

# Advanced Prompting Techniques

*Generative AI*

Module 1 – Lesson 4

# Today's activities

- Review: Basics of prompt engineering
- Chain-of-thought prompting
- Self-consistency
- Tree-of-thought



# Basics of prompt engineering

# Review: Good prompting practices

---

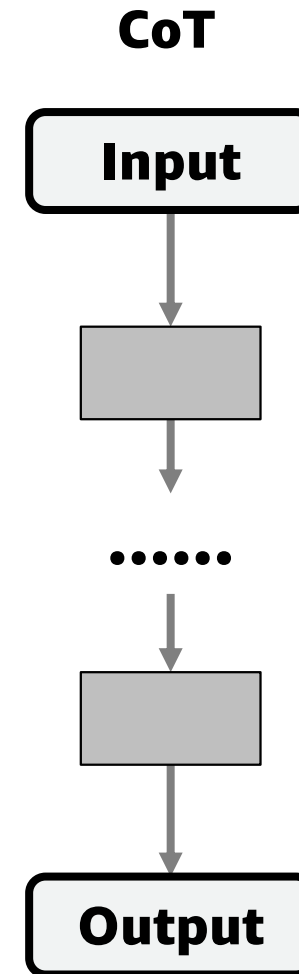
- Write **clear** and **specific** instructions (unambiguous and specific).
- Highlight or specify the part of the prompt **that the model should focus** on.
- Add relevant **details** or **restrictions** to your prompt.
- Separate the instruction, content, question, and output directions
- Prefer using positive instructions
- Try using examples to guide the model's response
  - **In-context learning**
- Finding the optimum prompt is usually an **iterative** process which may take a few attempts.

# Chain-of-thought prompting

# Chain-of-thought (CoT) prompting

---

- Technique that breaks down complex tasks through **intermediate reasoning steps**.
- Encourages model to explain its reasoning process by **decomposing the solution** into a series of steps.
  - This behaviour can be facilitated through various strategies (few-shot CoT, zero-shot CoT, ...).
  - CoT is the **basis for other prompting techniques** which separate out the task's decomposition and its solving.



# Zero-shot CoT

- Finding: Appending instructions like “**Let’s think step by step**” to the prompt elicits the generation of a sequential reasoning chain by the LLM. This derives more precise answers

## 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 **X**

Figure from [Kojima et al. \(2022\)](#)

## Zero-shot CoT

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.* **✓**

**CoT query  
augmentation**

# Few-shot CoT

- Show LLM examples with reasoning so that the LLM will also produce a reasoned answer. This **explanation of reasoning often leads to more accurate results.**

## Few-shot

### 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. ❌

## Few-shot CoT

### 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. ✅

CoT example(s)

Step-by-step answer

Figure from [Wei et al. \(2022\)](#)



# Benefits of CoT

---

- LLMs can benefit from **detailed and logical steps** to process the prompt.
- Such chains of thought can be facilitated by:
  - Addition of **smaller logical steps** in the prompt to break down the question or task.
  - A series of **demonstrations**, composed of question and reasoning chain leading to an answer.
  - Prompt augmentations like **“Let’s think step by step”**.
- CoT can **enhance performance** of LLMs on tasks requiring arithmetic, common-sense, and symbolic reasoning.
- Performance **gains in models of ~100B** parameters [Wei et al. 2022].

# Limitations of CoT

---

- CoT prompts (especially few-shot examples) are **specific to a problem type**.
- Smaller models might produce fluent but **illogical CoT**, with lower performance.
- Increased cost of generation

# Overview of prompting techniques

---

Method	Use case
Zero-shot prompting	Tasks that the LLM is capable of performing simply <b>leveraging the information</b> from the large amounts of data it has been pre-trained on.
One-shot prompting	Output needs to follow a certain structure that can be demonstrated via <b>one example</b> .
Few-shot prompting	Output needs to follow a certain structure that can be demonstrated via <b>various examples</b> .
Chain of thought	Complex tasks that require <b>reasoning</b> before a response can be generated.

