

CS11-711 Advanced NLP

# Prompting

Graham Neubig



**Carnegie Mellon University**

Language Technologies Institute

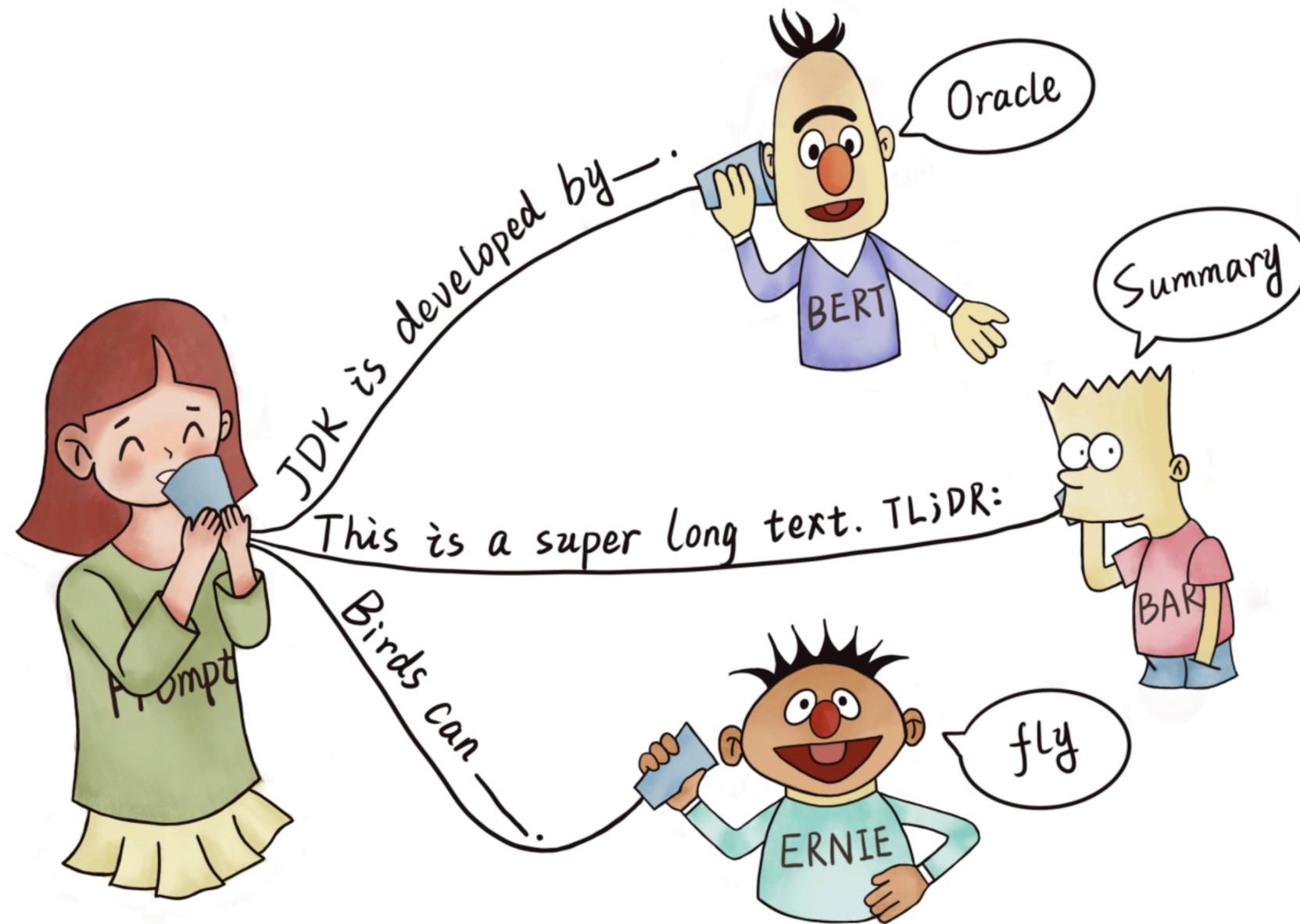
Site

<https://phontron.com/class/anlp2024/>

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 = **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.

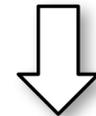
# Standard Prompting Workflow

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

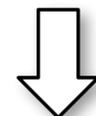
# Prompt Templates

- A template where you fill in with an actual input

**Input:**  $x = \text{"I love this movie"}$



**Template:**  $[x]$  Overall, it was  $[z]$



**Prompting:**  $x' = \text{"I love this movie. Overall it was [z]"}$

# 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

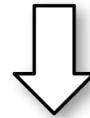
```
Sys. | ### Instruction:
      | You are an assistant that ...
User | ### Instruction:
      | This movie is great.
Asst.| ### Response:
      | Positive.
```

- **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

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

## Markdown Rendering

 **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

 **You**  
Output that in JSON format.

 **ChatGPT**  
Here is the information in JSON format:

```
json Copy code  
  
[  
  {"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;` `
```

