



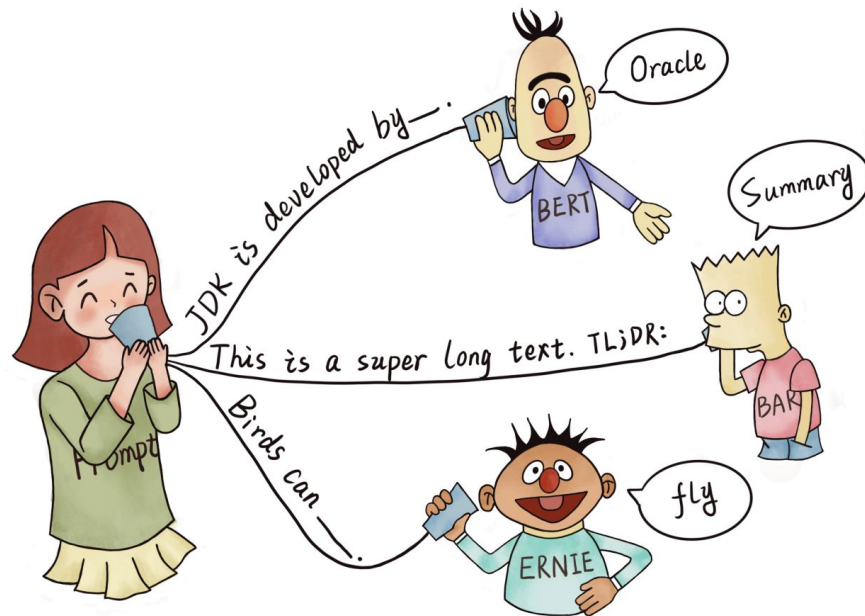
INFO-I590 Fundamentals and Applications of LLMs

Prompting

Jisun An

What is Prompting?

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



Decoding / Inference

Choosing the Next Token in Generation

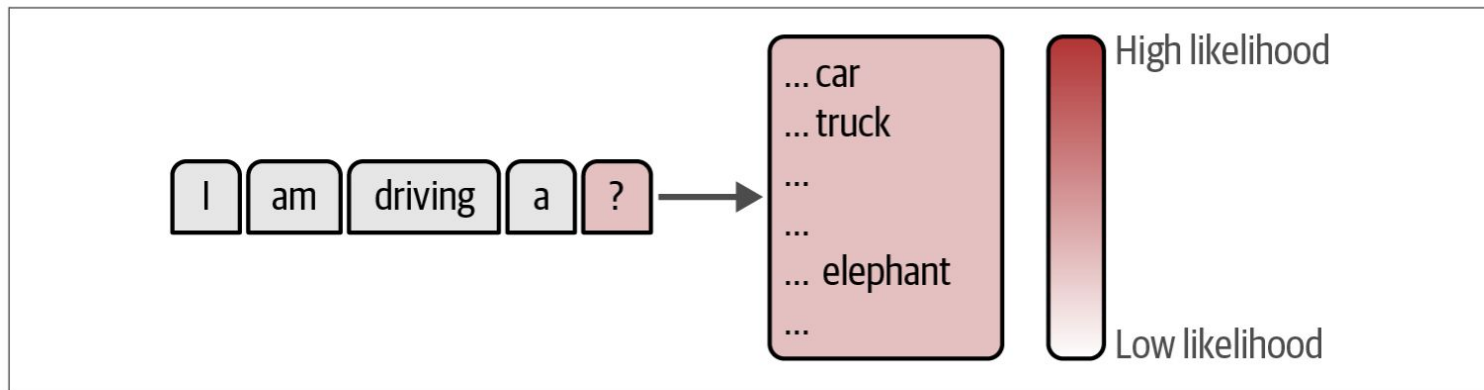
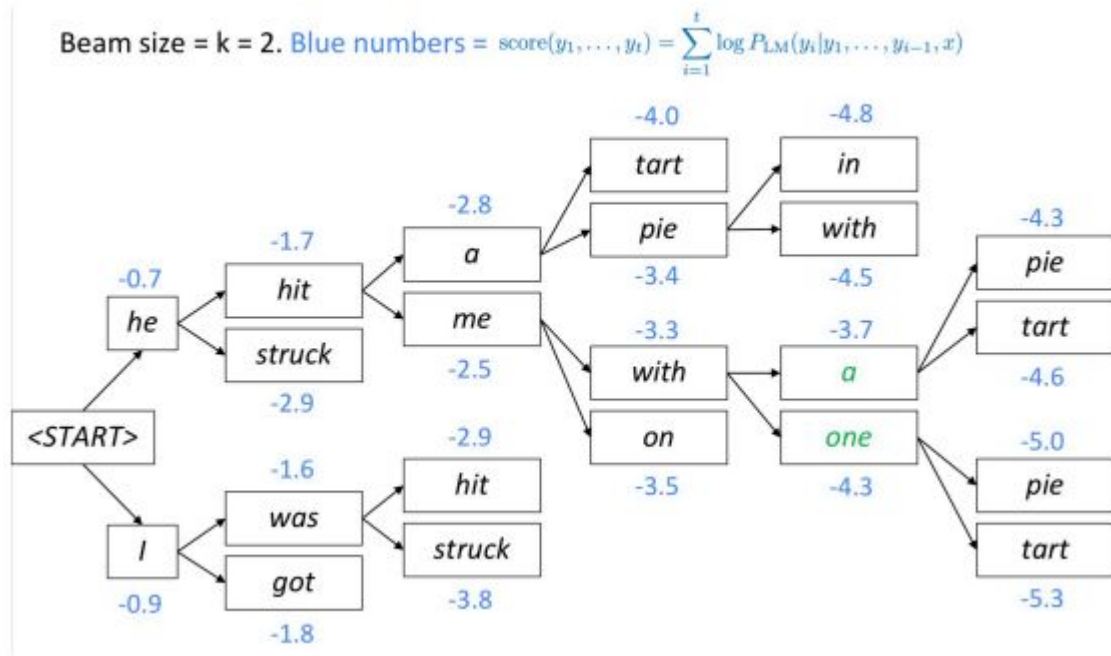


Figure 6-3. The model chooses the next token to generate based on their likelihood scores.

