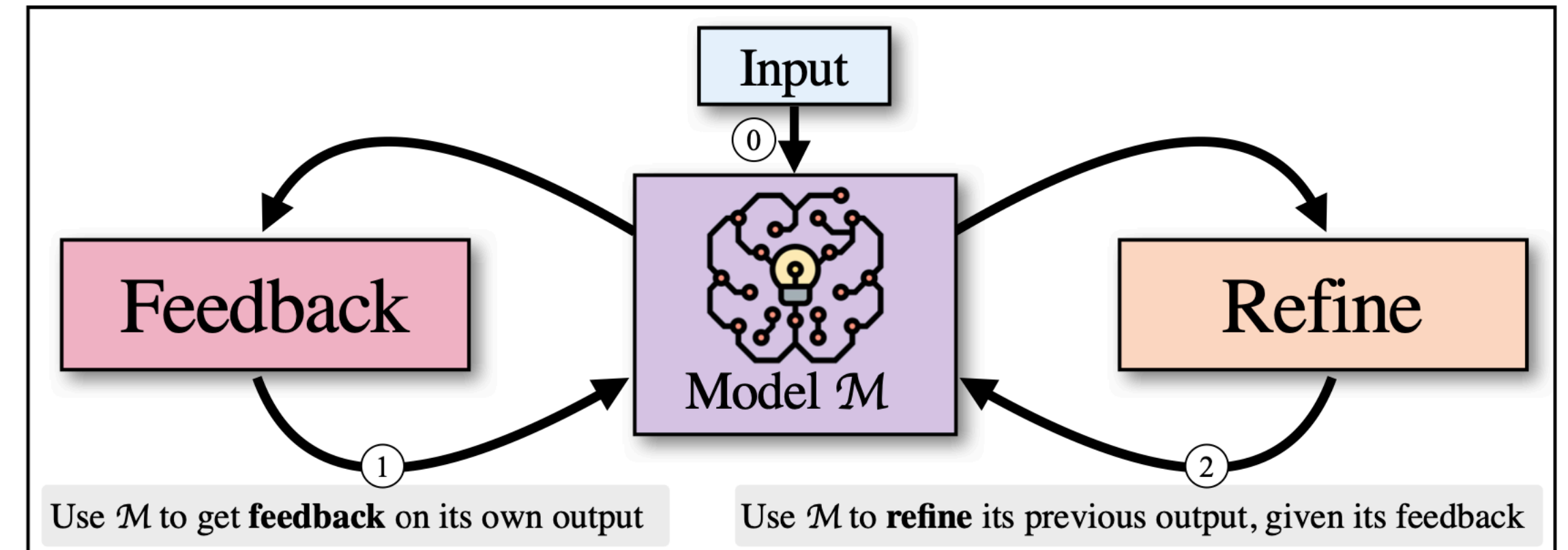
✓ Keep the high score candidates    ✗ Discard the low score candidates    ★ Final selected prompt with highest score





# Self-Refinement Prompting (Madaan et al. 2023)

- **Definition:** LMs revise their own outputs when presented with feedback generated by themselves
- **Advantage:**
  - Simple setup—a single model handles both generation and refinement
  - Reduces the likelihood of incorrect information or hallucinations in the final output



(a) Dialogue:  $x, y_t$

```
User: I am interested in playing Table tennis.  
Response: I'm sure it's a great way to socialize, stay active
```

(b) FEEDBACK fb

```
Engaging: Provides no information about table tennis or how to play it.  
User understanding: Lacks understanding of user's needs and state of mind.
```

(c) REFINE  $y_{t+1}$

```
Response (refined): That's great to hear (...) ! It's a fun sport requiring quick reflexes and good hand-eye coordination. Have you played before, or are you looking to learn?
```

(d) Code optimization:  $x, y_t$

```
Generate sum of 1, ..., N  
def sum(n):  
    res = 0  
    for i in range(n+1):  
        res += i  
    return res
```

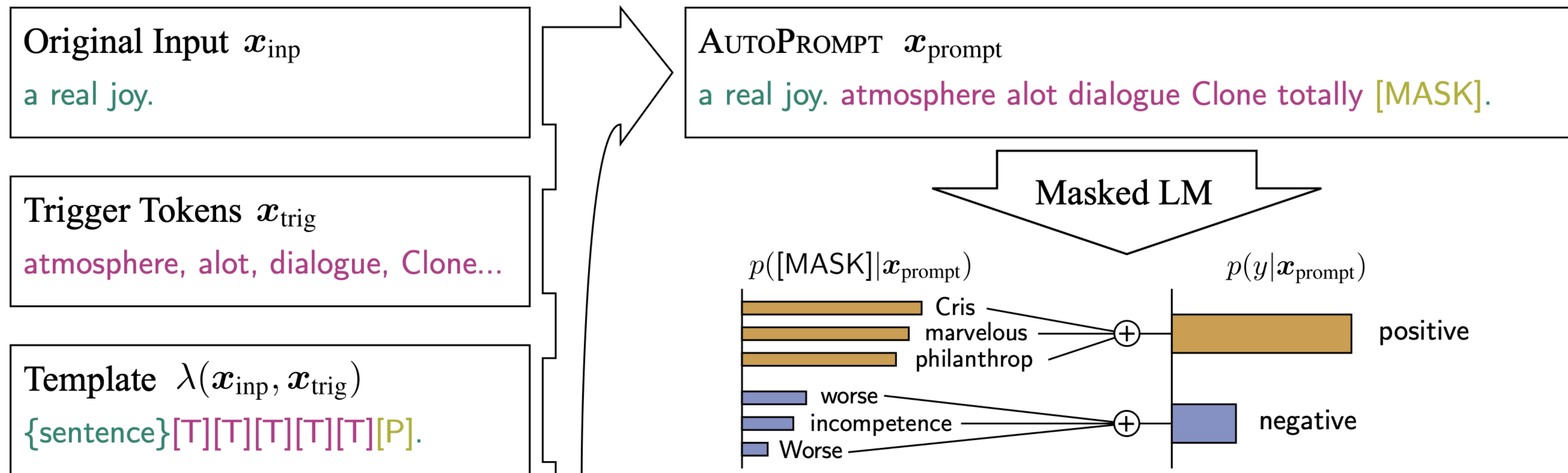
(e) FEEDBACK fb

```
This code is slow as it uses brute force. A better approach is to use the formula ... (n(n+1))/2.
```

(f) REFINE  $y_{t+1}$

```
Code (refined)  
def sum_faster(n):  
    return (n*(n+1))/2
```

# Gradient-Based Search (Shin et al. 2020)

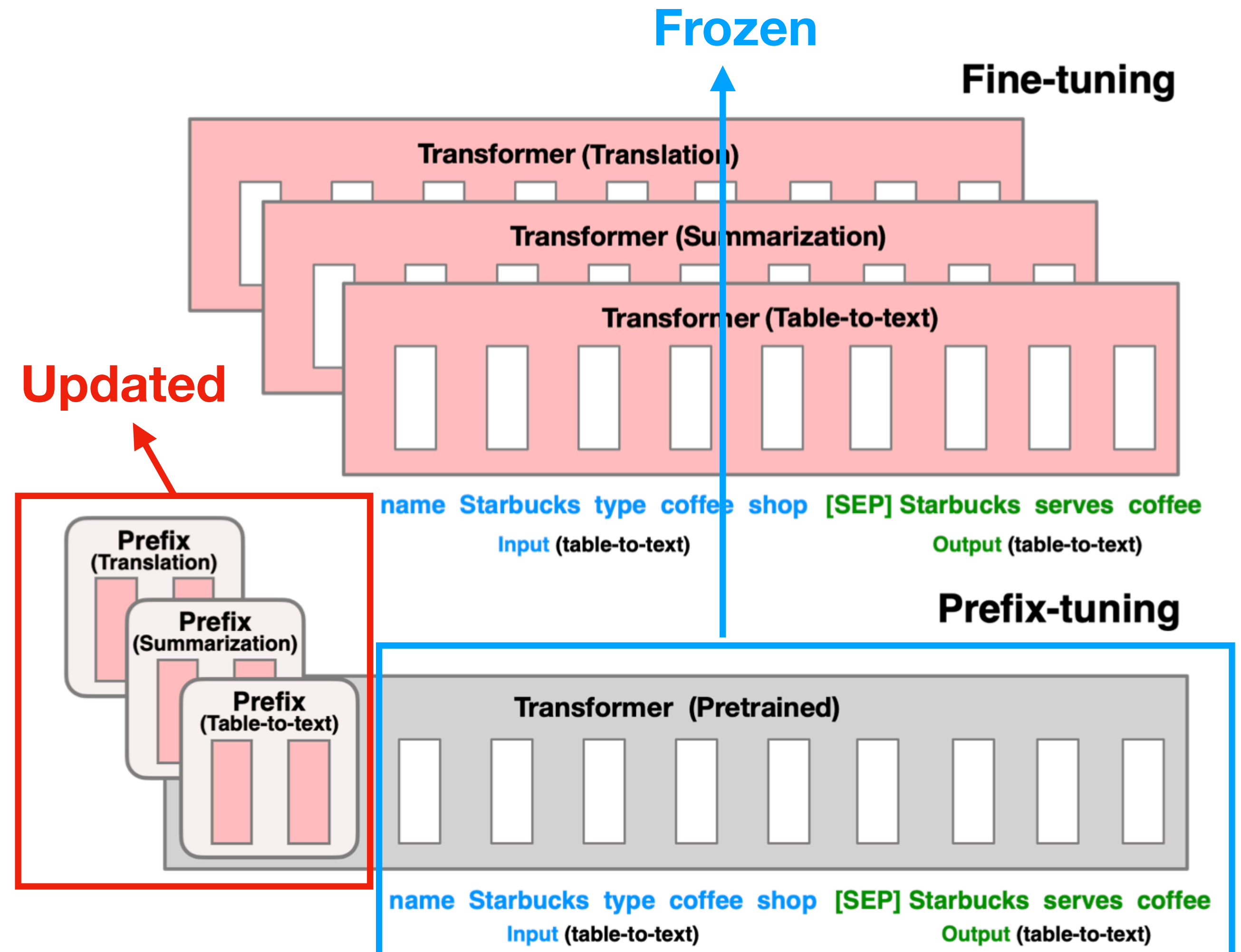


- **Advantage:** fully automating for tasks with clear-cut target outputs
- **Downside:**
  - Usually not interpretable (gibberish)
  - Optimization can be slow, or even impossible for large models

# Prefix-Tuning (Li and Liang, 2021)

Optimizes the prefix of **all layers**

- Learning a small *continuous task-specific* vector (called the prefix) to **each transformer block**, while keeping the pre-trained LM frozen
- With 0.1% parameter is comparable to full fine-tuning, especially under low-data regime

