

CS11-711 Advanced NLP

Prompting and Complex Reasoning

Graham Neubig



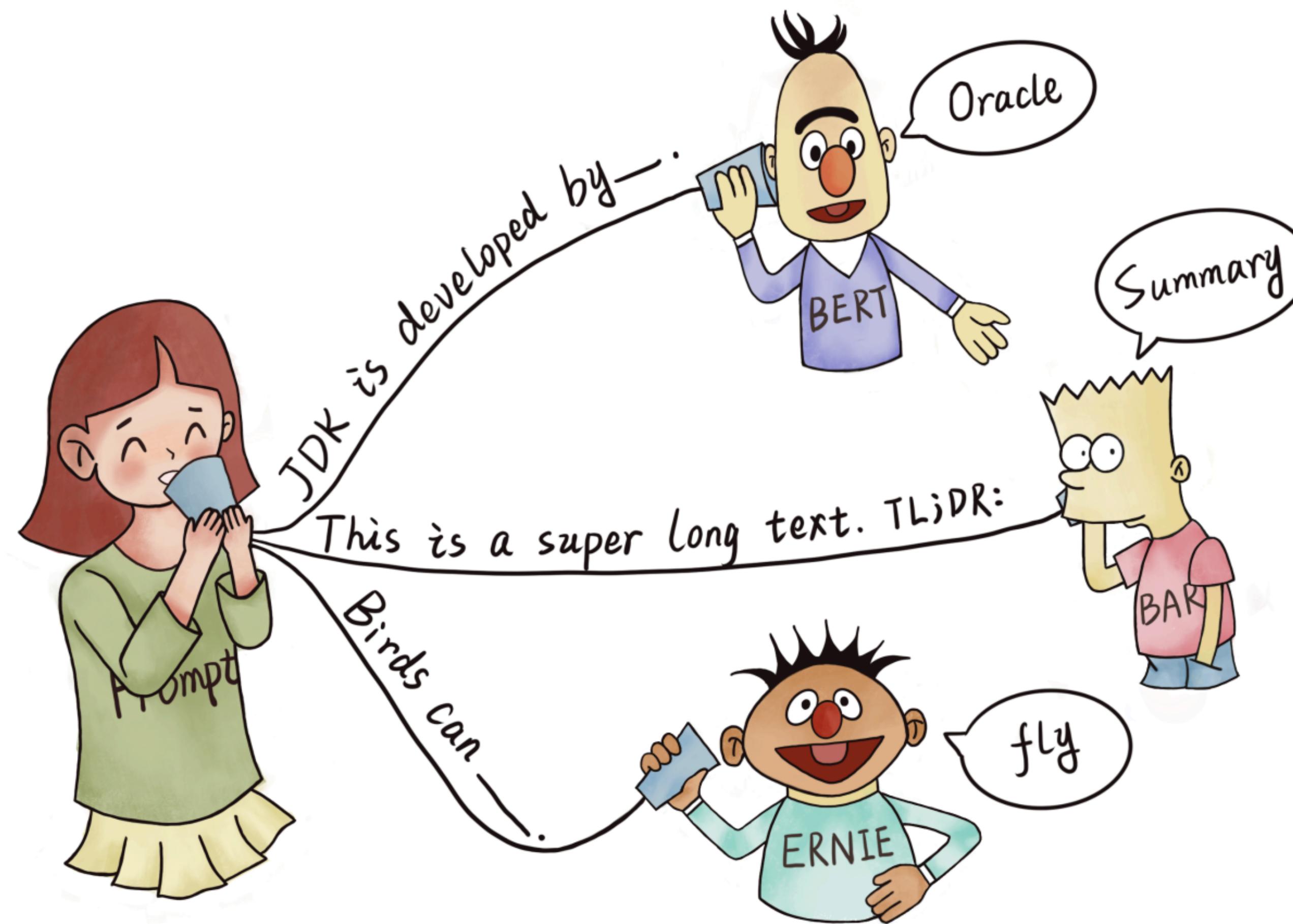
Carnegie Mellon University
Language Technologies Institute

<https://phontron.com/class/anlp-fall2024/>

Some Slides by Pengfei Liu

What is Prompting ?

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



Prompting Fundamentals

Basic Prompting

(Radford et al. 2018)

- Append a textual string to the beginning of the sequence and complete

$x = \text{When a dog sees a squirrel, it will usually}$

(GPT-2 Small)

be afraid of anything unusual. As an exception, that's when a squirrel is usually afraid to bite.

(GPT-2 XL)

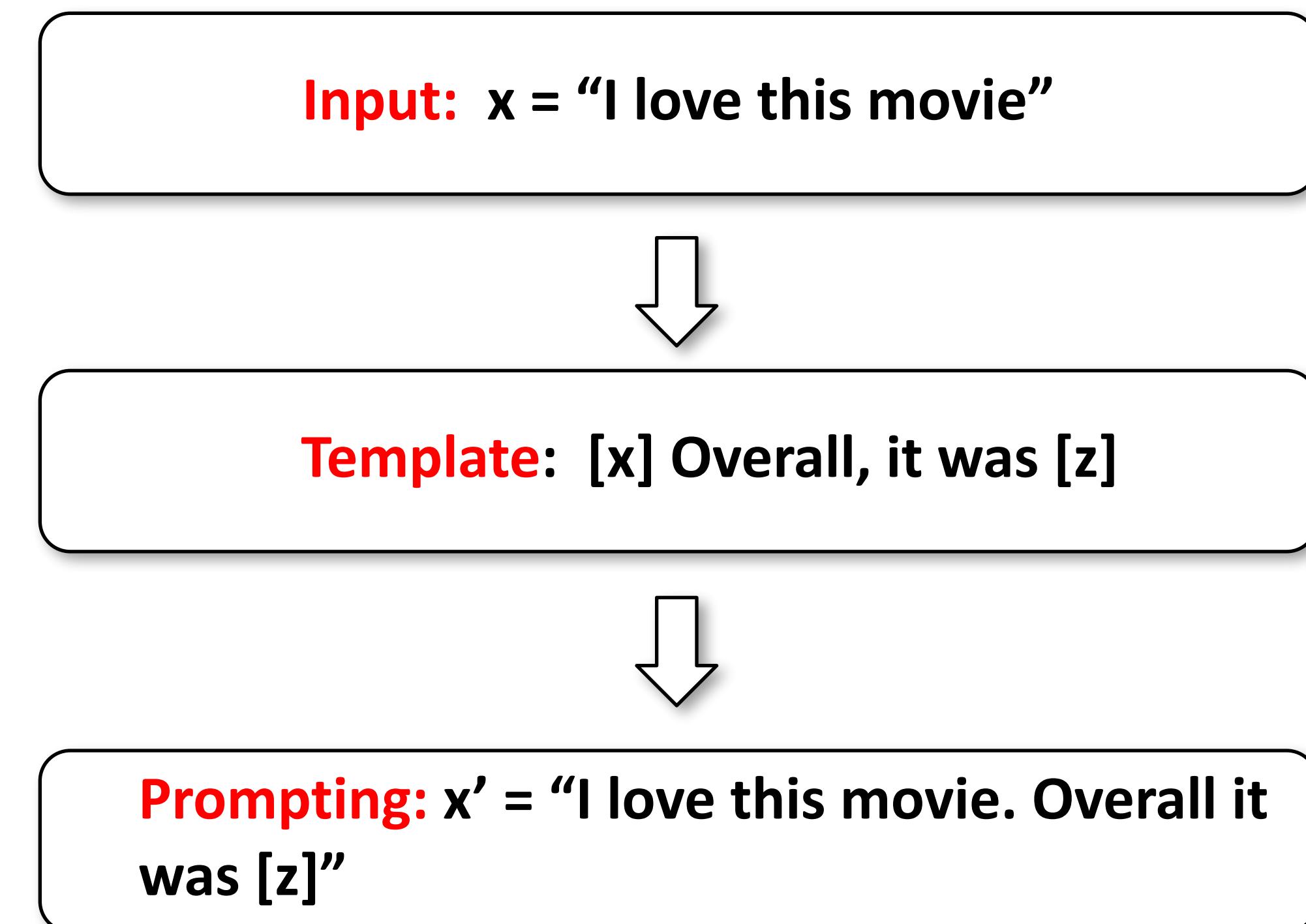
lick the squirrel. It will also touch its nose to the squirrel on the tail and nose if it can.

Standard Prompting Workflow

- Fill a prompt template
- Predict the answer
- Post-process the answer

Prompt Templates

- A template where you fill in with an actual input



Chat Prompts

- Recently, many models are trained as chatbots
- Usually inputs are specified in OpenAI messages format

```
messages=[  
  {  
    "role": "system",  
    "content": "Please classify movie reviews as 'positive' or 'negative'."  
  },  
  {  
    "role": "user",  
    "content": "This movie is a banger."  
  },  
]
```

- Roles:
 - **“system”**: message provided to the system to influence behavior
 - **“user”**: message input by the user
 - **“assistant”**: message output by the system

Chat Prompts Behind the Scenes

- Behind the scenes, messages are converted to token strings

LLaMa

Sys| [INST]
| <<SYS>>
| You are an assistant that ...
| <</SYS>>
| [/INST]

User| [INST]This movie is great.[/INST]

Asst.| Positive.

Alpaca

Sys| ### Instruction:
| You are an assistant that ...

User| ### Instruction:
| This movie is great.

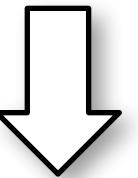
Asst.| ### Response:
| Positive.

- Software:** See LiteLLM Prompt Templates

Answer Prediction

- Given a prompt, predict the answer

Prompting: x' = “I love this movie. Overall it
was [z]”



Predicting: x' = “I love this movie. Overall it
was **fantastic**”

- Use any inference algorithms, as in generation class

Post-processing

- Based on the answer, select the actual output
- For instance:
 - Taking the output as-is
 - Formatting the output for easy visualization
 - Selecting only parts of the output that you want to use
 - Mapping the outputs to other actions

Output Formatting

Markdown Rendering

GN

You

Please write a table with the last 5 presidents and their birthdates.

ChatGPT

Here is a table with the last 5 U.S. Presidents and their birthdates:

President	Birthdate
Joe Biden	November 20, 1942
Donald Trump	June 14, 1946
Barack Obama	August 4, 1961
George W. Bush	July 6, 1946
Bill Clinton	August 19, 1946

Code

GN

You

Output that in JSON format.