Decoding algorithms

- **Greedy decoding** is a simple method; gives low quality output
 - Select the most probable word (argmax)
- **Sampling methods** are a way to get more diversity and randomness
 - **Pure sampling**: On each step t , randomly sample from the probability distribution P_t to obtain your next word.
 - **Top-n sampling**: On each step t , randomly sample from P_t , restricted to just the **top-n most probable words**
 - Good for open-ended / creative generation (poetry, stories)
- **Beam search** explores several different hypotheses instead of just a single one
 - keep track of k most probable partial translations at each decoder step
 - the beam size k is usually 5-10

Beam Search (k=2) Example



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 bit.

(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.
```

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 (sampling, greedy method, etc)

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

 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

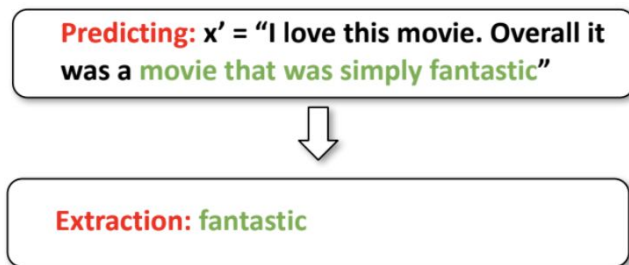
[
  {"President": "Joe Biden", "Birthdate": "November 20, 1942"},
  {"President": "Donald Trump", "Birthdate": "June 14, 1946"},
  {"President": "Barack Obama", "Birthdate": "August 4, 1961"},
  {"President": "George W. Bush", "Birthdate": "July 6, 1946"},
  {"President": "Bill Clinton", "Birthdate":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

''' &#8203;''' [oacite:0] '''&#8203;'''
```

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

Output Selection

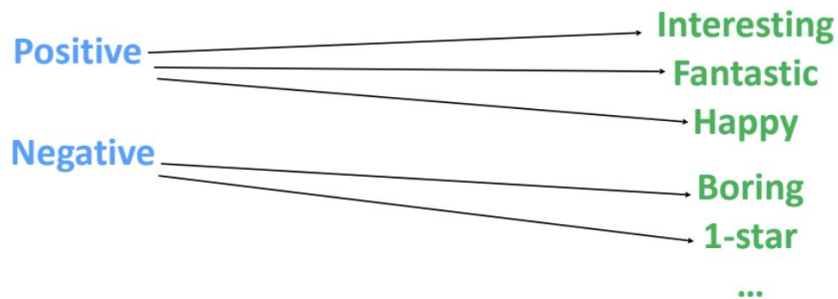
- From a longer response, select the information indicative of an answer



- 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



Zero-shot, Few-shot Prompting / In-context Learning

Zero-shot Prompting

- Solve a task by simply conditioning the models on an instruction describing the task

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

Few-shot Prompting (Brown et al. 2021)

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

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

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

Input: This movie is great!
Output: positive

LMs are Sensitive to Small Changes in In-context Examples

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

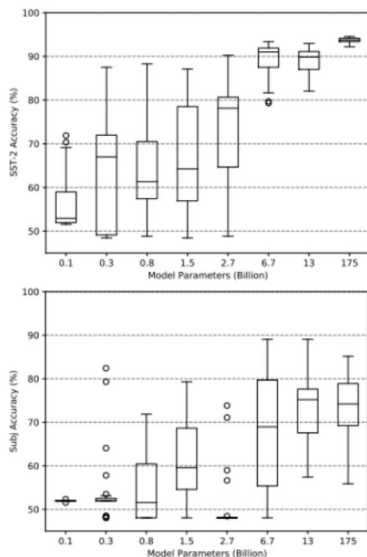
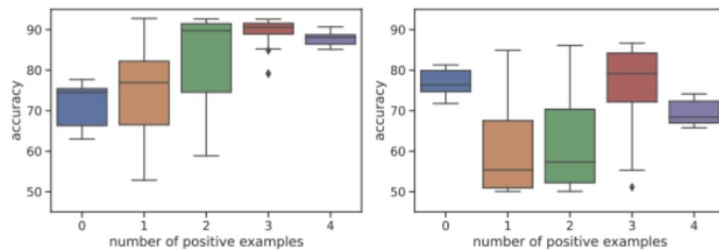


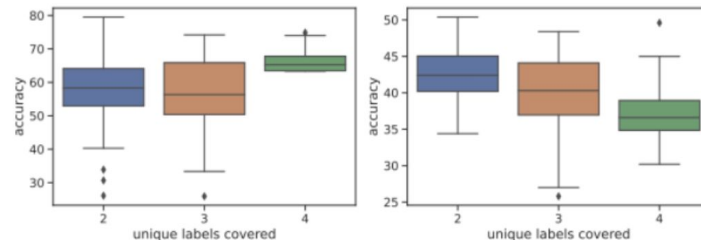
Figure 1: Four-shot performance for 24 different sample orders across different sizes of GPT-family models (GPT-2 and GPT-3) for the SST-2 and Subj datasets.



(a) Amazon

(b) SST-2

- Label coverage (Zhang et al. 2022)