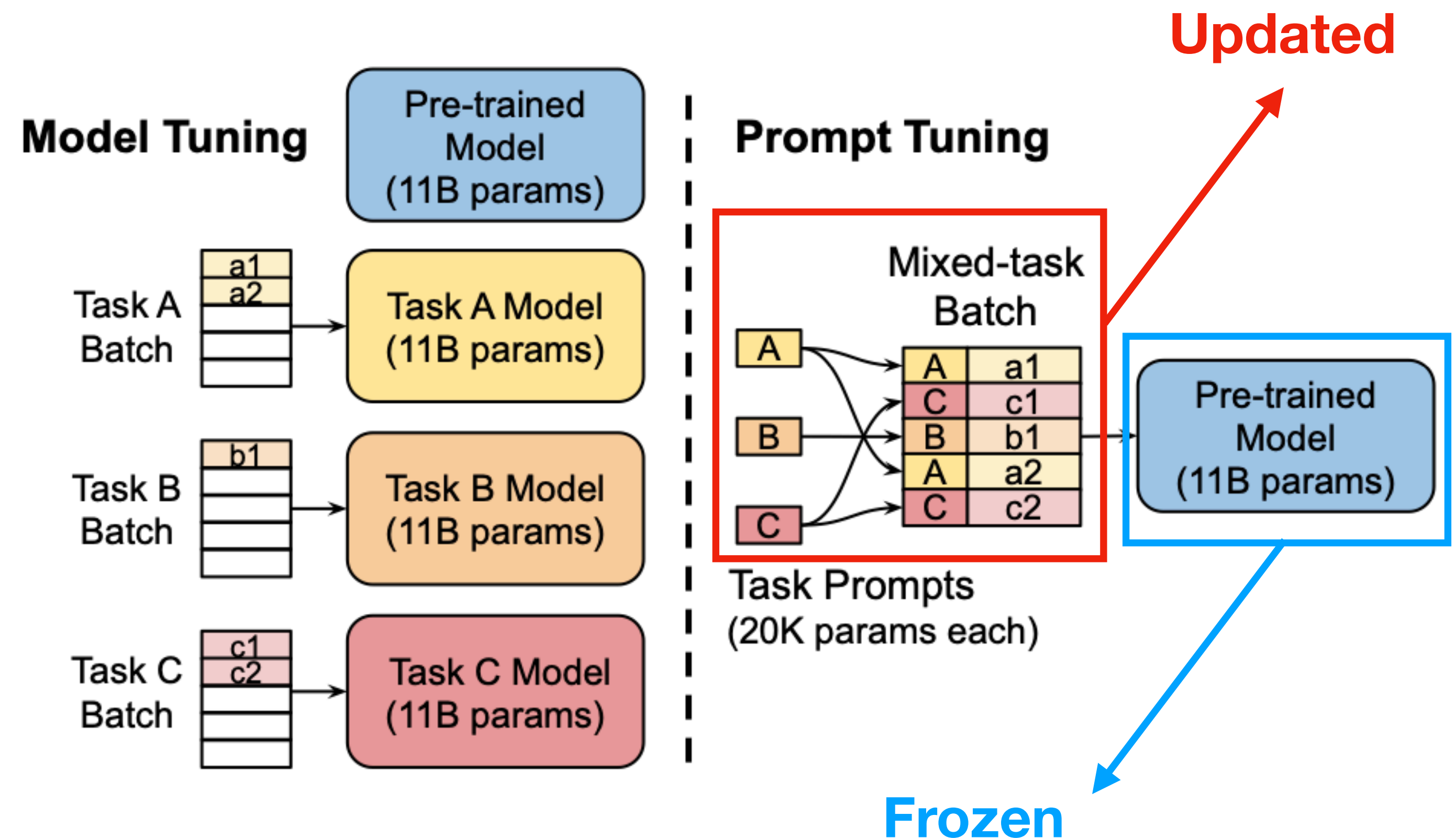




# Prompt-Tuning (Lester et al., 2021)

Optimizes only the **embedding layer**

- Contemporaneous work to prefix-tuning
- A **single** "soft prompt" representation that is prepended to the **embedded input** on the encoder side
- Require **fewer** parameters than prefix-tuning





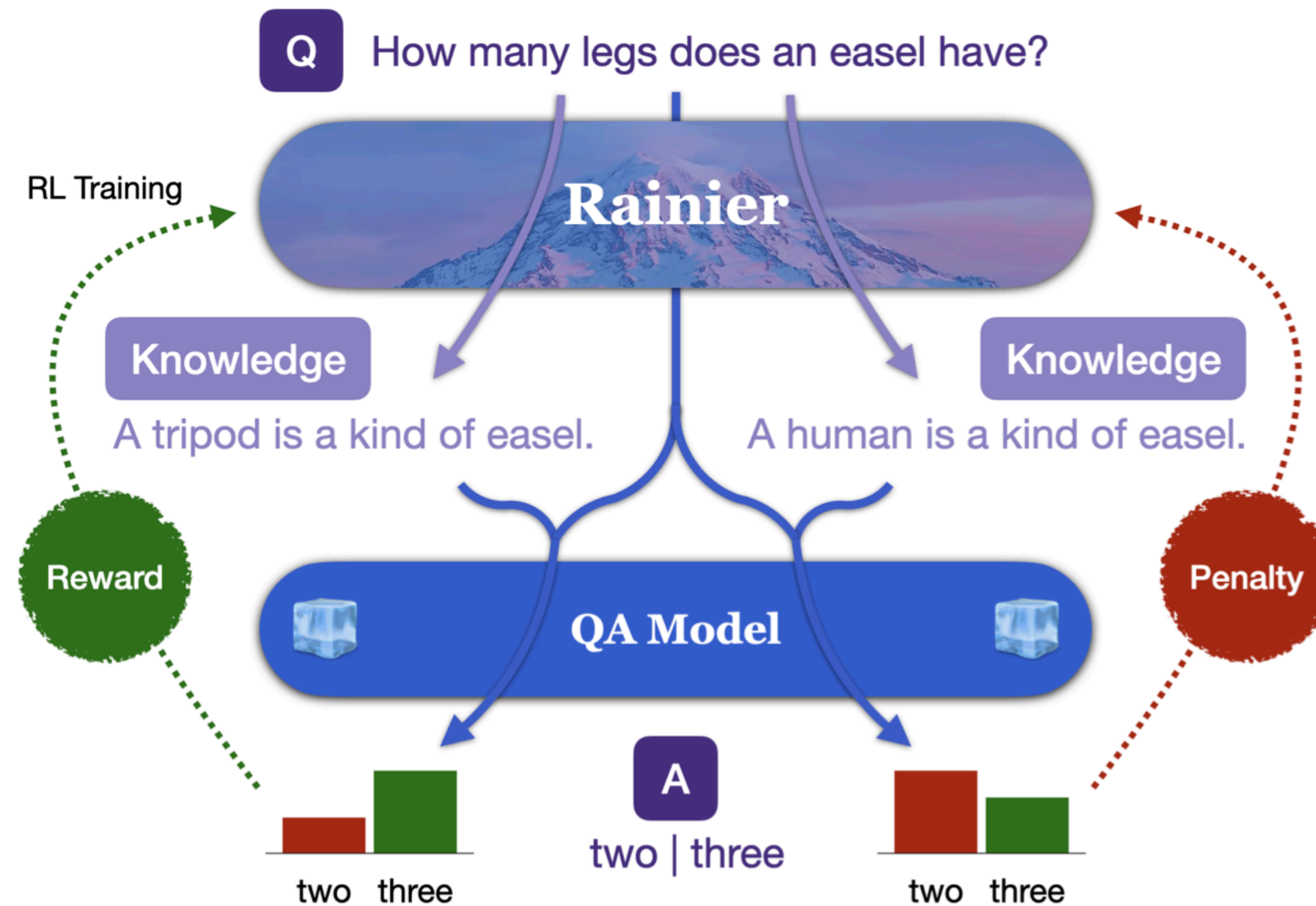
# More about Prompt Engineering

- Other prominent prompting techniques can be found at: <https://www.promptingguide.ai/techniques>
- Survey: [Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing](#)

Advanced Prompting:  
**Knowledge Enhanced Reasoning & Dialog**  
**Think-Before-Speaking**  
**Agent & Tool Use**  
**Preference Elicitation with Clarification Questions**

# Knowledge Enhanced Reasoning

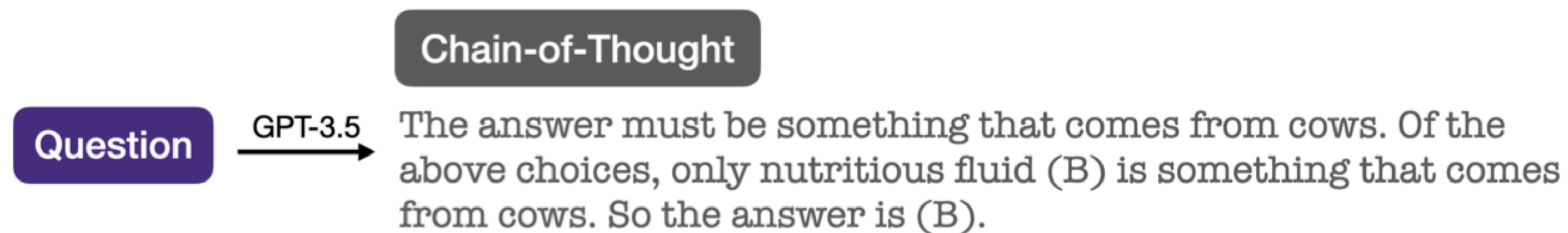
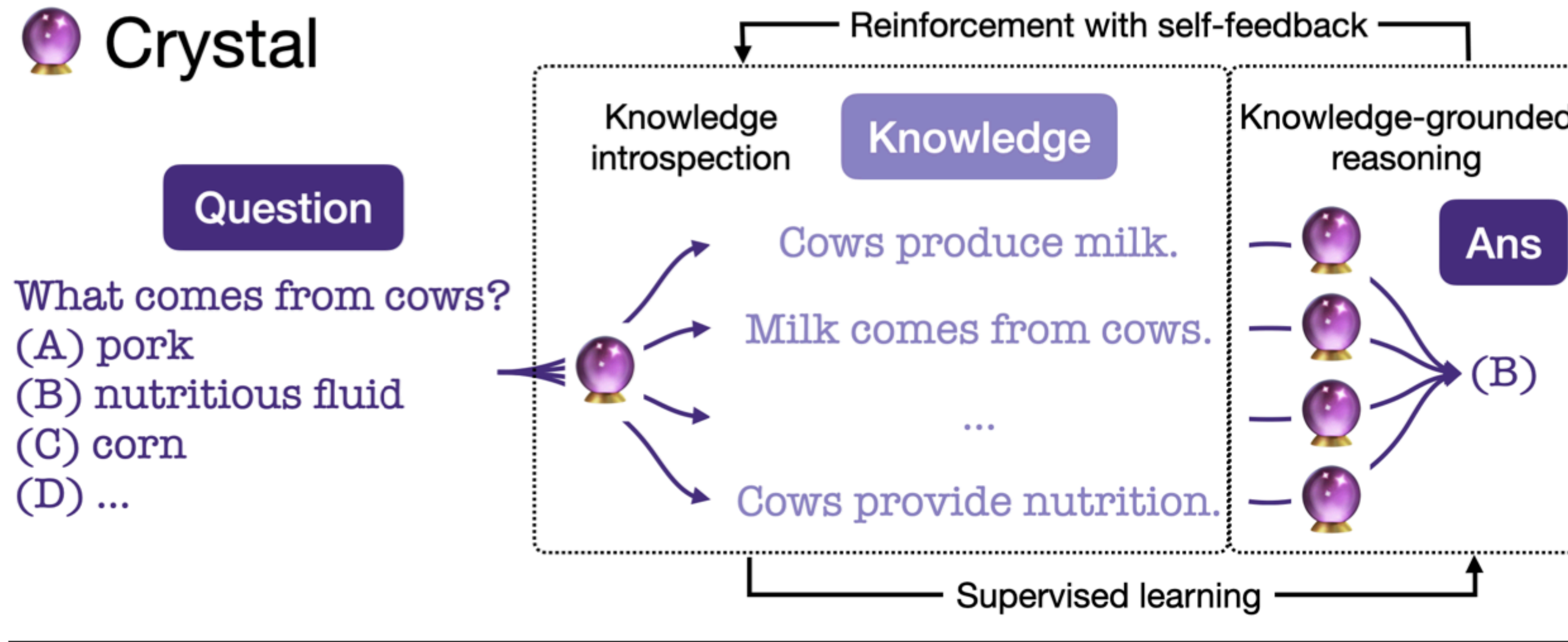
(Rainier, Liu et al. 2022)



Dataset → Method ↓	CSQA		QASC		PIQA		SIQA		WG		Avg.	
	dev	test	dev	test	dev	test	dev	test	dev	test	dev	test
UQA-large (0.77B)	61.43	53.00	43.09	45.65	63.66	65.50	53.84	57.21	53.35	54.67	55.07	55.21
+ Few-shot GPT-3-Curie (13B)	66.34	–	53.24	–	64.25	–	<b>58.29</b>	–	55.56	–	59.54	–
+ Self-talk GPT-3-Curie (13B)	63.31	–	49.89	–	65.23	–	51.89	–	52.96	–	56.66	–
+ DREAM (11B)	64.54	–	49.46	–	64.74	–	51.59	–	56.12	–	57.29	–
<b>+ RAINIER-large (0.77B) [ours]</b>	<b>67.24</b>	<b>60.18</b>	<b>54.97</b>	<b>54.13</b>	<b>65.67</b>	<b>67.09</b>	57.01	<b>59.01</b>	<b>56.91</b>	<b>57.39</b>	<b>60.36</b>	<b>59.56</b>

