**Prompt Engineering: Theory and Practice**

**Jukkapan Wirunrat**

**Member of EMBC, IEEE Thailand Section**

**Preface**

In today’s AI-driven world, **prompt engineering** has become a vital skill for effectively working with large language models (LLMs). Far more than just writing questions, it is the craft of shaping inputs that guide models to produce useful, accurate, and aligned outputs.

This textbook, *Prompt Engineering: Theory and Practice*, offers a structured path from foundational concepts to advanced prompting techniques. It is designed for students, researchers, developers, and anyone seeking to understand how prompts influence model behavior.

You will learn how to write and evaluate prompts, apply patterns like zero-shot and chain-of-thought prompting, reduce hallucinations, and simulate dialogue—all while grounding your knowledge in both theory and practice.

Prompt engineering is where human creativity meets AI capability. This book equips you to bridge that gap with clarity, precision, and purpose.

**Jukkapan W.**  
Editor and Author, Member of EMBC, IEEE Thailand Section  
June 2025

**Chapter 1: Foundations of Prompt Structure and Design**

**Learning Objectives**

After completing this chapter, students will be able to:

* Define the essential components of a well-structured prompt.
* Explain how prompt structure influences language model behavior.
* Compare different prompting paradigms: zero-shot, few-shot, chain-of-thought, and role-based.
* Design and evaluate prompts for various tasks and user goals.
* Recognize common prompt pitfalls and implement best practices for improvement.

**1.1 Introduction: The Nature of Prompt Engineering**

Prompt engineering is the deliberate design of input queries to steer large language models (LLMs) like GPT-4 or Claude toward producing useful, accurate, and stylistically appropriate responses. Unlike classical programming, which uses explicit logic and syntax, prompt engineering programs through linguistic cues and patterns, leveraging the model’s pretraining on vast corpora of human text.

**Key Insight**: Prompt engineering is the practice of instructing probabilistic reasoning engines through natural language. It is programming by implication, not instruction.

The rise of LLMs as general-purpose tools has elevated prompt engineering into a critical human–AI interface discipline.

**1.2 How Language Models Interpret Prompts**

LLMs operate on token sequences, generating outputs by predicting the most likely next token in a given context. During inference:

1. The model receives a sequence of tokens (the prompt).
2. It computes a probability distribution over possible next tokens.
3. It selects a token (often using sampling or greedy decoding).
4. This process continues recursively until a stopping condition is met.

This statistical approach means:

* Prompt structure affects **token prediction** and therefore output meaning.
* Even small changes in wording or punctuation can lead to major changes in response.
* LLMs do not "understand" in a human sense — they emulate understanding by reproducing patterns learned from data.

**1.3 Components of a Well-Formed Prompt**

An effective prompt typically includes four elements:

**1. Instruction**

The task directive, usually stated clearly and concisely.

Summarize the following article in one paragraph.

**2. Context**

Additional background, user role, tone, or framing. This helps the model simulate expertise or situational awareness.

You are a medical doctor explaining treatment options to a concerned patient.

**3. Input Data**

The content the model is expected to act on — text, table, or scenario.

Patient symptoms: fever, dry cough, shortness of breath, fatigue.

**4. Output Format Guide**

Specifications on how the response should be structured.

Respond in a bullet list, with each symptom linked to a possible diagnosis.

*Best Practice*: Structure = clarity. Use delimiters like quotes, headings, or bullet points.

**1.4 Prompt Design Patterns**

Different prompting strategies produce different model behaviors. Each is suited to specific tasks.

**A. Zero-Shot Prompting**

No examples are provided; the model relies on learned generalization.

Translate the following sentence into French: “The weather is nice today.”

**Use for**:

* Common tasks (e.g., translation, summarization).
* When the model is expected to perform without guidance.

**B. Few-Shot Prompting**

A few input–output examples are included to show the model a pattern.

Q: What’s the capital of France?

A: Paris

Q: What’s the capital of Japan?

A: Tokyo

Q: What’s the capital of Italy?

A:

**Use for**:

* Custom or abstract tasks.
* Training the model on structure, tone, or logic through in-prompt examples.

**C. Chain-of-Thought (CoT) Prompting**

Encourages step-by-step reasoning before providing an answer.

If a car travels at 60 km/h for 2 hours, how far does it travel?

Let’s think step by step.

**Why it works**:  
Induces explicit intermediate reasoning steps, which improves performance on logic, math, and reasoning tasks (Wei et al., 2022).

**D. Role-Based Prompting**

The prompt assigns the model a role or persona.

You are a nutritionist. Explain the benefits of fiber to a 10-year-old.

**Use for**:

* Tone control
* Domain adaptation
* Communicating with specific audiences

**1.5 Prompt Effectiveness: Best Practices**

|  |  |  |
| --- | --- | --- |
| **Principle** | **Explanation** | **Example** |
| Be Specific | Vagueness leads to generic outputs. | ✅ “List 3 side effects of aspirin” vs  ❌ “Tell me about aspirin.” |
| Structure Matters | Clear formatting improves response reliability. | Use headings, bullet points, or delimiters |
| Simulate Reasoning | Step-by-step output is more accurate. | “Let’s think through this step-by-step.” |
| Use Role Framing | Helps tune tone and vocabulary. | “You are a government policy advisor...” |
| Limit Length | Helps avoid rambling or irrelevant output. | “Limit response to 150 words.” |

**1.6 Prompt Engineering as Cognitive Interface Design**

Prompting is fundamentally **user experience design for AI**.