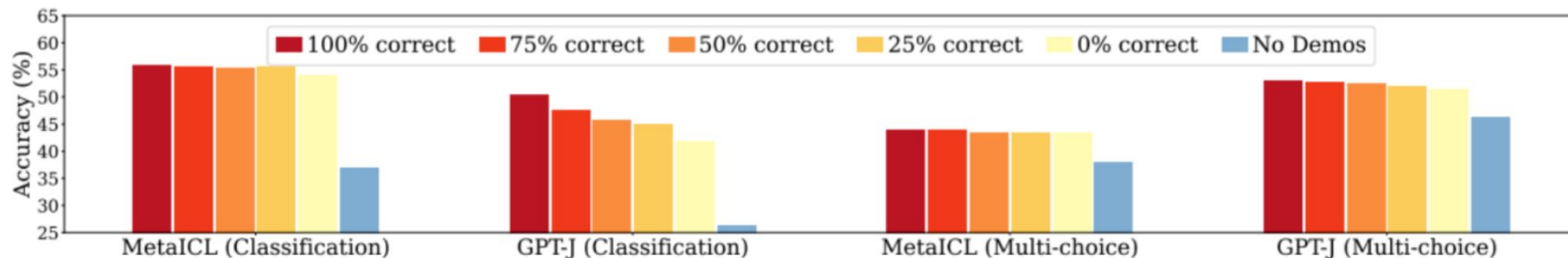


(a) AGNews

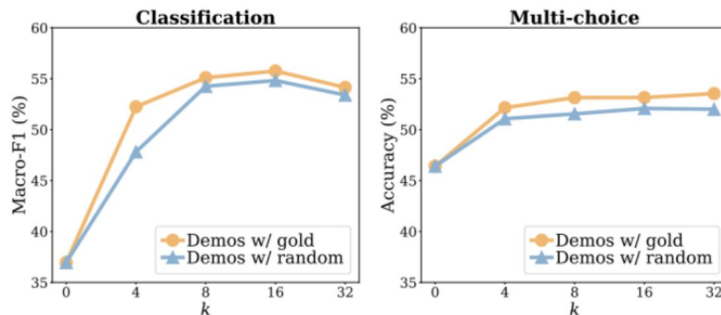
(b) TREC

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 Basics

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.

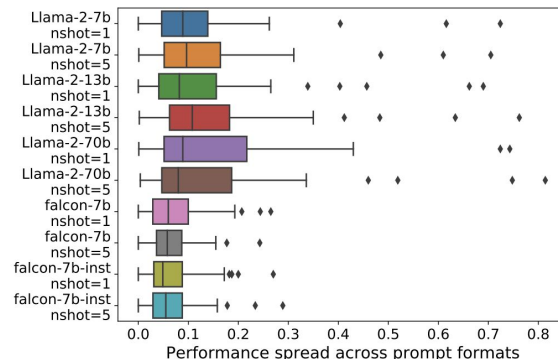
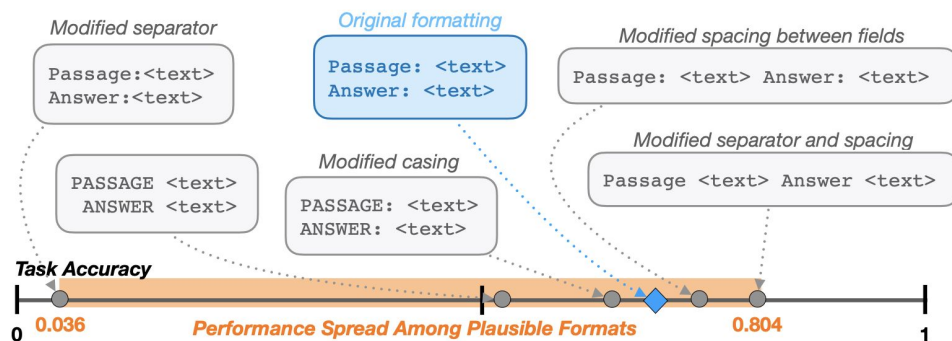
More Precise:

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

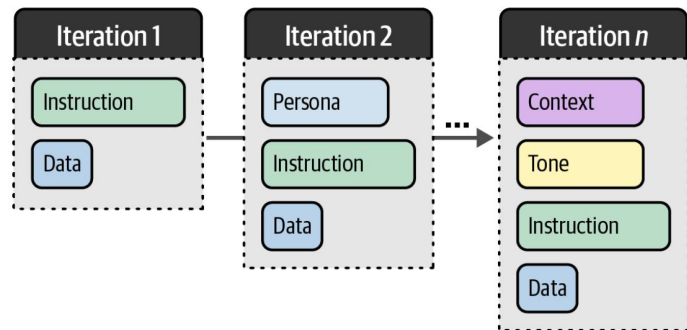
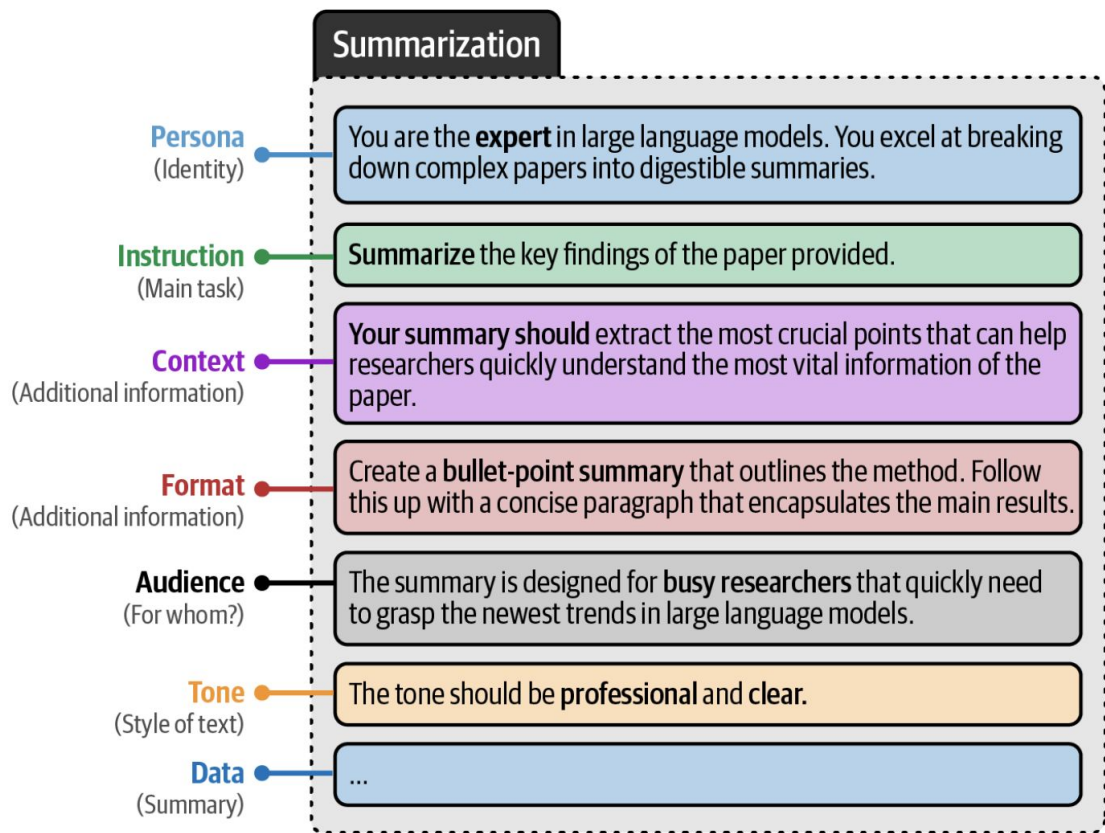
Paying attention to prompt formats

