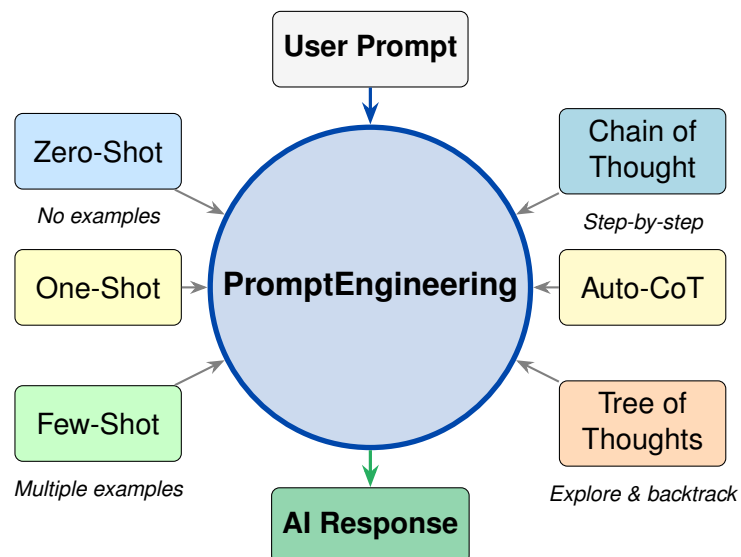


# Deep Learning for Perception

## Lecture 05: Prompt Engineering



### Topics Covered:

- Introduction to Prompt Engineering
- Zero-Shot Prompting
- One-Shot Prompting
- Few-Shot Prompting
- Chain-of-Thought (CoT) Prompting
- Zero-Shot CoT
- Automatic Chain-of-Thought (Auto-CoT)
- Tree of Thoughts (ToT)

**FAST-NUCES**

Department of Computer Science

# Contents

---

- 1 Introduction to Prompt Engineering** **2**
- 2 Zero-Shot, One-Shot, and Few-Shot Prompting** **4**
  - 2.1 Zero-Shot Prompting . . . . . 4
  - 2.2 One-Shot Prompting . . . . . 5
  - 2.3 Few-Shot Prompting . . . . . 5
- 3 Chain-of-Thought (CoT) Prompting** **7**
  - 3.1 Standard Prompting vs. Chain-of-Thought . . . . . 7
  - 3.2 Zero-Shot Chain-of-Thought Prompting . . . . . 8
  - 3.3 Automatic Chain-of-Thought (Auto-CoT) . . . . . 10
- 4 Tree of Thoughts (ToT)** **12**
  - 4.1 How ToT Works . . . . . 12
  - 4.2 Example: The Game of 24 . . . . . 13
  - 4.3 Simple ToT Prompt Template . . . . . 14
- 5 Summary** **15**
- 6 Glossary** **17**

## Advance Organizer — What You'll Learn

### The Big Picture

#### Building on prior knowledge:

- Understanding of Large Language Models (LLMs) and how they generate text
- Basic knowledge of natural language processing concepts
- Familiarity with the concept of machine learning inference

#### Learning Objectives:

1. Define **prompt engineering** and explain why it is essential for effective LLM interaction
2. Distinguish between **zero-shot**, **one-shot**, and **few-shot** prompting strategies
3. Apply **Chain-of-Thought (CoT)** prompting to improve reasoning in complex tasks
4. Understand the **Auto-CoT** algorithm for automatic demonstration generation
5. Explain the **Tree of Thoughts (ToT)** framework for exploration-based problem solving

**Real-World Applications:** Prompt engineering is used in chatbots, code generation tools, content creation systems, automated reasoning assistants, and virtually every application that interfaces with modern LLMs like GPT-4, Claude, or Gemini.

## 1 Introduction to Prompt Engineering

### Why It Matters

The quality of your prompt directly determines the quality of the AI's output. Many generative AI tools are freemium [*free with optional paid features*], meaning you may have limited queries. Crafting effective prompts ensures you get useful results without wasting attempts. Prompt engineering is the bridge between human intent and AI capability.