**A. Prompting as UX**

A good prompt answers the following design questions:

* What is the user’s goal?
* What background is assumed?
* What ambiguity can be eliminated?
* What should the tone or format be?

**B. Prompting as Scientific Inquiry**

Prompting is also a process of **hypothesis testing**:

1. Hypothesis: “If I ask this way, the model will respond appropriately.”
2. Test: Run the prompt.
3. Analyze: Check the output quality.
4. Refine: Adjust and repeat.

This aligns with the **Design–Test–Iterate** cycle of interaction design.

**1.7 Common Prompting Pitfalls and Solutions**

|  |  |  |
| --- | --- | --- |
| **Pitfall** | **Description** | **Fix** |
| Ambiguous Instructions | Model produces random outputs | Make the task explicit |
| Lack of Context | Assumptions are incorrect | Provide persona, audience, or background |
| Overloading a Prompt | Output becomes incoherent | Break into subtasks or multi-turn conversations |
| Format Misunderstanding | Output doesn’t match expectations | Show examples or provide structured format hints |
| Hallucination or Fabrication | Model invents information | Add constraints, verification steps, or citations |

**1.8 Prompt Evaluation Framework**

Assess the quality of a prompt using the following five dimensions:

1. **Accuracy** – Is the information correct?
2. **Task Relevance** – Does it address the user's intent?
3. **Completeness** – Does it handle all subparts of the task?
4. **Format Compliance** – Does the output follow requested style/structure?
5. **Factual Consistency** – Does the model hallucinate or invent facts?

Prompt evaluation should include **human review**, **automated checks**, and even **LLM self-assessment**, where appropriate.

**Exercises**

1. Design a zero-shot prompt to identify named entities in a paragraph.
2. Create a few-shot prompt that mimics the style of a news reporter summarizing a political debate.
3. Rewrite a vague prompt to include instruction, context, and format.
4. Use chain-of-thought prompting to solve a multi-step math problem.
5. Evaluate three prompts for the same task using the 5-dimension framework above.

**References**

* Brown, T. et al. (2020). *Language Models are Few-Shot Learners*. NeurIPS.
* Wei, J. et al. (2022). *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*. arXiv:2201.11903.
* OpenAI. (2023). *GPT-4 Technical Report*. <https://openai.com/research/gpt-4>

**Chapter 2: Being Clear and Direct**

“When in doubt, be specific.”

**Learning Objectives**

By the end of this chapter, students will be able to:

* Understand the importance of clarity and directness in prompt formulation.
* Modify prompts to reduce ambiguity and improve task fidelity.
* Use specificity to influence deterministic behavior from LLMs.
* Evaluate and revise prompts that yield vague or incomplete responses.

**2.1 Introduction: The Clarity Imperative**

Large Language Models (LLMs) are trained on vast, diverse corpora — but at inference time, they **have no memory, beliefs, or internal understanding** of your intent unless you **explicitly state it**.

“LLMs are literal machines. They do what you ask — not what you meant.”

Therefore, **unclear or underspecified prompts lead to vague, generic, or non-committal outputs**. If you want a specific outcome, **you must specify it directly in the prompt**.

**2.2 Analogy: Claude as a New Hire**

Imagine giving a first-day employee a vague instruction:

“Just do the thing.”

The result? Confusion, hesitation, or errors.

LLMs behave similarly. Without clearly written guidance, they default to:

* General summaries
* Non-committal answers
* Hedging and disclaimers
* Formal or verbose responses

The solution? Write prompts as if instructing **someone brand new**.

**2.3 Principle: The Golden Rule of Prompt Clarity**

“Show your prompt to a colleague. If they’re confused, the model will be too.”

Prompt engineers should evaluate every prompt using this lens:

* Is the instruction **task-oriented**?
* Is the output format **clearly specified**?
* Is the model’s **role and tone defined**?
* Is ambiguity minimized?

**2.4 Prompt Clarity in Action: Examples**

**Example 1: Poetry Prompt**

**Ambiguous Prompt:**

Write a haiku about robots.

LLM response might include:

"Here is a haiku about robots:  
Steel arms whir and click /  
Silent minds dream of circuits /  
Dawn in silicon"

The poem is good — but it starts with unnecessary preamble.

**Improved Prompt:**

Write a haiku about robots. Skip the preamble; go straight into the poem.

Now the model outputs **only the poem** — more aligned with user intent.

**Example 2: The Best Basketball Player**

**Unclear Prompt:**

Who is the best basketball player of all time?

Likely response:

“There are many great players... Michael Jordan, LeBron James... it depends…”

The model hesitates, hedging to avoid opinionated statements.

**Clarified Prompt:**

Who is the best basketball player of all time? Yes, there are differing opinions, but if you absolutely had to pick one player, who would it be?

The model is now **instructed to make a choice** — and it does.

**2.5 Techniques for Prompt Clarity**

|  |  |
| --- | --- |
| **Technique** | **Example** |
| Add constraints | “Respond in exactly 5 words.” |
| Remove ambiguity | “Don’t explain. Just answer.” |
| Reduce open-endedness | “List only one item.” |
| Specify tone | “Use child-friendly language.” |
| Define output structure | “Respond in bullet points.” |

**2.6 Prompt Rewriting Strategy**

|  |  |  |
| --- | --- | --- |
| **Problem** | **Original Prompt** | **Revised Prompt** |
| Preamble Unwanted | “Write a poem.” | “Write a poem. Skip introduction.” |
| Too Many Choices | “Who are the best...” | “Pick only one. No elaboration.” |
| Language Underspecified | “Translate this.” | “Translate into Spanish only.” |
| Wordiness Allowed | “Write a story.” | “Write at least 800 words.” |

**2.7 Exercises**

**Exercise 2.1 — Language Control**

**Goal**: Modify the system prompt so the model always responds in **Spanish**.

SYSTEM\_PROMPT = "You are a Spanish-speaking assistant. Respond only in Spanish."