- 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)



Prompt Components

- **Persona:** Define the role the LLM should assume. For example, specify “You are an astrophysics expert” to tailor the response to questions on that topic.
- **Instruction:** Clearly outline the task. Ensure it is detailed to minimize ambiguity.
- **Context:** Provide background information to clarify the task’s purpose and reasoning.
- **Format:** Specify the desired output format to ensure consistency, especially in automated systems.
- **Audience:** Identify the intended recipient or level of the output, such as using “ELI5” (Explain Like I’m 5) for simplified explanations.
- **Tone:** Indicate the tone of voice for the response. For formal communication, such as emails to a superior, avoid casual language.
- **Data:** Include any relevant data necessary for completing the task.



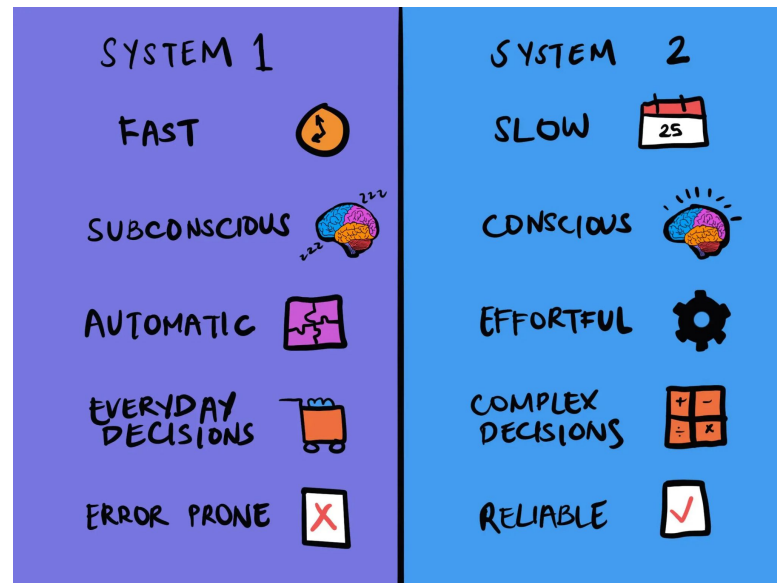
Other Key Factors for Prompt Design

- Hallucination:
 - LLMs may confidently produce incorrect information.
 - Solution: Prompt the LLM to respond with "I don't know" if unsure.
- Order Matters:
 - Place key instructions at the beginning or end of the prompt.
 - Why?
 - Primacy Effect: Focus on the start
 - Recency Effect: Focus on the end

Advanced Prompting Methods / Reasoning in LMs

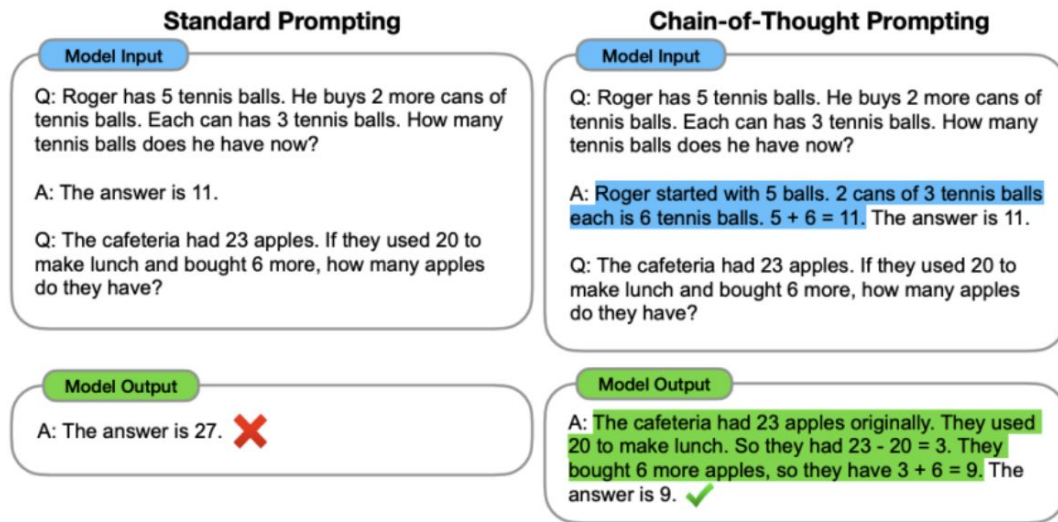
What is Reasoning?

- Using **evidence** and **logic** to arrive at conclusions and make judgment
- Can be
 - Formal: focusing on stric truth values
 - Informat: based on experience and intuition
- How can we design LLMs to mimic reasoning processes and enhance their outputs?



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

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



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

(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)

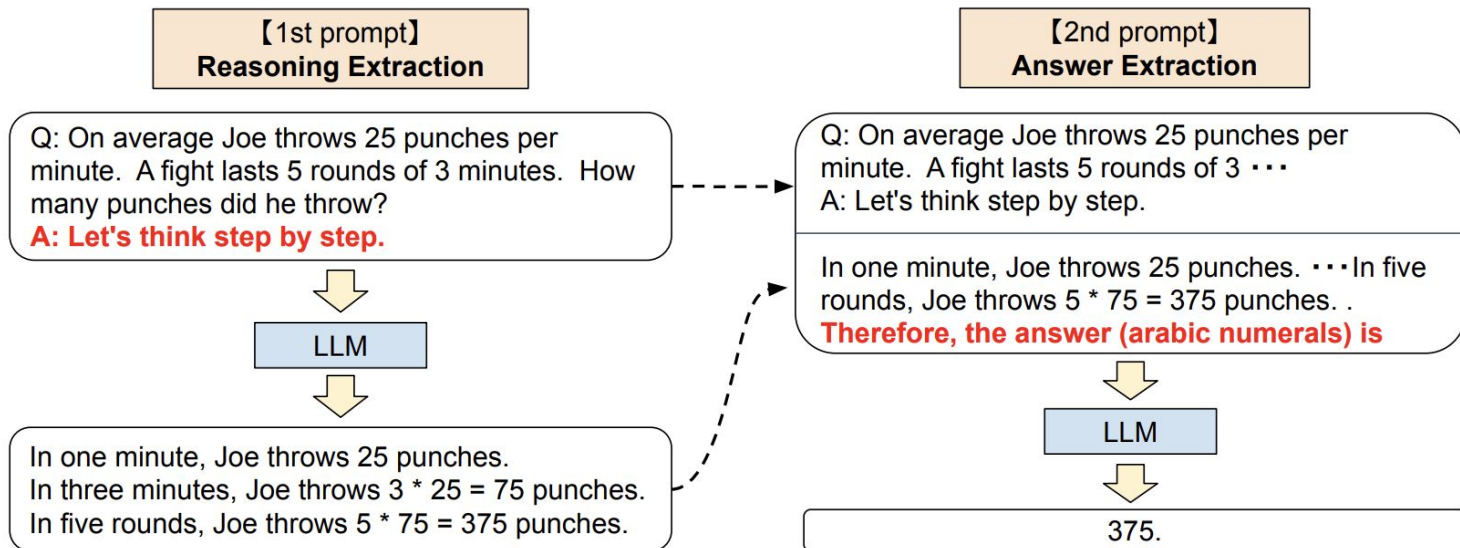
Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