ChatGPT

Here is the information in JSON format:

json

```
[  
  {"President": "Joe Biden", "Birthdate": "November 20, 1942"},  
  {"President": "Donald Trump", "Birthdate": "June 14, 1946"},  
  {"President": "Barack Obama", "Birthdate": "August 4, 1961"},  
  {"President": "George W. Bush", "Birthdate": "July 6, 1946"},  
  {"President": "Bill Clinton", "Birthdate": "August 19, 1946"}]  
` `` &#8203; `` [oaicite:0] `` &#8203;
```

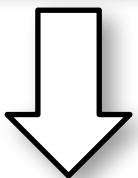
Copy code

- For user-facing applications, format in a pretty way

Output Selection

- From a longer response, select the information indicative of an answer

Predicting: x' = “I love this movie. Overall it was a movie that was simply fantastic”



Extraction: fantastic

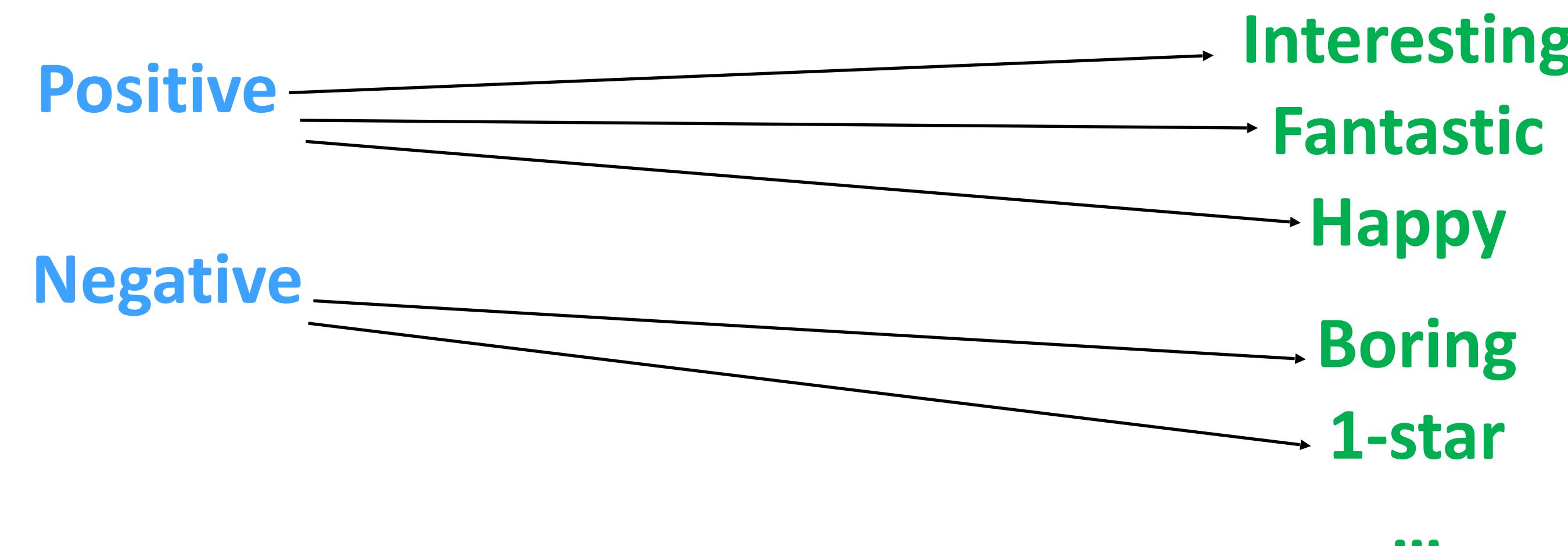
- Various methods for extraction
 - Classification:** identify keywords
 - Regression/numerical problems:** identify numbers
 - Code:** pull out code snippets in triple-backticks

Output Mapping

- Given an answer, map it into a class label or continuous value



- Often map many extracted words onto a single class



Few-shot Prompting / In-context Learning

Few-shot Prompting (Brown+ 2021)

- Provide a few examples of the task together with the instruction

Instruction | Please classify movie reviews as 'positive' or 'negative'.

Input: I really don't like this movie.
Output: negative

Examples

Input: This movie is great!
Output: positive

Few-shot Prompting w/ Chat Prompts (OpenAI Cookbook)

- For OpenAI models, add “role”: “system” and a “name”: “example_assistant” etc.

```
messages=[  
    {  
        "role": "system",  
        "content": "You are an assistant that translates corporate jargon into plain English."  
    },  
    {  
        "role": "system",  
        "name": "example_user",  
        "content": "New synergies will help drive top-line growth."  
    },  
    {  
        "role": "system",  
        "name": "example_assistant",  
        "content": "Things working well together will increase revenue."  
    },  
    ...  
    {  
        "role": "user",  
        "content": "This late pivot means we don't have time to boil the ocean for the client deliverable."  
    },  
]
```

LMs are Sensitive to Small Changes in In-context Examples

- Example ordering (Lu et al. 2021)
- Label balance (Zhang et al. 2022)

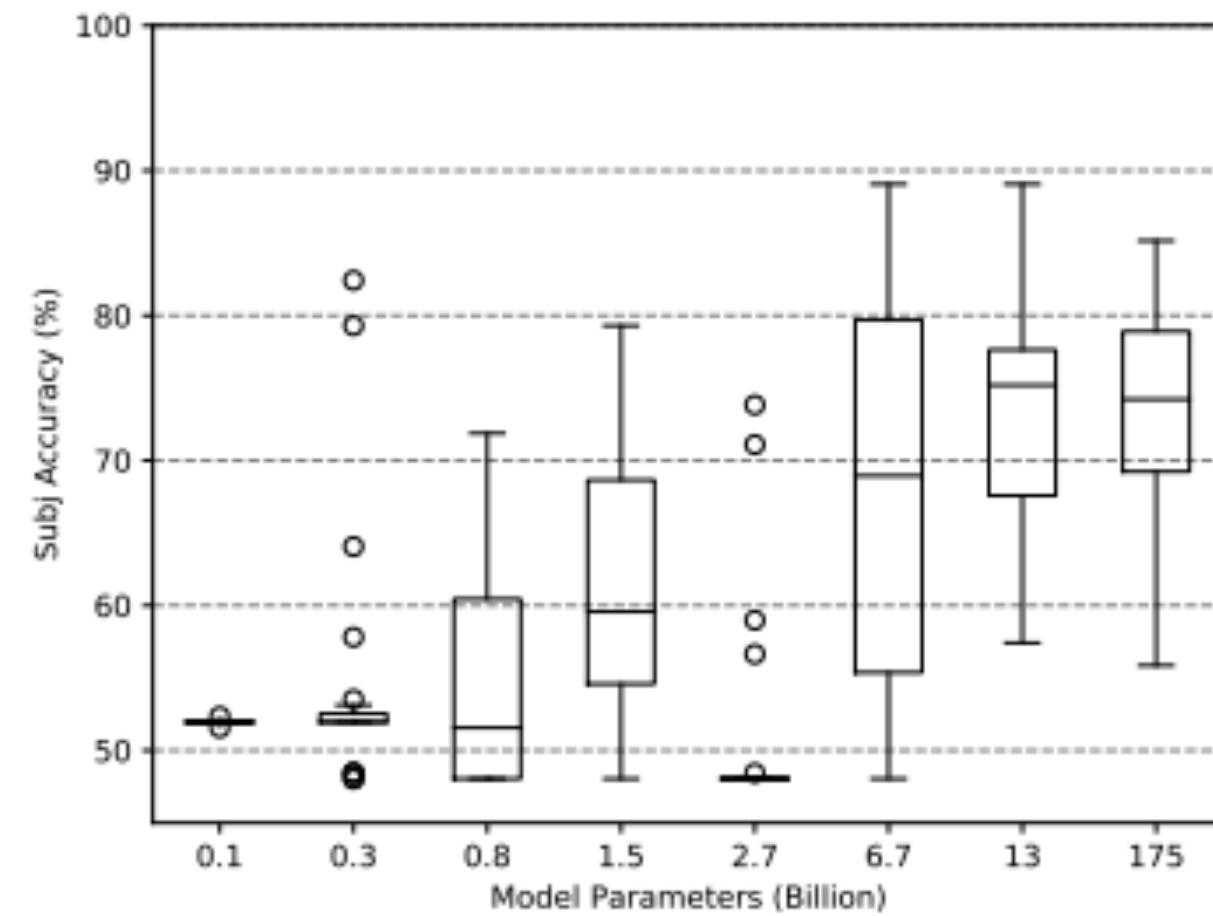
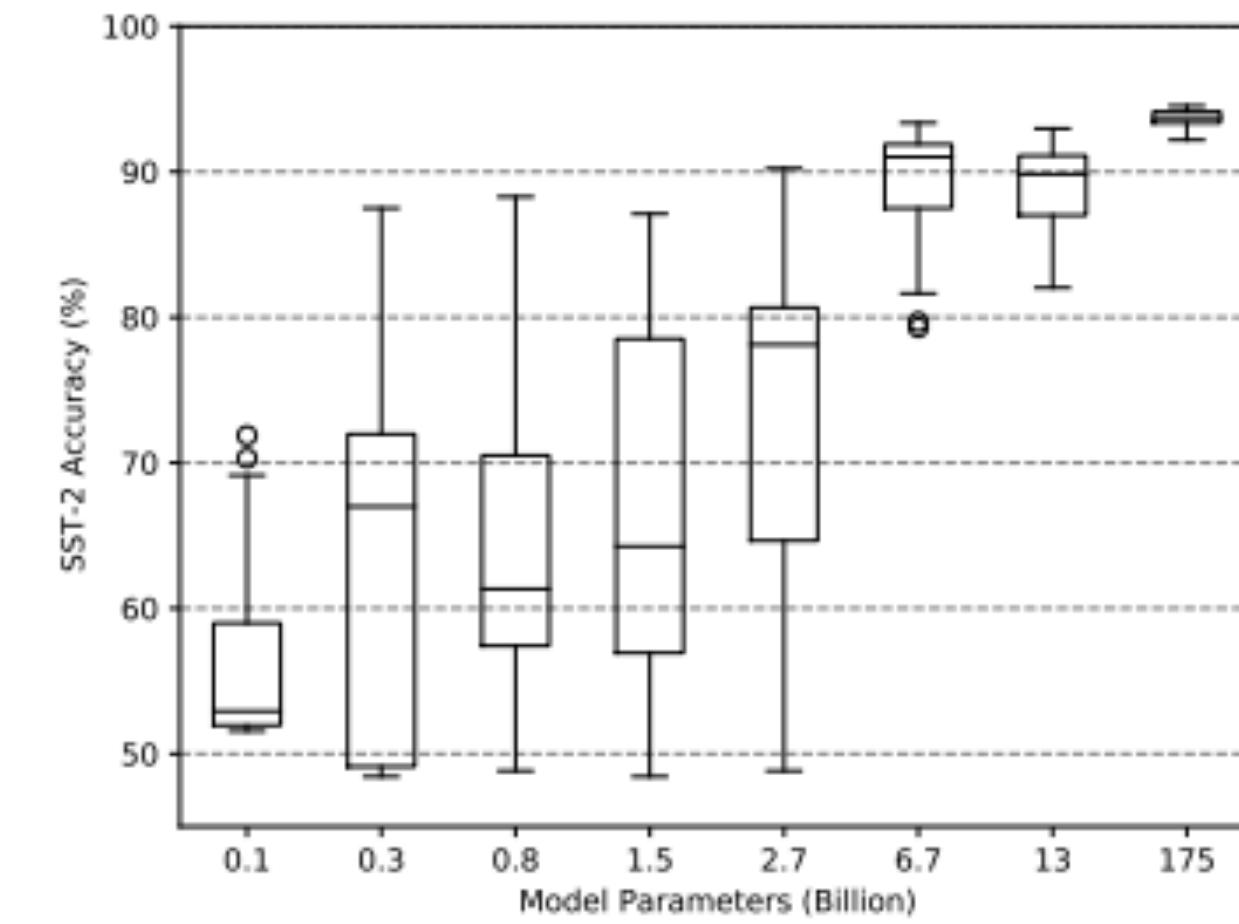
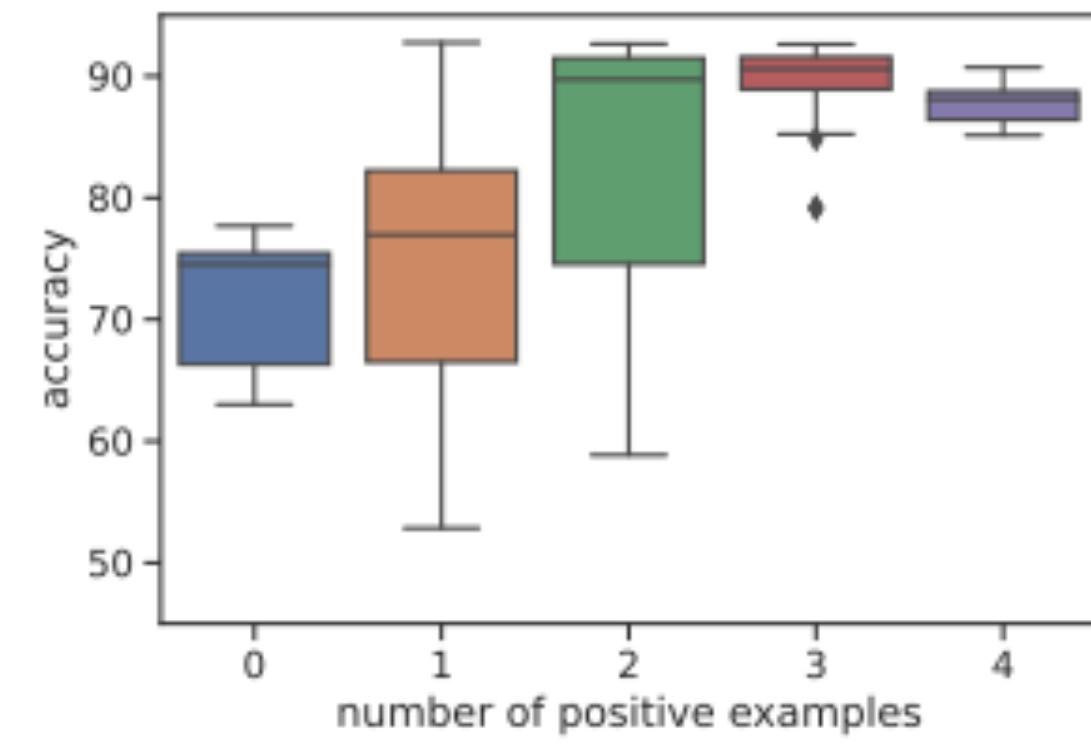
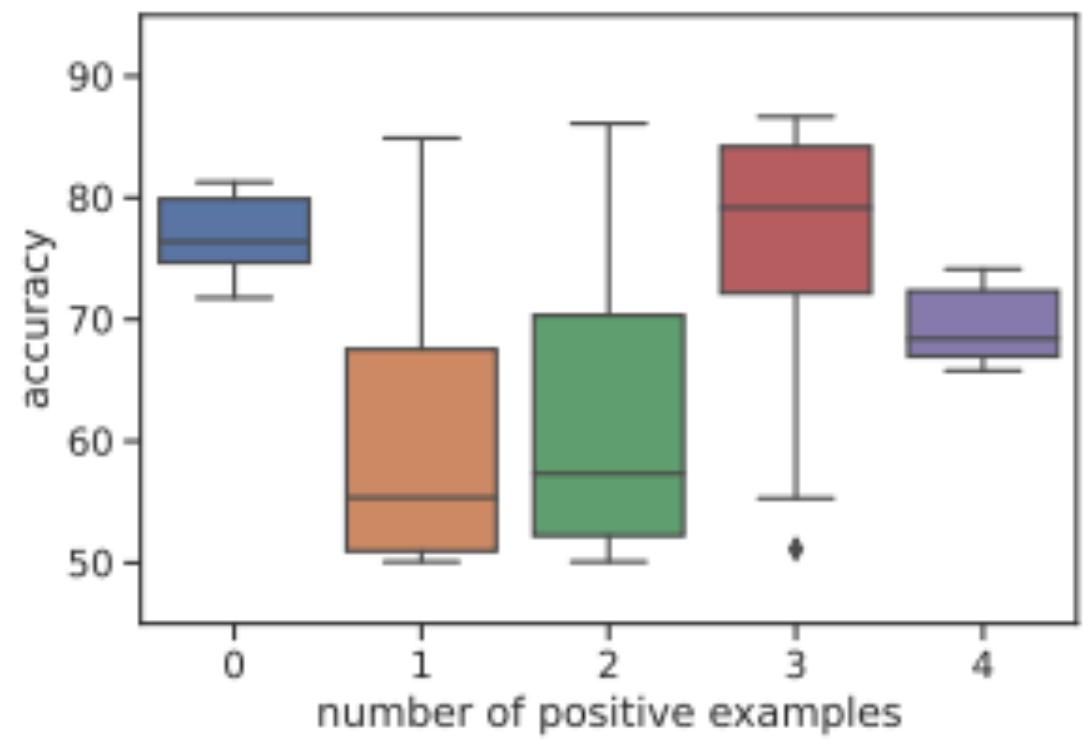