### Definition

**Prompt:** A **prompt** is a set of instructions a user creates to direct a generative AI tool's responses. It is the input text you provide to an AI model to elicit a desired response.

**Prompt Engineering:** **Prompt engineering** is a set of skills, strategies, and practices for creating prompts for generative AI tools to generate accurate, relevant content. It is the art and science of communicating effectively with AI systems.

### Analogy

#### The Taxi Driver Analogy

Imagine you are giving directions to a taxi driver in a foreign city. If you simply say "Take me somewhere nice," you might end up anywhere. But if you say "Take me to the

best-rated Italian restaurant within 2 km,” you will get exactly what you want. Prompt engineering is like learning to give precise directions to AI—the better your instructions, the more accurately the AI takes you where you want to go.

## 2 Zero-Shot, One-Shot, and Few-Shot Prompting

### Connection to Prior Learning

#### How This Connects to What You Know:

- In machine learning, models learn from training data. These prompting strategies control how much “training-like” information you provide at inference time.
- The terms “zero-shot,” “one-shot,” and “few-shot” come from the broader field of transfer learning and meta-learning.

When working with generative AI, one of the most fundamental decisions is whether to include examples in your prompt. This leads to three distinct prompting strategies.

### 2.1 Zero-Shot Prompting

#### Definition

**Zero-shot prompting** is when you give a generative AI tool instructions *without* giving any examples. It relies on the assumption that the AI will understand your request based solely on its pre-trained knowledge.

Zero-shot prompting works because modern LLMs have been trained on massive amounts of text data. They learned patterns, facts, and relationships from billions of documents during training.

#### Concrete Example

##### Zero-shot Prompt:

“Classify this statement as positive or negative: ‘The customer service was incredibly unhelpful and I waited for 2 hours.’”

##### Model Output:

“Negative. The statement expresses frustration about unhelpful service and a long wait time.”

Notice we provided **no examples** of what positive or negative statements look like. We relied completely on the AI’s pre-trained knowledge. Zero-shot is a good starting point—if results are unsatisfactory, consider adding examples.

#### Memory Hook

**Zero-shot = Zero examples.** The AI uses only its pre-trained knowledge. Always start here first!

## 2.2 One-Shot Prompting

### Definition

**One-shot prompting** is when you give the generative AI tool a *single example* before giving instructions. This example serves as a template that shows the AI the format, style, or pattern you expect.

### Concrete Example

**One-shot Prompt:**

Slang: “No cap, that movie was amazing.”

Formal: “That movie was amazing, for real.”

Now translate: “I think I spent time with family at mall.”

**Model Output:**

Formal: “I believe I spent quality time with my family at the mall.”

By providing one example, we showed the model exactly what transformation we wanted. The model learned from this single demonstration and applied the same pattern.

## 2.3 Few-Shot Prompting

### Definition

**Few-shot prompting** is when you provide *multiple examples* (typically 2–5) instead of just one. This creates a mini dataset within your prompt to help the AI recognize patterns more reliably.

### Concrete Example

**Few-shot Prompt:**

Generate five more romantasy [*romantic fantasy*] novel titles in this style.

Examples:

- The Tender Alchemist’s Gambit
- A Court of Whispers and Thorns
- The Shadow Prince’s Redemption

**Model Output:**

- The Wicked Enchanter’s Promise
- A Dance of Devils and Dreamers
- The Broken Vampire’s Redemption
- A Symphony of Shadows and Starlight
- The Forbidden Witch’s Reckoning