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. ✓

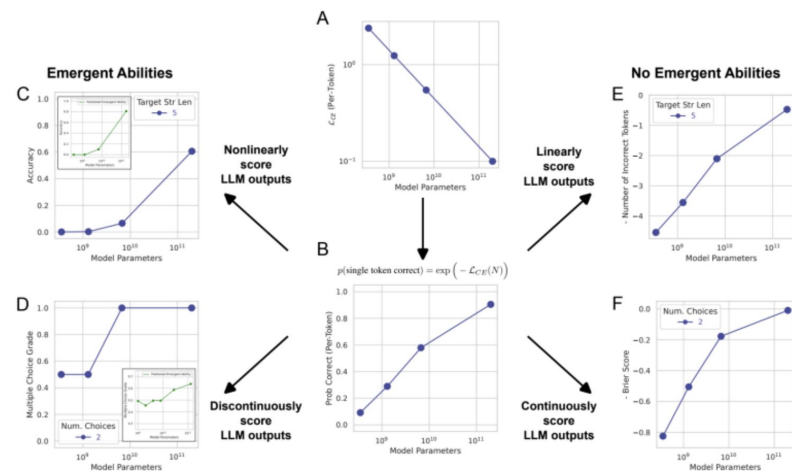
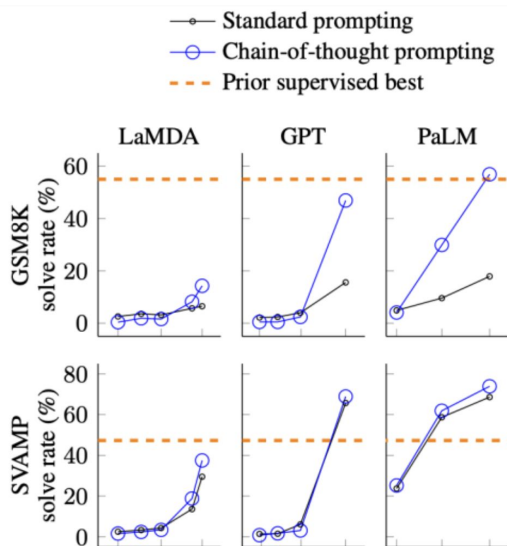
Unsupervised Chain-of-thought Prompting (Kojima et al. 2022)

Simply appending “Let’s think step by step” to the end of each prompt can improve the performance of LLMs on several reasoning tasks.



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)



Problem Decomposition

- Solving a complex problem by breaking it down into simpler problem-solving tasks

“Please write a blog about the risks of AI”

You are a blog writer. Please follow the provided outline below to write a blog about the risks of AI.

- Introduction
Introduce AI, its relevance, and the importance of understanding its risks for youth.
- Privacy Concerns
Discuss how AI might compromise personal privacy through interactions online.
- Misinformation
Explore AI's role in spreading misinformation and influencing young people's decisions.
- Cyberbullying
Highlight how AI tools can be utilized in cyberbullying and the impact on mental health.
- Tips for Safe AI Use
Offer guidelines for responsible AI usage and promote critical thinking.
- Conclusion
Recap main points and encourage proactive engagement with AI ethics.

Problem Decomposition Framework

A general framework for problem decomposition consists of two key elements:

- Sub-problem Generation
 - Breaks down the input problem into smaller, manageable sub-problems.
- Sub-problem Solving
 - Solves each sub-problem and derives intermediate and final conclusions through reasoning.

Least-to-Most Prompting (Zhou et al. 2023)

- Sub-problem generation is performed by prompting an LLM with instructions and/or demonstrations.
- Solving each subproblem is facilitated by the answers to previously solved subproblems.

Stage 1: Decompose Question into Subquestions

Q: It takes Amy 4 minutes to climb to the top of a slide. It takes her 1 minute to slide down. The water slide closes in 15 minutes. How many times can she slide before it closes?

Language Model

A: To solve "How many times can she slide before it closes?", we need to first solve: "How long does each trip take?"

Stage 2: Sequentially Solve Subquestions

Subquestion 1

It takes Amy 4 minutes to climb to the top of a slide. It takes her 1 minute to slide down. The slide closes in 15 minutes.

Q: How long does each trip take?

Language Model

A: It takes Amy 4 minutes to climb and 1 minute to slide down. $4 + 1 = 5$. So each trip takes 5 minutes.

Append model answer to Subquestion 1

It takes Amy 4 minutes to climb to the top of a slide. It takes her 1 minute to slide down. The slide closes in 15 minutes.

Q: How long does each trip take?

A: It takes Amy 4 minutes to climb and 1 minute to slide down. $4 + 1 = 5$. So each trip takes 5 minutes.

