PROMPTING TECHNIQUES

USED BY

PROMPT ENGINEERS

- Zero-shot Prompting
- Few-shot Prompting
- Chain-of-Thought Prompting
- Meta Prompting
- Self-Consistency
- Generate Knowledge Prompting
- Prompt Chaining
- Tree of Thoughts

SLIDE TO EXPLORE

1

ZERO-SHOT PROMPTING

Large Language Models (LLMs) like GPT-3.5 Turbo, GPT-4, and Claude 3 are trained on massive datasets and fine-tuned to follow instructions.

This enables zero-shot prompting — where you ask a model to perform a task without providing any examples.

Example :

Prompt:

```
Classify the text into neutral, negative or positive.
Text: I think the vacation is okay.
Sentiment:
```

Output:

```
Neutral
```

Note that in the prompt above we didn't provide the model with any examples of text alongside their classifications, the LLM already understands "sentiment" -- that's the zero-shot capabilities at work.

When zero-shot doesn't work, it's recommended to provide demonstrations or examples in the prompt which leads to **few-shot prompting.**

FEW-SHOT PROMPTING

While LLMs shine in zero-shot settings, they often struggle with more complex tasks.

👉 That's where few-shot prompting helps — by adding examples in the prompt, we guide the model toward better performance. For example :

Prompt:

```
A "whatpu" is a small, furry animal native to Tanzania. An example of a
sentence that uses the word whatpu is:
We were traveling in Africa and we saw these very cute whatpus.

To do a "farduddle" means to jump up and down really fast. An example
of a sentence that uses the word farduddle is:
```

Output:

```
When we won the game, we all started to farduddle in celebration.
```

With just 1 example, the model picks up pattern — this is 1-shot learning in action.

What Improves Few-Shot Performance? (Insights from Min et al. (2022))
✅ Keep the format consistent
✅ The label space and input distribution matter
✅ Even random labels or imperfect demonstrations are better than none!

2

FEW-SHOT PROMPTING

Surprisingly... Format Matters More Than Accuracy

Prompt:

```
This is awesome! // Negative
This is bad! // Positive
Wow that movie was rad! // Positive
What a horrible show! //
```

Output:

```
Negative
```

Even with random labels, the model predicts correctly — likely because of the structured format.

Even Messy Formats Work (Sometimes)

Prompt:

```
Positive This is awesome!
This is bad! Negative
Wow that movie was rad!
Positive
What a horrible show! --
```

Output:

```
Negative
```

Despite the messy structure, the model still gets it right — newer LLMs are becoming more robust to noisy prompts.

2 LIMITATIONS OF FEW-SHOT PROMPTING

Few-shot prompting is great for many tasks, but it often falls short when complex reasoning is involved.

Let’s look at an example 👇

Prompt:

```
The odd numbers in this group add up to an even number: 15, 32, 5, 13,
82, 7, 1.
A:
```

Output:

```
Yes, the odd numbers in this group add up to 107, which is an even
number.
```

This shows that the model struggles with multi-step reasoning, even with clear instructions.

Let's try to add few examples to see if few-shot prompting improves result

Prompt:

```
The odd numbers in this group add up to an even number: 4, 8, 9, 15, 12, 2, 1.
A: The answer is False.
The odd numbers in this group add up to an even number: 17,  10, 19, 4, 8, 12, 24.
A: The answer is True.
The odd numbers in this group add up to an even number: 16,  11, 14, 4, 8, 13, 24.
A: The answer is True.
The odd numbers in this group add up to an even number: 17,  9, 10, 12, 13, 4, 2.
A: The answer is False.
The odd numbers in this group add up to an even number: 15, 32, 5, 13, 82, 7, 1.
A:
```

Output:

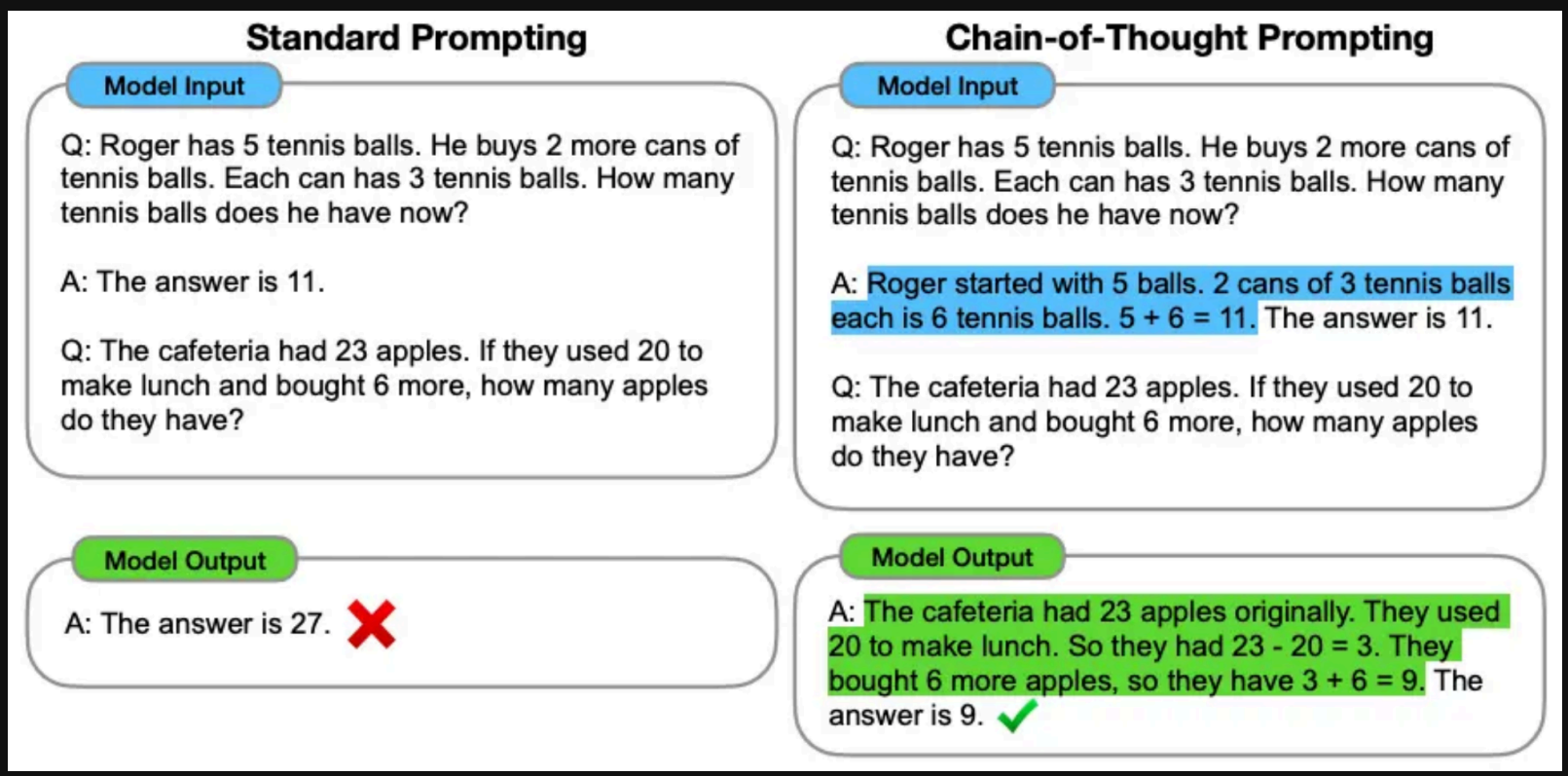
```
The answer is True.
```

What’s Going On? This task requires **more than pattern recognition….**

3

CHAIN-OF-THOUGHT PROMPTING

Chain of Thought (CoT) prompting improves performance on complex reasoning tasks by breaking them down into intermediate steps.



It's even more powerful when combined with few-shot prompting.
Let’s Revisit the Same Task (With CoT)

Prompt:

```
The odd numbers in this group add up to an even number: 4, 8, 9, 15, 12, 2, 1.
A: Adding all the odd numbers (9, 15, 1) gives 25. The answer is False.
The odd numbers in this group add up to an even number: 17,  10, 19, 4, 8, 12, 24.
A: Adding all the odd numbers (17, 19) gives 36. The answer is True.
The odd numbers in this group add up to an even number: 16,  11, 14, 4, 8, 13, 24.
A: Adding all the odd numbers (11, 13) gives 24. The answer is True.
The odd numbers in this group add up to an even number: 17,  9, 10, 12, 13, 4, 2.
A: Adding all the odd numbers (17, 9, 13) gives 39. The answer is False.
The odd numbers in this group add up to an even number: 15, 32, 5, 13, 82, 7, 1.
A:
```

Output:

```
Adding all the odd numbers (15, 5, 13, 7, 1) gives 41. The answer is False.
```

✅ Correct! This time, the model nailed it — because we helped it think step by step.

3

CHAIN-OF-THOUGHT PROMPTING

🔁 With Just One Example? Still Works.