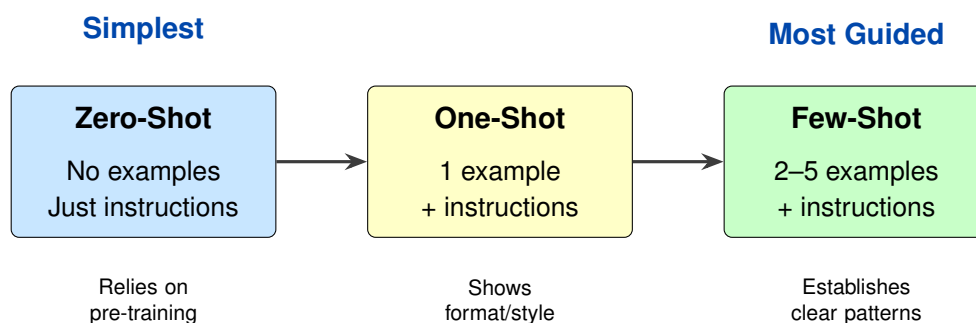


Figure 1: Progression from zero-shot to few-shot prompting. Each step adds more guidance for the model.

### Key Comparison

#### When to Use Each Approach

Approach	Best Used When...
Zero-shot	The task is simple, well-defined, or the model likely understands it from training (e.g., translation, summarization, basic classification).
One-shot	You need to demonstrate a specific format or style, but the pattern is straightforward.
Few-shot	The task requires understanding subtle patterns, the output format is complex, or zero/one-shot produces inconsistent results.

### Self-Test

**Q1:** What is the main difference between zero-shot and few-shot prompting?

*A: Zero-shot provides no examples; few-shot provides multiple examples to establish patterns.*

**Q2:** If you wanted an AI to classify emails as “urgent” or “non-urgent,” which strategy would you start with?

*A: Start with zero-shot (simplest). If results are inconsistent, add examples (few-shot).*

**Q3:** True or False: Few-shot prompting always produces better results than zero-shot.

*A: False. For simple tasks, zero-shot may work equally well and is more efficient.*

### 3 Chain-of-Thought (CoT) Prompting

#### Why It Matters

Standard prompting often fails on tasks requiring multi-step reasoning (arithmetic word problems, logical puzzles, complex analysis). This happens because the model tries to produce the final answer immediately without working through intermediate steps. **Chain-of-Thought prompting** solves this by encouraging the model to “think out loud.”

#### Definition

**Chain-of-Thought (CoT) prompting** is a technique that enables complex reasoning capabilities in language models by encouraging them to generate *intermediate reasoning steps* before arriving at a final answer. Instead of jumping from question to answer, the model “shows its work.”

#### Analogy

##### The “Show Your Work” Analogy

Imagine solving  $347 \times 28$  in your head. If you try to compute directly, you will likely make errors. But if you break it down— $347 \times 20 = 6940$ , then  $347 \times 8 = 2776$ , then  $6940 + 2776 = 9716$ —you can solve it reliably.

Chain-of-Thought prompting teaches AI to use this same “show your work” approach that your math teacher always asked for!

#### 3.1 Standard Prompting vs. Chain-of-Thought

The key difference lies in what the model outputs. With standard prompting, the model outputs only the final answer. With CoT, the model outputs the reasoning process followed by the answer.