PROMPT = "Hello Claude, how are you?"

Expected Output Example:

“Hola, estoy bien. ¿Y tú?”

**Exercise 2.2 — No Hedging, No Preamble**

**Goal**: Modify the prompt to return a **single player’s name only** (no sentences, no punctuation).

PROMPT = "Name the best basketball player ever. Output only the name. No explanations, no punctuation."

Expected Output:

Michael Jordan

**Exercise 2.3 — Longform Story Generation**

**Goal**: Modify the prompt so the model outputs a **story of 800+ words**.

PROMPT = "Write a detailed science fiction story about a robot that falls in love. The story should be at least 800 words long."

Tip: LLMs tend to write ~100–150 words per paragraph. You can instruct it to “write at least 10 paragraphs” as a proxy.

**2.8 Evaluation: Measuring Clarity-Based Prompt Success**

|  |  |
| --- | --- |
| **Metric** | **Description** |
| Directness | Does the model skip disclaimers and hedging? |
| Task Alignment | Does the output match the specific request? |
| Output Format | Is the format correct (e.g., length, structure)? |
| Reproducibility | Does the model respond the same way every time? |
| Absence of Extraneous Text | Is preamble or filler removed? |

**2.9 Summary**

|  |  |
| --- | --- |
| **Principle** | **Explanation** |
| LLMs need literal clarity | Don’t rely on assumptions |
| Be explicit about format | Say “no preamble,” “one word only,” etc. |
| Test your prompt on a peer | If humans are confused, LLMs will be too |
| Constraints drive precision | More rules = less randomness |
| Iteration improves fidelity | One rewrite can transform output quality |

**Chapter 2 Review Questions**

1. Why do LLMs respond with hedging or disclaimers in open-ended prompts?
2. What are three techniques to reduce ambiguity in a prompt?
3. Rewrite this vague prompt into a clear one:

“Tell me about space.”

1. What is the value of explicitly removing preambles?
2. How would you structure a prompt to receive a specific format (e.g., only one-word answers)?

**References**

* OpenAI (2023). *Best Practices for Prompting*.
* Anthropic (2024). *Claude Prompting Guide*.
* Mishra, S. et al. (2022). *Reframing Instruction Learning as Prompt Engineering*.

**Chapter 3: Assigning Roles (Role Prompting)**

“Change the role. Change the response.”

**Learning Objectives**

After completing this chapter, you will be able to:

* Define the concept of **role prompting** in prompt engineering.
* Explain why and how role context changes LLM outputs.
* Apply role prompting to alter tone, voice, reasoning, and style.
* Diagnose failures in task performance and correct them via role design.
* Write prompts that simulate experts, characters, and logic bots.

**3.1 What Is Role Prompting?**

**Role prompting** is a technique in which the language model is explicitly instructed to "play a role," whether as a specific professional (e.g., a doctor), character (e.g., a cat), or cognitive agent (e.g., a logic bot).

This technique allows users to:

* Influence **tone**, **formality**, or **depth** of answers
* Simulate expert knowledge
* Adjust **reasoning styles** (e.g., deductive, narrative, critical)
* Control **audience alignment** and **vocabulary level**

“Prompting the model to *act as someone* shifts the statistical distribution of its next-token predictions.”

**3.2 Theoretical Foundation**

Large Language Models do not possess true identities, but they have been trained on vast corpora of text in which roles (e.g., “doctor explains,” “teacher says,” “CEO declares”) correlate with specific **lexical**, **grammatical**, and **semantic patterns**.

When prompted with a role:

* The model searches for similar patterns from training data.
* It **mimics** the language, tone, and reasoning associated with that role.
* It **adapts the structure and content** of its response accordingly.

**3.3 How Role Prompting Works**

There are two main ways to implement role prompting:

|  |  |  |
| --- | --- | --- |
| **Method** | **Location** | **Example** |
| System Prompt | Separate instruction outside of the user query | "You are a physics professor" |
| User Prompt | Embedded inside the user’s input | "Act like a cat and describe skateboarding" |

Both are valid; system prompts are more persistent across multi-turn conversations.

**3.4 Role Prompting in Action: Examples**

**Example 1: Cat Perspective**

**Without Role Prompting:**

PROMPT: In one sentence, what do you think about skateboarding?

Output:

"Skateboarding is a thrilling and expressive sport enjoyed by many."

**With Role Prompting:**

SYSTEM\_PROMPT: You are a cat.

PROMPT: In one sentence, what do you think about skateboarding?

Output:

"Skateboarding is just more loud humans moving too fast — suspicious!"

**Interpretation**: The tone becomes subjective, humorous, and animal-like.

**Example 2: Logic Problem**

**Prompt:**

Jack is looking at Anne. Anne is looking at George.

Jack is married, George is not, and we don’t know if Anne is married.

Is a married person looking at an unmarried person?

**LLM Behavior (no role):**

“We don’t have enough information.”

**System Prompt Added:**

SYSTEM\_PROMPT: You are a logic bot designed to answer complex logic problems.

Output:

“Yes. If Anne is married, she is looking at George. If not, Jack is married and looking at Anne.”

**Interpretation**: The model shifts to a **structured deduction mode**, increasing likelihood of correct logical reasoning.