# 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**

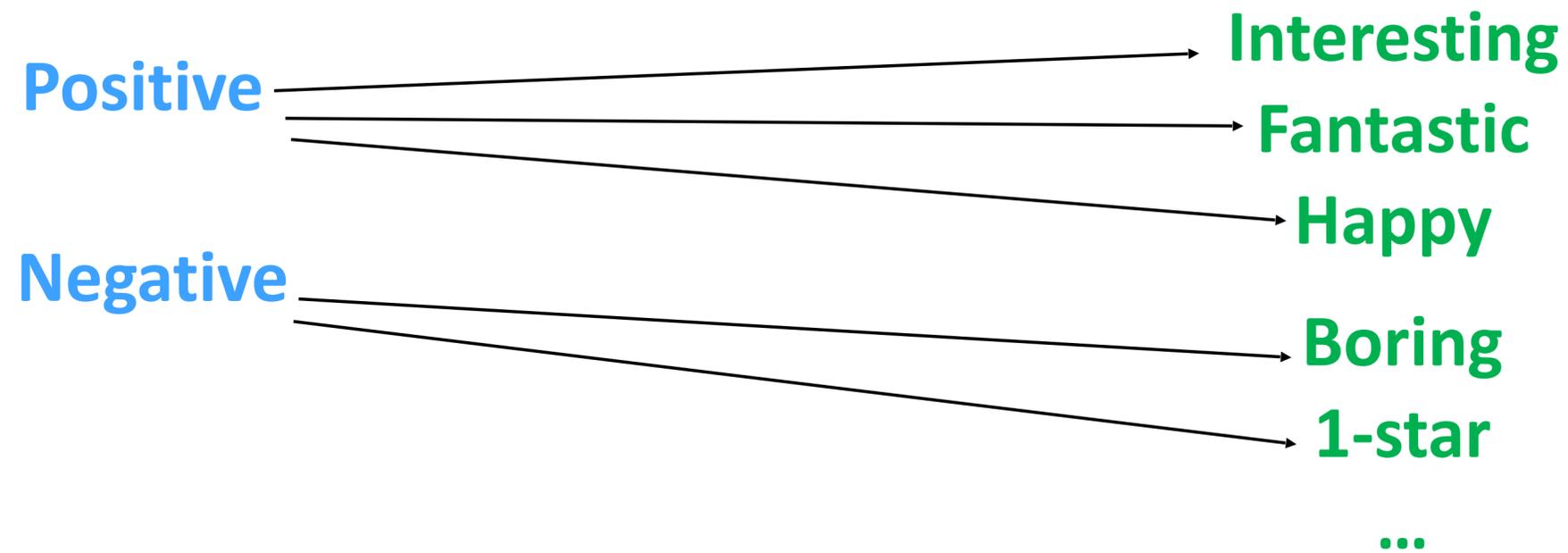
- Various methods for extraction
  - **Classification:** identify keywords
  - **Regression/numerical problems:** identify numbers
  - **Code:** pull out code snippets in triple-backticks

# 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)

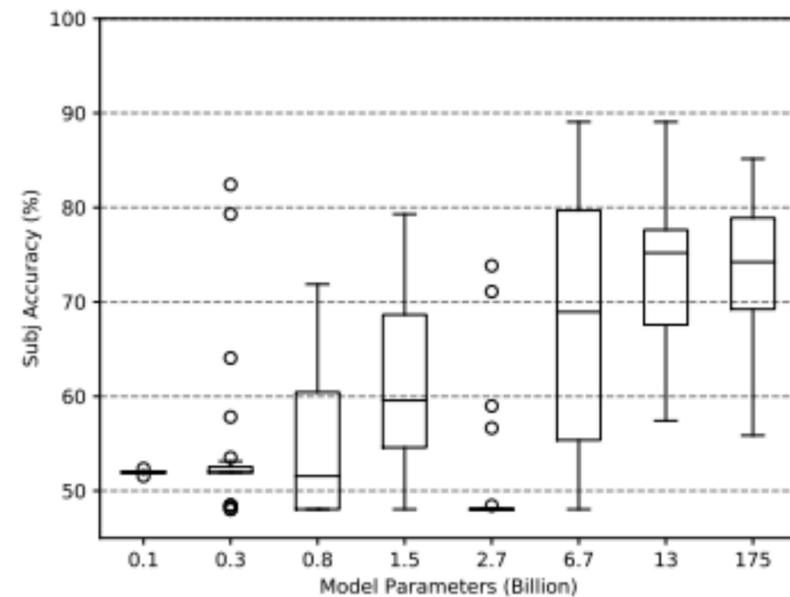
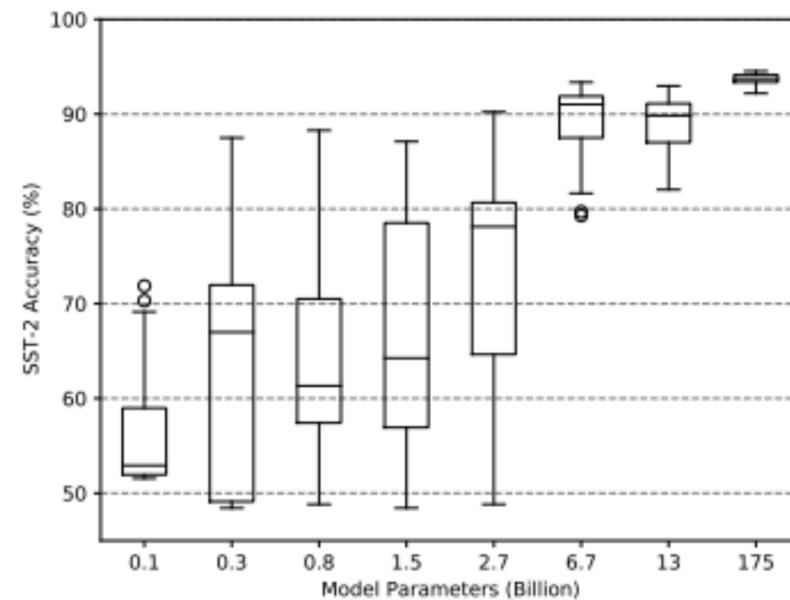
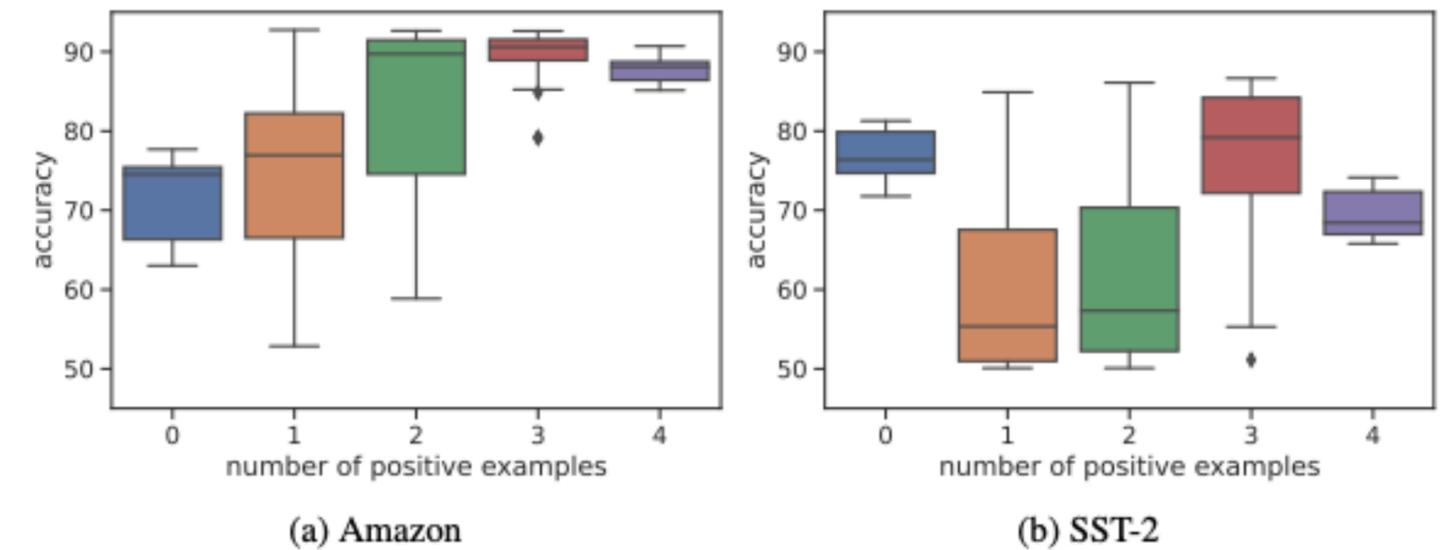
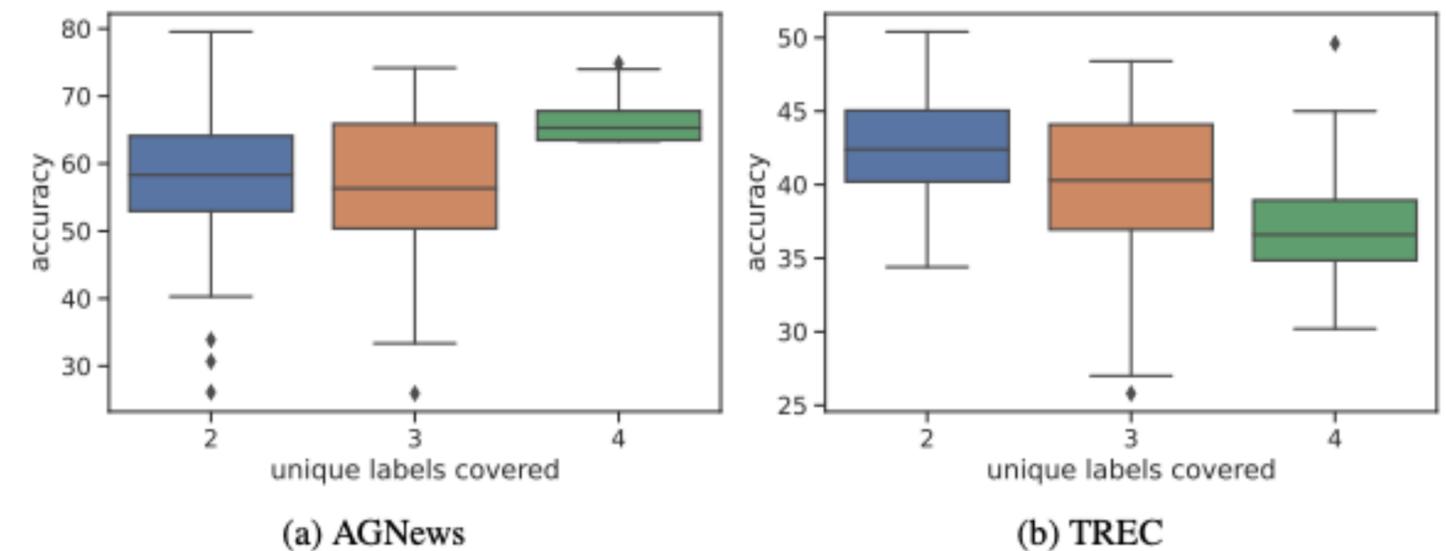