Figure 1: Four-shot performance for 24 different sample orders across different sizes of GPT-family models (GPT-2 and GPT-3) for the SST-2 and Subj datasets.

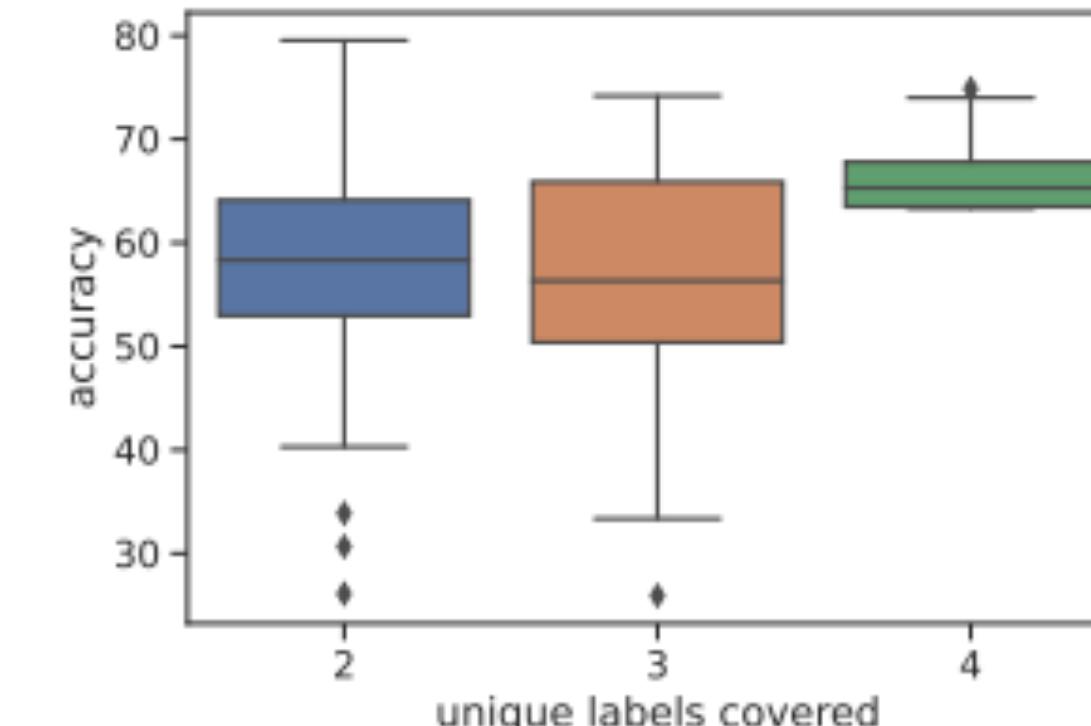


(a) Amazon

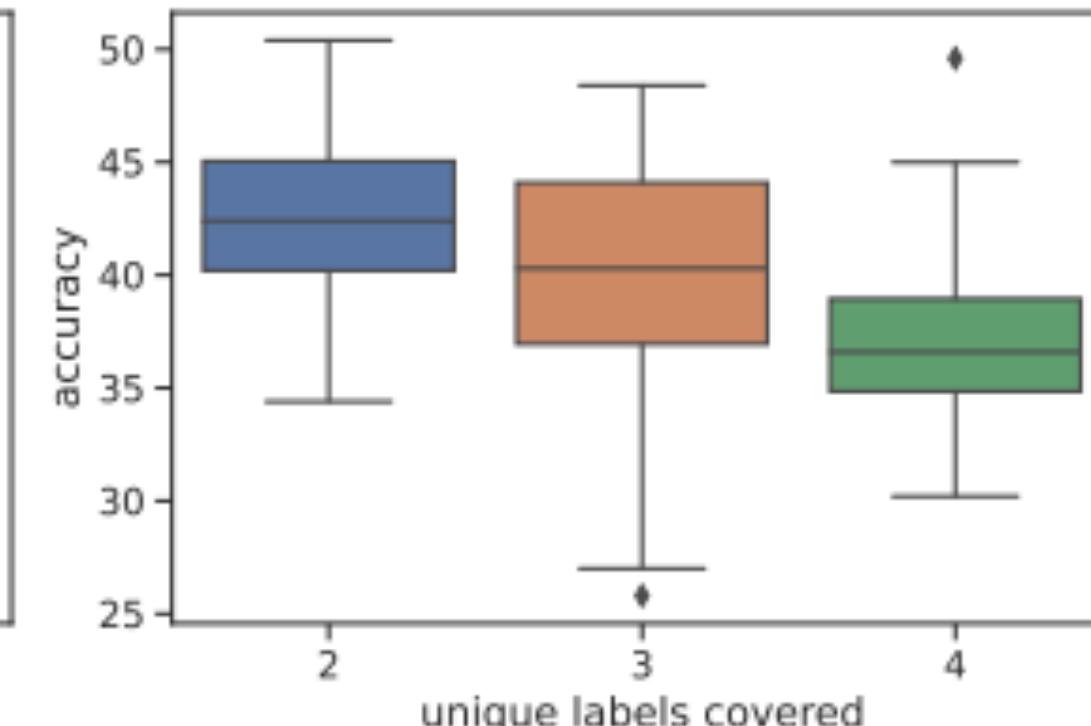


(b) SST-2

- Label coverage (Zhang et al. 2022)