# Knowledge Enhanced Reasoning

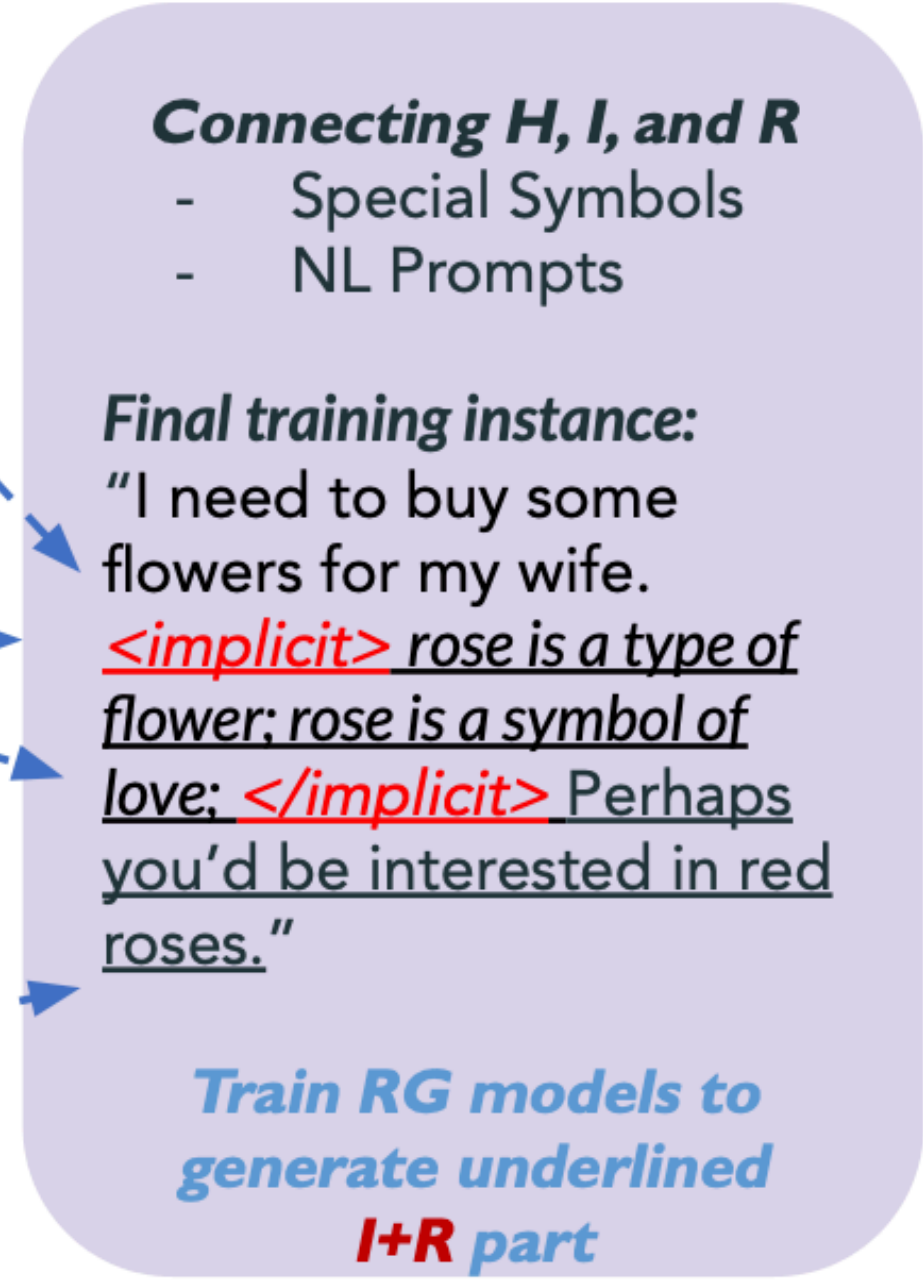
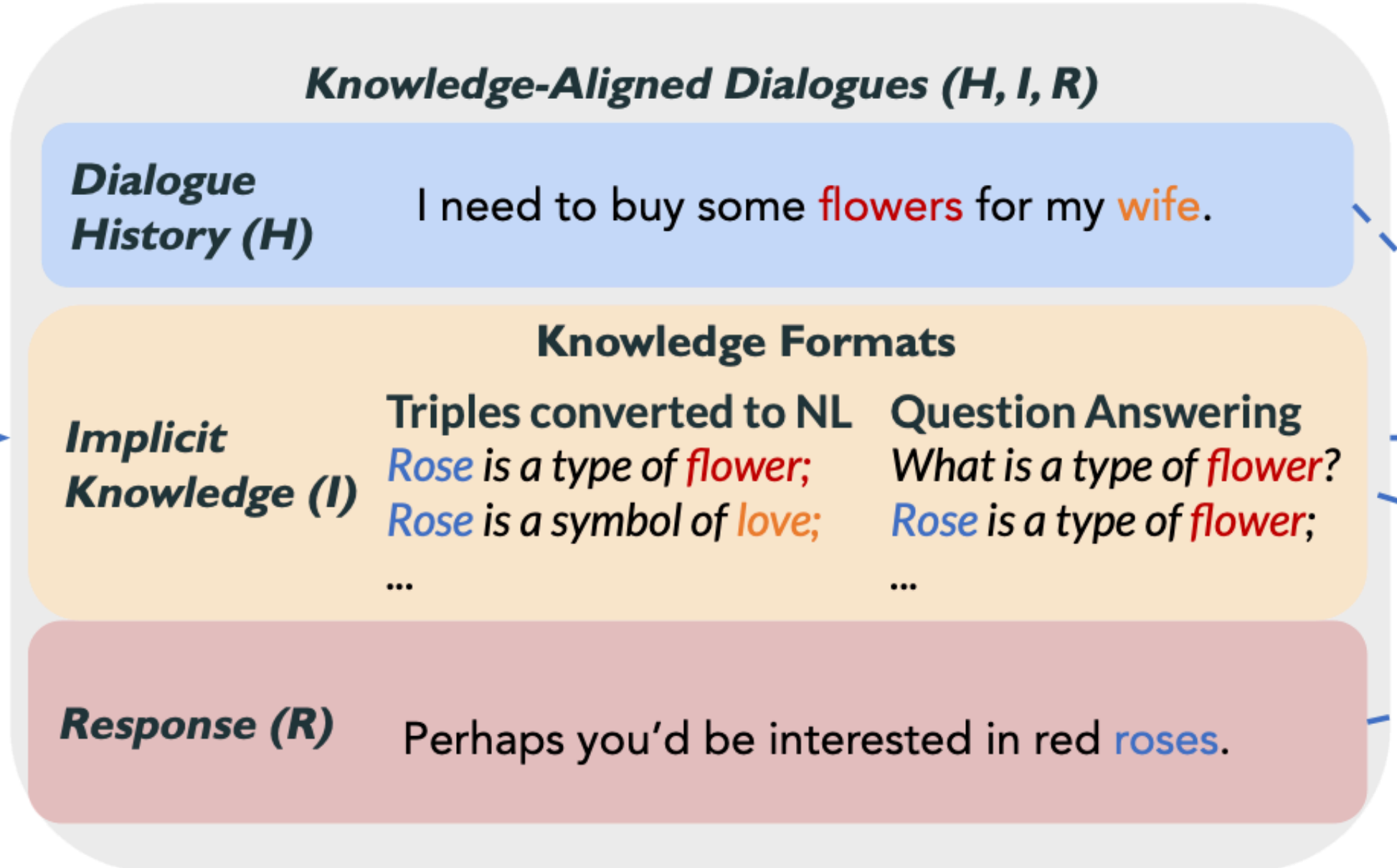
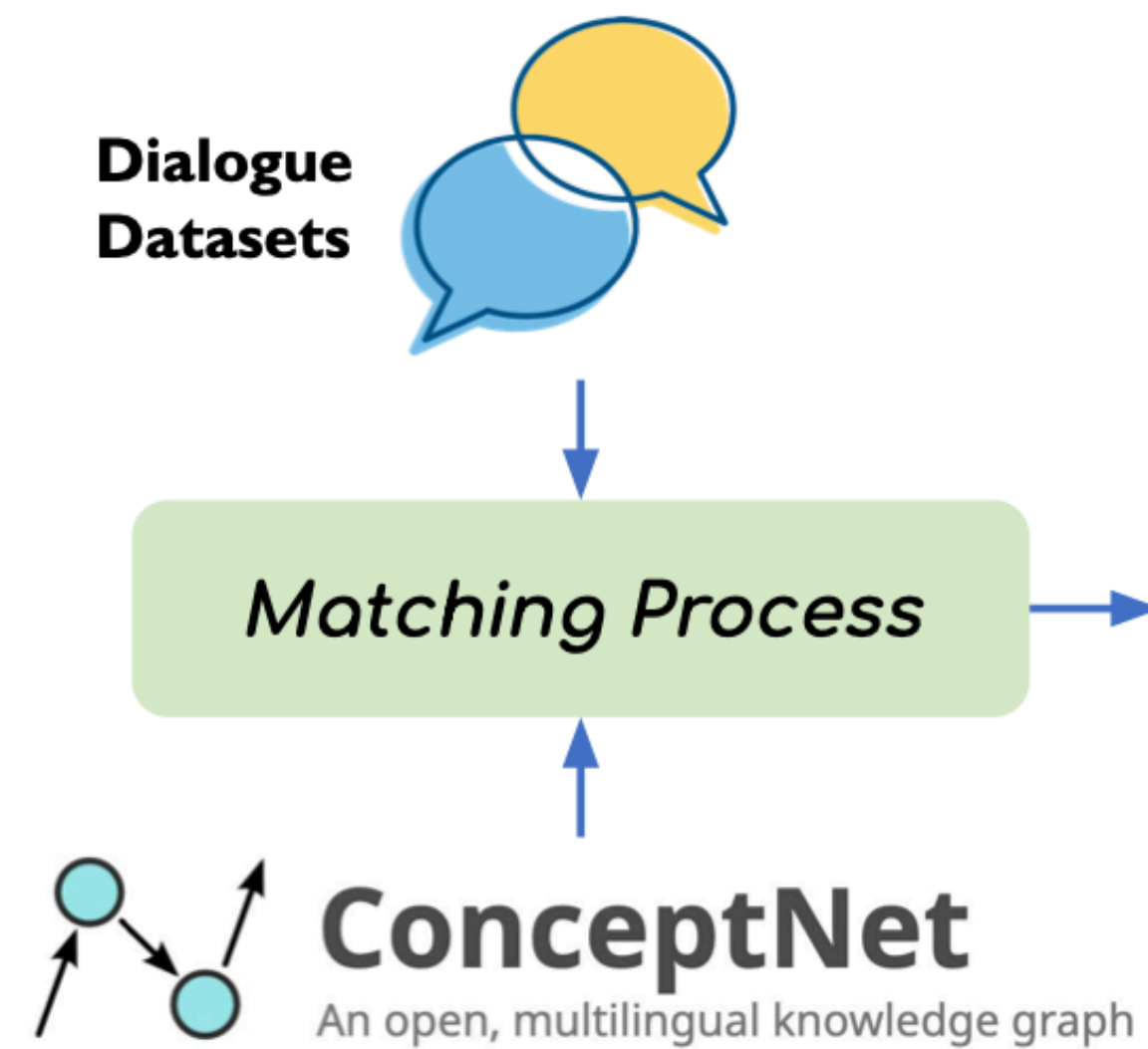
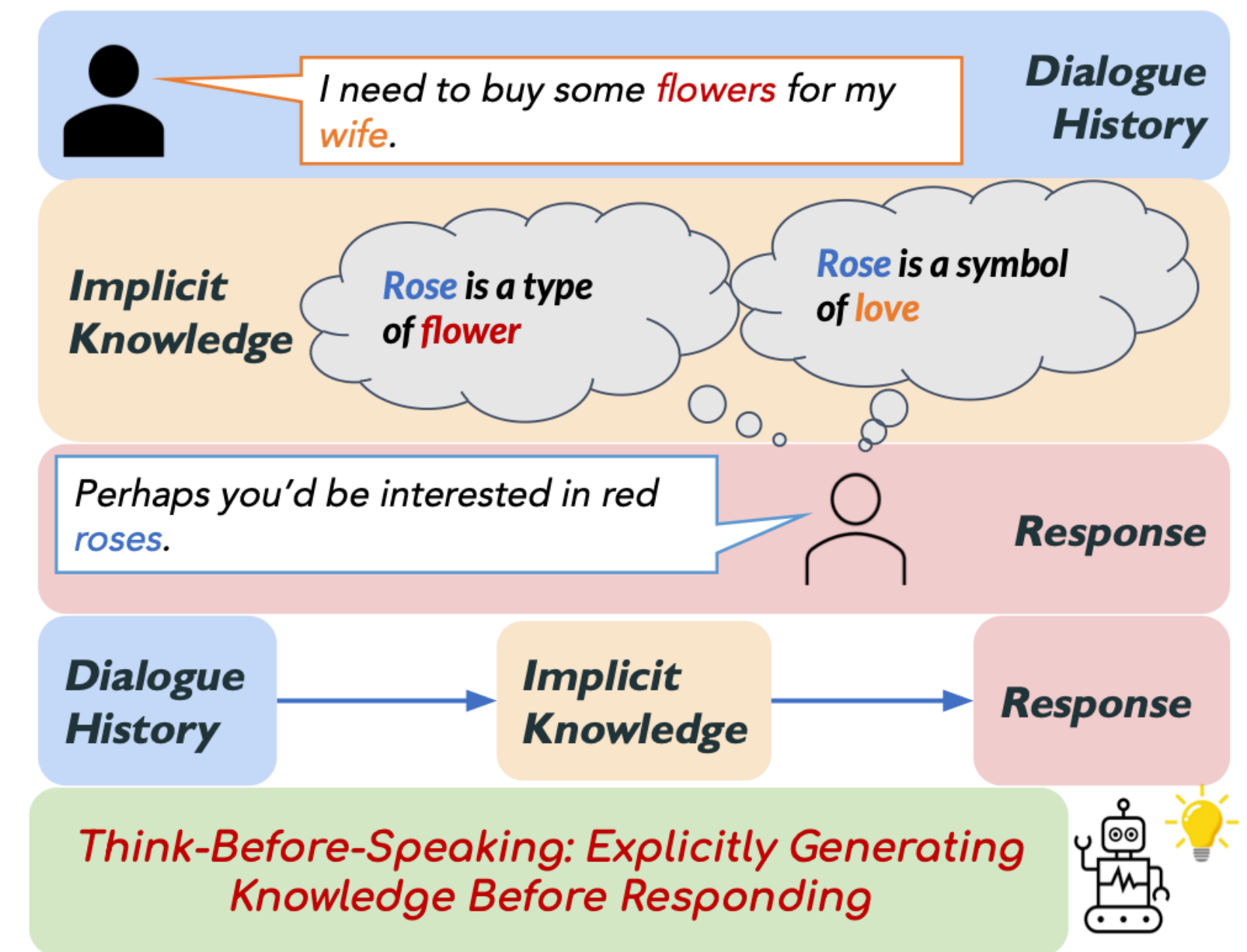
(Crystal, Liu et al. 2023)