Prompt:

```
The odd numbers in this group add up to an even number: 4, 8, 9, 15, 12, 2, 1.
A: Adding all the odd numbers (9, 15, 1) gives 25. The answer is False.
The odd numbers in this group add up to an even number: 15, 32, 5, 13, 82, 7, 1.
A:
```

Output:

```
Adding all the odd numbers (15, 5, 13, 7, 1) gives 41. The answer is False.
```

📈 Even 1-shot CoT prompting performs well!

🌱 Why It Works

✅ Breaks the task into manageable steps
✅ Makes the model's internal reasoning explicit
✅ Enables better performance on multi-step logic, math, and symbolic reasoning

✨ Emergent Ability

CoT prompting is considered an emergent behavior — one that appears only in sufficiently large models.

Up next: We'll look at **Meta Prompting**

4

META PROMPTING

Meta Prompting is an advanced technique that focuses on the structure and syntax of tasks, rather than specific examples or content.
🟦 Think format-first, not content-first.

Key Characteristics (Zhang et al., 2024)

1. Structure-Oriented – Emphasizes problem format over content
2. Syntax-Focused – Uses syntax to shape model output
3. Abstract Examples – Shows how to think, not what to think
4. Versatile – Works across many domains
5. Categorical – Leverages logical structures and type theory

Meta Prompting vs. Few-Shot Prompting

Feature	Meta Prompting	Few-Shot Prompting
Focus	Structure & Syntax	Content & Examples
Token Efficiency	✅ High	❌ Can be verbose
Zero-Shot Friendly	✅ Yes	❌ Needs examples
Comparison Fairness	✅ Consistent across models	❌ Influenced by specifics

The next example obtained from Zhang et al. (2024) demonstrates the difference between a structured meta prompt and a few-shot prompt for solving problems from the MATH benchmark:

META PROMPTING

Example: MATH Benchmark Prompting

Problem Statement:
- **Problem**: [question to be answered]

Solution Structure:
1. Begin the response with "Let's think step by step."
2. Follow with the reasoning steps, ensuring the solution process is broken down clearly and logically.
3. End the solution with the final answer encapsulated in a LaTeX-formatted box, $\boxed{\ldots}$, for clarity and emphasis.
4. Finally, state "The answer is [final answer to the problem].", with the final answer presented in LaTeX notation.

Figure 1: A structure meta prompt presented in markdown format for solving MATH [17] problems.

Problem: Find the domain of the expression $\frac{\sqrt{x-2}}{\sqrt{5-x}}$.
Solution: The expressions inside each square root must be non-negative. Therefore, $x-2 \geq 0$, so $x \geq 2$, and $5-x \geq 0$, so $x \leq 5$. Also, the denominator cannot be equal to zero, so $5-x > 0$, which gives $x < 5$. Therefore, the domain of the expression is $\boxed{[2,5)}$. Final Answer: The final answer is $[2,5)$. I hope it is correct.

Problem: If $\det \mathbf{A} = 2$ and $\det \mathbf{B} = 12$, then find $\det(\mathbf{AB})$.
Solution: We have that $\det(\mathbf{AB}) = (\det \mathbf{A})(\det \mathbf{B}) = (2)(12) = \boxed{24}$. Final Answer: The final answer is 24. I hope it is correct.

...

🧩 Few-Shot Prompt:
– Gives specific math problems and their solutions.
📐 Meta Prompt:
– Describes how problems are solved in structured steps,
without relying on exact examples.

This allows LLMs to generalize better across problems with less bias from example content.

4 META PROMPTING

💡 **Why Use Meta Prompting?**

✅ Saves tokens

✅ Boosts generalization

✅ Enables structured thinking

✅ Supports fair evaluation of models

Use it in:

🔢 Math & logic problems

👨‍💻 Coding challenges

💭 Abstract reasoning

📚 Theoretical questions

📌 Reminder:

Meta prompting assumes the LLM already has some internal understanding of the task.

Performance may vary on novel or unseen tasks — similar to zero-shot settings.

5

SELF CONSISTENCY

Proposed by Wang et al. (2022), Self-Consistency improves over standard Chain-of-Thought (CoT) prompting by:

✨ Replacing greedy decoding with diverse reasoning paths and selecting the most consistent answer.

🚫 Problem with Greedy Reasoning
Let’s start with a common mistake from naive prompting:

Prompt:

```
When I was 6 my sister was half my age. Now
I’m 70 how old is my sister?
```

Output:

```
35
```

This is incorrect, a reminder that complex reasoning needs better strategy.

✅ Self-Consistency to the Rescue
Let’s try again, but this time use CoT prompting with multiple sampled outputs.