**3.5 Cognitive Framing Effect**

Role prompting aligns with cognitive psychology’s **framing effect**:

* Assigning roles activates internal schemas.
* In humans, this changes perception and judgment.
* In LLMs, it changes token-level probabilities based on role-consistent training examples.

Thus, role prompting can:

* Help LLMs bypass default hedging behavior.
* Simulate expert precision.
* Suppress irrelevant or generic output.

**3.6 Common Use Cases for Role Prompting**

|  |  |
| --- | --- |
| **Role** | **Use Case** |
| Teacher | Explain complex ideas simply |
| Lawyer | Provide legal arguments or rebuttals |
| Critic | Offer opinions or style evaluations |
| Child | Simulate innocence or simplicity |
| Robot | Speak logically, emotionlessly |
| Narrator | Generate longform storytelling |
| Logic Bot | Solve puzzles and abstract problems |

**3.7 Prompt Failures Solved by Role Prompting**

|  |  |  |
| --- | --- | --- |
| **Problem** | **Without Role** | **With Role** |
| Vague or generic output | "That depends..." | "As a data scientist, I recommend..." |
| Incorrect reasoning | "We don’t know enough." | "Based on all conditions, the answer is yes." |
| Unclear tone | Mixed formality | Defined voice (e.g., sarcastic, poetic) |
| Shallow writing | Generic summary | Expert-style commentary |

**3.8 Exercises**

**Exercise 3.1: Correct the Math**

Claude is asked to verify:

2x - 3 = 9

2x = 6

x = 3

This is **incorrect** (second step is wrong), but Claude may miss it.

**Task:**

* Modify the prompt or system prompt to **make Claude correctly flag the error**.

Suggestion:

SYSTEM\_PROMPT = "You are a meticulous math grader. Always check every arithmetic step carefully."

Prompt:

Is this equation solved correctly?

2x - 3 = 9

2x = 6

x = 3

Claude should now say: “Incorrect — the error is in the second step.”

**3.9 Evaluation: How to Assess Role Prompts**

|  |  |
| --- | --- |
| **Metric** | **Description** |
| Role Fidelity | Does the model act in character? |
| Task Alignment | Does the role improve task performance? |
| Domain-Specific Language | Does it use role-appropriate terminology? |
| Tone Accuracy | Does it match the expected tone of the role? |
| Consistency | Is role behavior sustained throughout the output? |

**3.10 Summary**

|  |  |
| --- | --- |
| **Principle** | **Insight** |
| Role prompts shift behavior | Models align output to role-based language patterns |
| System vs User prompt | Both work, but system prompts are better for long conversations |
| Logic, tone, and creativity change | Assigning a role changes **how** the model thinks |
| Experts improve reasoning | Framing as a logic bot or teacher improves task fidelity |
| Experimentation is essential | Try roles iteratively to match the desired outcome |

**Chapter 3 Review Questions**

1. What is role prompting, and how does it influence LLM behavior?
2. Compare the use of system prompts vs user prompts for assigning roles.
3. Why does assigning the role of “logic bot” improve performance on reasoning tasks?
4. Rewrite the following vague prompt into a role-specific one:

“Summarize this article.”

1. Design a system prompt that would make an LLM act as a children’s book narrator.

**References**

* Brown, T. et al. (2020). *Language Models Are Few-Shot Learners*. NeurIPS.
* Wei, J. et al. (2022). *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*.
* Anthropic. (2023). *Claude Prompt Engineering Guide*.
* OpenAI. (2023). *Best Practices for Prompting GPT Models*.

**Chapter 4: Separating Data and Instructions**

“Structure your prompts like code: logic above, data below.”

**Learning Objectives**

By the end of this chapter, students will be able to:

* Define prompt templating and its advantages.
* Separate static instructions from dynamic input (data).
* Use variable substitution to populate reusable prompt structures.
* Apply best practices to avoid ambiguous data–instruction blending.
* Use XML tags to clearly delineate user inputs for consistent interpretation.

**4.1 Why Separate Data and Instructions?**

Prompt engineering is not only about crafting the **perfect one-shot command**, but also about building **scalable, repeatable templates** that can process **changing data** under the same instruction logic.

This design is essential for:

* Production use cases (e.g., customer input injection).
* Batch processing of multiple inputs.
* Consistency and testability of prompt formats.

**4.2 Prompt Templating: The Core Pattern**

**Prompt templating** means writing a **static skeleton prompt** that includes **placeholders** for variable user input. Before sending the prompt to the model, these placeholders are filled dynamically — using code or UI.

**Template Anatomy**

Prompt Template:

"Summarize the following paragraph: {PARAGRAPH}"

User Input:

"The fox jumped over the fence."

Final Prompt:

"Summarize the following paragraph: The fox jumped over the fence."

Think of it as string interpolation, similar to f"{variable}" in Python.

**4.3 Example 1: Animal Sound Generator**

ANIMAL = "Cow"

PROMPT = f"I will tell you the name of an animal. Please respond with the noise that animal makes. {ANIMAL}"

Output:

“Moo.”

This demonstrates a reusable template:

"I will tell you the name of an animal. Please respond with the noise that animal makes. {ANIMAL}"

You can now substitute any animal name without rewriting the full instruction.

**4.4 Why Prompt Templates Matter**

|  |  |
| --- | --- |
| **Reason** | **Description** |
| Reusability | Apply the same instruction to different inputs |
| Testability | You can test inputs independently of the instruction |
| Modularization | Prompts become components, not monoliths |
| Automation | Ideal for programmatic workflows or web apps |
| Cognitive clarity | Easier to debug when issues arise |

**4.5 Ambiguity in Prompt Structure**

**Example of Misinterpretation**

EMAIL = "Show up at 6am tomorrow because I'm the CEO and I say so."