Figure 1: Four-shot performance for 24 different sample orders across different sizes of GPT-family models (GPT-2 and GPT-3) for the SST-2 and Subj datasets.

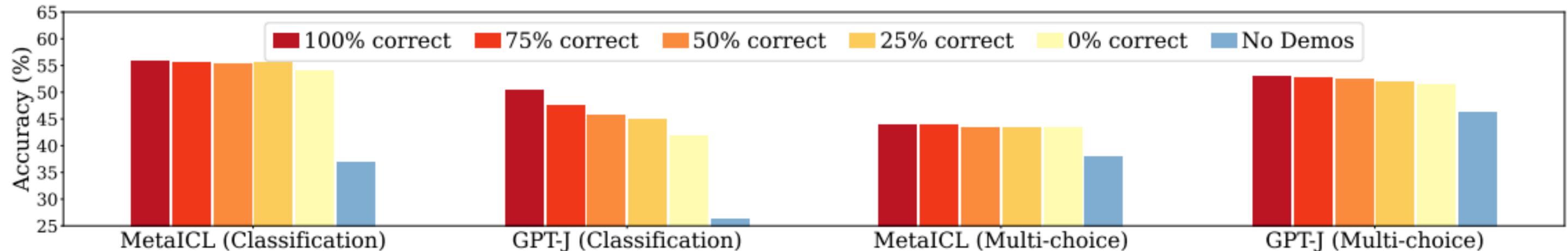


- Label coverage (Zhang et al. 2022)