# Knowledge Enhanced Dialog

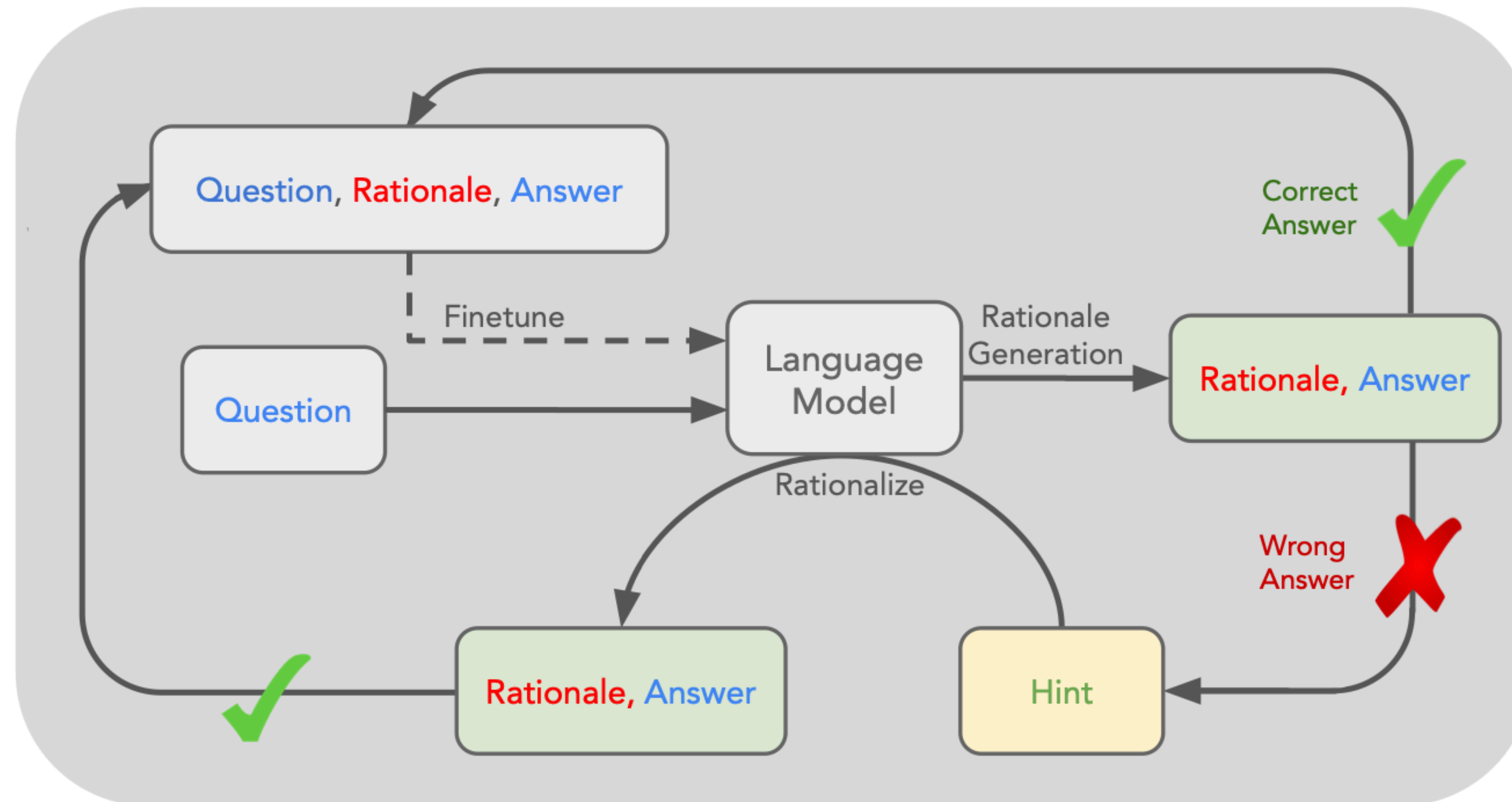
(Zhou et al. 2023)



# STaR: Self-Taught Reasoner

(STaR, Zelikman et al. 2022)

## Bootstrapping Reasoning With Reasoning



Q: What can be used to carry a small dog?

Answer Choices:

- (a) swimming pool
- (b) basket
- (c) dog show
- (d) backyard
- (e) own home

A: The answer must be something that can be used to carry a small dog. Baskets are designed to hold things. Therefore, the answer is basket (b).

Figure 1: An overview of STaR and a STaR-generated rationale on CommonsenseQA. We indicate the fine-tuning outer loop with a dashed line. The questions and ground truth answers are expected to be present in the dataset, while the rationales are generated using STaR.



# STaR: Self-Taught Reasoner

(STaR, Zelikman et al. 2022)

## Bootstrapping Reasoning With Reasoning

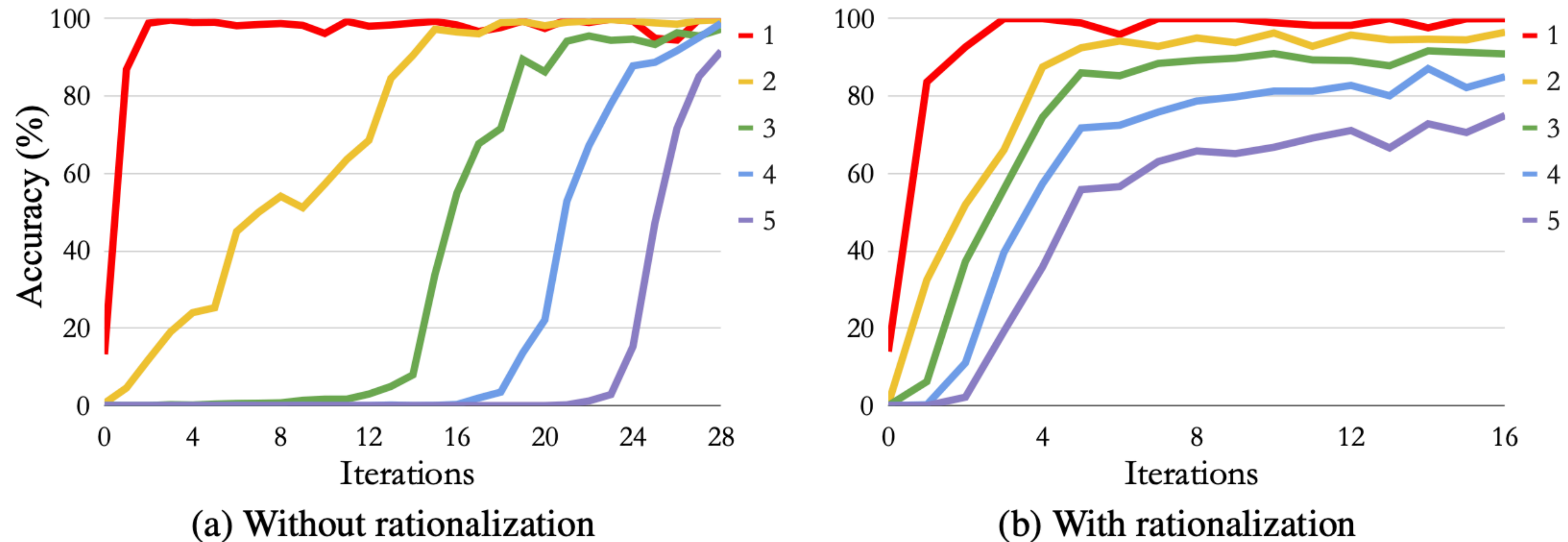


Figure 4: A visualization of the accuracy of  $n$ -digit summation with each iteration of STaR with and without rationalization for arithmetic. Each series corresponds to the accuracy of summing two  $n$ -digit numbers.