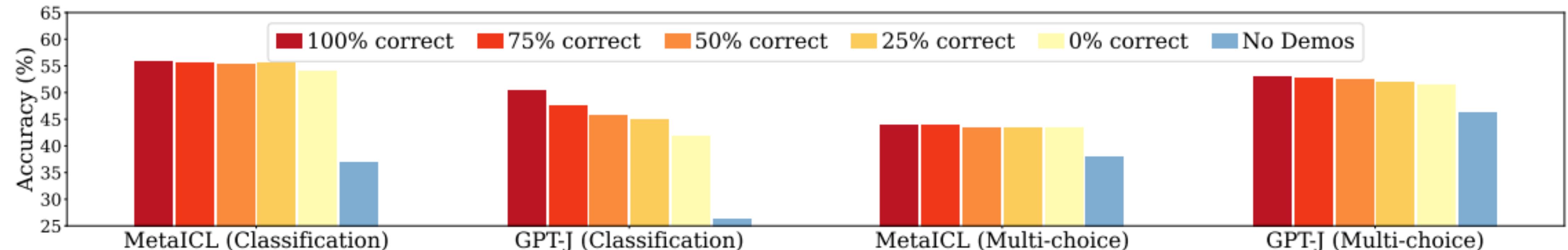
(a) AGNews



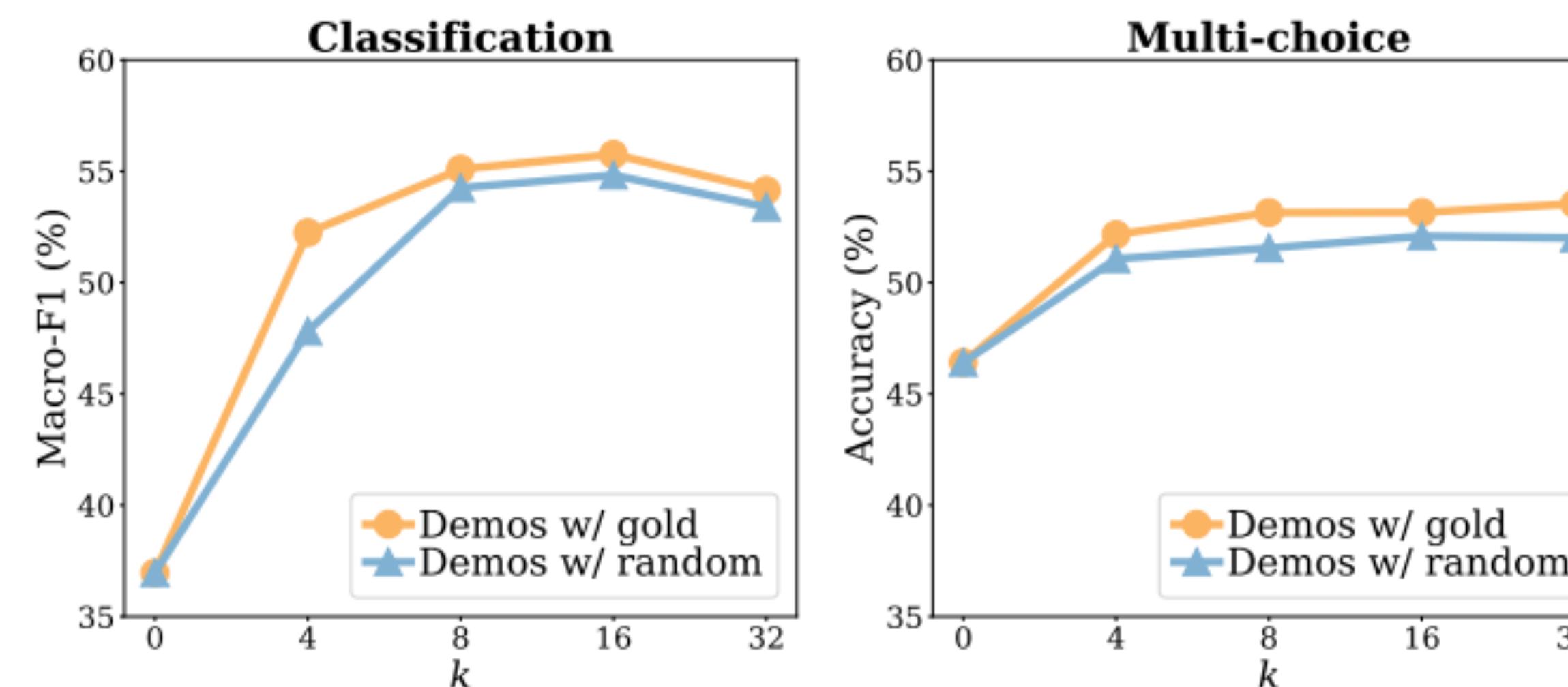
(b) TREC

But Effects are Sometimes Counter-intuitive (Min et al. 2022)

- Replacing correct labels with random labels sometimes barely hurts accuracy



- More demonstrations can sometimes hurt accuracy



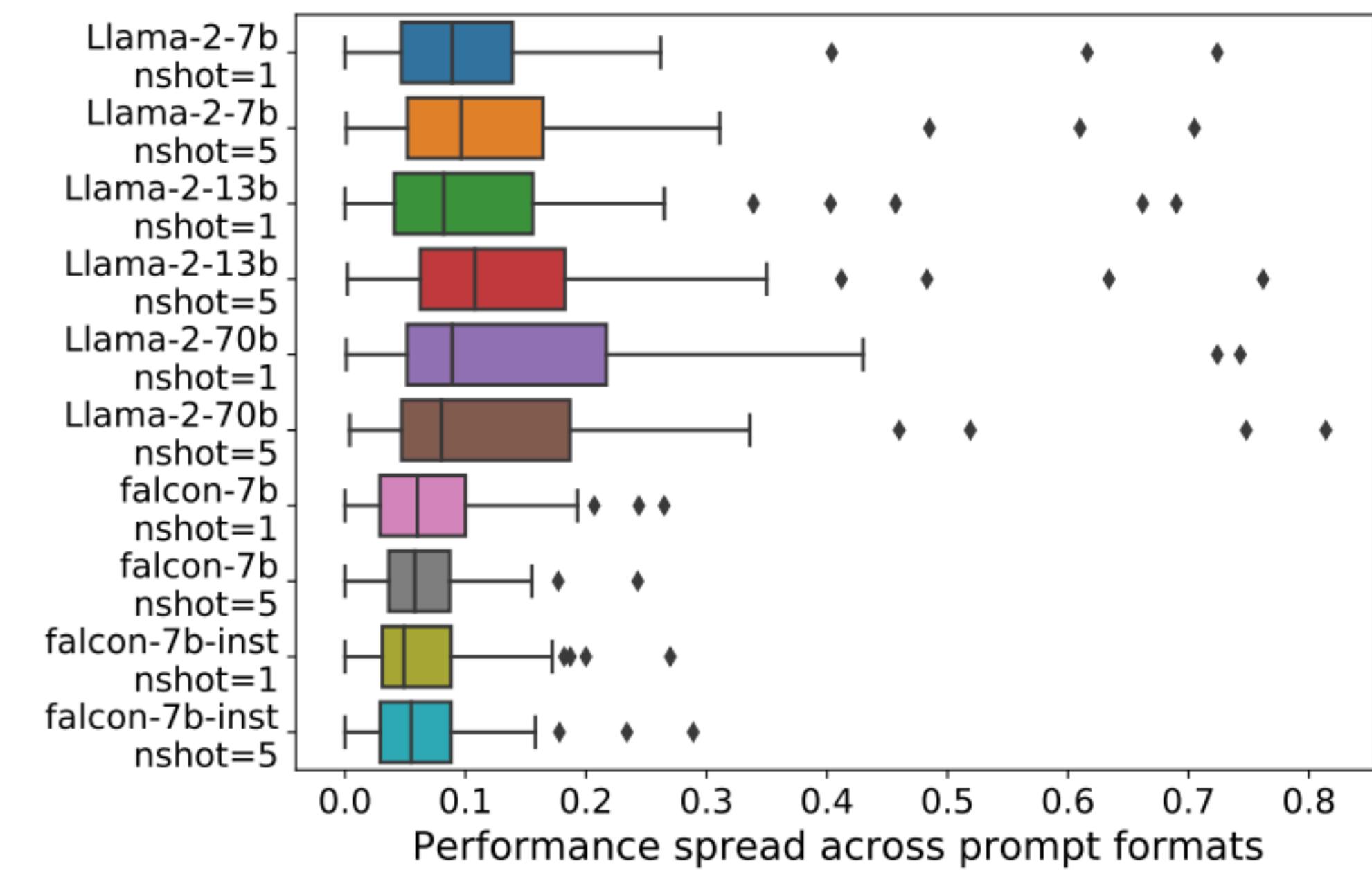
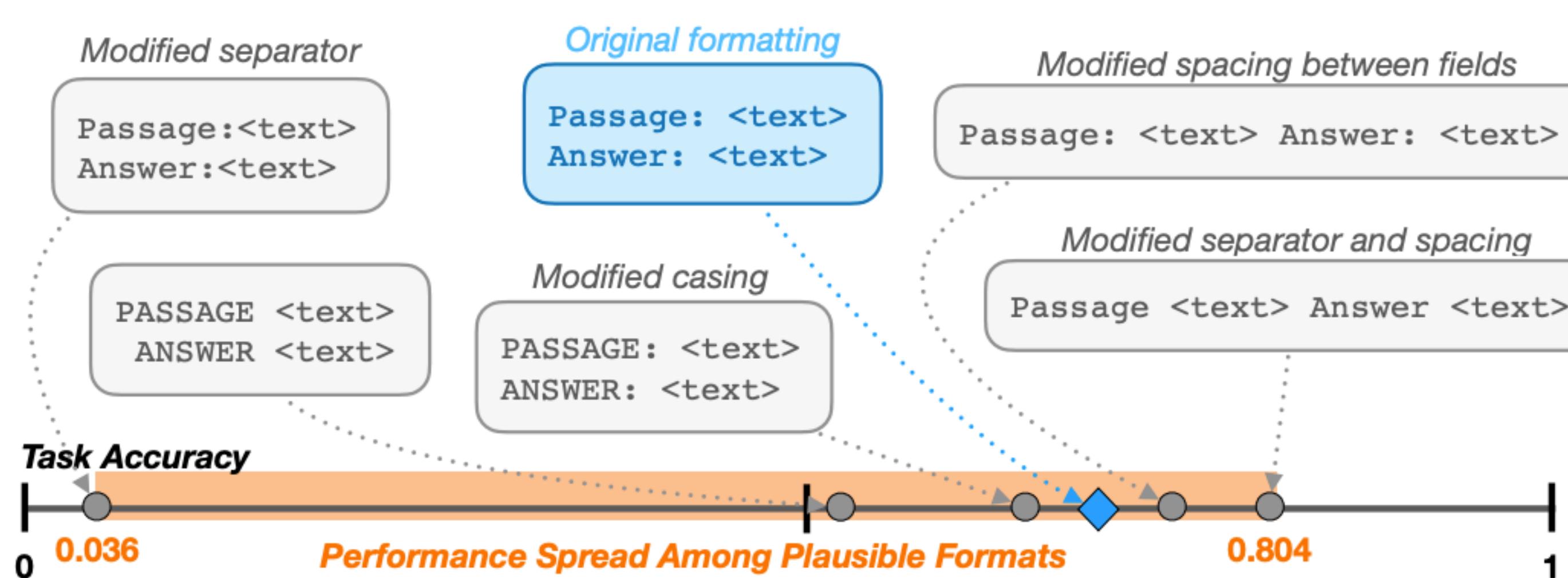
Prompt Engineering

Design of Prompts

- Manual
 - Configure a manual template based on the characteristics of the task
- Automated search
 - Search in discrete space
 - Search in continuous space

Manual Engineering: Format

- Make sure that the format matches that of a trained model (e.g. chat format)
- This can have a large effect on models! (Sclar et al. 2023)



Manual Engineering: Instructions

- Instructions should be clear, concise and easy to understand
- Good examples: <https://www.promptingguide.ai/introduction/tips>

Less Precise:

Explain the concept prompt engineering. Keep the explanation short, only a few sentences, and don't be too descriptive.

More Precise:

Use 2-3 sentences to explain the concept of prompt engineering to a high school student.