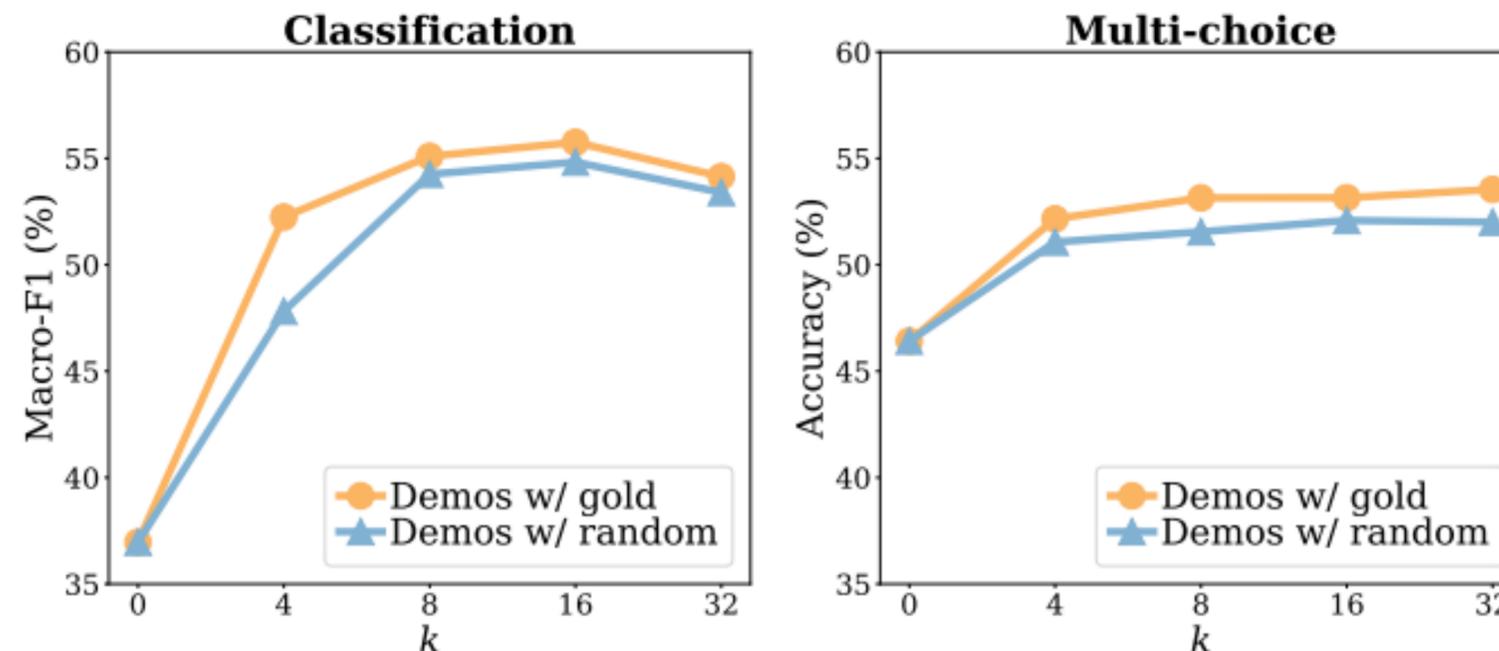


# But Effects are Sometimes Counter-intuitive (Min et al. 2022)

- Replacing correct labels with random labels sometimes barely hurts accuracy



- More demonstrations can sometimes hurt accuracy



# Chain-of-thought Prompting

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

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

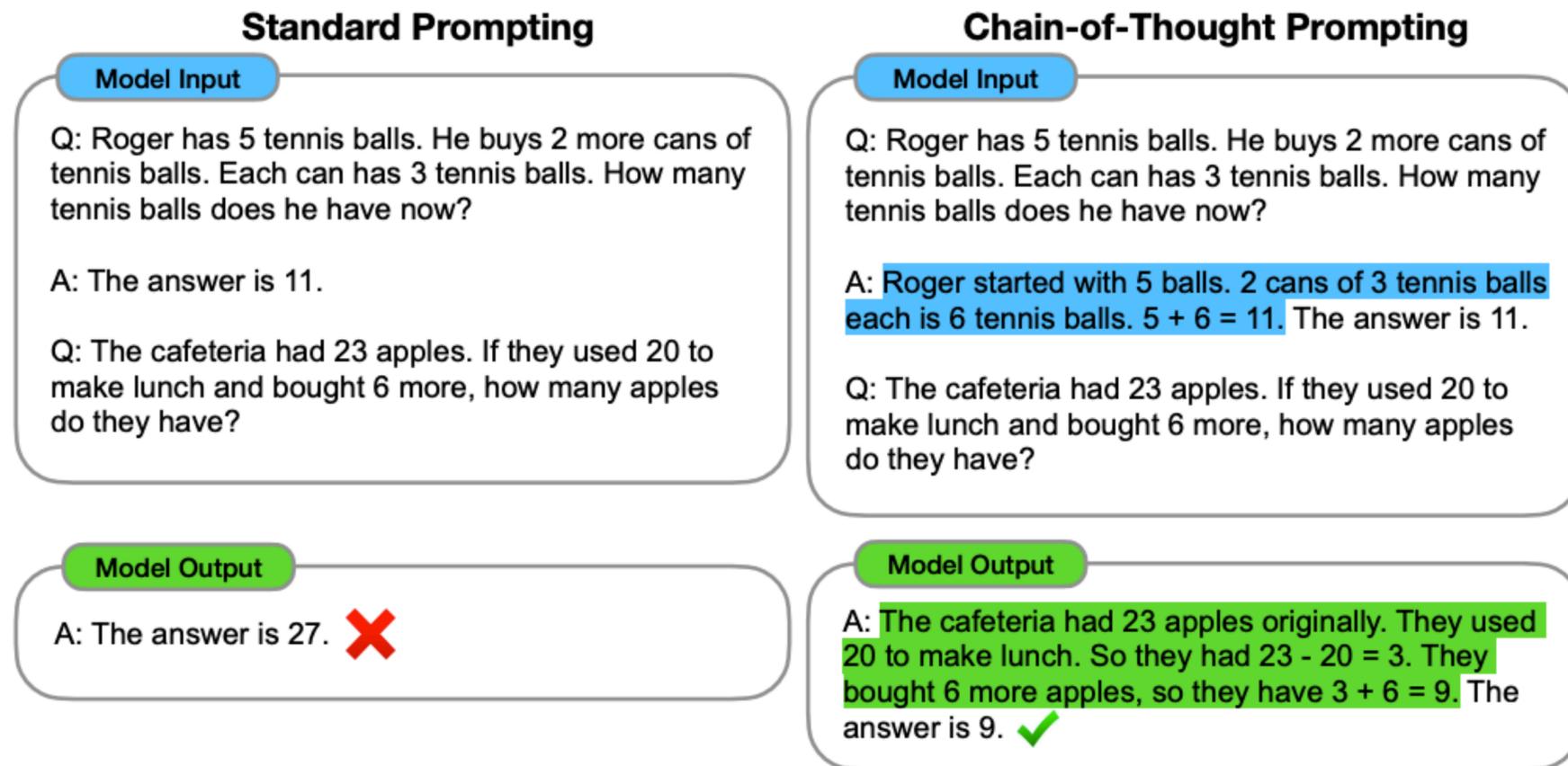


Figure 1: Chain-of-thought prompting enables large language models to tackle complex arithmetic, commonsense, and symbolic reasoning tasks. Chain-of-thought reasoning processes are highlighted.