By sampling multiple reasoning paths and aggregating them, we land on a more accurate, reliable output.

5

SELF CONSISTENCY

Prompt:

```
Q: There are 15 trees in the grove. Grove workers will plant trees in the grove
today. After they are done,
there will be 21 trees. How many trees did the grove workers plant today?
A: We start with 15 trees. Later we have 21 trees. The difference must be the
number of trees they planted.
So, they must have planted 21 - 15 = 6 trees. The answer is 6.
Q: If there are 3 cars in the parking lot and 2 more cars arrive, how many cars
are in the parking lot?
A: There are 3 cars in the parking lot already. 2 more arrive. Now there are 3 +
2 = 5 cars. The answer is 5.
Q: Leah had 32 chocolates and her sister had 42. If they ate 35, how many pieces
do they have left in total?
A: Leah had 32 chocolates and Leah's sister had 42. That means there were
originally 32 + 42 = 74
chocolates. 35 have been eaten. So in total they still have 74 - 35 = 39
chocolates. The answer is 39.
Q: Jason had 20 lollipops. He gave Denny some lollipops. Now Jason has 12
lollipops. How many lollipops
did Jason give to Denny?
A: Jason had 20 lollipops. Since he only has 12 now, he must have given the rest
to Denny. The number of
lollipops he has given to Denny must have been 20 - 12 = 8 lollipops. The answer
is 8.
Q: Shawn has five toys. For Christmas, he got two toys each from his mom and
dad. How many toys does
he have now?
A: He has 5 toys. He got 2 from mom, so after that he has 5 + 2 = 7 toys. Then
he got 2 more from dad, so
in total he has 7 + 2 = 9 toys. The answer is 9.
Q: There were nine computers in the server room. Five more computers were
installed each day, from
monday to thursday. How many computers are now in the server room?
A: There are 4 days from monday to thursday. 5 computers were added each day.
That means in total 4 * 5 =
20 computers were added. There were 9 computers in the beginning, so now there
are 9 + 20 = 29 computers.
The answer is 29.
Q: Michael had 58 golf balls. On tuesday, he lost 23 golf balls. On wednesday,
he lost 2 more. How many
golf balls did he have at the end of wednesday?
A: Michael initially had 58 balls. He lost 23 on Tuesday, so after that he has
58 - 23 = 35 balls. On
Wednesday he lost 2 more so now he has 35 - 2 = 33 balls. The answer is 33.
Q: Olivia has $23. She bought five bagels for $3 each. How much money does she
have left?
A: She bought 5 bagels for $3 each. This means she spent $15. She has $8 left.
Q: When I was 6 my sister was half my age. Now I'm 70 how old is my sister?
A:
```

Output 1:

```
When I was 6 my sister was half my age, so she was 3. Now I am 70, so she is 70
- 3 = 67. The answer is 67.
```

GENERATED KNOWLEDGE PROMPTING

LLMs are getting smarter — but they still make mistakes when the task requires commonsense or factual world knowledge. For example:

Prompt:

```
Part of golf is trying to get a higher point total than others. Yes or No?
```

Output:

```
Yes.
```

That’s wrong! In golf, lower scores win, not higher ones.

📚 Can We Teach the Model First?

Inspired by Liu et al. (2022), we try Knowledge Generation — prompting the model to generate background knowledge first, then answer the question.

Step 1: Generate Knowledge
We give the model a few samples like:

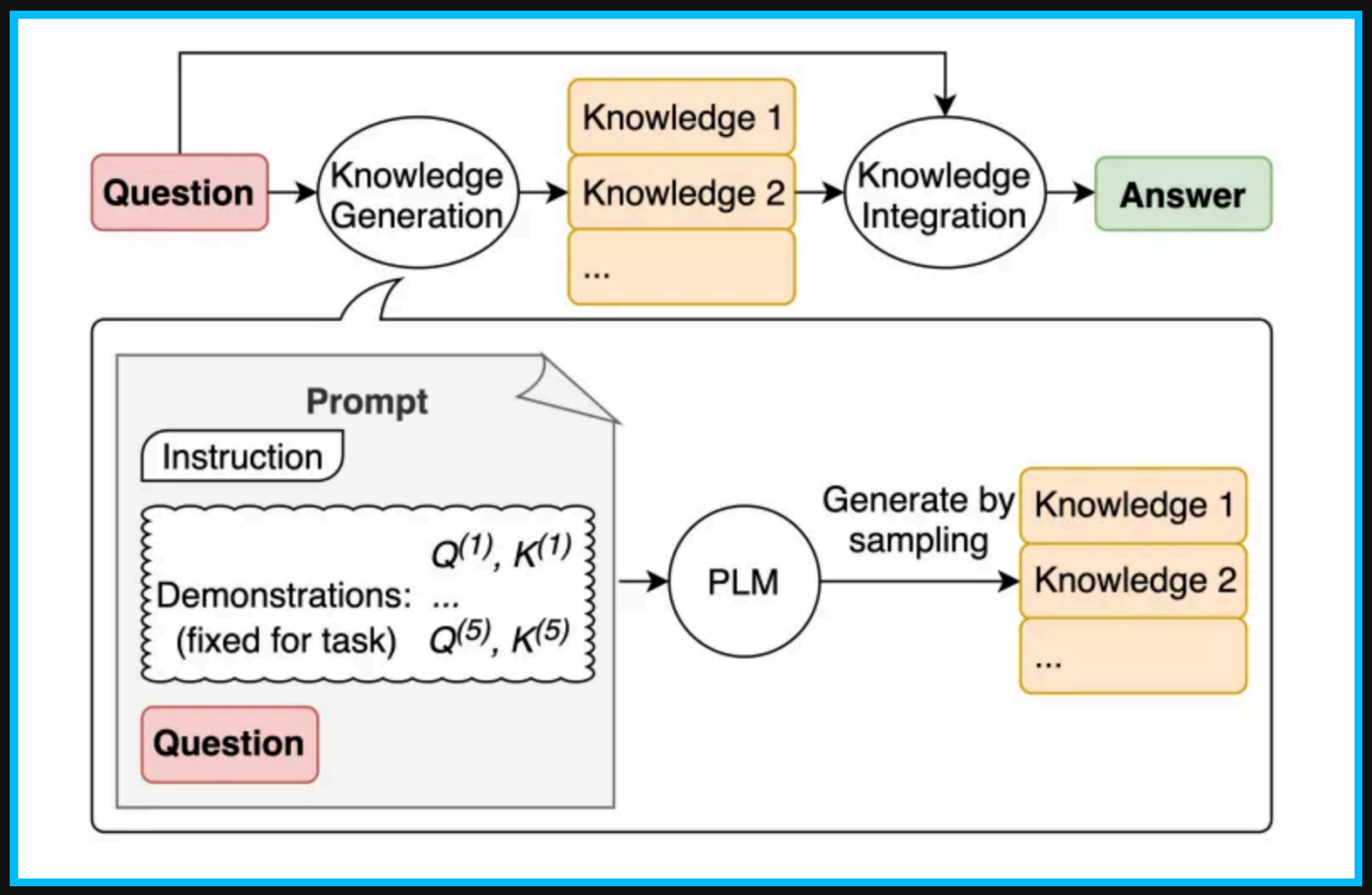
- Input: A fish is capable of thinking.
- ➤ Knowledge: Fish are more intelligent than they appear...
- Input: Part of golf is trying to get a higher point total than others.
- ➤ Knowledge 1: The goal is the least number of strokes.
- ➤ Knowledge 2: The player with the lowest score wins.

Now we inject this knowledge into the prompt

6

GENERATED KNOWLEDGE PROMPTING

Step 2: Use the Knowledge to Answer
Prompt:
Question: Part of golf is trying to get a higher point total than others. Yes or No?
Knowledge: The objective of golf is to play a set of holes in the least number of strokes...
Explain and Answer:
Answer 1 (Confidence: High):
No, the goal is to get the lowest score, not the highest.
✅ Correct!



🤔 Key Takeaway
Knowledge generation helps...
🧩 But how it's integrated matters.

- ✅ Properly structured knowledge → better accuracy
- ⚠️ Poorly phrased or misaligned knowledge → confusion

This technique works best when the model's prior knowledge is fuzzy or insufficient — especially in commonsense and factual reasoning tasks.

7

PROMPT CHAINING

Sometimes single prompt isn't enough. Prompt Chaining is a technique to improve accuracy, transparency, and control in LLM applications.

🧩 **What is Prompt Chaining?**

- You break a big task into smaller, manageable subtasks.
- Each subtask becomes its own prompt.
- The output of one prompt → becomes input to the next.

Why use prompt chaining?