# Review: Standard prompting strategies

---

- Standard prompt engineering techniques helps get **improved results** from LLMs for different tasks. Best practices:
  - Write clear, concise, positive, and precise instructions.
  - Supplement with examples (one/few-shot prompting).
  - Ask the model to decompose a task into reasoning steps (CoT prompting).

# Towards advanced prompting

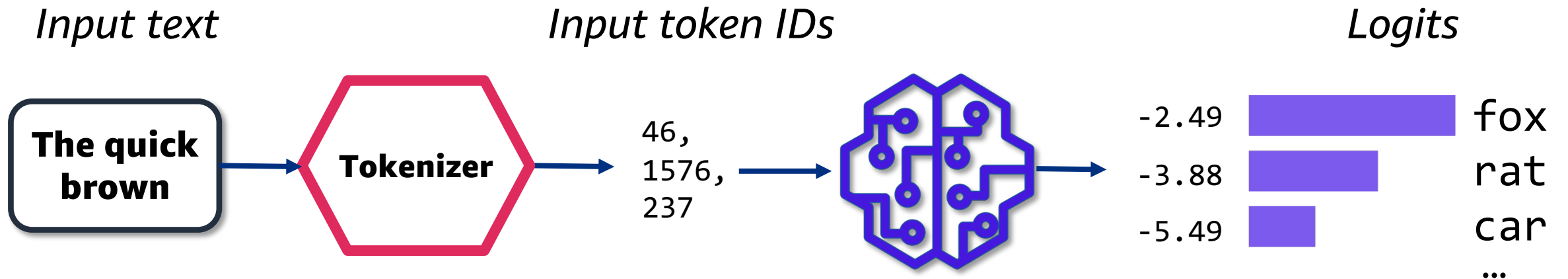
---

- Difficult questions might require the LLM to follow complex instructions and perform multi-step reasoning. **Standard prompting techniques are often not enough.**
  - Few-shot learning requires the **limited context window** of most LLMs to be occupied with exemplars.
  - CoT does **not guarantee correct reasoning paths** and can give rise to both correct and incorrect answers.
  - LLMs can be **tricked** into providing harmful output if safeguards aren't put in place.

# Self-consistency

# Decoding for text generation

- Decoder-only models generate text **word by word**.
- LLMs calculate logits: **scores** assigned to every possible token in their vocabulary via a probability distribution.
- **Decoding** is the process of turning the logits coming from the probability distributions generated by the model into actual text.