# Quiet-STaR:

## Language Models Can Teach Themselves to Think Before Speaking

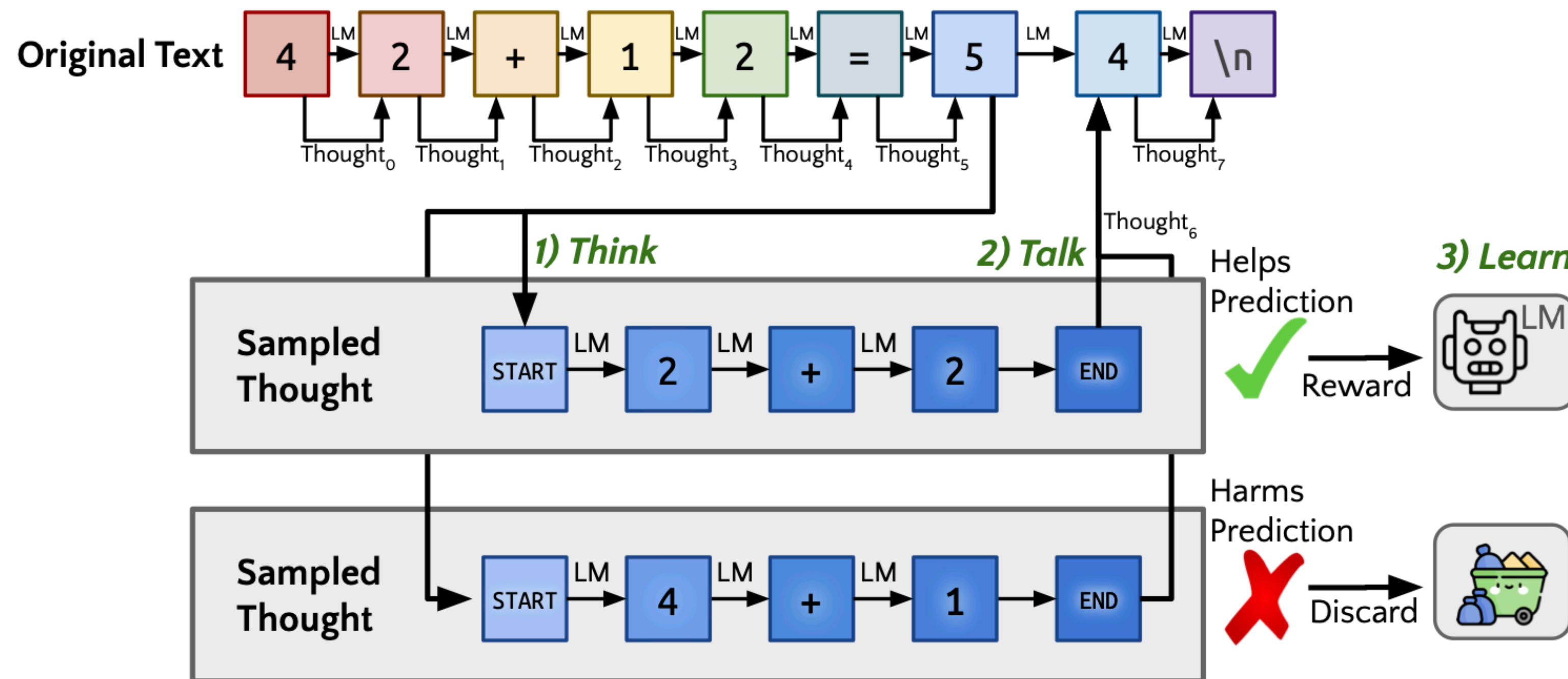
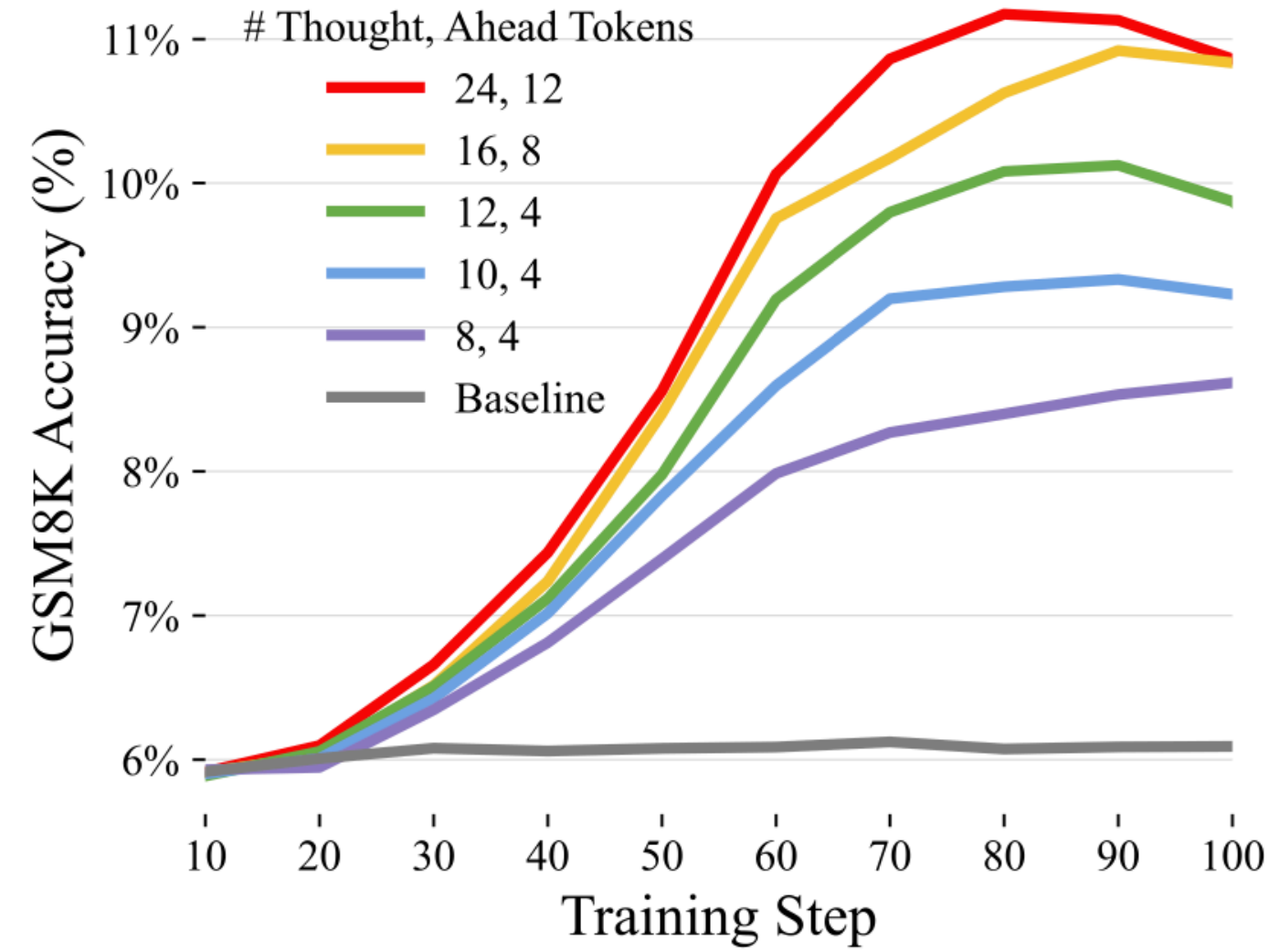


Figure 1: **Quiet-STaR**. We visualize the algorithm as applied during training to a single thought. We generate thoughts, in parallel, following all tokens in the text (**think**). The model produces a mixture of its next-token predictions with and without a thought (**talk**). We apply REINFORCE, as in STaR, to increase the likelihood of thoughts that help the model predict future text while discarding thoughts that make the future text less likely (**learn**).

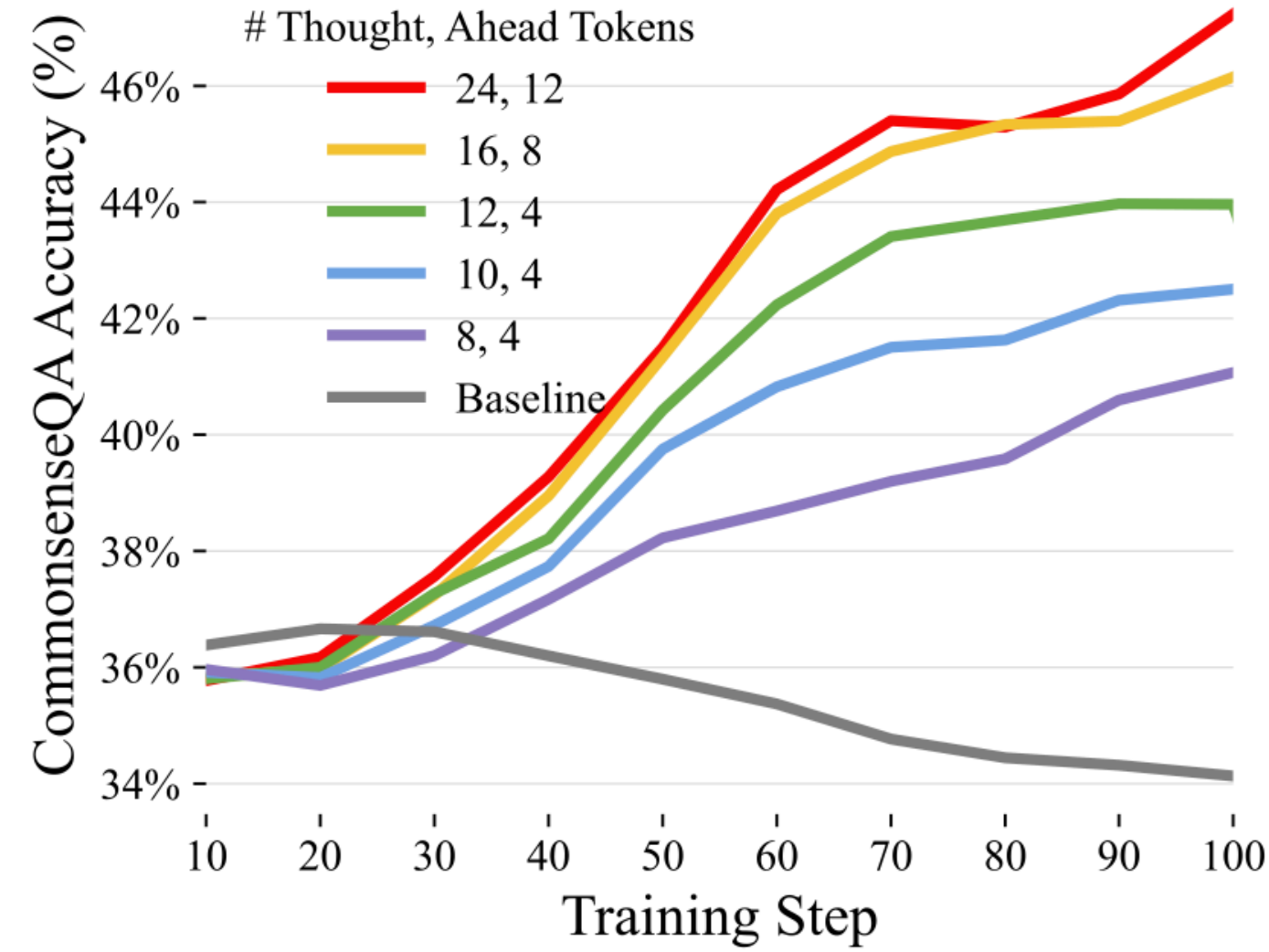


# Quiet-STaR:

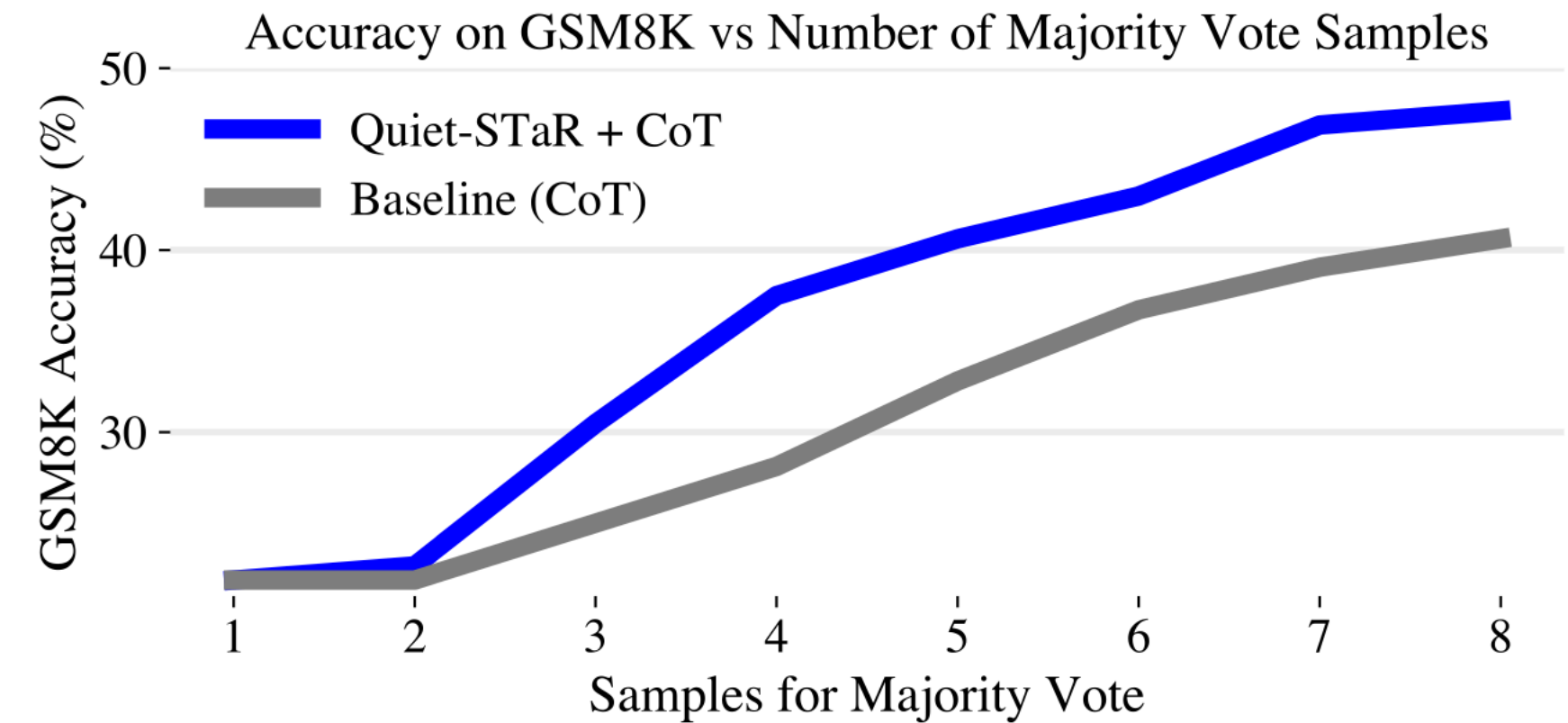
## Language Models Can Teach Themselves to Think Before Speaking