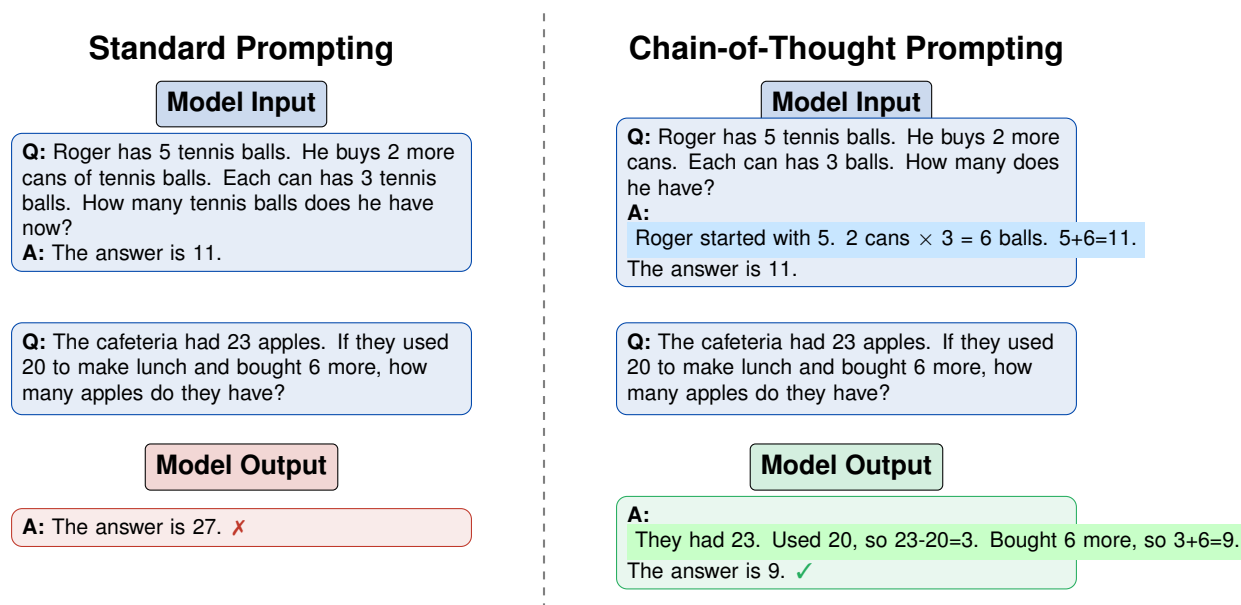


Figure 2: Comparison of standard prompting (left) and Chain-of-Thought prompting (right). The highlighted reasoning steps in CoT lead to correct answers.

### Memory Hook

**CoT = “Show your work.”** Just like a math teacher asks students to show their reasoning, CoT prompting asks the AI to show its reasoning steps. This makes errors visible and often prevents them.

## 3.2 Zero-Shot Chain-of-Thought Prompting

A remarkable discovery is that you do not always need examples to trigger chain-of-thought reasoning. By simply appending the phrase **“Let’s think step by step”** to your prompt, you can achieve similar results.

### Definition

**Zero-shot CoT prompting** involves adding the instruction “Let’s think step by step” (or similar phrase) to the prompt. This triggers the model to generate intermediate reasoning steps without requiring any examples.

### Concrete Example

#### Standard Zero-shot Prompt:

I went to the market and bought 10 apples. I gave 2 apples to the neighbor and 2 to the repairman. I then went and bought 5 more apples and ate 1. How many apples did I remain with?

**Output:** 10 ✗ (incorrect)

**Zero-shot CoT Prompt:**

I went to the market and bought 10 apples. I gave 2 apples to the neighbor and 2 to the repairman. I then went and bought 5 more apples and ate 1. How many apples did I remain with? **Let's think step by step.**

**Output:**

First, you started with 10 apples.

You gave 2 to the neighbor:  $10 - 2 = 8$  apples.

You gave 2 to the repairman:  $8 - 2 = 6$  apples.

You bought 5 more:  $6 + 5 = 11$  apples.

You ate 1:  $11 - 1 = 10$  apples.

**The answer is 10. ✓**

**(a) Few-shot**

**Q:** Roger has 5 tennis balls. He buys 2 cans of 3 balls. How many?

**A:** The answer is 11.

**Q:** A juggler juggles 16 balls. Half are golf balls, half of those are blue. How many blue golf balls?

**A:**

(Output) The answer is 8. ✗

**(c) Zero-shot**

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

**A:** The answer (arabic numerals) is

(Output) 8 ✗

**(b) Few-shot-CoT**

**Q:** Roger has 5 balls. Buys 2 cans of 3 balls.

**A:**  $5 \text{ balls} + (2 \times 3) = 6$ . Total: 11. Answer: 11.

**Q:** 16 balls. Half are golf balls, half of golf balls are blue. Blue golf balls?

**A:**

(Output)  $16/2=8$  golf.  $8/2=4$  blue golf. Answer: 4. ✓

**(d) Zero-shot-CoT**

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

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

(Output)

16 total. Half=8 golf. Half of 8=4 blue. Answer: 4. ✓

Figure 3: Comparison of four prompting strategies. Only methods with chain-of-thought reasoning (b and d) produce correct answers on multi-step problems.