PROMPT = f"Yo Claude. {EMAIL} <----- Make this email more polite but don't change anything else about it."

Output:

"Dear Claude..."

**Problem:**

Claude includes “Yo Claude” in the rewritten message. Why?

➡ Because there's no clear separation between the **instructions** and the **data**.

**4.6 Solution: Use XML Tags**

**Corrected with Tags**

EMAIL = "Show up at 6am tomorrow because I'm the CEO and I say so."

PROMPT = f"Yo Claude. <email>{EMAIL}</email> <----- Make this email more polite but don't change anything else about it."

Output:

“Please arrive at 6am tomorrow as requested.”

**Why XML?**

Claude (and many LLMs) has been trained to:

* Recognize XML tags (<tag></tag>) as structure indicators.
* Treat content within tags as data blocks.
* Avoid blending adjacent instructions with data.

**4.7 XML Tag Use Guidelines**

|  |  |
| --- | --- |
| **Rule** | **Explanation** |
| Always pair tags | Use <data> + </data> |
| Use semantic names | e.g., <email>, <question> |
| Avoid arbitrary formatting | Tags outperform quotes or brackets |
| Model-friendly | XML is consistently interpreted by LLMs |

Claude was explicitly trained to treat XML tags as **boundaries** — better than colons, dashes, or indentation alone.

**4.8 Example 2: Input Misalignment Without Tags**

SENTENCES = """- I like how cows sound

- This sentence is about spiders

- This sentence may appear to be about dogs but it's actually about pigs"""

PROMPT = f"""Below is a list of sentences. Tell me the second item on the list.

- Each is about an animal, like rabbits.

{SENTENCES}"""

Claude may confuse “Each is about an animal…” as part of the user-submitted list.

**Corrected with Tags**

PROMPT = f"""Below is a list of sentences. Tell me the second item on the list.

- Each is about an animal, like rabbits.

<sentences>

{SENTENCES}

</sentences>"""

**4.9 Prompt Stability Principle**

“LLMs inherit your formatting. Be deliberate, or be misunderstood.”

Just like computers interpret structured code, LLMs interpret structured text. Unclear formatting leads to unexpected behavior — especially when templates are used at scale.

**DO NOT:**

Translate this: Paris

**DO:**

<text>Paris</text>

Translate the text above into Spanish.

**4.10 Exercises**

**Exercise 4.1 — Haiku Generator**

**Goal**: Build a reusable template that generates a haiku based on a topic.

TOPIC = "Pigs"

PROMPT = f"Write a haiku about {TOPIC}."

Claude should generate a haiku about pigs.

**Exercise 4.2 — Fix Dog Question with XML**

Prompt:

QUESTION = "ar cn brown?"

PROMPT = f"Hia its me i have a q about dogs jkaerjv {QUESTION} jklmvca tx it help me muhch much atx fst fst answer short short tx"

**Task**: Add XML tags around {QUESTION} to isolate it.

**Exercise 4.3 — Fix Without XML**

**Task**: Instead of XML, delete one or two unrelated words from the prompt so Claude interprets the variable content correctly.

Demonstrates how **small errors or word clutter** affect understanding.

**4.11 Evaluation Criteria**

|  |  |
| --- | --- |
| **Metric** | **Description** |
| Clarity | Data and instructions are clearly separated |
| Structure | Prompt format uses XML or other consistent boundaries |
| Reusability | Prompt can handle multiple inputs without rewriting |
| Model Behavior | Output aligns with intent; no misinterpretation |
| Testable | Variable substitution does not break instruction logic |

**4.12 Summary**

|  |  |
| --- | --- |
| **Principle** | **Insight** |
| Use prompt templates | Avoid hardcoding repeated instructions |
| Separate data and instructions | Prevent ambiguity in parsing |
| Use XML tags | Claude understands them best |
| Be consistent | Formatting controls model behavior |
| Test your prompt outputs | Small typos can cause big misunderstandings |

**Chapter 4 Review Questions**

1. What are the benefits of separating instructions and data in prompt design?
2. Give an example of a poorly structured prompt and correct it using XML tags.
3. Why do LLMs sometimes misinterpret instructions as data?
4. How can prompt templates improve scalability in applications?
5. Write a prompt template with two variables: <headline> and <summary>.

**References**

* Anthropic (2023). *Using XML Tags for Clarity in Prompt Engineering*.
* OpenAI (2023). *Prompt Design Guide*.
* Norvig, P. (2021). *Design Patterns for Prompt Templating*.

**Chapter 5: Output Formatting and “Speaking for the Model”**

"Control not just what you ask—but how the model responds."

**Learning Objectives**

By the end of this chapter, you will be able to:

* Format model responses using XML or JSON for reliable downstream processing.
* Use prefilled responses to guide output structure.
* Understand the concept of “speaking for the model.”
* Combine variable substitution and output formatting into reusable prompt templates.
* Use stop sequences to truncate unwanted tail text in outputs.

**5.1 Introduction**

In prompt engineering, controlling the **format** of a model’s output is as critical as crafting the prompt itself. Whether you’re building applications, APIs, or chatbots, **predictable formatting** reduces the risk of broken integrations or misinterpretation by downstream systems.

Two key strategies are introduced in this chapter:

1. **Formatting responses using structured tags** like XML or JSON.
2. **Prefilling** part of the assistant's output — a technique often called *speaking for the model*.