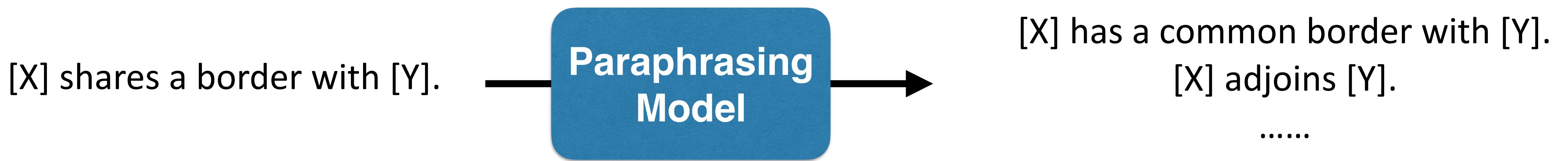
- Similar to humans, but (right now) LMs don't complain when you're vague

Methods for Automatic Prompt Engineering

- Prompt paraphrasing
- Gradient-based discrete prompt search
- Prompt tuning
- Prefix tuning

Prompt Paraphrasing

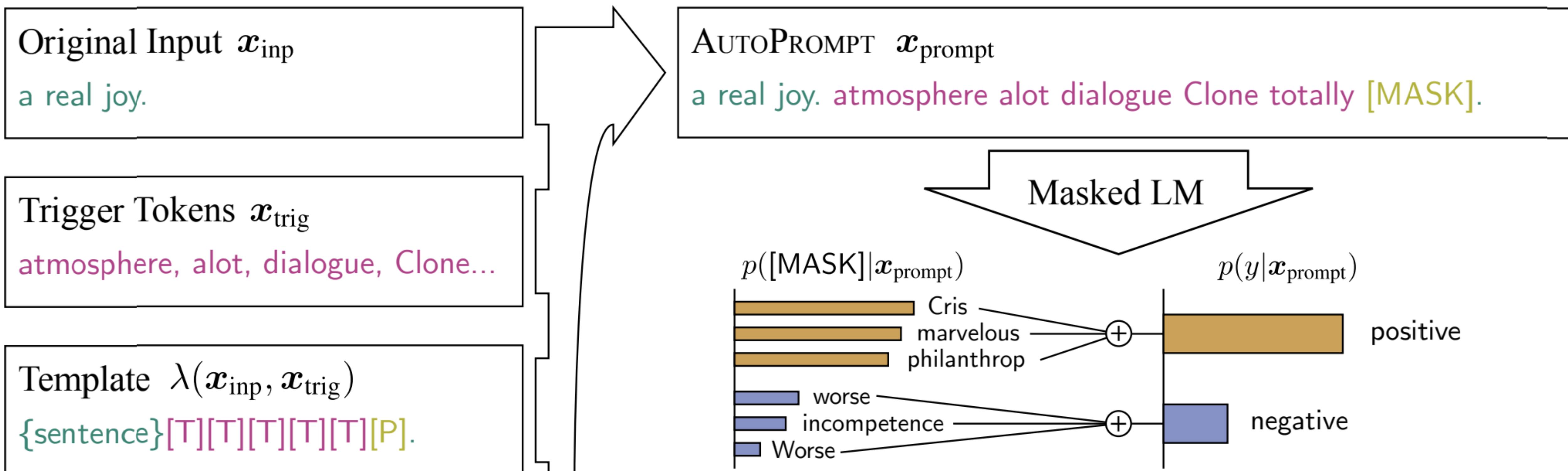
- Paraphrase an existing prompt to get other candidates (Jiang et al. 2019)



- Can be done through iterative paraphrasing (Zhou et al. 2021)

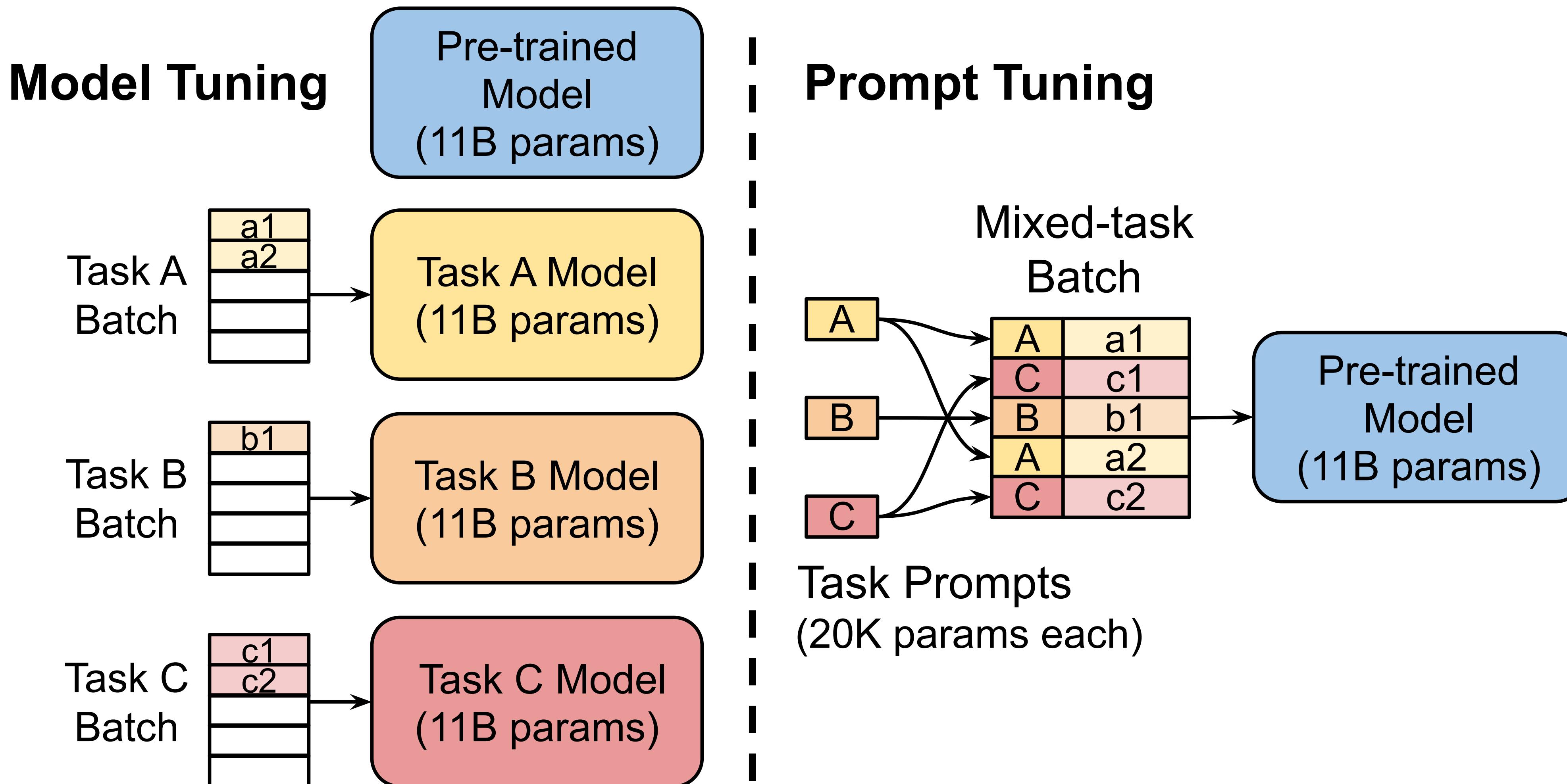
Gradient-based Search (Shin et al. 2020)

- Automatically optimize arbitrary prompts based on existing



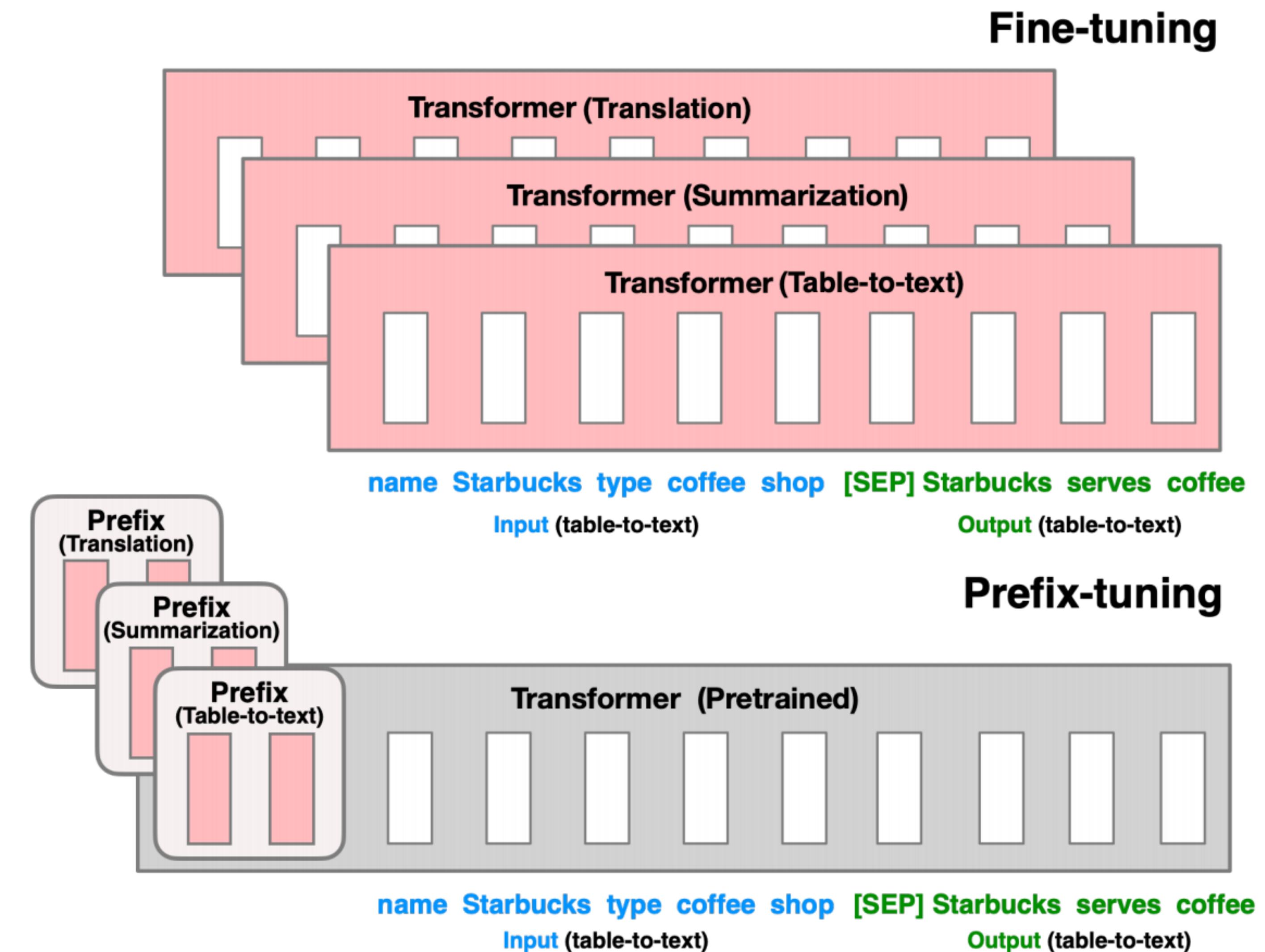
Prompt Tuning (Lester et al. 2021)

- Optimize the embeddings of a prompt, instead of the words.



Prefix Tuning (Li and Liang 2021)

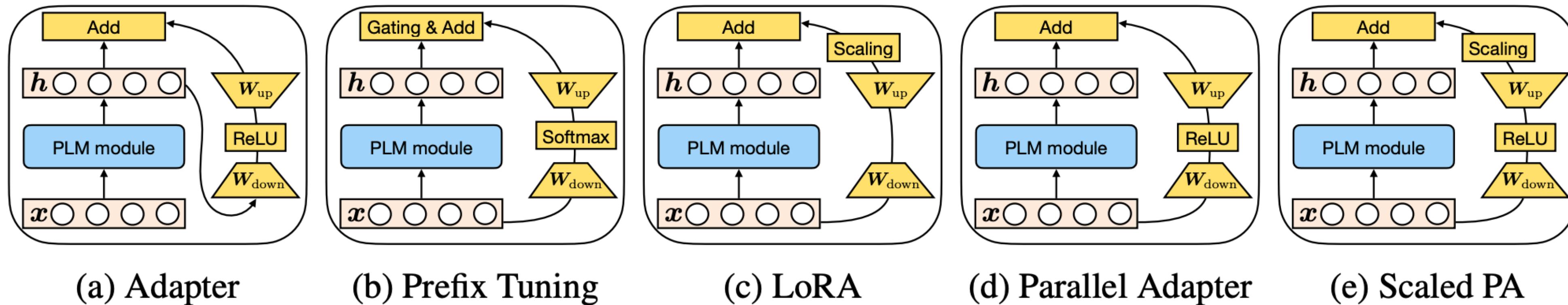
- "Prompt Tuning" optimizes only the embedding layer
- "Prefix Tuning" optimizes the prefix of all layers



Prompting and Fine Tuning

Connection to Other Training Methods

- We will be covering LoRA, Adapters, and other parameter-efficient training methods in a future class
- But prefix tuning is actually very similar, with a few design differences! (He et al. 2021)