(a) GSM8K



(b) CommonsenseQA

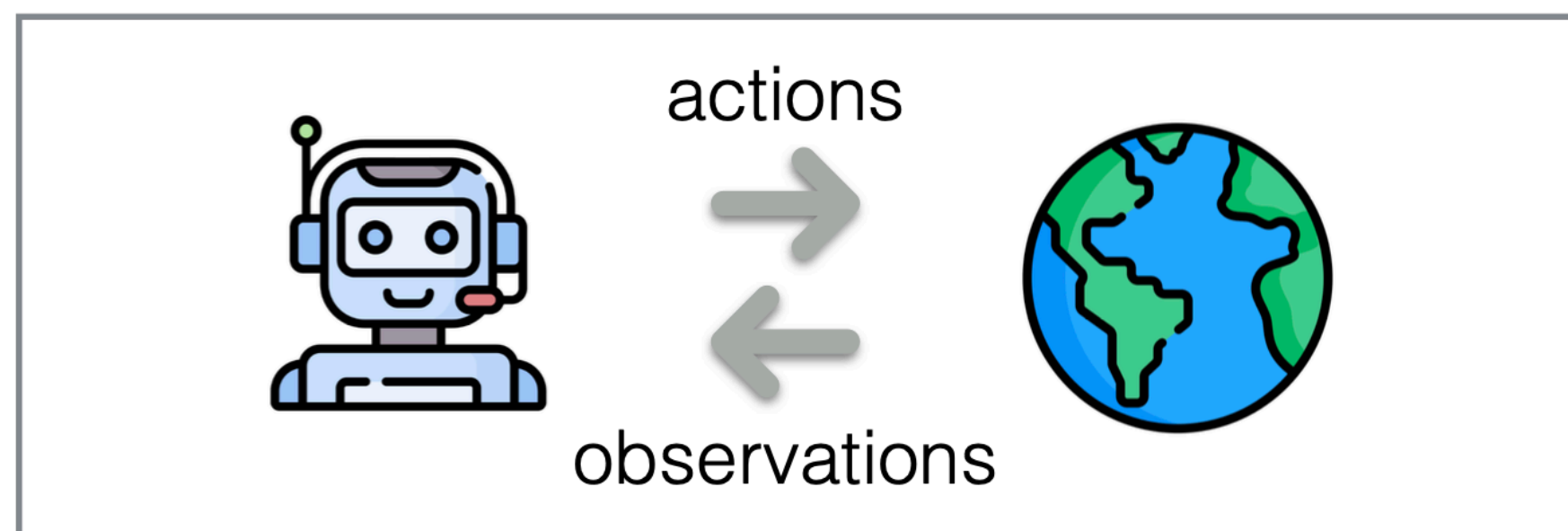


# What are LLM-Powered Agents?

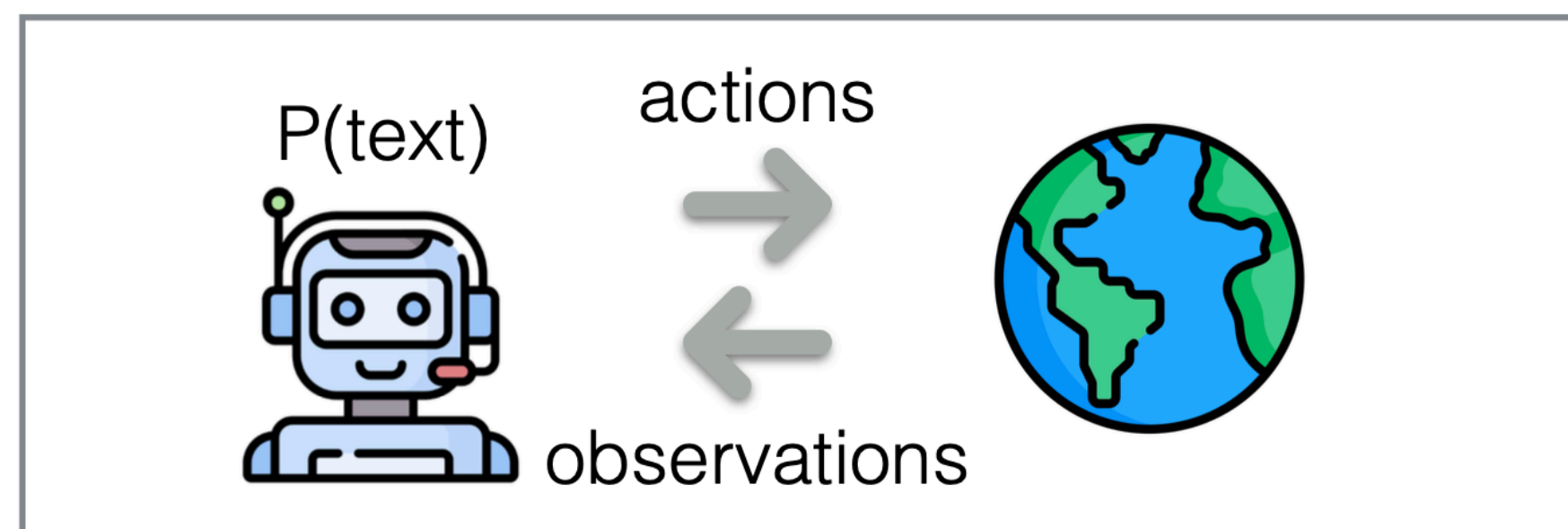
Language models predict text

$P(\text{text})$

AI agents iteratively perform actions in the world



LM agents are an agent with a an LM backbone



## Minimal Components of LLM Agents:

- Underlying LLM
- Prompt
- Action/Observation Space

# Things that LLMs Are Bad At...

## **Numerical/symbolic operations**

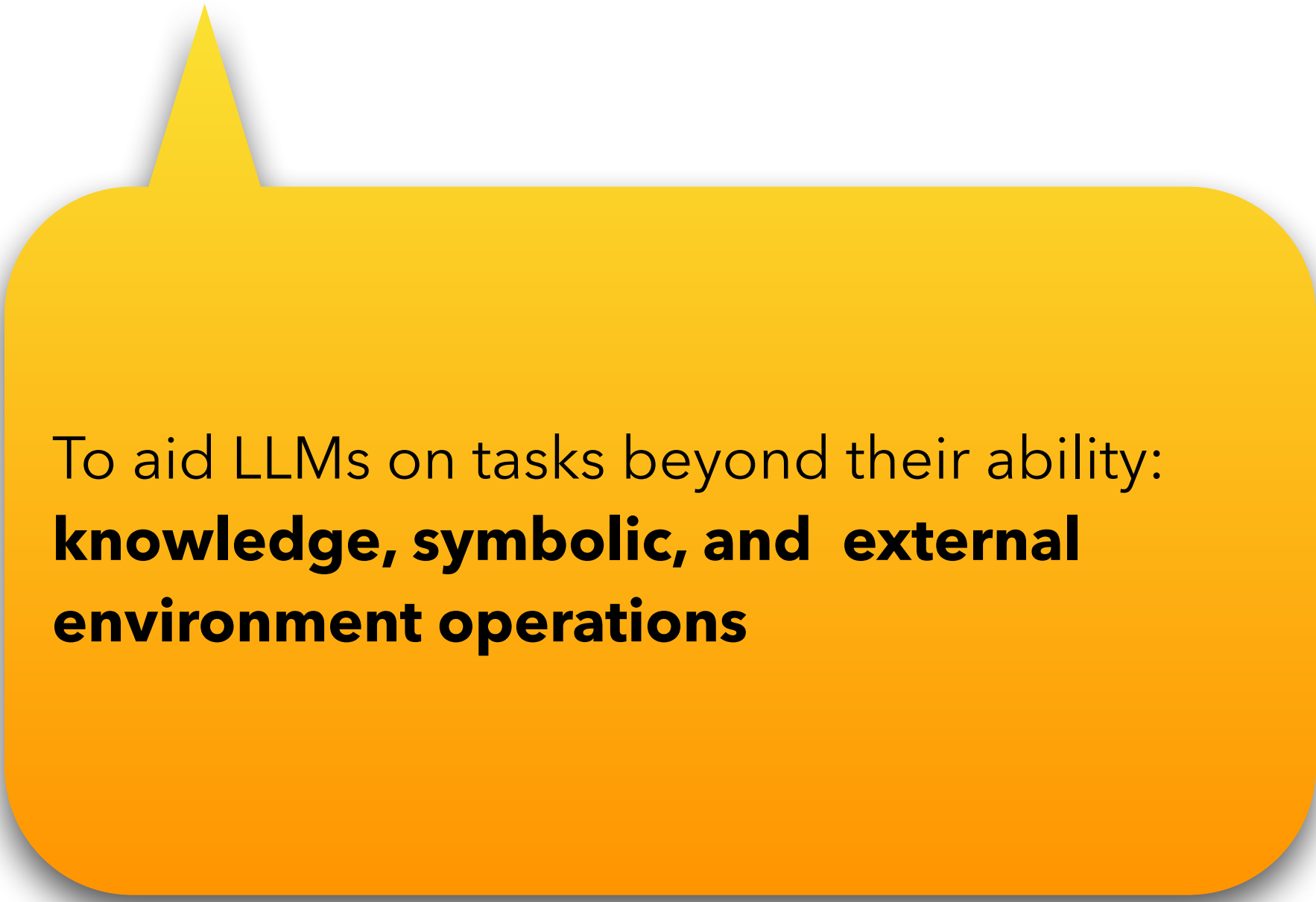
1. Calculation
2. Logic deduction
3. Exact operations