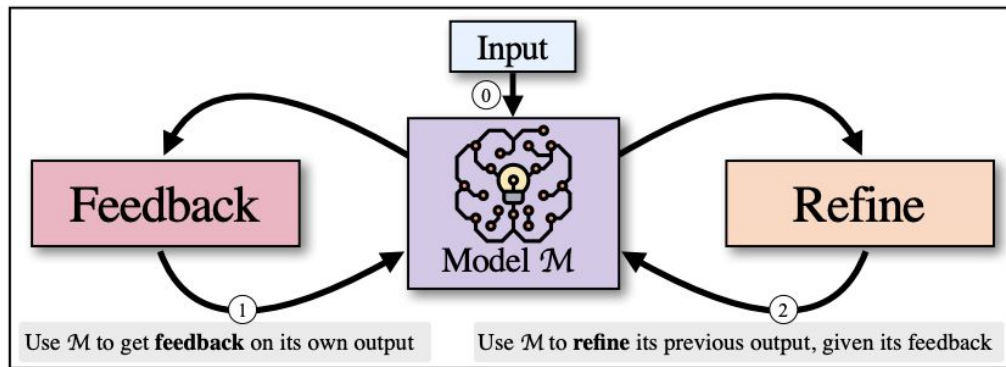
Language Model

A: The water slide closes in 15 minutes. Each trip takes 5 minutes. So Amy can slide $15 \div 5 = 3$ times before it closes.

Subquestion 2

Q: How many times can she slide before it closes?

Self-refinement (Madaan et al., 2024)



- **Prediction.** We use an LLM to produce the initial model output.
- **Feedback Collection.** We obtain feedback on the model output.
- **Refinement.** We use the LLM to refine the model output based on the feedback.

Self-refinement (Madaan et al., 2024)

(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

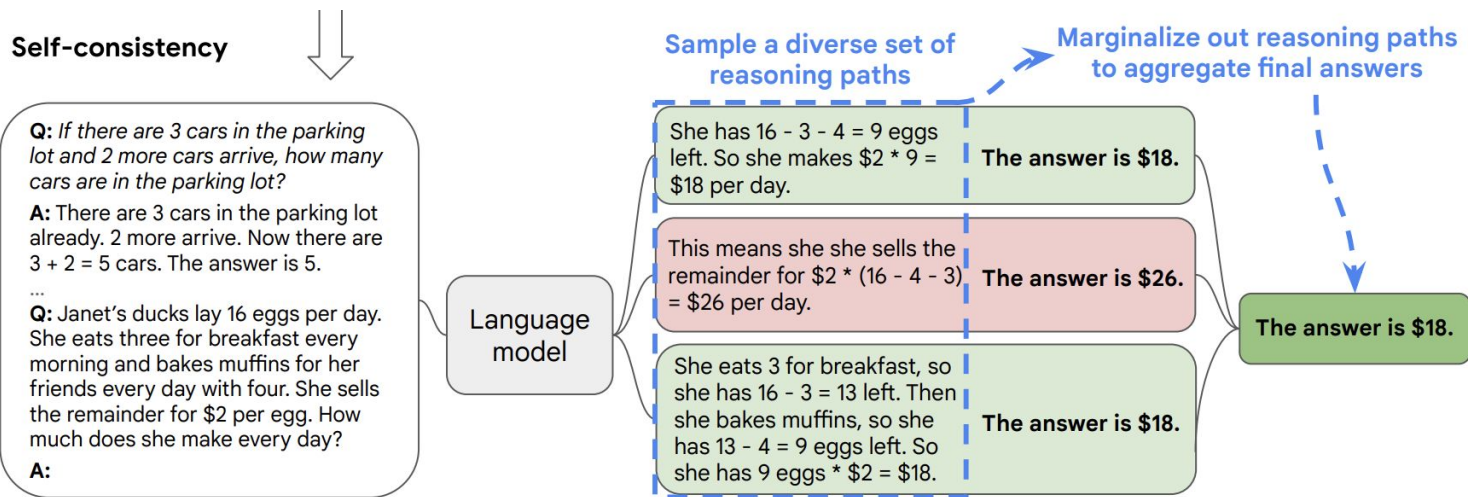
This code is slow as it uses brute force. A better approach is to use the formula ... $(n(n+1))/2$.

(f) REFINE y_{t+1}

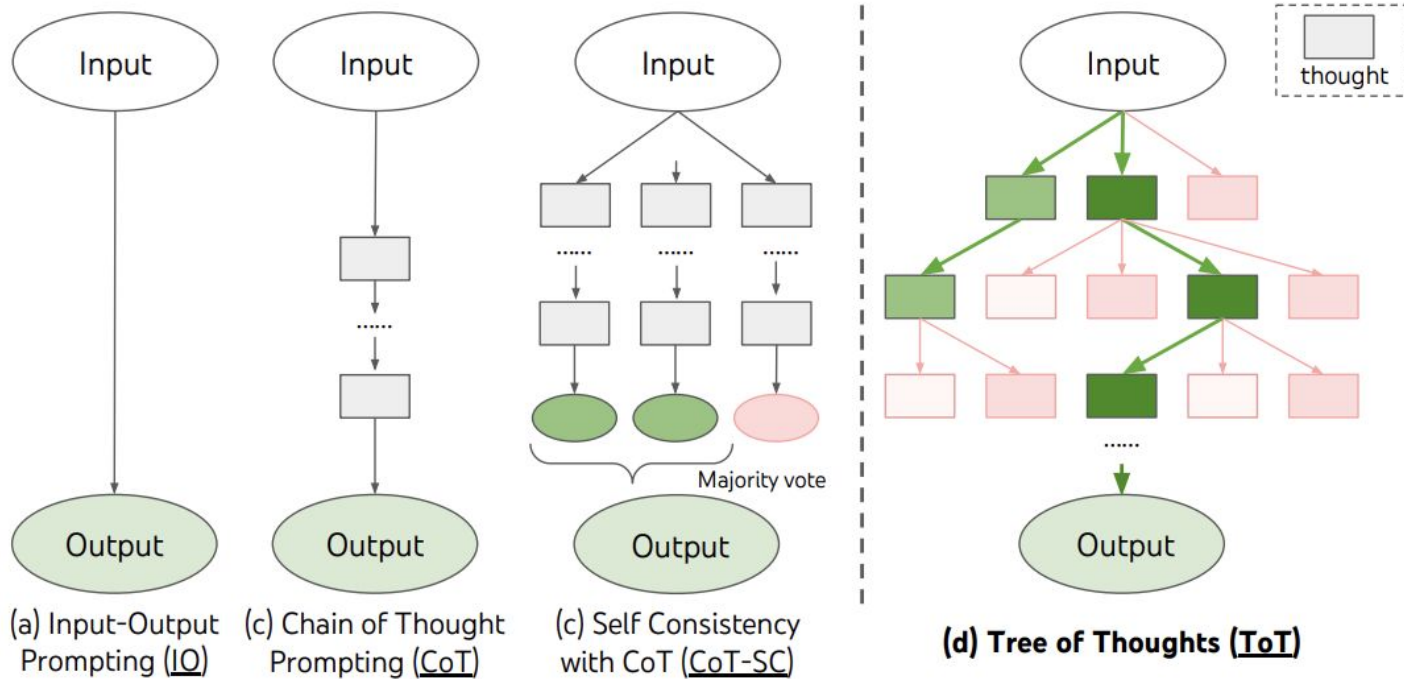
```
Code (refined)
def sum_faster(n):
    return (n*(n+1))//2
```