**Key Insight**

The comparison above demonstrates that chain-of-thought reasoning—whether triggered by examples (few-shot-CoT) or by a simple phrase (zero-shot-CoT)—dramatically improves performance on reasoning tasks. Both few-shot and zero-shot *without* CoT fail, while both CoT variants succeed.

### 3.3 Automatic Chain-of-Thought (Auto-CoT)

#### Definition

**Automatic Chain-of-Thought (Auto-CoT)** is an approach that automatically generates diverse reasoning demonstrations. Instead of hand-crafting examples, Auto-CoT uses the LLM itself to generate reasoning chains, combined with intelligent sampling strategies.

Auto-CoT eliminates manual effort by leveraging LLMs with “Let’s think step by step” to generate reasoning chains automatically. The key innovation is in how it selects demonstration questions.

#### Algorithm

##### Auto-CoT Algorithm (Two Stages)

##### Stage 1: Question Clustering

1. Take all questions from a given dataset
2. Partition *[divide]* questions into  $k$  clusters based on semantic similarity *[how similar their meanings are]*
3. Each cluster represents a different “type” of question

##### Stage 2: Demonstration Sampling

1. For each cluster, select a representative question using simple heuristics *[rules of thumb]*
2. Heuristics include: question length (not too long or short), number of reasoning steps
3. Generate a reasoning chain for each selected question using Zero-Shot-CoT
4. Compile these question-reasoning pairs as demonstrations

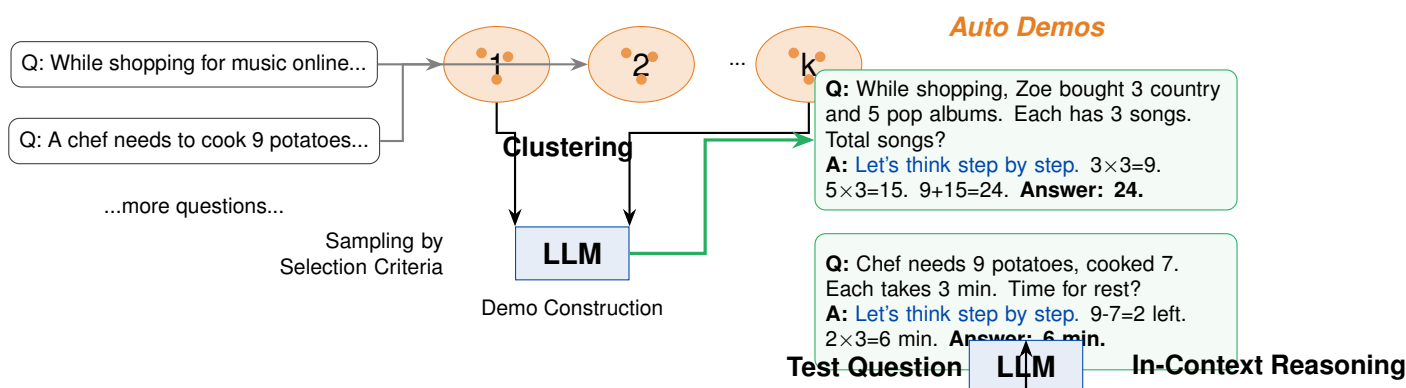


Figure 4: Auto-CoT architecture. Questions are clustered, representative questions are selected from each cluster, reasoning chains are generated automatically, and these become demonstrations for in-context learning.

### Why It Matters

Auto-CoT addresses two key limitations of manual CoT: (1) the labor required to write detailed reasoning chains, and (2) the risk of selecting non-representative or biased examples. By clustering and sampling, Auto-CoT ensures diversity in demonstrations, improving generalization.