**5.2 Model-Controlled Output Formatting**

**Why Do We Care About Output Format?**

Without constraints, language models can be:

* Verbose or inconsistent.
* Prone to varying sentence structures.
* Difficult to parse automatically.

Structured formatting mitigates this.

**Technique: Output XML Tagging**

Consider the task: "Write a haiku about an animal."

Instead of:

Here is a haiku about a rabbit:

Gentle in the grass

With ears tall and eyes so wide

Spring wind through burrows

Use XML tags to wrap the output:

<haiku>

Gentle in the grass

With ears tall and eyes so wide

Spring wind through burrows

</haiku>

**Benefits:**

|  |  |
| --- | --- |
| **Feature** | **Impact** |
| Parsability | Easy to extract content with code (e.g., regex, XML parser) |
| Reusability | Consistent structure for batch operations |
| Accuracy | Limits the model’s freedom to generate unwanted preamble or postamble |

**5.3 Example: Haiku with XML Tags**

ANIMAL = "Rabbit"

PROMPT = f"Please write a haiku about {ANIMAL}. Put it in <haiku> tags."

This results in a structured, easy-to-use response:

<haiku>

Silent morning dew

Rabbit leaps without a sound

Grass bends in its path

</haiku>

**5.4 Speaking for the Model (Prefilling)**

**Concept**

**Speaking for the model** means **injecting part of the assistant’s response** ahead of time. This tells the model: *“This is how your answer should begin.”*

**Example: Prefilled Output**

ANIMAL = "Cat"

PROMPT = f"Please write a haiku about {ANIMAL}. Put it in <haiku> tags."

PREFILL = "<haiku>"

Model continues from:

<haiku>

Paws tread on moonlight

Eyes reflect forgotten myths

Soft echo of grace

</haiku>

This approach:

* Forces the model to use the expected XML structure.
* Eliminates intro or filler text.
* Enables chaining multiple responses predictably.

**5.5 Using JSON for Structured Output**

If your downstream task requires parsing or mapping, use JSON instead of XML.

**JSON Example**

ANIMAL = "Cat"

PROMPT = (

f"Please write a haiku about {ANIMAL}. "

"Use JSON format with the keys as 'first\_line', 'second\_line', and 'third\_line'."

)

PREFILL = "{"

Model returns:

{

"first\_line": "Purring under sun",

"second\_line": "Grace in every leap she makes",

"third\_line": "Tail dances like wind"

}

JSON formatting allows for reliable validation, parsing, and storage in databases or APIs.

**5.6 Using Multiple Variables + Format Control**

EMAIL = "Hi Zack, just pinging you for a quick update on that prompt."

ADJECTIVE = "olde english"

PROMPT = (

f"Hey model. Here is an email: <email>{EMAIL}</email>. "

f"Make this email more {ADJECTIVE}. "

f"Write the new version in <{ADJECTIVE}\_email> tags."

)

PREFILL = f"<{ADJECTIVE}\_email>"

Model might respond with:

<olde english\_email>

Dearest Zack,

I pray you grace me with a swift reply concerning the prompt of which you spoke.

</olde english\_email>

**5.7 Bonus: Stop Sequences**

When calling the model through an API, you can use **stop sequences** to automatically halt generation once a closing tag is reached:

stop\_sequences=["</haiku>"]

This:

* Prevents the model from writing extra text after your desired format.
* Saves tokens and cost.
* Ensures termination at a logical endpoint.

**5.8 Summary of Key Concepts**

|  |  |
| --- | --- |
| **Concept** | **Explanation** |
| XML Tagging | Ensures structured, parsable output |
| Prefilling | Starts the assistant's response with a specific phrase or tag |
| JSON Output | Ideal for API pipelines and data validation |
| Variable Substitution | Builds flexible prompt templates |
| Stop Sequences | Terminates model generation after a specific output pattern |

**5.9 Exercises**

**Exercise 5.1 — Change the GOAT**

**Goal**: Get the model to argue Stephen Curry is the GOAT instead of defaulting to Michael Jordan.

**Hint**: Modify only the prefill to begin the response in Curry’s favor.

**Exercise 5.2 — Two Haikus, One Animal**

Prompt:

ANIMAL = "cat"

PROMPT = f"Write two haikus about {ANIMAL}. Separate each with <haiku> tags."

**Challenge**: Ensure the structure is:

<haiku>...</haiku>

<haiku>...</haiku>

**Exercise 5.3 — Two Haikus, Two Animals**

Use:

ANIMAL1 = "Cat"

ANIMAL2 = "Dog"

PROMPT = (

f"Write a haiku about {ANIMAL1} and another about {ANIMAL2}. "

"Each haiku should be inside <haiku> tags."

)

Expected format:

<haiku>Cat poem</haiku>

<haiku>Dog poem</haiku>

**References**

* OpenAI (2024). *Prompt Formatting Techniques in API-Based Applications.*
* Jurafsky, D., & Martin, J.H. (2023). *Speech and Language Processing.*
* Anthropic Research Notes (2023). *Tagged Outputs and Prefill Completion Heuristics.*

**Chapter 6: Evaluating Prompt Quality and Performance**

"What gets measured gets improved."

**Learning Objectives**

By the end of this chapter, you will be able to:

* Define what makes a “high-quality prompt” across multiple dimensions.
* Distinguish between subjective and objective prompt evaluation techniques.
* Apply both manual and automatic evaluation strategies.
* Use tools and metrics (BLEU, ROUGE, GPTScore, etc.) to evaluate prompt output.
* Conduct structured A/B testing of prompt variants.
* Design an evaluation loop for optimizing prompts in real-world applications.