- 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.**

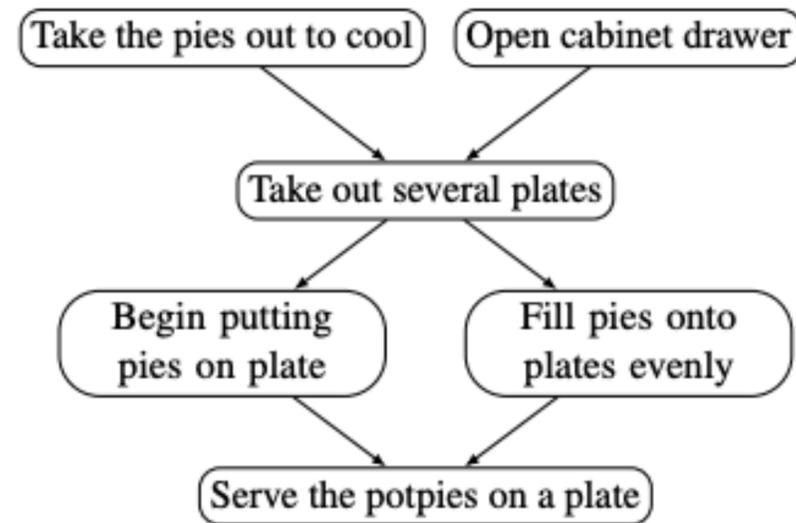
(Output) There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls. ✓

- Note: GPT models reason even w/o specific instructions now (probably due to instruction tuning)

# 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.



## 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



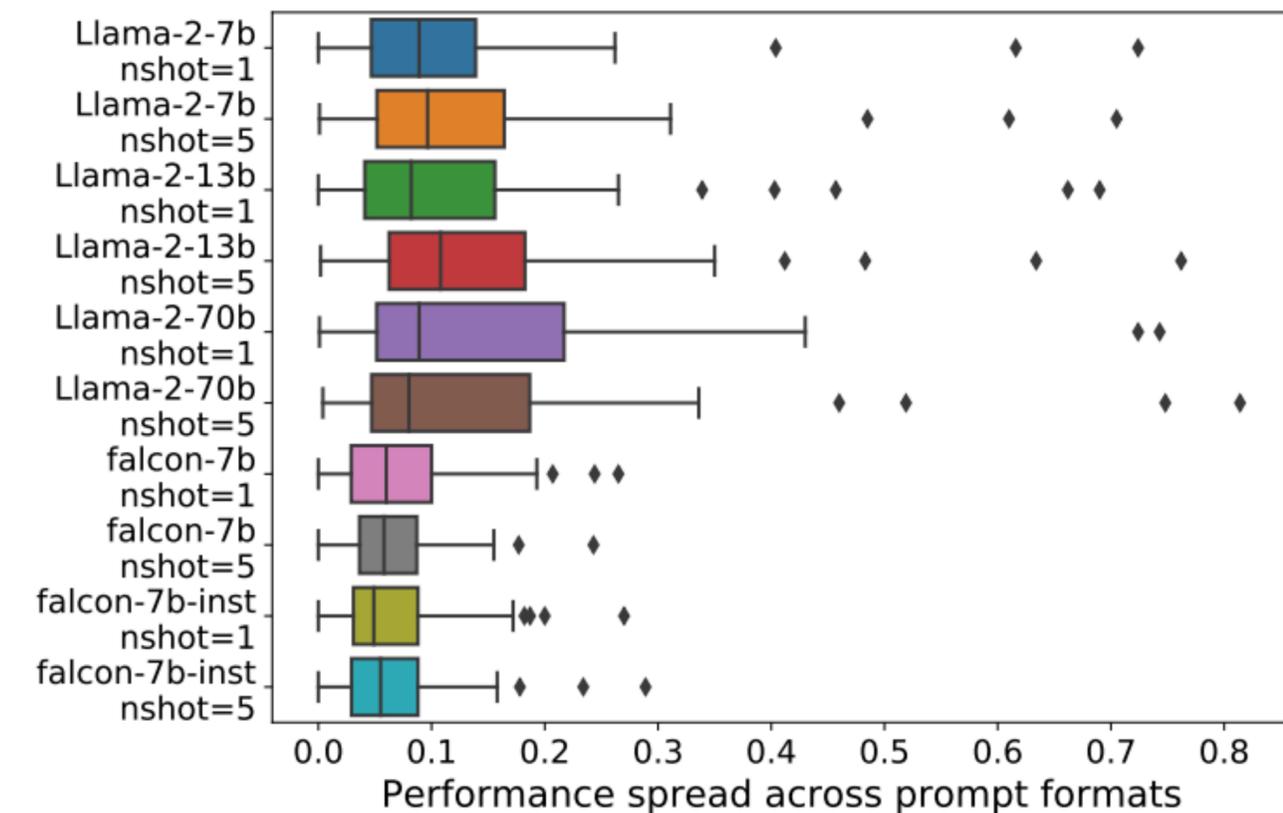
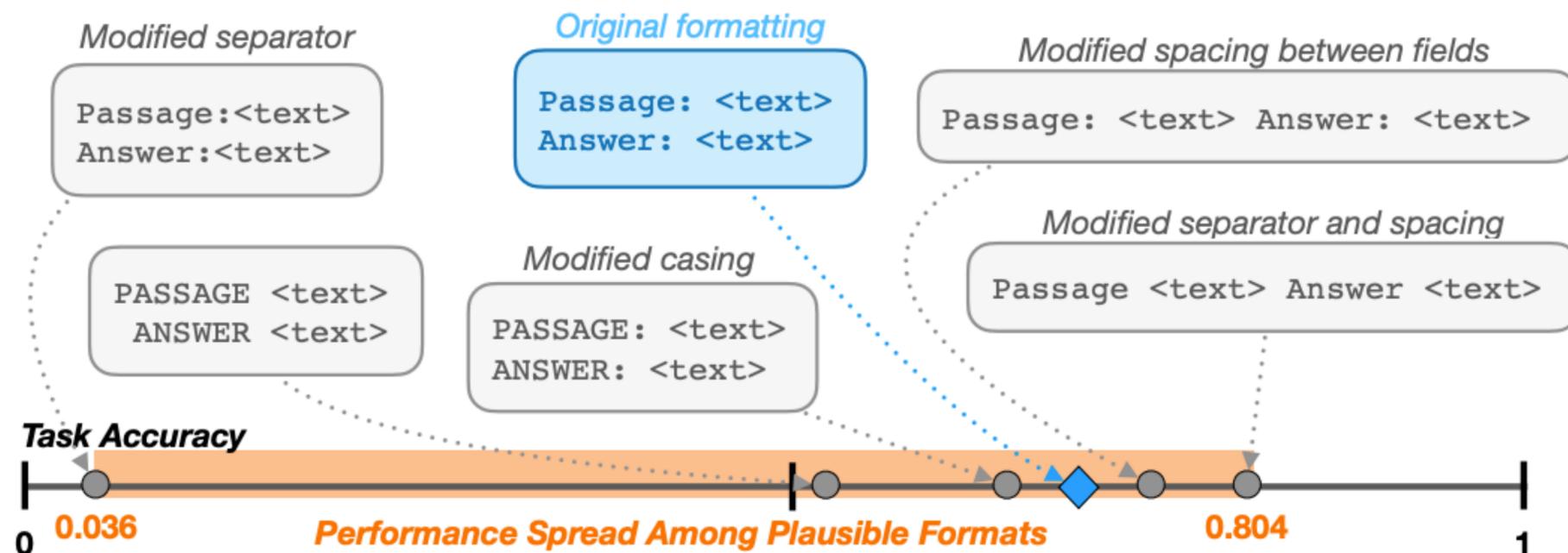
# Prompt Engineering