# Greedy vs stochastic decoding

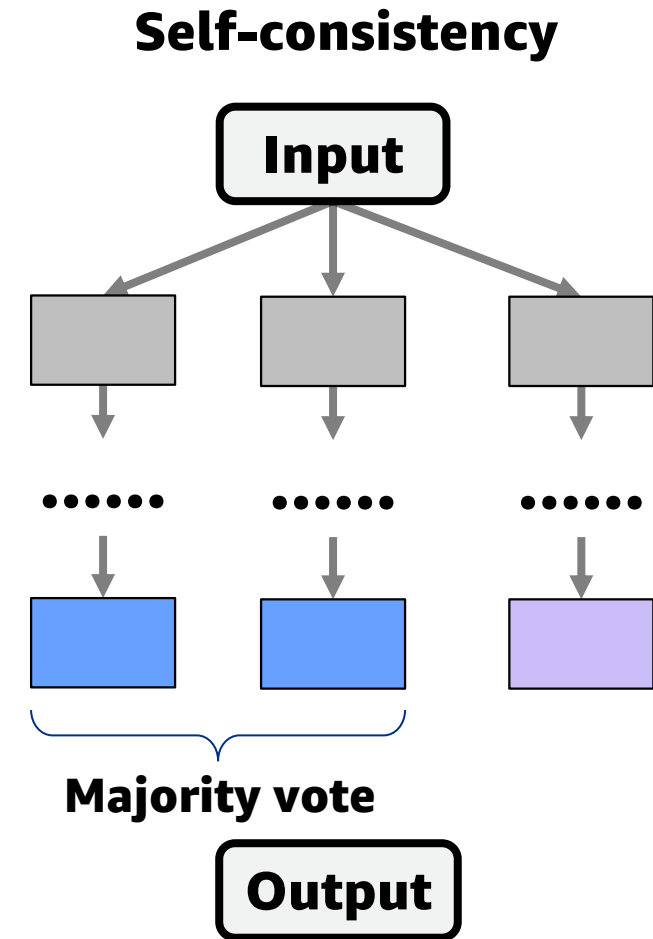
---

Greedy	Stochastic
<b>Deterministic</b> decoding method	<b>Non-deterministic</b> decoding method
At each step: select the token with the <b>highest probability</b>	At each step: select next token based on the <b>probability distribution</b>
<b>Fast and efficient</b> , as it doesn't keep track of multiple sequences	Sampled token is <b>not guaranteed</b> to have the highest individual probability.
Can get stuck in repetitive loops; generated output is <b>not "creative"</b>	Allows for <b>greater diversity</b> in the generated output
Equivalent to sampling with <b><math>T = 0</math></b>	Sampling controlled by <b>several params</b>



# Self-consistency

- Technique that builds on chain-of-thought prompting.
- Idea: generate **multiple, diverse reasoning paths** through few-shot CoT, and use them to verify the **consistency** of the responses.
- Model gains the ability to explore multiple possibilities for the elicited reasoning chain and helps boost performance on arithmetic and common-sense reasoning tasks.



# Self-consistency improves performance

---

- Usual baseline for comparison: Chain-of-thought.
- Self-consistency generates **multiple chains of thought** and takes the **majority vote** of these multiple outputs as the final answer.
  - This improves reliability and accuracy.
  - The idea reminds of “model ensembling” in machine learning.
- Self-consistency improves CoT when used in a range of **common arithmetic and common-sense reasoning benchmarks**.
  - Use to improve on tasks that admit a unique correct answer, such as quantitative business questions.