(a) Adapter

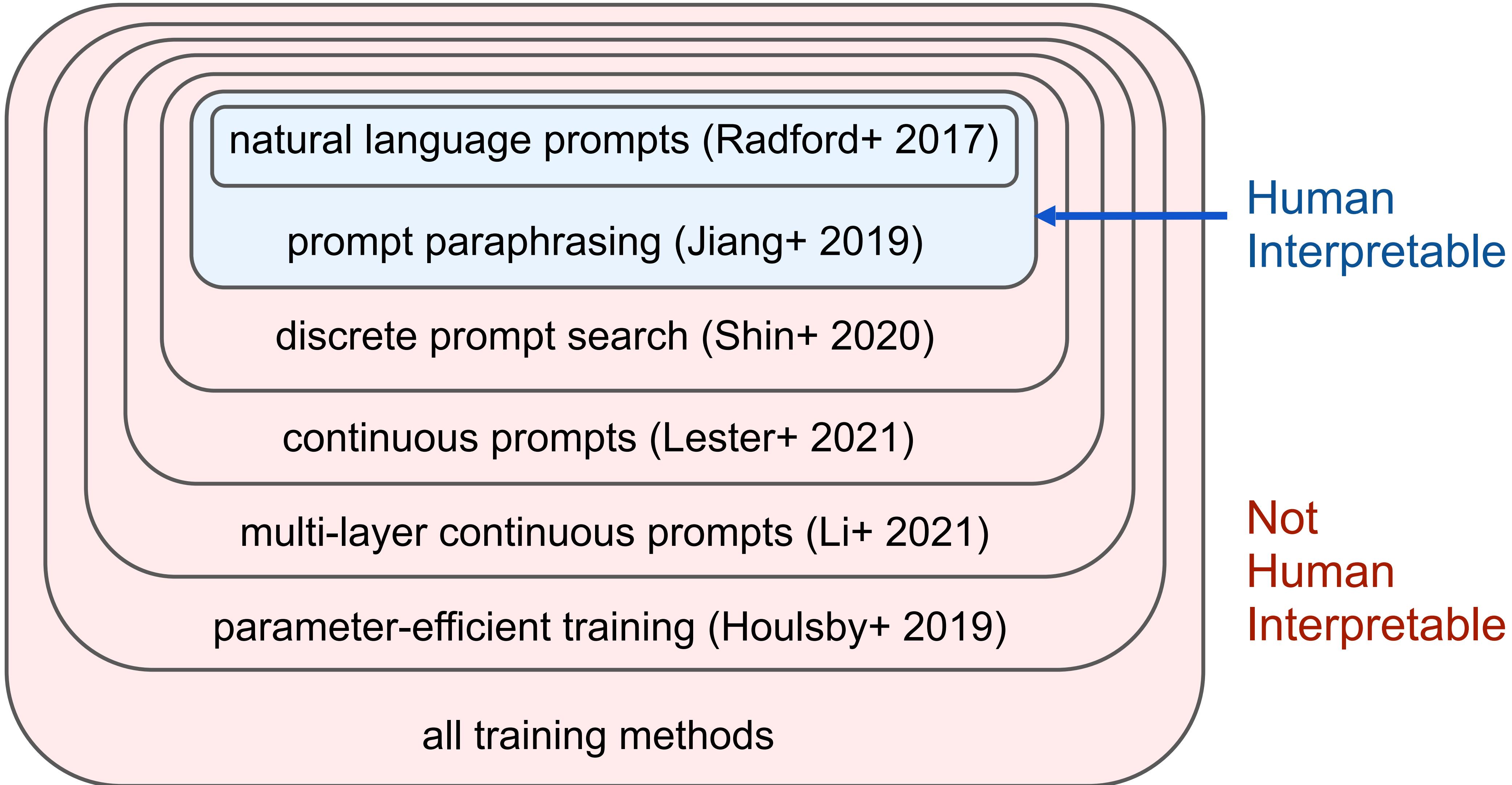
(b) Prefix Tuning

(c) LoRA

(d) Parallel Adapter

(e) Scaled PA

A Taxonomy of Prompting Methods



Prompting as a Prior

- Prompts are a human-interpretable method for providing a prior over the parameters of the model!
- This can be used in place of fine-tuning, or combined with fine-tuning (e.g. Schick and Schütze 2020)
 - More in future classes!

Reasoning in LMs

What is Reasoning?

- Using **evidence** and **logic** to arrive at conclusions and make judgments
- Can be
 - **Formal:** focusing on strict truth values
 - **Informal:** based on experience and intuition

Types of Reasoning

(examples: Huang and Chang 2023)

- Using **evidence** and **logic** to arrive at conclusions and make judgments (Huang and Chang 2023)

Deductive: Use logic to go from premise to firm conclusion.

Premise: All mammals have kidneys.

Premise: All whales are mammals.

Conclusion: All whales have kidneys.

Inductive: From observation, predict a likely conclusion.

Observation: When we see a creature with wings, it is usually a bird.

Observation: We see a creature with wings.

Conclusion: The creature is likely to be a bird.

Abductive: From observation, predict the most likely explanation.

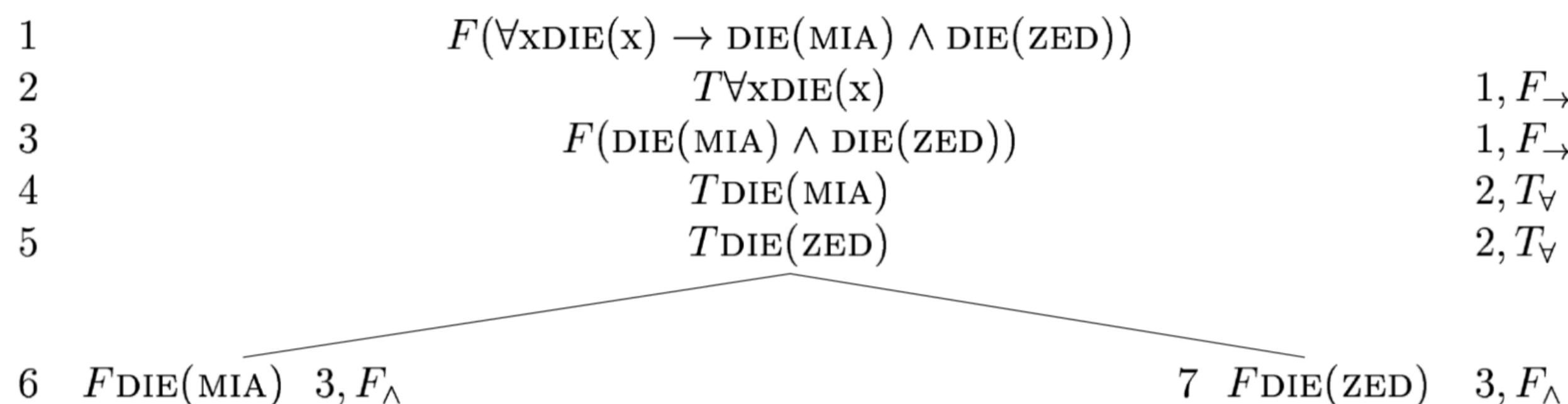
Observation: The car cannot start and there is a puddle of liquid under the engine.

Likely Explanation: The car has a leak in the radiator

Symbolic Reasoning

- Reasoning is something that traditional semantic representations are good at! (e.g. Prolog)

$$\forall x \text{DIE}(x) \rightarrow \text{DIE(MIA)} \wedge \text{DIE(ZED)}$$



- See "Representation and Inference for Natural Language" (Blackburn & Bos 1999)

Chain-of-thought Prompting

Chain of Thought Prompting (Wei et al. 2022)

- Get the model to explain its reasoning before making an answer

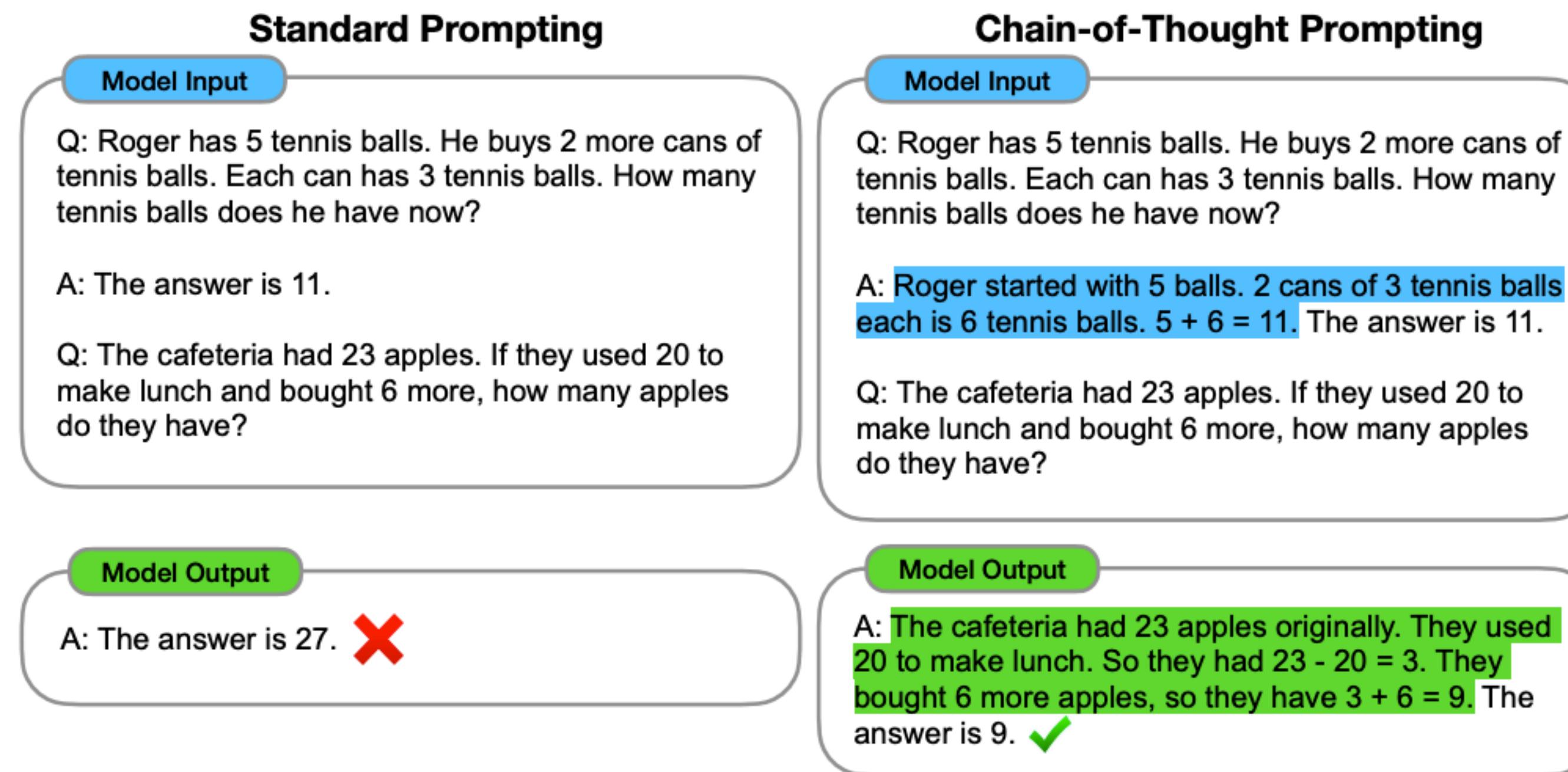


Figure 1: Chain-of-thought prompting enables large language models to tackle complex arithmetic, commonsense, and symbolic reasoning tasks. Chain-of-thought reasoning processes are highlighted.

- Provides the model with adaptive computation time

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?