Self-consistency (Wang et al., 2023)

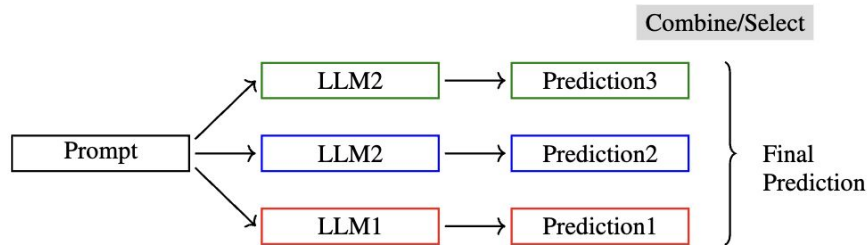
- It first samples a diverse set of reasoning paths instead of only taking the greedy one, and then selects the most consistent answer by marginalizing out the sampled reasoning paths.



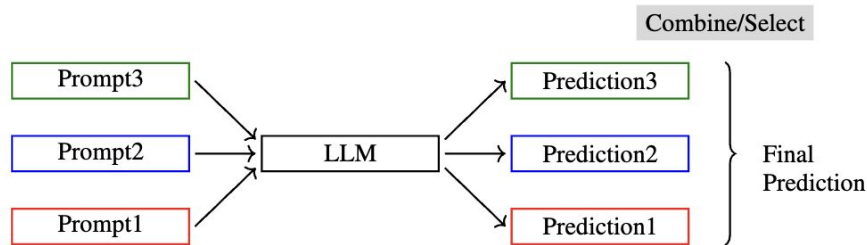
Tree-of-Thought: Exploring Intermediate Steps



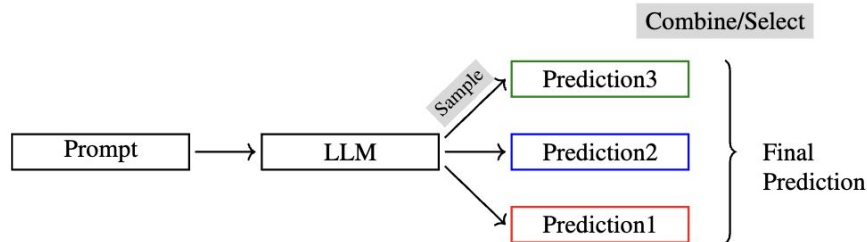
Ensembling methods for LLMs



(a) Model Ensembling



(b) Prompt Ensembling



(c) Output Ensembling

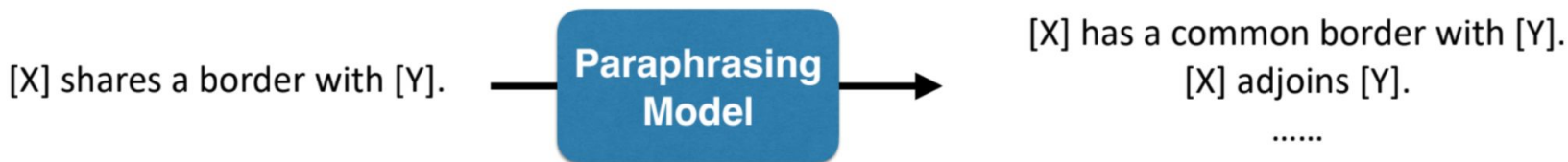
Learning to Prompt

Methods for Automatic Prompt Engineering

- Prompt paraphrasing
- Prompt tuning
- Prefix tuning

Prompt Paraphrasing

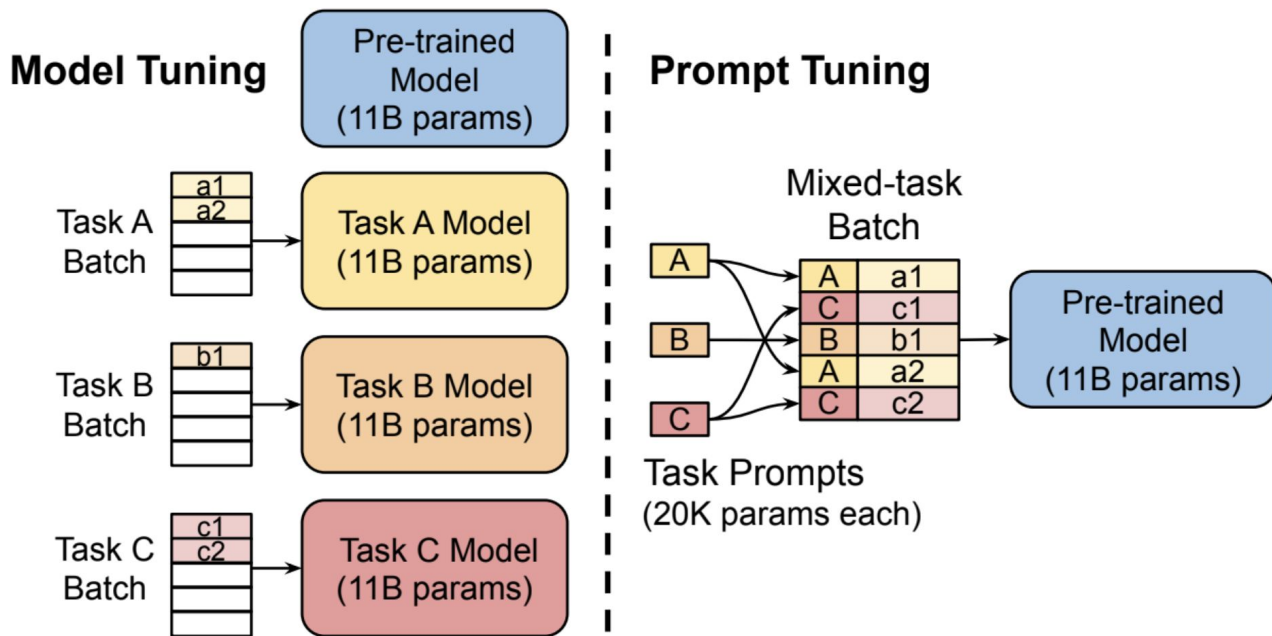
- Paraphrase an existing prompt to get other candidates (Jiang et al. 2019)