### Self-Test

**Q1:** What is the key phrase that triggers Zero-Shot CoT reasoning?

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

**Q2:** Why does Chain-of-Thought prompting improve performance on math word problems?

*A: It forces the model to show intermediate steps, making reasoning explicit and reducing errors.*

**Q3:** In Auto-CoT, why is question clustering performed before selecting demonstrations?

*A: To ensure diversity—selecting from different clusters covers different question types.*

## 4 Tree of Thoughts (ToT)

### Why It Matters

For complex tasks requiring exploration or strategic lookahead [*thinking ahead about consequences*], even Chain-of-Thought falls short. These include:

- Mathematical puzzles with multiple solution paths
- Planning problems where early decisions affect later options
- Creative tasks requiring consideration of alternatives
- Strategic games requiring multi-step thinking

### Definition

**Tree of Thoughts (ToT)** is a framework that generalizes Chain-of-Thought prompting by allowing the model to explore multiple reasoning paths simultaneously. Unlike CoT's single linear chain, ToT maintains a tree structure where each node is a "thought" (intermediate step), and the model can explore, evaluate, and **backtrack** [*go back to try a different path*] as needed.

### Analogy

#### The Maze Analogy

Chain-of-Thought is like walking through a maze following one path—if you hit a dead end, you are stuck.

Tree of Thoughts is like having a bird's-eye view of the maze, where you can see multiple paths, evaluate which ones look promising, and backtrack when you realize a path leads nowhere. You explore multiple branches and choose the best route.

### 4.1 How ToT Works

ToT combines the model's ability to generate and evaluate thoughts with search algorithms for systematic exploration.

### Algorithm

#### Tree of Thoughts Framework (Four Components)

##### 1. Thought Decomposition

Break down the problem into intermediate steps. Each "thought" should be small enough to evaluate but large enough to be meaningful.

##### 2. Thought Generation

At each step, generate multiple candidate thoughts (typically 3–5). The model proposes different ways to proceed.

##### 3. Thought Evaluation

Use the LLM to evaluate how promising each thought is. Rate thoughts as "sure"

(definitely leads to solution), “likely” (probably good), or “impossible” (definitely wrong).

#### 4. Search Algorithm

- **Breadth-First Search (BFS):** Explore all thoughts at one level before going deeper. Good for considering many alternatives.
- **Depth-First Search (DFS):** Go deep into one path before trying others. Good when solutions require many steps.

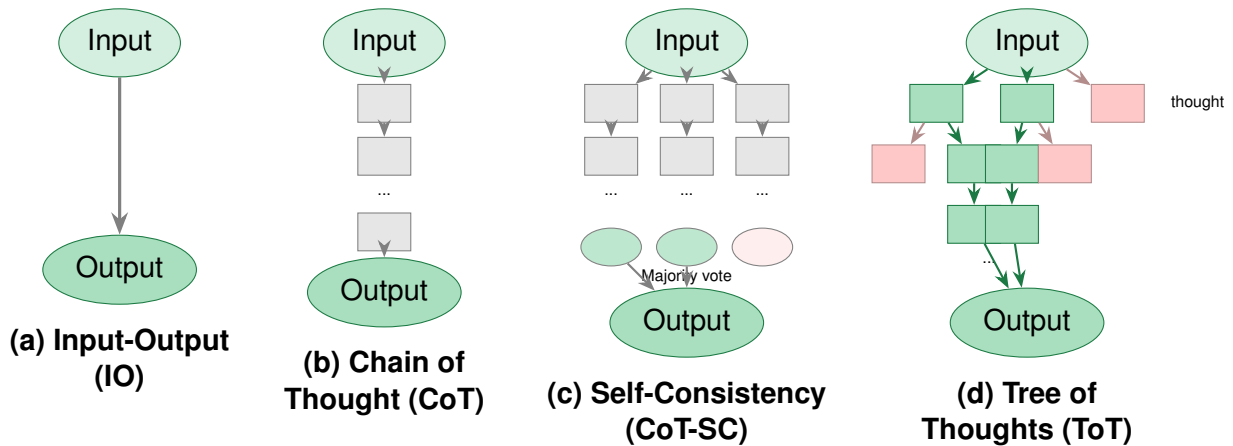


Figure 5: Evolution of prompting strategies. (a) IO: direct mapping; (b) CoT: single reasoning chain; (c) CoT-SC: multiple chains with voting; (d) ToT: tree exploration with evaluation and backtracking.