✅ Handles complex tasks more effectively
✅ Boosts reliability and transparency
✅ Easier to debug and fine-tune
✅ Improves UX for LLM-powered apps
✅ Perfect for building personalized assistants

🟦 Use Case: Document Q&A with Prompt Chaining

Let's say you want to answer questions based on a large document. Rather than throwing the whole thing at the model in one go, break it into steps:

Prompt 1:

```
You are a helpful assistant. Your task is to help answer a question given in a document. The
first step is to extract quotes relevant to the question from the document, delimited by ####.
Please output the list of quotes using <quotes></quotes>. Respond with "No relevant quotes
found!" if no relevant quotes were found.
####
{{document}}
####
```

7

PROMPT CHAINING

Output of Prompt 1:

```
<quotes>
- Chain-of-thought (CoT) prompting[27]
- Generated knowledge prompting[37]
- Least-to-most prompting[38]
- Self-consistency decoding[39]
- Complexity-based prompting[41]
- Self-refine[42]
- Tree-of-thought prompting[43]
- Maieutic prompting[45]
- Directional-stimulus prompting[46]
- Textual inversion and embeddings[59]
- Using gradient descent to search for prompts[61][62][63][64]
- Prompt injection[65][66][67]
</quotes>
```

Prompt 2:

```
Given a set of relevant quotes (delimited by <quotes></quotes>) extracted from a document and the original document (delimited by ####), please compose an answer to the question. Ensure that the answer is accurate, has a friendly tone, and sounds helpful.
####
{{document}}
####
<quotes>
- Chain-of-thought (CoT) prompting[27]
- Generated knowledge prompting[37]
- Least-to-most prompting[38]
- Self-consistency decoding[39]
- Complexity-based prompting[41]
- Self-refine[42]
- Tree-of-thought prompting[43]
- Maieutic prompting[45]
- Directional-stimulus prompting[46]
- Textual inversion and embeddings[59]
- Using gradient descent to search for prompts[61][62][63][64]
- Prompt injection[65][66][67]
</quotes>
```

7

PROMPT CHAINING

Output of Prompt 2:

```
The prompting techniques mentioned in the document include:
1. Chain-of-thought (CoT) prompting[27]
2. Generated knowledge prompting[37]
3. Least-to-most prompting[38]
4. Self-consistency decoding[39]
5. Complexity-based prompting[41]
6. Self-refine[42]
7. Tree-of-thought prompting[43]
8. Maieutic prompting[45]
9. Directional-stimulus prompting[46]
10. Textual inversion and embeddings[59]
11. Using gradient descent to search for prompts[61][62][63][64]
12. Prompt injection[65][66][67]
Each of these techniques employs unique strategies to enhance or specify the interactions with
large language models to produce the desired outcomes.
```

Boom 💥 — clean, targeted answer powered by modular reasoning.

🧪 Bonus Exercise : **Want to polish it further?**
Design a third prompt to clean up citations like [27] from the final output before showing it to the user.

🚀 Pro Tip:
Prompt chaining works beautifully with tools like Claude, GPT-4, or other long-context models.