## **Knowledge not in their pre-training corpus**

1. Tail factual knowledge
2. New information
3. Private information

## **Interaction with the external world**

1. Non natural language interfaces
2. Physical world
3. Environmental information (e.g., time)

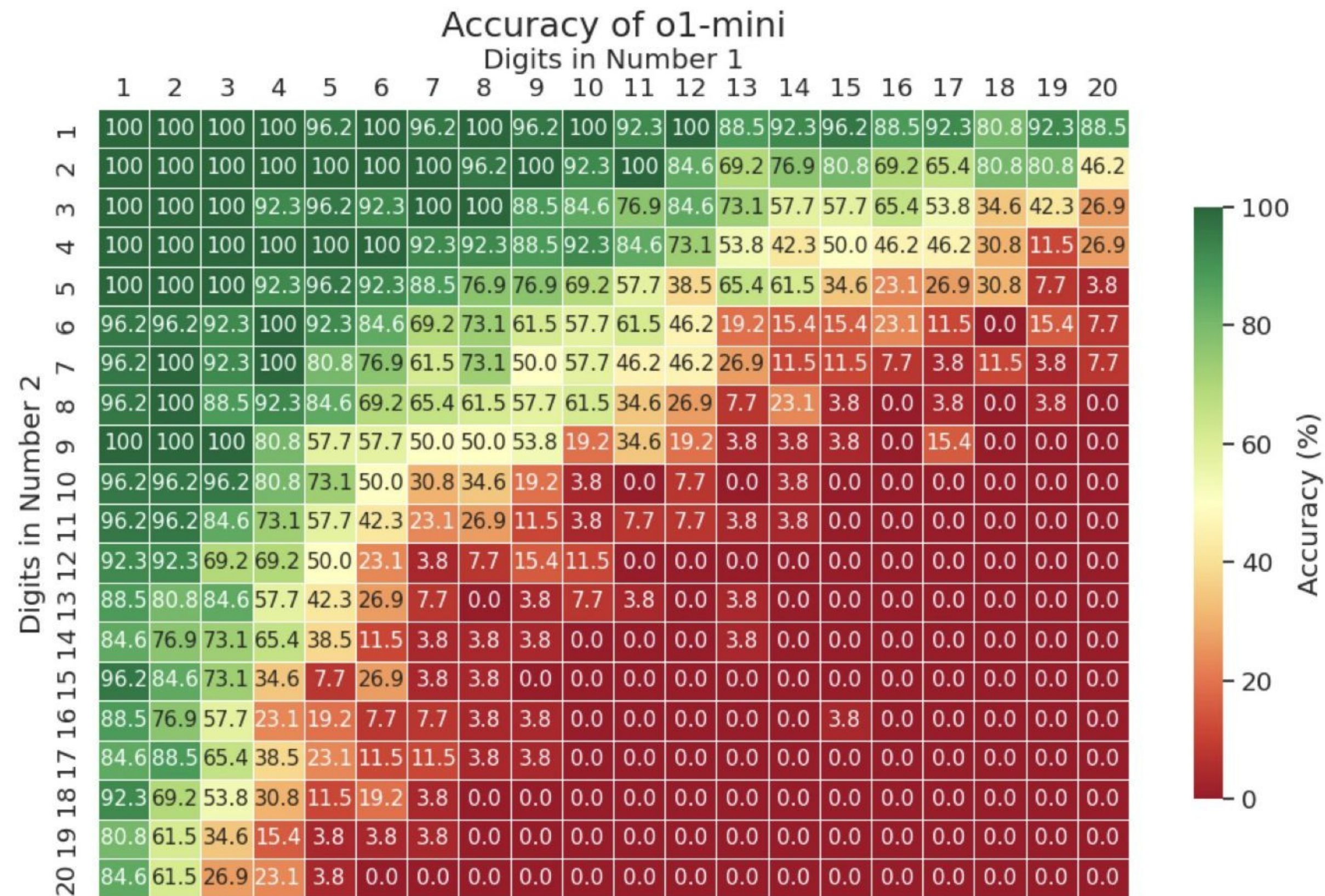


To aid LLMs on tasks beyond their ability:  
**knowledge, symbolic, and external environment operations**



# Why Tools?

LLMs are not the solution for everything. (Not AGI yet. Surprise?)



**O1 cannot solve multiplications of 10+ digits...**

**But why should we expect LLMs to do so?**

**Humans cannot do this on-the-fly either... but we can use calculator to solve it easily.**

**Can LLMs use tools too?**

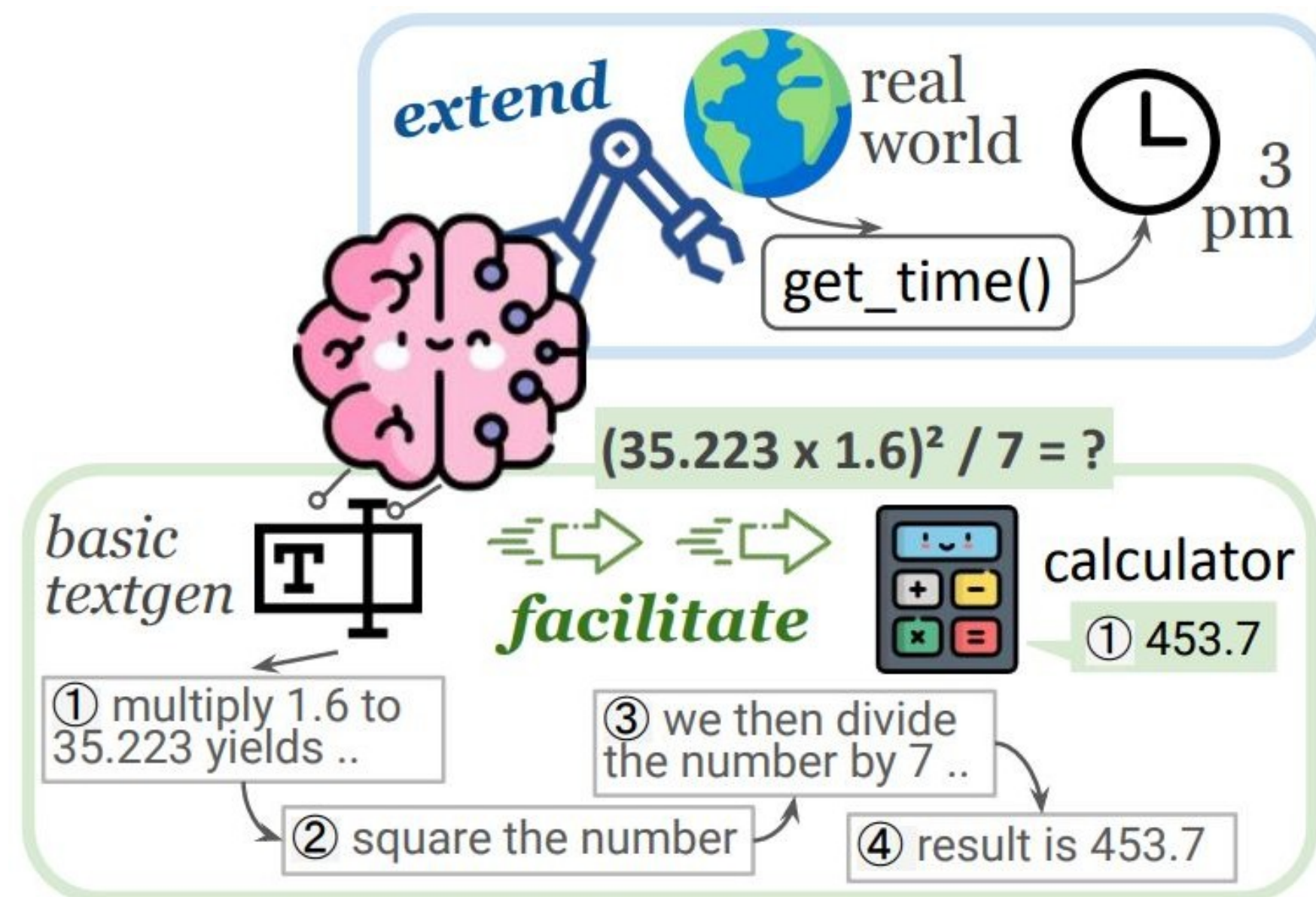
Multiplication Accuracy of OpenAI O1 (Yuantian Deng, X)



# What are Tools?

(Wang et al. 2024.)

**Definition:** An LM-used tool is a function interface to a computer program that runs externally to the LM, where the LM generates the function calls and input arguments in order to use the tool.








## A tool is:

- A Computer Program
- External to the LM
- Used through generated function calls



# What are Tools?

(Wang et al. 2024.)

Category	Example Tools
 Knowledge access	<code>sql_executor(query: str) -&gt; answer: any</code> <code>search_engine(query: str) -&gt; document: str</code> <code>retriever(query: str) -&gt; document: str</code>
 Computation activities	<code>calculator(formula: str) -&gt; value: int   float</code> <code>python_interpreter(program: str) -&gt; result: any</code> <code>worksheet.insert_row(row: list, index: int) -&gt; None</code>
 Interaction w/ the world	<code>get_weather(city_name: str) -&gt; weather: str</code> <code>get_location(ip: str) -&gt; location: str</code> <code>calendar.fetch_events(date: str) -&gt; events: list</code> <code>email.verify(address: str) -&gt; result: bool</code>
 Non-textual modalities	<code>cat_image.delete(image_id: str) -&gt; None</code> <code>spotify.play_music(name: str) -&gt; None</code> <code>visual_qa(query: str, image: Image) -&gt; answer: str</code>
 Special-skilled LMs	<code>QA(question: str) -&gt; answer: str</code> <code>translation(text: str, language: str) -&gt; text: str</code>

# Tool Use & Agent

- **Agent Definition**
  - Disagreement on what “agent” or “agentic” means
- **Requirements:**
  - *Probably*: Proactive use of tools
  - *Probably*: An iterative, multi-step process
  - *Maybe*: Interaction with the outside world



# Tool Usage Performance



(Toolformer, Snihck et al. 2023)

Significantly Improving GPT's Performances



# Can Models Ask Clarification Questions?

Similar to humans, but LMs (as-is) don't complain when the instructions are unclear

 What is a good pasta recipe?

Cook pasta, add chicken broth... [**wasted tokens**] 

 I am vegetarian! 😡

Here is a vegetarian... [**relevant tokens**] 

Ineffective Conversation

 What is a good pasta recipe?

To start off, do you have any dietary restrictions? 

 I am vegetarian. 🥑

Here is a vegetarian... [**relevant tokens**] 

Effective Conversation

- **Task ambiguity**
- Teaching the model to ask questions that best **elicit a particular user's preferences**

(STaR-GATE, Andukuri et al. 2024)

# STaR-GATE

(STaR-GATE, Andukuri et al. 2024)

Response generated by knowing both (Task + Persona)

$P(\text{Gold Response} \mid \text{Conversations})$

The base LM that's responsible for answering questions + asking clarification questions

E.g., A user named Zara has approached you with a request for help.

...  
What must I do to prepare for a job interview?

esponse

Conversations

Elicitation

Filtering

Regularization

w/ Gold Response

Finetune

Questioner

Questions

Roleplayer

Answers

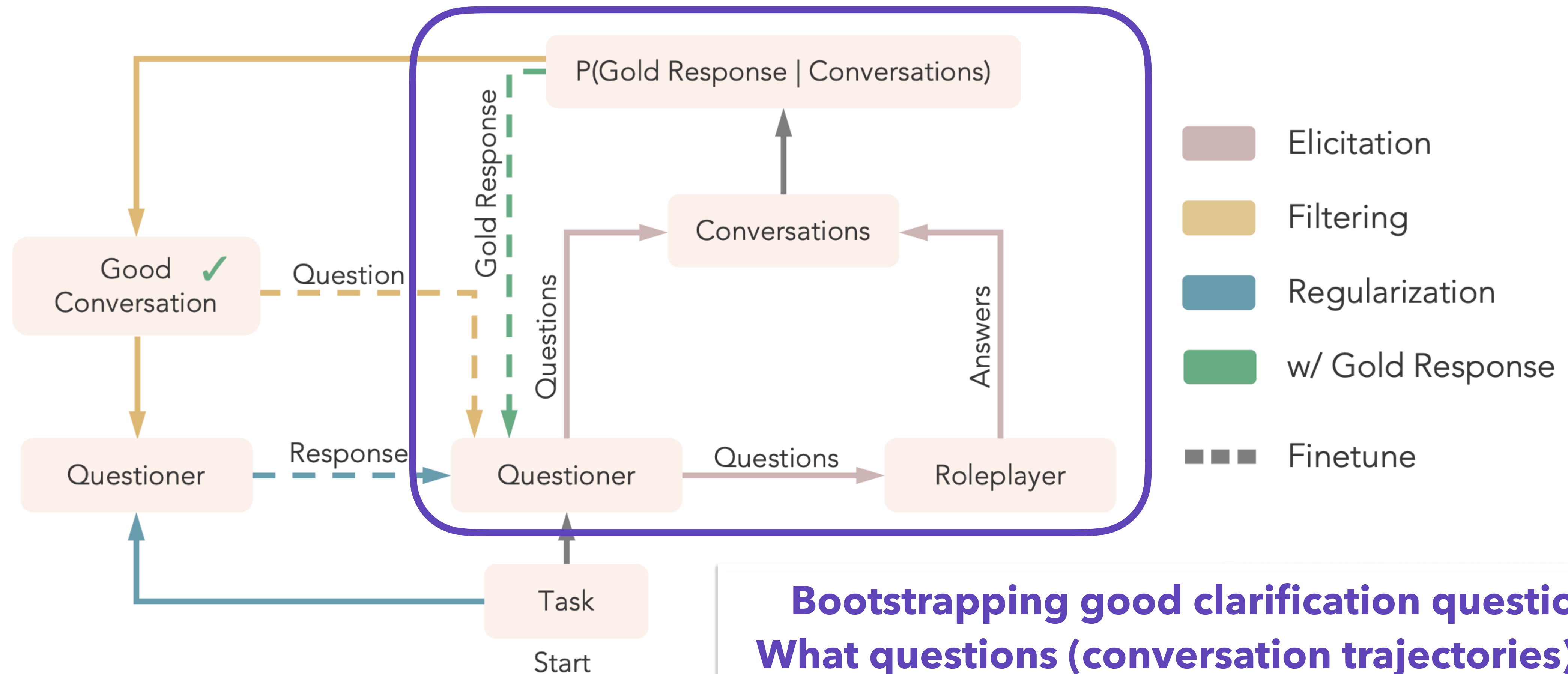
Task

Start

A simulated user with a pre-specified persona who makes a request

# STaR-GATE

(STaR-GATE, Andukuri et al. 2024)