### Key Comparison

#### Understanding the Evolution

Method	Key Characteristic
IO	Single forward pass from input to output. No intermediate steps.
CoT	Single chain of reasoning steps. Cannot explore alternatives.
CoT-SC	Multiple independent chains, then majority vote. Chains do not interact.
ToT	Tree structure with evaluation at each step. Can explore, evaluate, and backtrack. Thoughts can build on other thoughts.

## 4.2 Example: The Game of 24

The Game of 24 demonstrates ToT’s power. Given four numbers, use +, -, ×, ÷ to make exactly 24. For example, with 4, 9, 10, 13:  $(10 - 4) \times (13 - 9) = 6 \times 4 = 24$ .

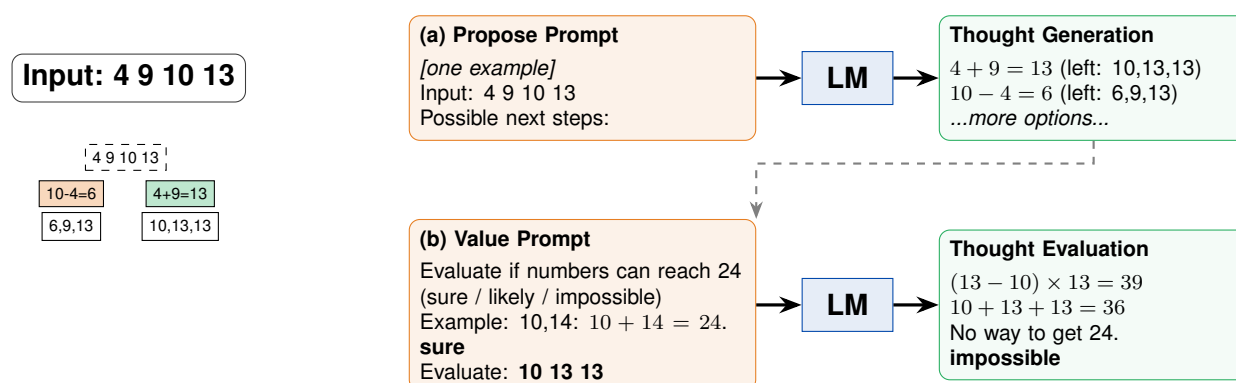


Figure 6: Tree of Thoughts applied to the Game of 24. The Propose Prompt generates candidate steps; the Value Prompt evaluates whether remaining numbers can reach 24. Thoughts rated “impossible” are pruned.

### Memory Hook

**ToT = Explore + Evaluate + Backtrack.** The key insight is that ToT does not just generate thoughts—it evaluates them and can abandon paths that are not working. This “lookahead” ability is what makes ToT powerful.

## 4.3 Simple ToT Prompt Template

For tasks where you want ToT-like reasoning without implementing the full framework:

### Prompt

Imagine three different experts are answering this question.  
 All experts will write down 1 step of their thinking, then share it with the group.  
 Then all experts will go on to the next step, etc.  
 If any expert realises they’re wrong at any point then they leave.  
 The question is: [YOUR QUESTION HERE]

This template simulates ToT by having the model role-play as multiple “experts” who share reasoning, evaluate progress, and abandon incorrect paths.

### Self-Test

**Q1:** What limitation of CoT does Tree of Thoughts address?

*A: CoT follows a single path and cannot backtrack. ToT explores multiple paths and can abandon wrong ones.*

**Q2:** Name the two search algorithms commonly used in ToT.

*A: Breadth-First Search (BFS) and Depth-First Search (DFS).*

**Q3:** In the Game of 24 example, why is the Value Prompt important?

*A: It evaluates whether the remaining numbers can possibly reach 24, allowing pruning of impossible paths.*

## 5 Summary

### Key Takeaways

#### 1. Prompt Engineering Fundamentals

- A **prompt** is instructions given to an AI; **prompt engineering** is the skill of crafting effective prompts
- Quality of prompt directly determines quality of output