# Limitations of self-consistency

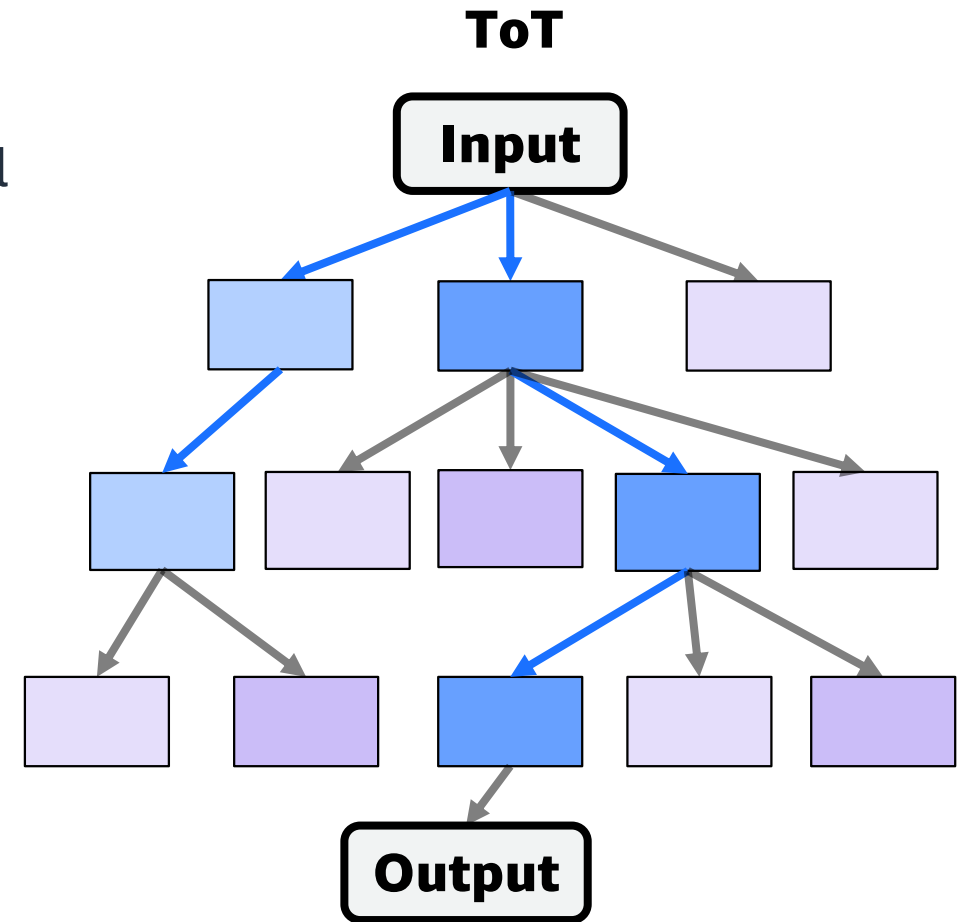
---

- Self-consistency incurs **more computational cost** than CoT.
  - In practice, generate a small number of paths (e.g. 5-10); in most cases the performance saturates quickly.
  - Potential solution: use self-consistency to generate better supervised data to fine-tune the model, and use the fine-tuned model to get improved accuracy in a single inference run.
- LLMs can often generate **incorrect or non-sensical** reasoning.
  - If most CoT paths are erroneous, self-consistency marginalization will fail.
  - Further work needed to understand what elicits proper reasoning

# Tree-of-thought

# Tree-of-thoughts prompting

- Strategy that guides LLMs to generate, evaluate, expand on, and decide among **multiple solutions**.
- Similar to problem-solving by humans: evaluate potential solutions before deciding on the most promising one.
- ToT builds a tree where ***thoughts*** represent coherent language sequences that serve as intermediate steps toward solving a problem.
  - Creative writing tasks such as ad copy generation
  - Mathematical reasoning tasks, crosswords, ...



# ToT helps decision-making

---

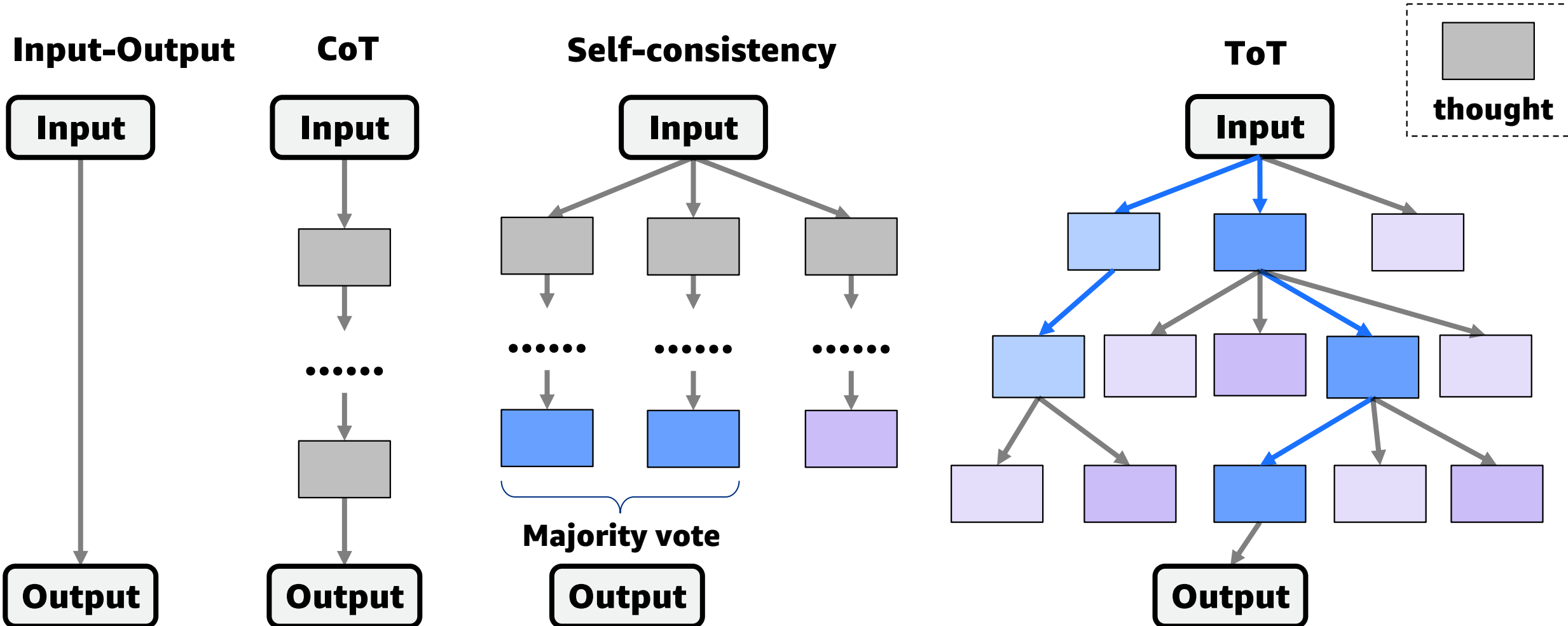
- Tree-of-thoughts prompting generalizes over chain of thought.
- ToT encourages exploration over "*thoughts*" that serve as intermediate steps for general problem solving with LLMs.
  - LLM generates thoughts (via CoT prompting).
  - Add tree-branching technique (breadth-first and depth-first search).
  - This enables **systematic exploration** with look-ahead and back-tracking.