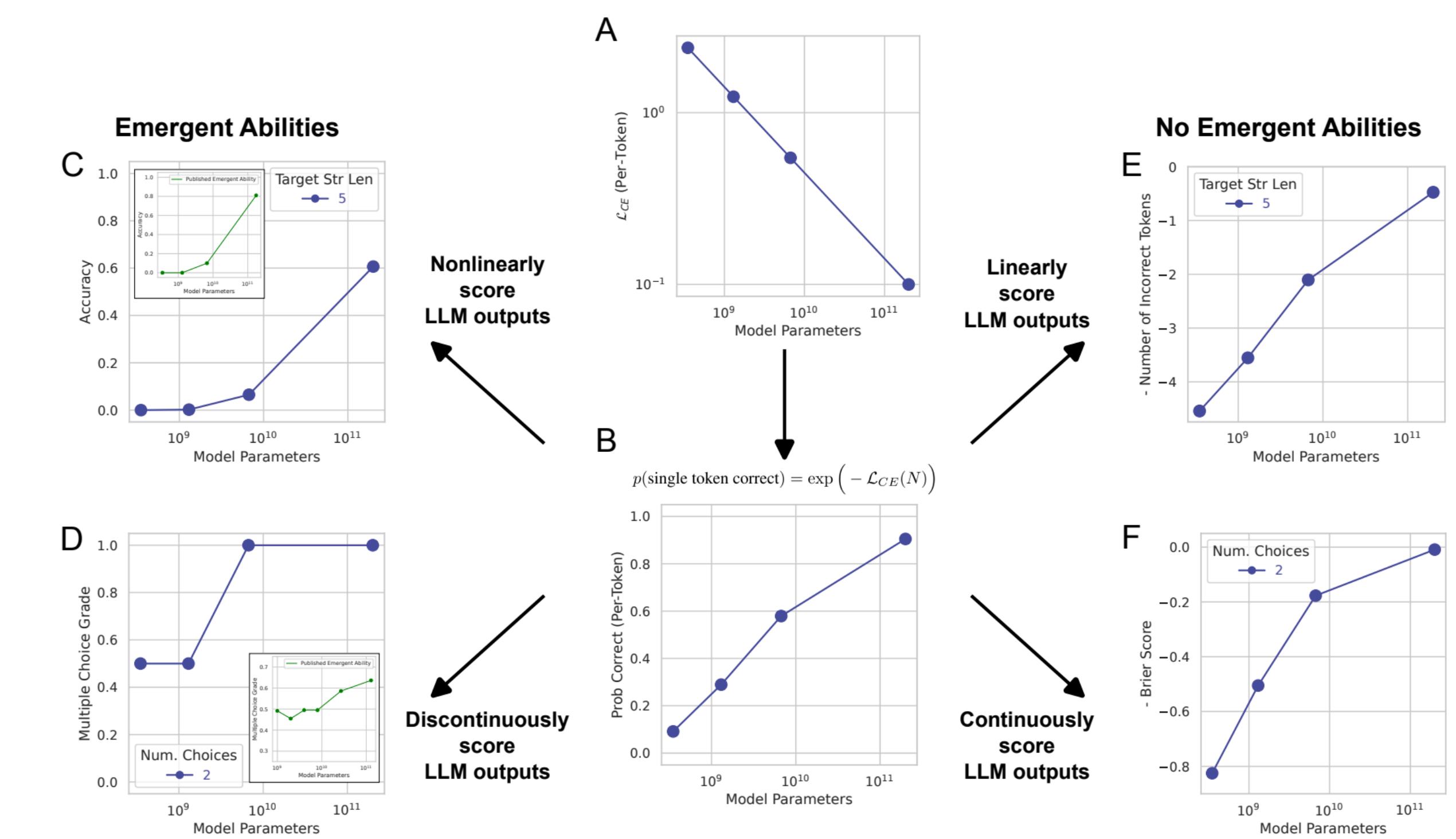
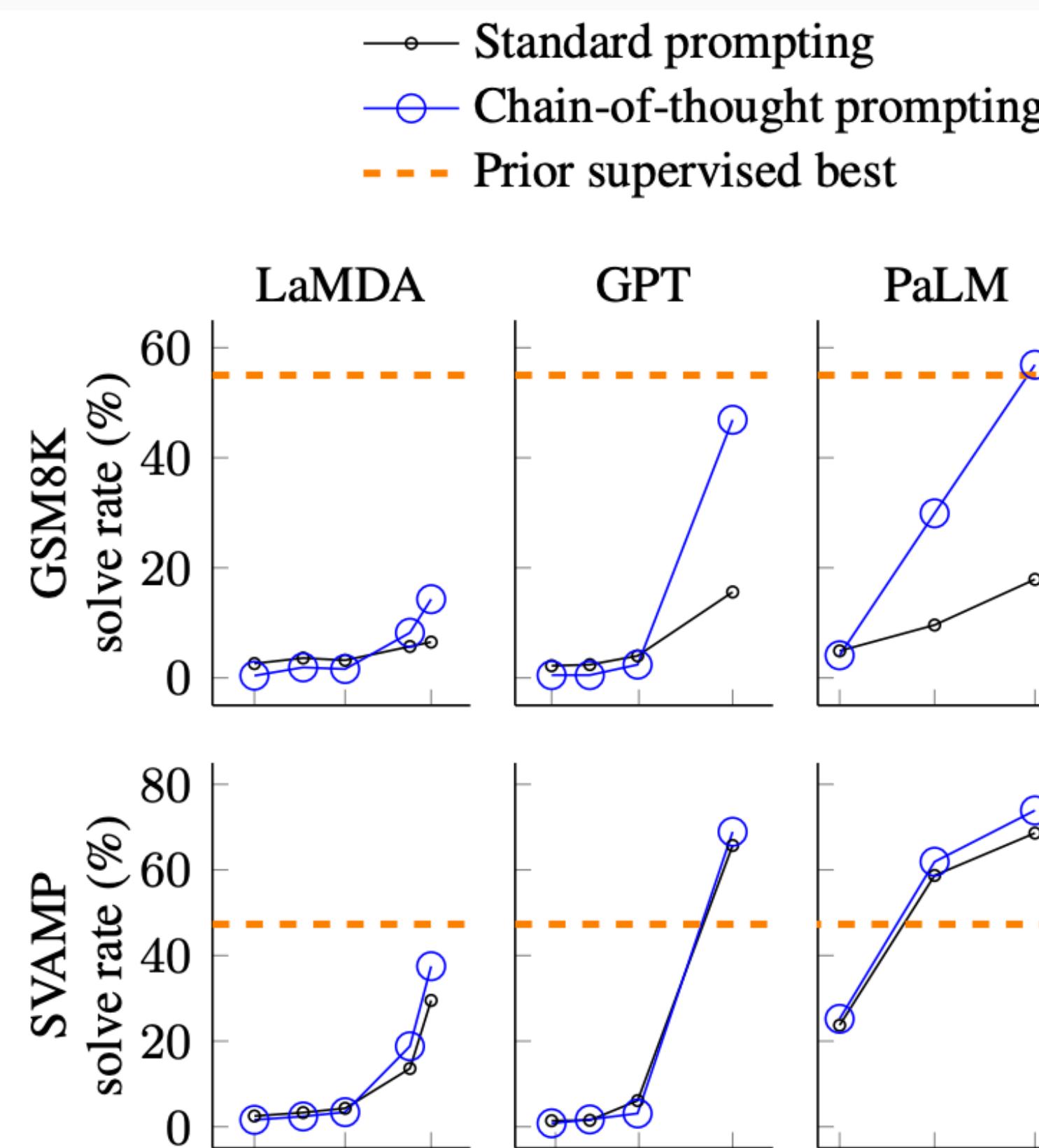
A: **Let's think step by step.**

(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 reason even w/o specific instructions now (probably due to instruction tuning)

Reasoning is an “Emergent” Ability (Wei et al. 2022)

- Emergent abilities — only appear when models are very large
- Note: emergent abilities are somewhat an artifact of how we measure accuracy (Schaeffer et al. 2023)



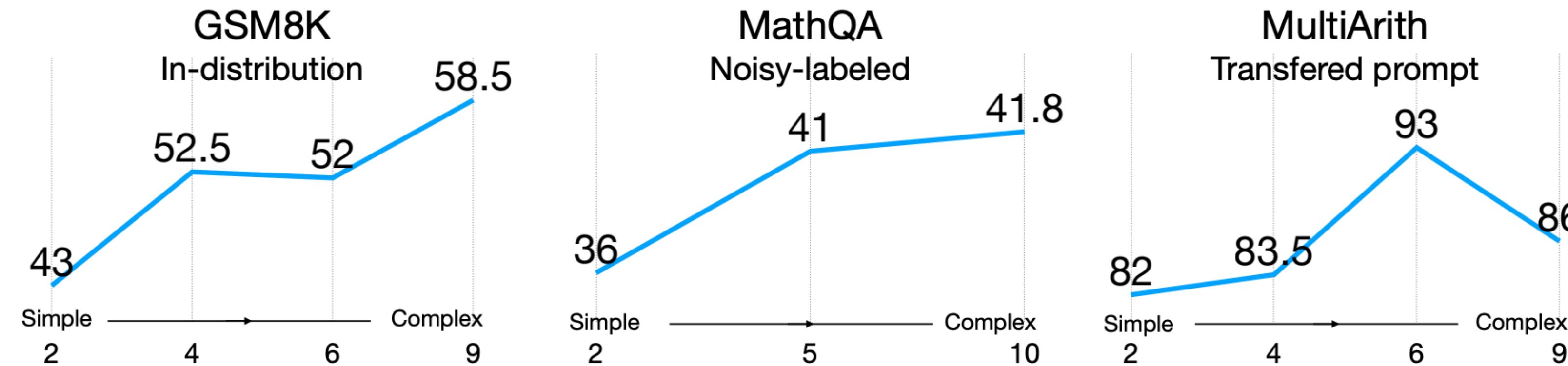
Reliability of Explanations (Ye and Durrett 2022)

- Examine the relationship between
 - accuracy of the answer
 - factuality of the explanation
 - consistency of the answer+explanation
- E-P: explain then predict (CoT)
- P-E: predict then explain

		Acc	Fac	Con	Acc=Fac	Acc=Con
<i>reliability of explanations generated by InstructGPT</i>						
InstructGPT	SYNTH (E-P)	58.4	72.8	64.8	66.5	68.8
	SYNTH (P-E)	54.8	51.6	95.2	89.6	57.2
	ADVHP (E-P)	62.0	79.6	91.2	80.0	68.4
	ADVHP (P-E)	54.0	69.2	82.0	77.6	67.2
	E-SNLI (P-E)	62.0	—	98.8	—	62.0
<i>reliability of explanations generated by other LLMs on SYNTH</i>						
OPT (175B)	SYNTH (E-P)	30.0	77.2	47.2	45.6	58.8
	SYNTH (P-E)	39.6	64.0	81.2	69.2	49.6
GPT-3	SYNTH (E-P)	46.8	59.2	64.8	66.8	61.2
	SYNTH (P-E)	52.4	52.4	83.2	78.4	58.0
text-davinci-002	SYNTH (E-P)	86.0	91.6	85.2	91.2	84.8
	SYNTH (P-E)	81.6	83.2	96.4	95.8	82.8

Complexity-based Prompting (Fu et al. 2022)

- Interestingly, for some tasks a larger number of reasoning steps is indicative of improved accuracy