#### 2. Shot-Based Prompting Strategies

- **Zero-shot**: No examples, relies on pre-trained knowledge
- **One-shot**: Single example to demonstrate format/style
- **Few-shot**: Multiple examples to establish clear patterns

#### 3. Chain-of-Thought (CoT) Prompting

- Encourages intermediate reasoning steps before final answer
- **Zero-shot CoT**: Triggered by “Let’s think step by step”
- **Auto-CoT**: Automatically generates diverse demonstrations via clustering

#### 4. Tree of Thoughts (ToT)

- Explores multiple reasoning paths as a tree structure
- Uses search algorithms (BFS/DFS) with evaluation and backtracking
- Best for complex problems requiring strategic exploration

Strategy	Best For	Example Tasks
Zero-Shot	Simple, well-defined tasks	Translation, summarization, basic Q&A
One-Shot	Format/style demonstration	Format conversion, style matching
Few-Shot	Pattern recognition	Categorization, styled text generation
CoT	Multi-step reasoning	Math word problems, logical puzzles
Zero-Shot CoT	Reasoning without examples	General reasoning tasks
Auto-CoT	Large-scale deployment	Automated systems, varied domains
ToT	Exploration & backtracking	Puzzles, planning, creative problem-solving

### Memory Hook

#### Decision Flow:

1. Start with **Zero-Shot**. If results are unsatisfactory...
2. Try **One-Shot or Few-Shot** to demonstrate patterns. If reasoning is needed...
3. Add **Chain-of-Thought** (examples or “Let’s think step by step”). If exploration helps...
4. Consider **Tree of Thoughts** for the most complex problems.

### Self-Test

#### Final Review Questions:

**Q1:** Define prompt engineering in your own words.

*A: The practice of crafting effective instructions for AI systems to produce desired outputs.*

**Q2:** What is the difference between few-shot and few-shot CoT prompting?

*A: Few-shot shows examples with answers only; few-shot CoT shows examples with reasoning steps.*

**Q3:** Why does “Let’s think step by step” improve model performance?

*A: It triggers the model to generate intermediate reasoning, making the thought process explicit.*

**Q4:** Compare CoT-SC and ToT. What can ToT do that CoT-SC cannot?

*A: ToT can evaluate thoughts mid-process and backtrack. CoT-SC runs independent chains without interaction.*

## 6 Glossary

Term	Definition
<b>Prompt</b>	A set of instructions given to an AI model to elicit a desired response.
<b>Prompt Engineering</b>	The skill and practice of crafting effective prompts for generative AI systems.
<b>Zero-shot Prompting</b>	Giving instructions to an AI without providing any examples.
<b>One-shot Prompting</b>	Providing a single example along with instructions to guide the AI's response format.
<b>Few-shot Prompting</b>	Providing multiple examples (typically 2–5) to establish patterns for the AI to follow.
<b>Chain-of-Thought (CoT)</b>	A prompting technique that encourages the model to generate intermediate reasoning steps.
<b>Zero-shot CoT</b>	Triggering chain-of-thought reasoning by adding “Let's think step by step” without examples.
<b>Auto-CoT</b>	An algorithm that automatically generates diverse CoT demonstrations via clustering and sampling.
<b>Tree of Thoughts (ToT)</b>	A framework that explores multiple reasoning paths as a tree, with evaluation and backtracking.
<b>Backtracking</b>	The ability to abandon a reasoning path and return to try an alternative approach.
<b>BFS (Breadth-First Search)</b>	A search algorithm that explores all options at one level before going deeper.
<b>DFS (Depth-First Search)</b>	A search algorithm that explores one path completely before trying alternatives.
<b>Heuristics</b>	Rules of thumb or practical guidelines used for decision-making.
<b>Semantic Similarity</b>	A measure of how similar the meanings of two pieces of text are.
<b>LLM (Large Language Model)</b>	An AI model trained on large amounts of text data to understand and generate language.

---

*End of Lecture Notes*

For additional resources, visit: <https://www.promptingguide.ai/>