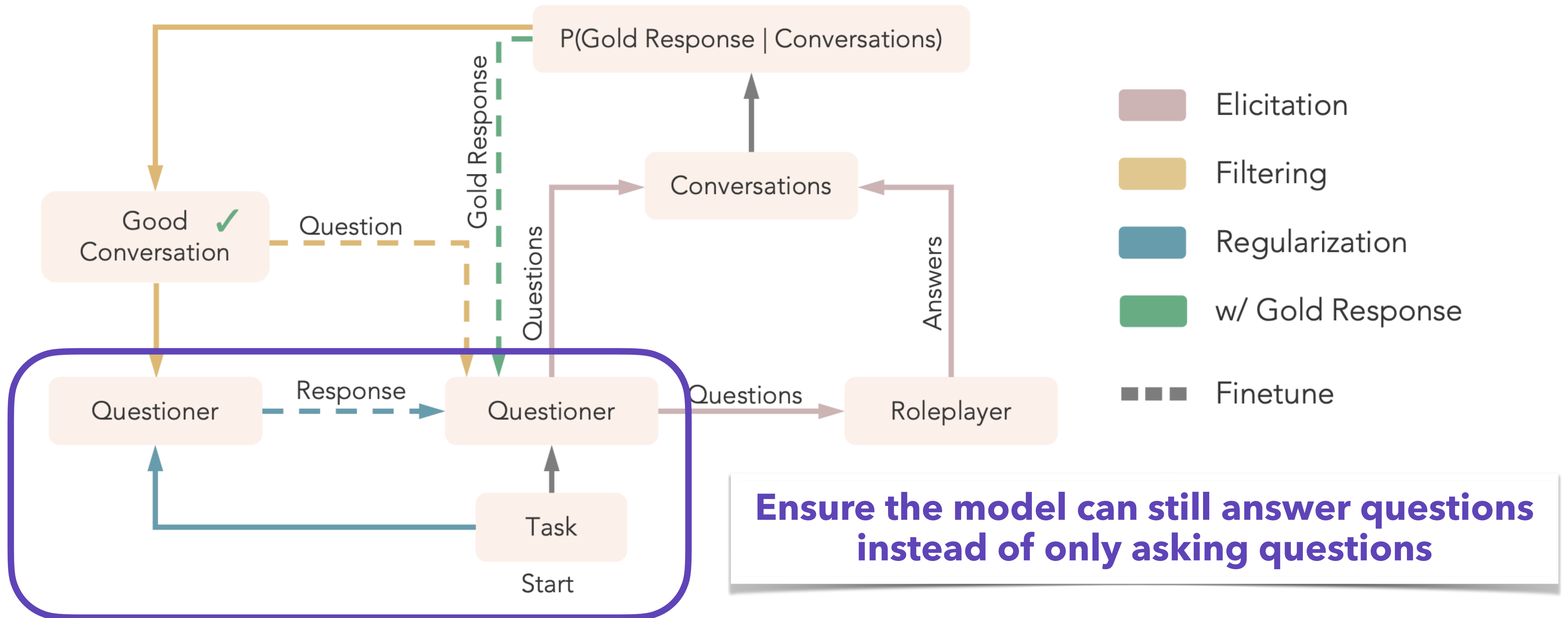


**Bootstrapping good clarification question.  
What questions (conversation trajectories) are  
most likely to elicit gold responses?**



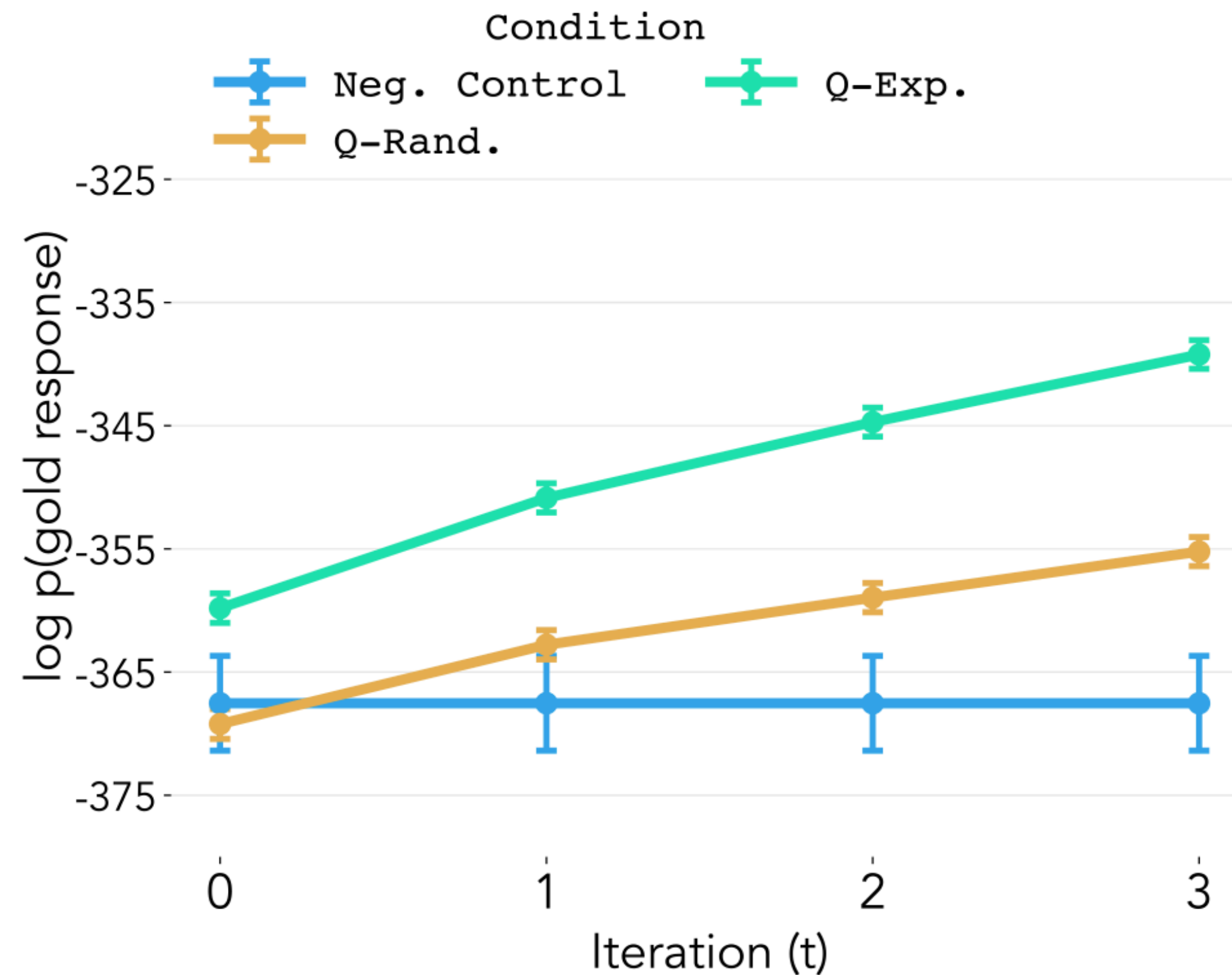
# STaR-GATE

(STaR-GATE, Andukuri et al. 2024)

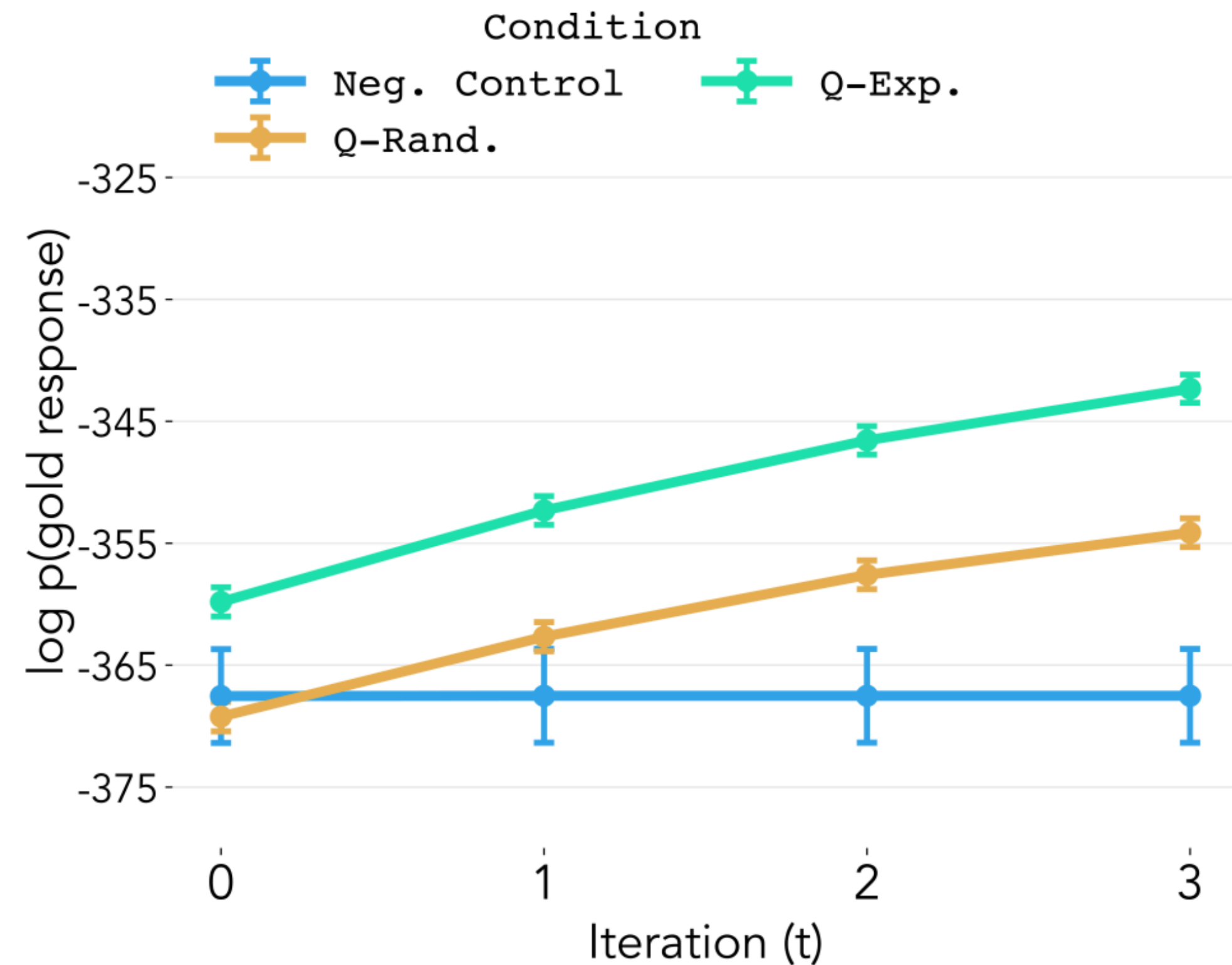


# Can Models Ask Clarification Questions?

(STaR-GATE, Andukuri et al. 2024)



[a] STaR-GATE



[b] w/o Regularization

**Thank you!**