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

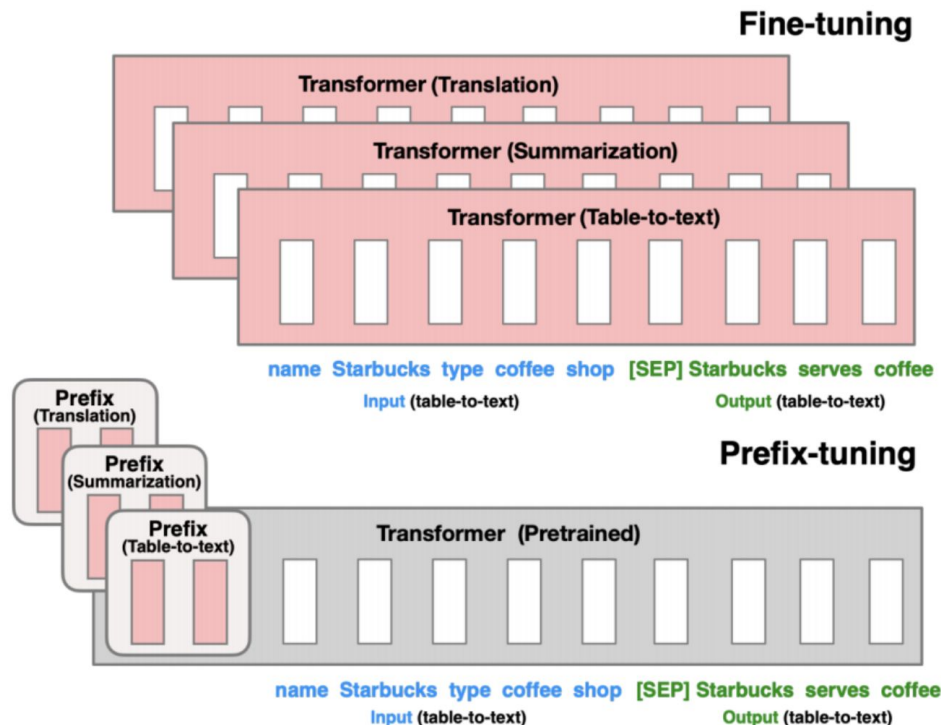
Prompt Tuning (Lester et al. 2021)

- Optimize the embeddings of a prompt

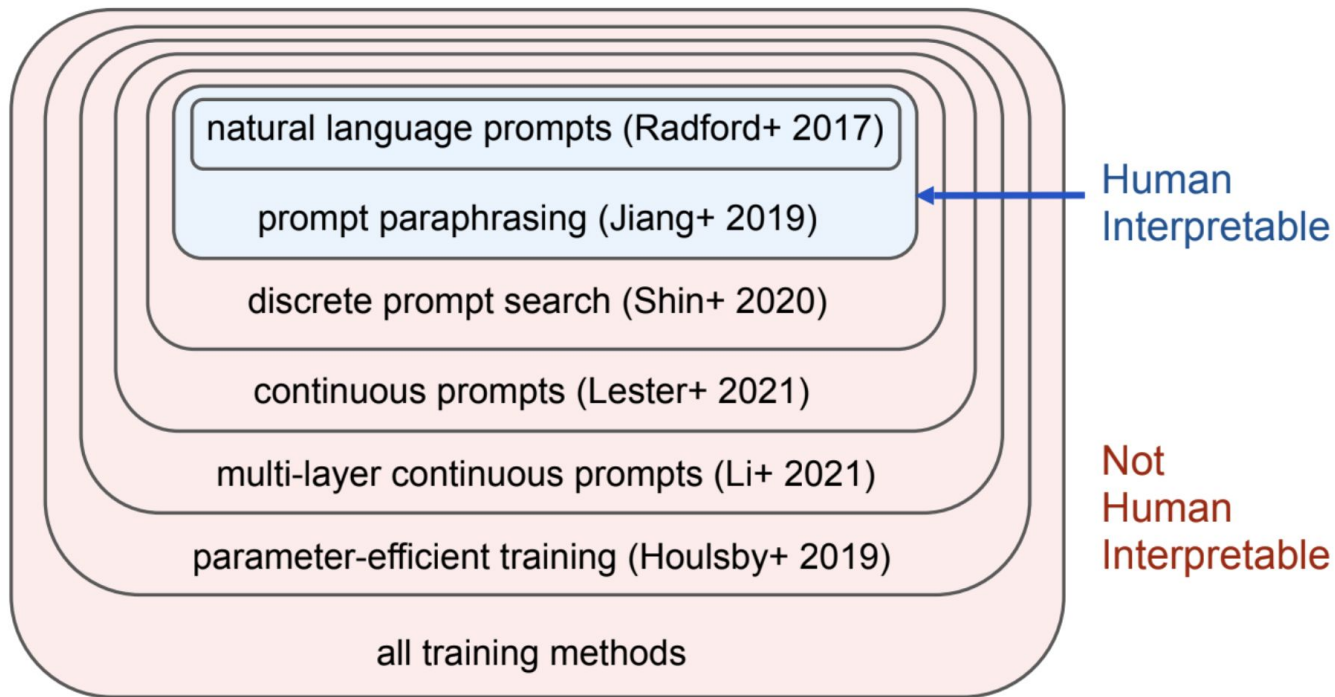


Prefix Tuning (Li and Liang 2021)

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



A Taxonomy of Prompting Methods



Any Questions?