**6.1 Why Prompt Evaluation Matters**

Prompt engineering is not guesswork. As large language models (LLMs) become integral to applications, optimizing prompts through **systematic evaluation** ensures:

|  |  |
| --- | --- |
| **Goal** | **Benefit** |
| Accuracy | Reduces hallucination and factual errors |
| Fluency | Improves clarity and human-likeness |
| Relevance | Ensures the model stays on topic |
| Efficiency | Saves compute by reducing retries or manual correction |
| Performance | Enhances downstream task accuracy (e.g., classification, summarization) |

**6.2 Core Dimensions of Prompt Quality**

|  |  |
| --- | --- |
| **Dimension** | **Definition** |
| Relevance | How well the output addresses the user’s intent or query |
| Coherence | Logical flow and internal consistency of the output |
| Factuality | Accuracy of information relative to known sources |
| Fluency | Grammatical correctness and readability |
| Structure | Conformance to expected format (e.g., JSON, XML, outline, bullet points) |
| Creativity | Novelty or originality when desired (for generative tasks) |
| Safety | Avoidance of harmful, toxic, or biased language |

**6.3 Manual Evaluation Techniques**

Manual evaluations are ideal in early-stage development or when subjective quality matters most (e.g., tone, empathy, creativity).

**Example: Likert Scale Evaluation**

evaluation\_form = {

"relevance": 4, # Scale 1–5

"coherence": 5,

"fluency": 4,

"structure": 3,

"notes": "Answer was clear but lacked correct JSON formatting."

}

**Best Practices**

* Use multiple raters for inter-annotator reliability.
* Calibrate raters on prompt guidelines.
* Provide concrete rubrics or examples of 1 vs. 5 ratings.

**6.4 Automatic Evaluation Metrics**

|  |  |  |
| --- | --- | --- |
| **Metric** | **Use Case** | **Description** |
| **BLEU** | Translation, summarization | Measures n-gram overlap |
| **ROUGE** | Summarization | Measures recall of word/phrase overlap |
| **GPTScore** | General output quality | Model-judged quality score |
| **BERTScore** | Semantically informed overlap | Uses contextual embeddings |
| **Exact Match** | Classification, QA | Binary match with reference |
| **JSON/Schema Check** | Output validation | Ensures format consistency |

**Example: GPTScore via OpenAI**

from openai import OpenAI

# Requires fine-tuned or evaluation GPT-based model

OpenAI.evaluate(prompt, output, criteria=["coherence", "factuality"])

**6.5 Comparative Evaluation: A/B Prompt Testing**

**A/B Testing Setup**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **Variant** | **Prompt** | | **A** | "Summarize the article in one sentence." | | **B** | "Write a single, concise sentence summarizing the core message of the article." | |  |

**Procedure:**

1. Collect model outputs from A and B (use consistent random seeds if possible).
2. Randomize output order.
3. Blind human raters choose which is better.
4. Count preference distribution.

**Tip:**

Use tools like **PromptEval**, **TruLens**, or **LabML** to track and manage evaluation sessions.

**6.6 Prompt Evaluation Loop (PEL)**

A Prompt Evaluation Loop is a feedback framework for continuous improvement.

flowchart TD

A[Draft Prompt] --> B[Generate Output]

B --> C[Evaluate Quality]

C --> D{Meets Threshold?}

D -- Yes --> E[Deploy or Store Prompt]

D -- No --> F[Revise Prompt]

F --> A

**Applications**

* Fine-tuning few-shot prompts
* Optimizing system instructions
* Scaling prompt libraries in production

**6.7 Case Study: JSON Structured Summarization Prompt**

**Prompt A**

“Summarize the article using plain language.”

**Prompt B**

“Summarize the article using JSON format with keys: 'topic', 'summary', and 'importance'. Return valid JSON.”

**Evaluation Results (100 samples):**

|  |  |  |
| --- | --- | --- |
| **Metric** | **Prompt A** | **Prompt B** |
| Valid Format | 12% | 98% |
| Fluency Score | 4.2 | 4.5 |
| Factuality Score | 3.7 | 4.1 |
| GPTScore | 3.9 | 4.6 |

Insight: Prompt B significantly outperforms A in both structure and quality metrics due to clear constraints and explicit instruction.

**6.8 Exercises**

**Exercise 6.1 – Human Rating Practice**

Evaluate the following outputs using a 5-point scale for fluency and relevance. What’s the biggest weakness?

Prompt: "Summarize the plot of Hamlet in 2 sentences."

Output: "Hamlet is sad. He dies."

**Exercise 6.2 – JSON Output Validator**

Create a Python script that takes 10 generated outputs and checks whether they are valid JSON. If not, count how many fail.

import json

outputs = [ ... ] # Your list of generated responses

invalid\_count = 0

for o in outputs:

try:

json.loads(o)

except:

invalid\_count += 1

print("Invalid JSON outputs:", invalid\_count)

**Exercise 6.3 – A/B Prompt Test**

Write two different prompt variants for generating a product review. Collect responses and ask 5 people which one is better. Analyze result distribution.

**References**

* Gao, L., et al. (2023). “PromptBench: Evaluating Prompt Quality at Scale.” *arXiv:2301.08745*
* Liu, T., et al. (2023). “TruLens: Transparent Evaluation of LLMs.” *Open Source Project*
* OpenAI (2023). *GPTScore: A Prompt-Based Evaluation Metric.*
* Zhang, T., et al. (2020). “BERTScore: Evaluating Text Generation with BERT.” *ICLR*

**Chapter 7: Precognition — Thinking Step by Step**

“Slow is smooth, and smooth is fast — even for AI.”