# 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.

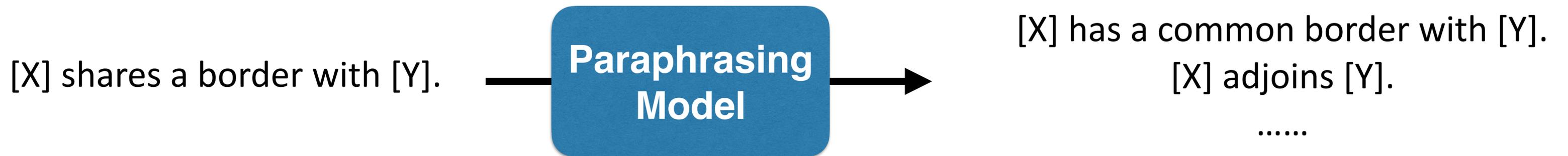
- Similar to humans, but (right now) LMs don't complain when you're vague

# Methods for Automatic Prompt Engineering

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

# Prompt Paraphrasing

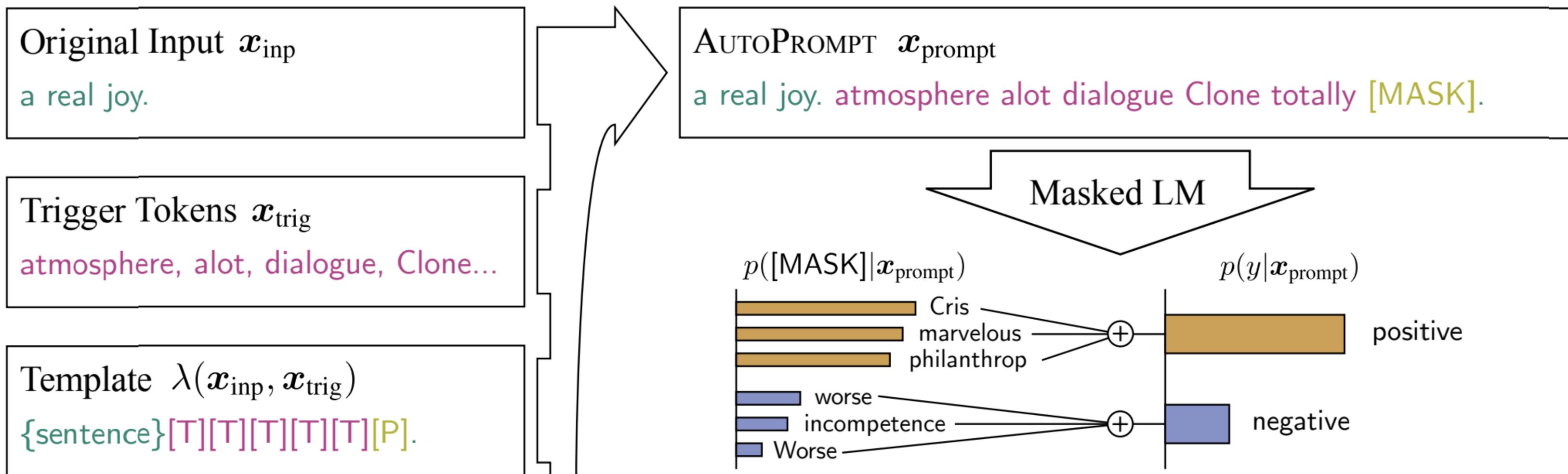
- Paraphrase an existing prompt to get other candidates (Jiang et al. 2019)