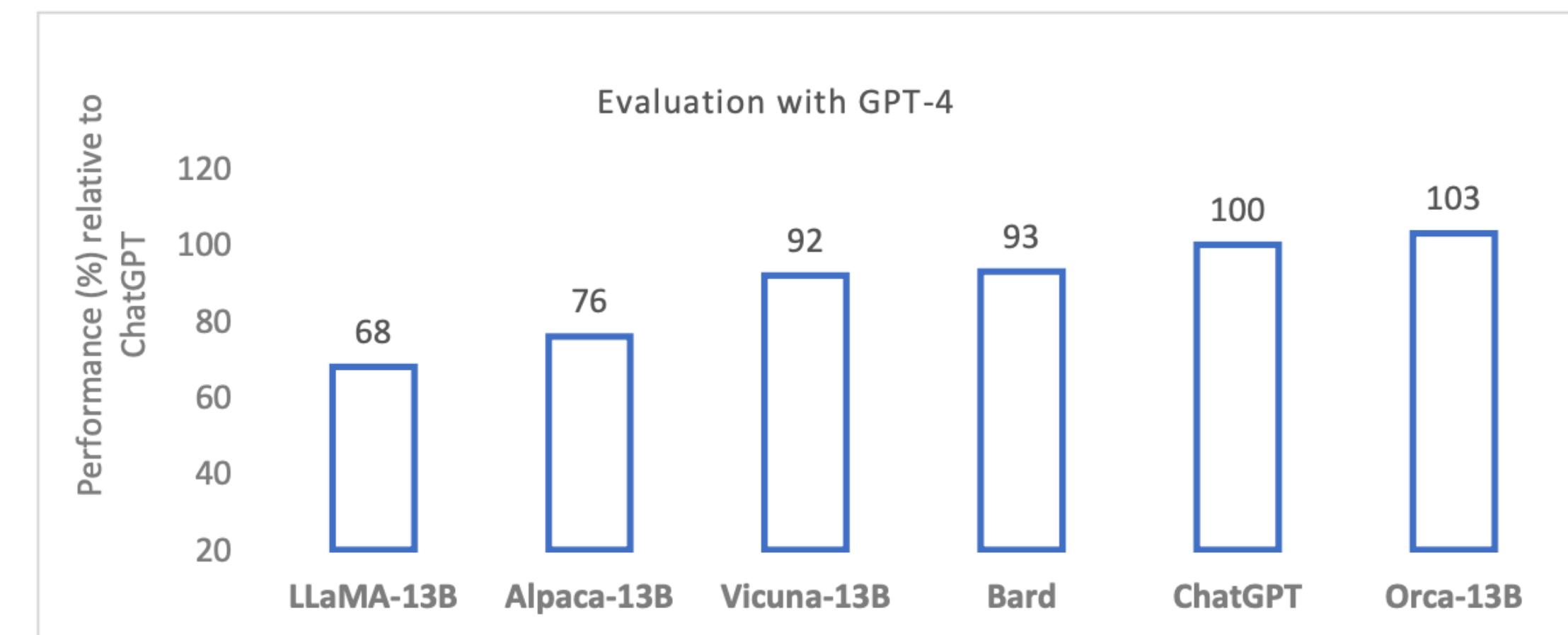


- This can be used to simply improve accuracy:
 - Sample multiple reasoning paths
 - Perform self-consistency over the longer reasoning paths

Training for Chain of Thought

ORCA: Training Small Models for Reasoning (Mukherjee et al. 2024)

- Generates a large and diverse chain-of-thought dataset from GPT-3.5 and GPT-4
- 5M complex instructions + CoT explanations
- Corresponding high accuracy compared to other datasets



- Replicated in OpenORCA dataset

Chain of Thought Reward Models (Lightman et al. 2023)

- Get human supervision on the steps
- Train reward model, give feedback on each sentence

The denominator of a fraction is 7 less than 3 times the numerator. If the fraction is equivalent to $\frac{2}{5}$, what is the numerator of the fraction? (Answer:

   Let's call the numerator x.

   So the denominator is $3x - 7$.

   We know that $x/(3x - 7) = \frac{2}{5}$.

   So $5x = 2(3x - 7)$.

   $5x = 6x - 14$.

   So $x = 7$.

Let

$$x^8 + 3x^4 - 4 = p_1(x)p_2(x) \cdots p_k(x),$$

where each non-constant polynomial $p_i(x)$ is monic with integer coefficients, and cannot be factored further over the integers. Compute $p_1(1) + p_2(1) + \cdots + p_k(1)$.

I notice that the given polynomial has even degree and only even powers of x , so I can try to make a substitution to simplify it.

Let $y = x^4$, then the polynomial becomes $y^2 + 3y - 4$, which is a quadratic equation.

I can factor this quadratic equation as $(y + 4)(y - 1)$, so the original polynomial is $(x^4 + 4)(x^4 - 1)$.

Now I need to factor each of these factors further, if possible, over the integers.

For the first factor, $x^4 + 4$, I recall the Sophie Germain identity, which says that $a^4 + 4b^4 = (a^2 + 2b^2 + 2ab)(a^2 + 2b^2 - 2ab)$.

Applying this identity with $a = x$ and $b = 1$, I get $x^4 + 4 = (x^2 + 2x + 2)(x^2 - 2x + 2)$.

For the second factor, $x^4 - 1$, I can use the difference of squares formula, which says that $a^2 - b^2 = (a + b)(a - b)$.

Applying this formula with $a = x^2$ and $b = 1$, I get $x^4 - 1 = (x^2 + 1)(x^2 - 1)$.

Then I can apply the difference of squares formula again with $a = x$ and $b = 1$, to get $x^2 - 1 = (x + 1)(x - 1)$.

So the original polynomial is $(x^4 + 4)(x^4 - 1) = (x^2 + 2x + 2)(x^2 - 2x + 2)(x^2 + 1)(x + 1)(x - 1)$.

These are all the irreducible factors over the integers, so $p_1(x) = x^2 + 2x + 2$, $p_2(x) = x^2 - 2x + 2$, $p_3(x) = x^2 + 1$, $p_4(x) = x + 1$, and $p_5(x) = x - 1$.

To find the sum of their values at $x = 1$, I just need to plug in $x = 1$ into each factor and add them up.

I get $p_1(1) = 1^2 + 2 \cdot 1 + 2 = 5$, $p_2(1) = 1^2 - 2 \cdot 1 + 2 = 1$, $p_3(1) = 1^2 + 1 = 2$, $p_4(1) = 1 + 1 = 2$, and $p_5(1) = 1 - 1 = 0$.

So the final answer is $5 + 1 + 2 + 2 + 0 = 10$.

Answer: 10

Let

$$x^8 + 3x^4 - 4 = p_1(x)p_2(x) \cdots p_k(x),$$

where each non-constant polynomial $p_i(x)$ is monic with integer coefficients, and cannot be factored further over the integers. Compute $p_1(1) + p_2(1) + \cdots + p_k(1)$.

This problem is about factoring a polynomial over the integers, which can be a tricky task.

One possible strategy is to look for rational roots of the polynomial, using the rational root theorem.

However, that would involve checking a lot of possible factors of -4 , and there is no guarantee that the polynomial has any rational roots at all.

Another possible strategy is to use some algebraic tricks to simplify the polynomial or reduce its degree.

I notice that the polynomial has only even powers of x , so I can try to substitute $y = x^2$ and see what happens.

Then the polynomial becomes $y^4 + 3y^2 - 4$, which is a quadratic equation in y^2 .

I can try to factor this quadratic equation by looking for two numbers that multiply to -4 and add to 3 .

Those numbers are 4 and -1 , so I can write $y^4 + 3y^2 - 4 = (y^2 + 4)(y^2 - 1)$.

Now I can substitute back $x^2 = y$ and get $x^8 + 3x^4 - 4 = (x^4 + 4)(x^4 - 1)$.

This looks promising, because I have reduced the degree of the factors from 8 to 4 .

I can try to factor each of these factors further, using the difference of squares formula.

I get $x^8 + 3x^4 - 4 = (x^2 + 2)(x^2 - 2)(x^2 + 1)(x^2 - 1)$.

I can apply the difference of squares formula again to the last factor and get $x^8 + 3x^4 - 4 = (x^2 + 2)(x^2 - 2)(x^2 + 1)(x + 1)(x - 1)$.

Now I have factored the polynomial completely into monic linear and quadratic factors with integer coefficients.

These are the $p_i(x)$'s that the problem is asking for.

To find the sum of their values at $x = 1$, I just need to plug in $x = 1$ into each factor and add them up.

I get $p_1(1) + p_2(1) + \cdots + p_k(1) = (1^2 + 2)(1^2 - 2)(1^2 + 1)(1 + 1)(1 - 1)$.

Simplifying, I get $p_1(1) + p_2(1) + \cdots + p_k(1) = (3)(-1)(2)(2)(0)$.

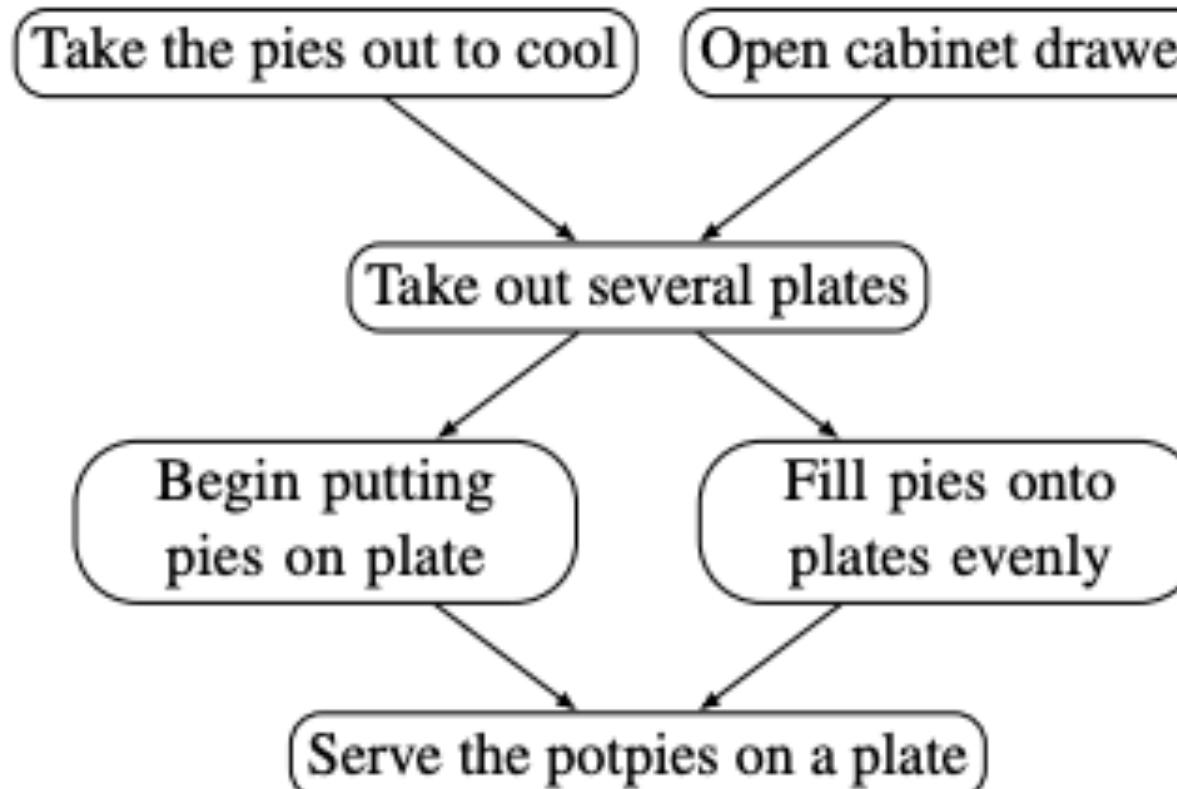
Multiplying, I get $p_1(1) + p_2(1) + \cdots + p_k(1) = 0$.

Answer: 0

Prompting and Programs

Structuring Outputs as Programs can Help (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 asking the LM to do so
- Especially useful for numeric questions
- See ChatGPT code interpreter, Bard code execution
- (More on agents/tools later)

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.

X

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

✓

Questions?