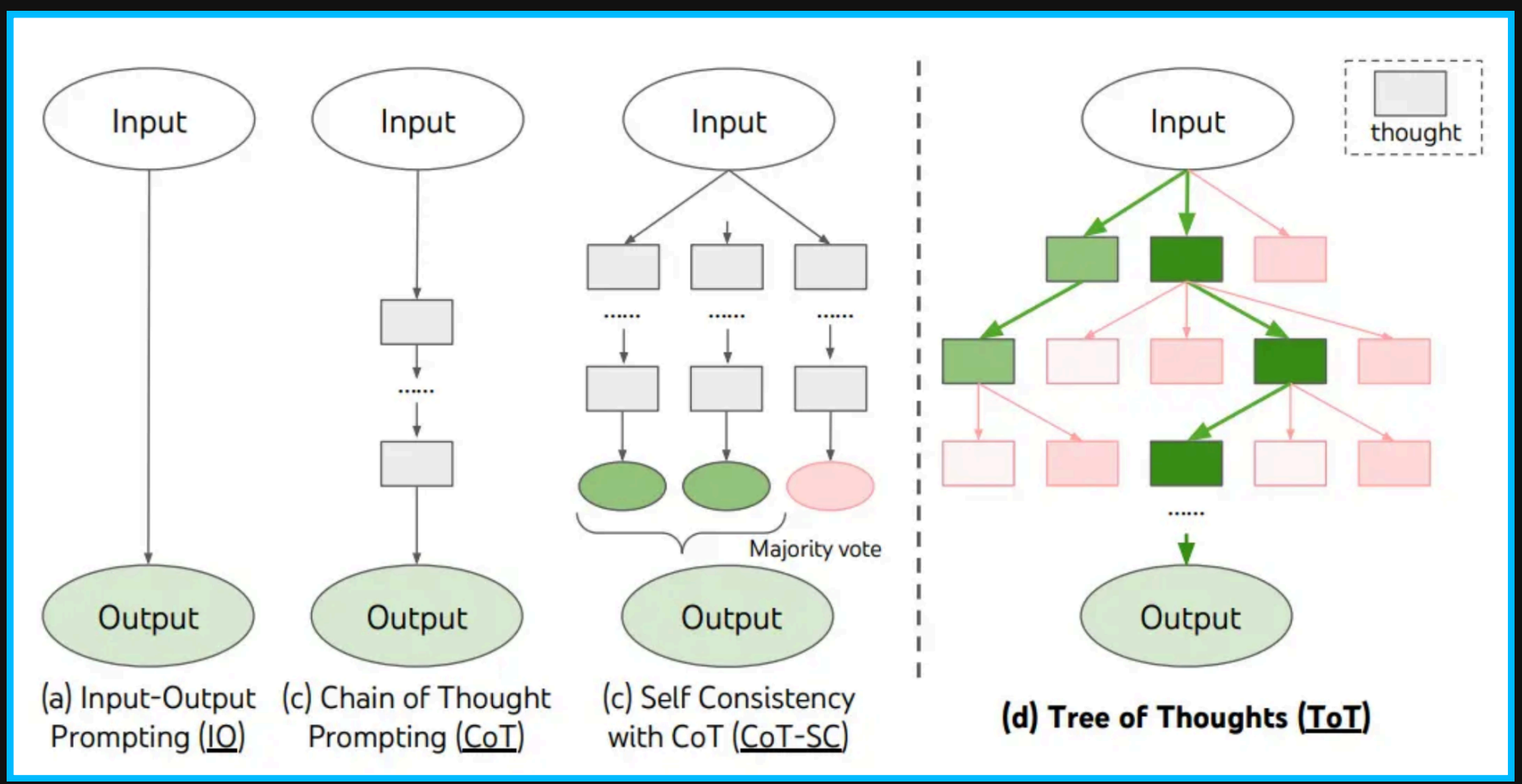
8

TREE OF THOUGHTS

Simple prompting isn't always enough for complex problem-solving.
That's where Tree of Thoughts (ToT) comes in — a new framework from Yao et al. (2023) and Long (2023).

🧠 **What is Tree of Thoughts?**
ToT treats reasoning like exploring a tree of possible thoughts.
Each "thought" is a step toward solving a problem.
LLMs generate & evaluate thoughts at each step — using search algorithms like BFS or DFS to explore multiple reasoning paths.

📐 Why It Matters
✅ Structured thinking with lookahead & backtracking
✅ Great for strategic tasks like math problems or puzzles
✅ Supports intermediate self-evaluation to discard bad paths
✅ Can boost accuracy beyond classic Chain-of-Thought prompting



TREE OF THOUGHTS

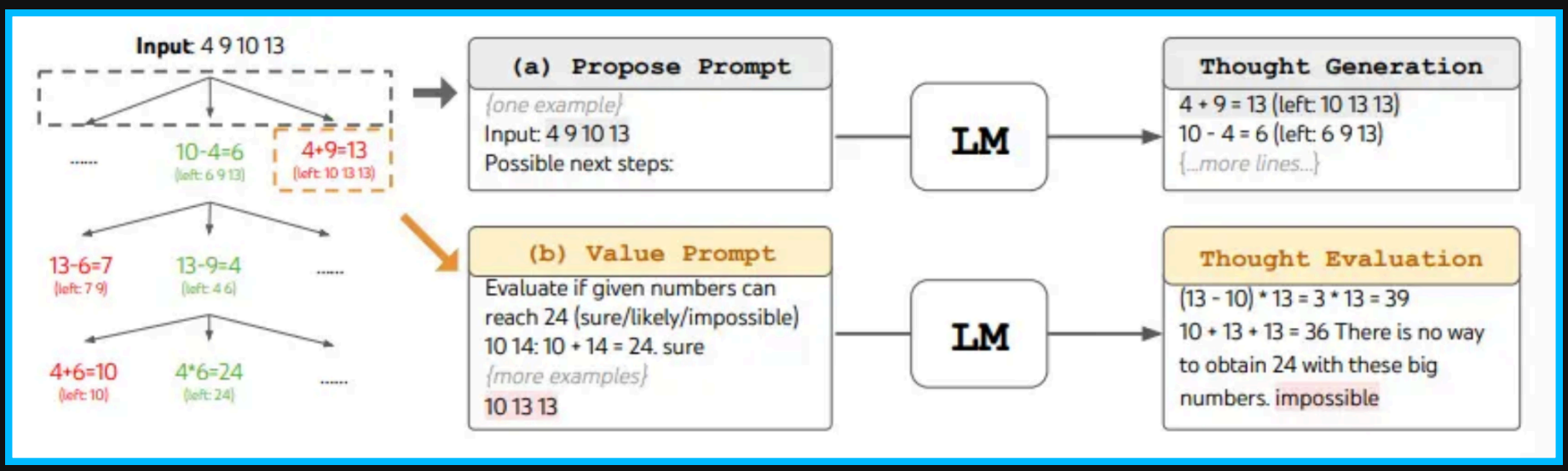
In the Game of 24 task, the problem is broken into 3 steps, each generating intermediate equations. At every step, the model keeps the top 5 (b=5) thought candidates.