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

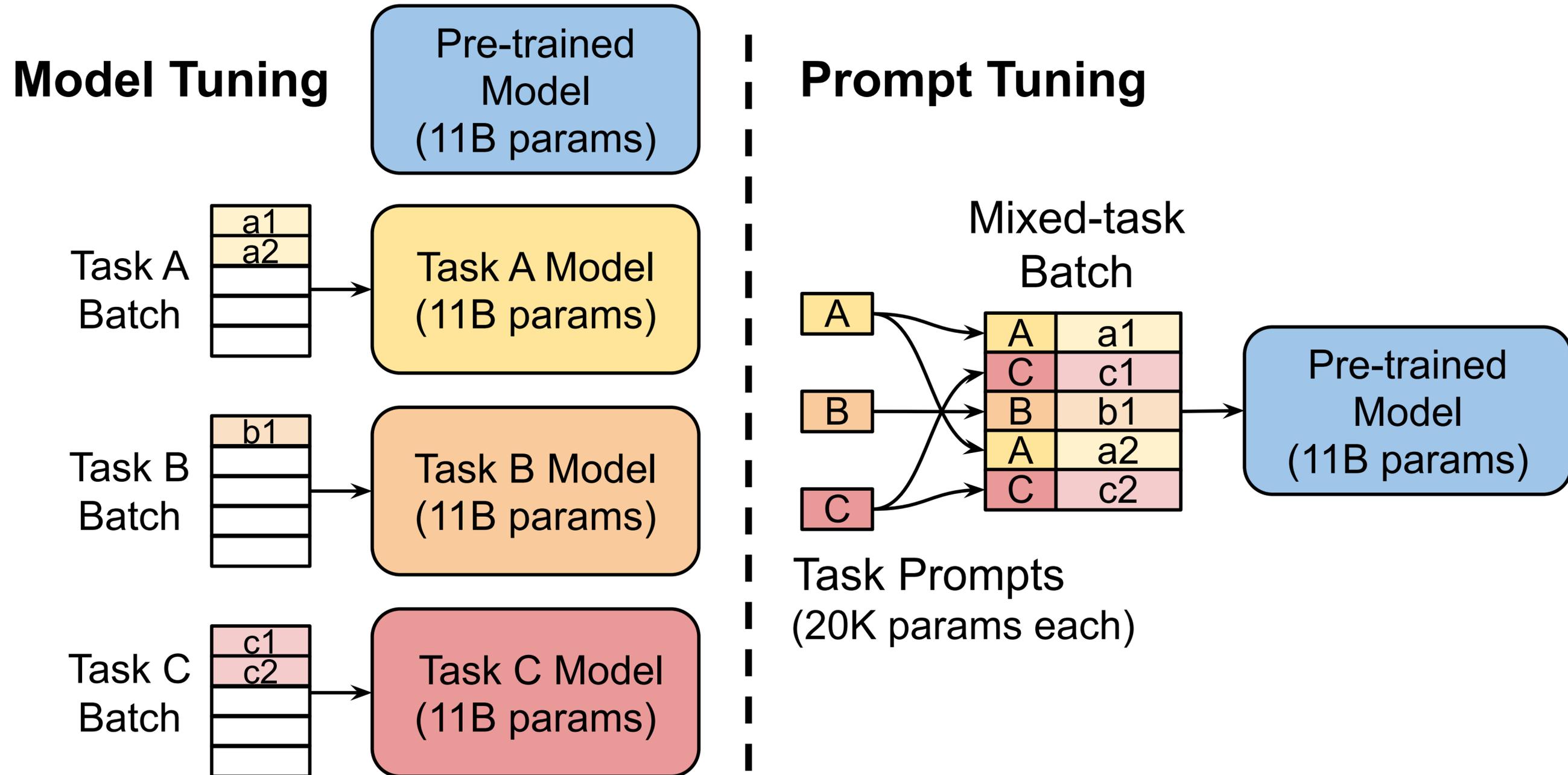
# Gradient-based Search (Shin et al. 2020)

- Automatically optimize arbitrary prompts based on existing



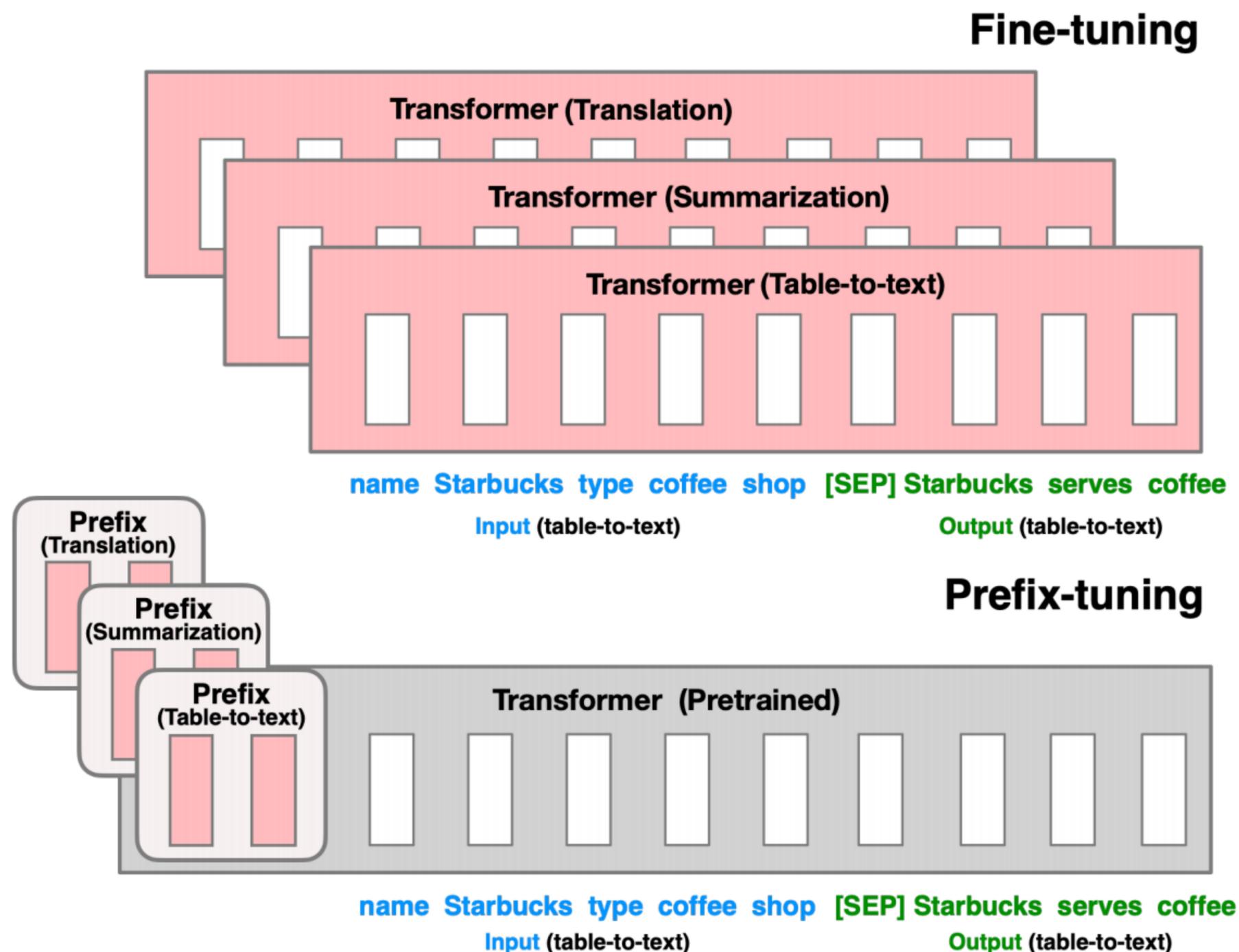
# Prompt Tuning (Lester et al. 2021)