# Benefits and limitations of ToT

---

- ToT is **especially effective** for tasks that involve important initial decisions, strategies for the future, and exploration of multiple solutions.
- Deliberate search **might not be necessary** for tasks for which powerful models already excel at.
- ToT requires **more resources** than sampling methods
  - Flexibility in tree construction allows users to customize cost-performance trade-offs.

# From standard to ToT prompting

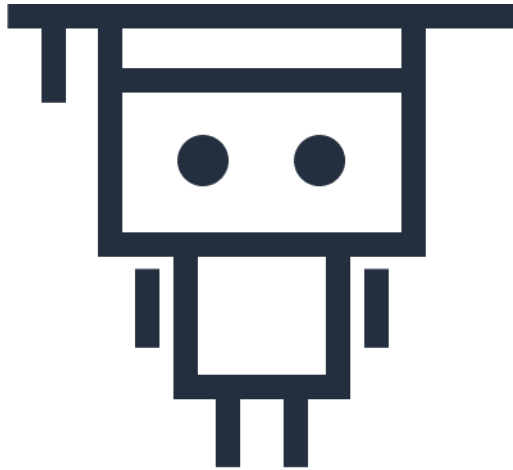


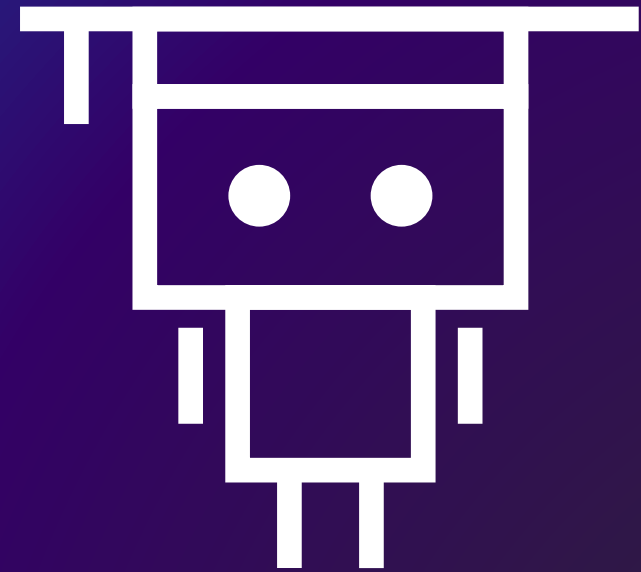


# Next lesson

---

- This lesson covered some advanced prompt engineering techniques.
- In the next lesson, you will explore multimodal solutions using foundation models.





Thank you!