To explore solutions, ToT uses BFS, prompting the model to evaluate each thought as:

“sure”, “maybe”, or “impossible”

based on whether it's likely to lead to 24.

This helps promote promising paths, discard bad ones, and keep uncertain options for further exploration. Each thought is sampled 3 times to assess reliability.



From the results reported in the figure below, ToT substantially outperforms the other prompting methods:

Method	Success
IO prompt	7.3%
CoT prompt	4.0%
CoT-SC (k=100)	9.0%
ToT (ours) (b=1)	45%
ToT (ours) (b=5)	**74%**
IO + Refine (k=10)	27%
IO (best of 100)	33%
CoT (best of 100)	49%

Table 2: Game of 24 Results.

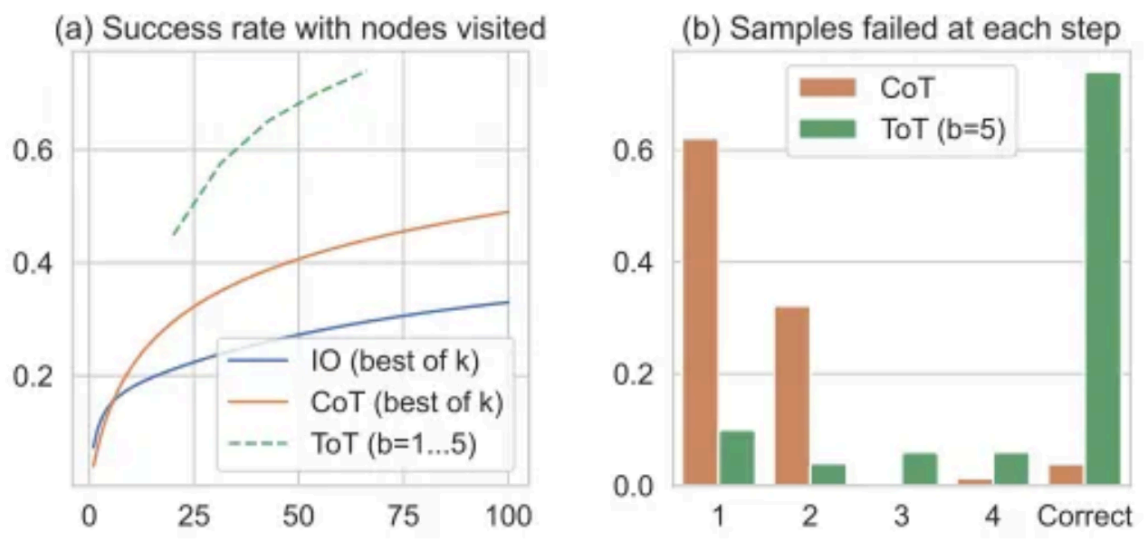


Figure 3: Game of 24 (a) scale analysis & (b) error analysis.

8

TREE OF THOUGHTS

📊 ToT vs Other Methods
🎯 ToT outperforms simple CoT prompting in complex tasks.
It brings deliberate search & reasoning into LLM workflows.
✅ Results show better success rates, especially in math & logic reasoning.

🤖 **RL-Enhanced ToT (Long 2023)**

- Adds a ToT Controller trained via reinforcement learning
- Learns when to backtrack and how far
- More adaptive than generic DFS/BFS
- Similar to how AlphaGo used learned strategies vs brute force

🗣️ ToT as a Prompting Style
You can also apply ToT as a lightweight prompting method:

> Imagine 3 experts each solving the problem step-by-step.
> If one realizes they're wrong, they leave.
> Now answer the question...

This is called Tree-of-Thought Prompting (Hulbert 2023).

💡 In Summary:
Tree of Thoughts =
🧱 **Multi-step thinking +** 🔍 **Search & evaluation +** 🤖 **Optional RL-driven control**
Perfect for building smarter, more strategic LLM systems.

Interested in more content like this?

Follow me :

OM NALINDE

linkedin.com/in/that-aum