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



# Prefix Tuning (Li and Liang 2021)

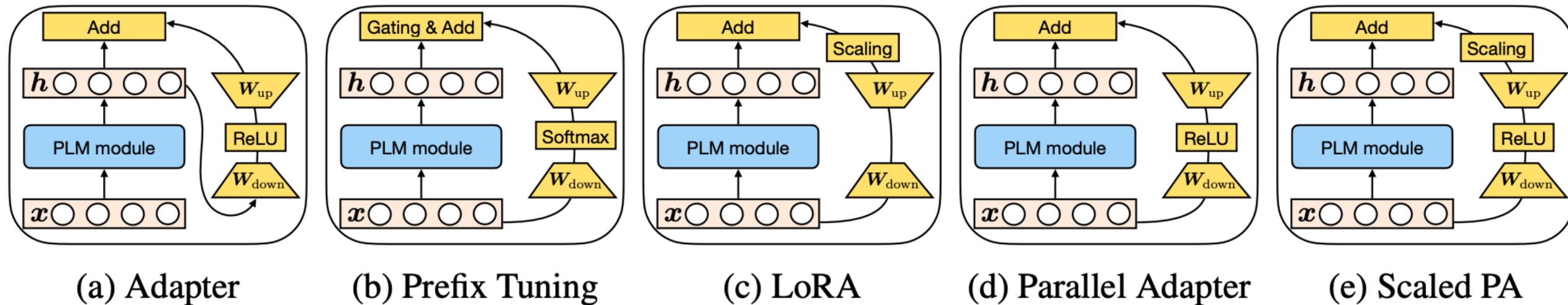
- "Prompt Tuning" optimizes only the embedding layer
- "Prefix Tuning" optimizes the prefix of all layers



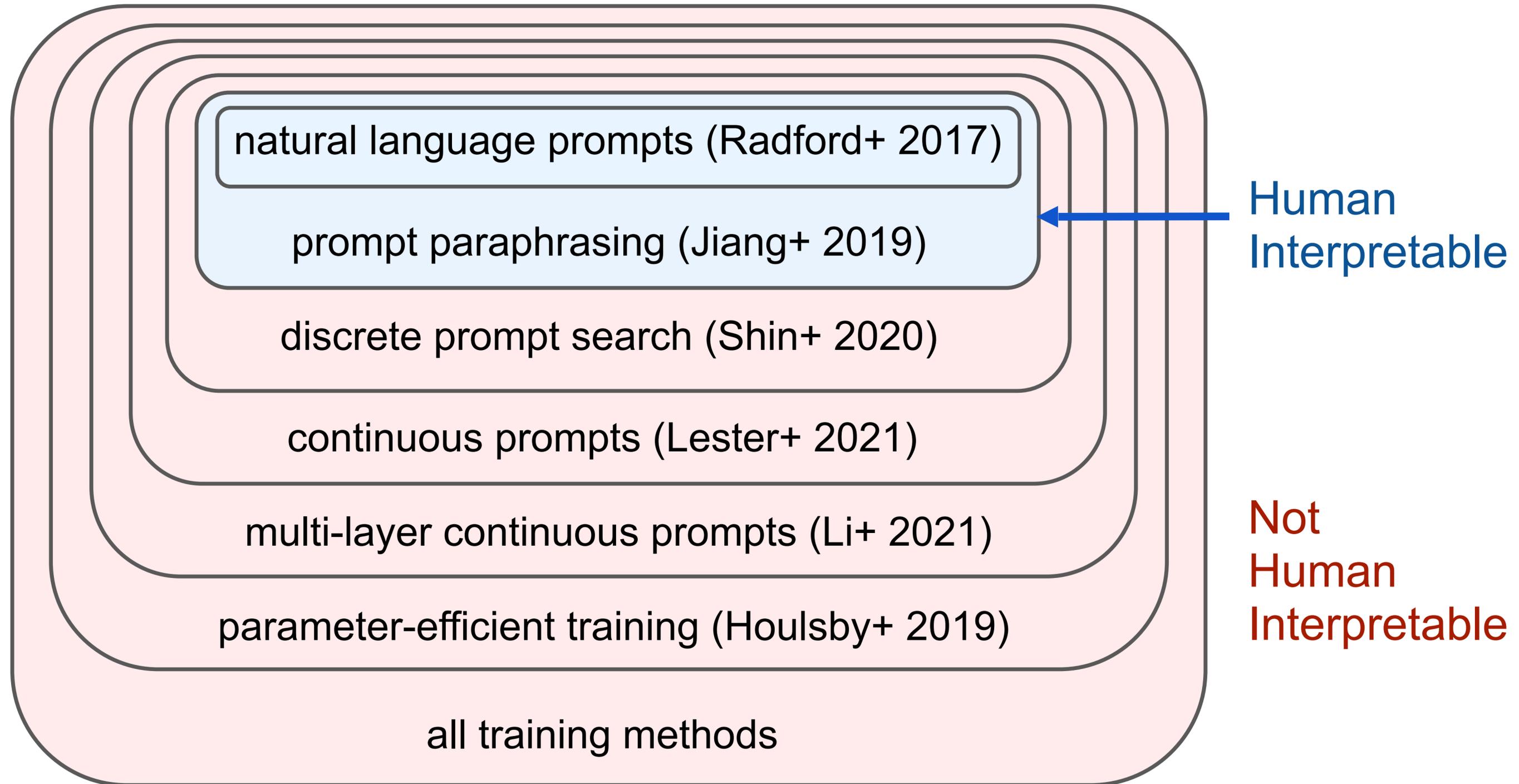
# Prompting and Fine Tuning

# 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 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!

Questions?