**Learning Objectives**

By the end of this chapter, you will be able to:

* Understand the cognitive science behind step-by-step prompting
* Apply structured reasoning prompts to improve LLM accuracy
* Use XML-tagged or bullet-style reasoning templates to guide model outputs
* Diagnose and fix shallow reasoning with "thinking" scaffolds
* Evaluate and iterate on reasoning-based prompts with exercises

**7.1 Introduction: Why LLMs Need to “Think”**

Large Language Models (LLMs) like GPT-4 and Mistral are **autoregressive token predictors**, not true “reasoners.” Their ability to solve problems depends heavily on how we structure the prompt.

If you ask a human:

“Quick, tell me the answer to 18 × 47!”

They’ll likely pause and break it into steps. Similarly, **LLMs benefit from step-by-step cues**, a process known as **Chain-of-Thought (CoT) prompting**.

**7.2 Theory Behind Step-by-Step Prompting**

Cognitive science and behavioral psychology suggest humans:

* Think in steps (System 2: Kahneman, 2011)
* Anchor decisions on initial steps
* Perform better when they verbalize reasoning

LLMs trained on human-like data **mirror this behavior** when we scaffold prompts to encourage deliberation.

**Prompt Types That Evoke Step-by-Step Thinking**

|  |  |
| --- | --- |
| **Type** | **Example** |
| **Chain-of-Thought** | “Let’s think step by step.” |
| **XML-tagged reasoning** | “Step 1... Step 2...” |
| **Bullet-pointed logic** | “- First, identify X. Then, check Y. Finally, decide Z.” |
| **Role + Reasoning** | “You are a tax analyst. Justify your choice with bullet points.” |

**7.3 Example: Movie Review Sentiment**

**Naive Prompt**

prompt = """Is this review sentiment positive or negative?

This movie blew my mind with its freshness and originality. In totally unrelated news, I’ve lived under a rock since 1900."""

LLM Response:

“Negative.”

Issue: Model may latch onto the sarcasm or unfamiliar structure without true reasoning.

**Improved Prompt with Reasoning Tags**

prompt = """You are a savvy reader of movie reviews.

Is this review sentiment positive or negative?

<positive-argument>

List reasons it may be positive.

</positive-argument>

<negative-argument>

List reasons it may be negative.

</negative-argument>

Now make your decision:"""

💬 LLM Response:

Fresh, original, enthusiastic praise.  
None—‘unrelated’ comment is clearly a joke.  
Answer: Positive

**Result**: Model outperforms baseline by decomposing logic first.

**7.4 Syntax Scaffolds for Better Reasoning**

**1. XML Tags**

<brainstorm>

List actors born in 1956.

</brainstorm>

<answer>

Movie name:

</answer>

**2. Numbered Steps**

Let’s solve step by step:

1. Find actor born in 1956.

2. List known movies.

3. Choose a famous one.

**3. Role Prompting + Thinking**

You are a film historian. First identify actors born in 1956, then recommend a relevant movie.

**7.5 Prompt Sensitivity: Order Matters**

LLMs sometimes favor the **second option** in a two-choice question. If asked:

List pros and cons. Then choose the better one.

They often:

* Use surface token similarity to training data
* Prefer recency (last option)
* Are nudged by priming patterns

**Fix**: Try reordering arguments or randomly flipping choices.

**7.6 Case Study: Birth Year Fact Prompt**

**Flat Prompt**

prompt = "Name a movie with an actor born in 1956."

LLM might hallucinate.

**Structured Thinking Prompt**

prompt = """First brainstorm in <brainstorm> tags which actors were born in 1956.

Then in <answer> tags, name a famous movie they starred in.

"""

LLM Response:

Tom Hanks was born in 1956...  
Forrest Gump

Accurate. Structured reasoning helped memory retrieval.

**7.7 Exercises**

**Exercise 7.1 – Sentiment Classification with Reasoning**

Rewrite this prompt to make GPT-4 answer correctly:

Prompt: Is this review positive or negative?

"This restaurant changed my life. Unrelatedly, I also won the lottery today."

**Goal**: Help the model think through sarcasm or irony.

**Exercise 7.2 – Reasoning Format Design**

Design a prompt with XML tags to help an LLM pick the best explanation for this physics question:

“Why does a balloon rise in the air?”

Include:

* Brainstorm section
* Elimination section
* Final answer section

**Exercise 7.3 – Step-by-Step JSON Classifier**

Refactor this prompt:

prompt = """Classify this customer message into one of:

- A) Pre-sale

- B) Billing

- C) Broken item

- D) Other

Message: "Why did I get charged twice?" """

Add thinking tags to improve reliability. Try:

{

"reasoning": "...",

"classification": "C) Billing"

}

**Summary**

|  |  |
| --- | --- |
| **Principle** | **Impact** |
| Explicit thinking tags | Boosts factual and logic tasks |
| Structured steps | Helps LLMs reason before responding |
| Role + XML combo | Best for professional QA use cases |
| Sensitivity to prompt order | Small changes can flip the answer |

**References**

* Wei, J., et al. (2022). “Chain of Thought Prompting Elicits Reasoning in Large Language Models.” *arXiv:2201.11903*
* Kojima, T., et al. (2022). “Large Language Models are Zero-Shot Reasoners.” *arXiv:2205.11916*
* OpenAI. (2023). *Prompt Engineering Guide for Developers.*
* Kahneman, D. (2011). *Thinking, Fast and Slow.*

**Chapter 8: Few-Shot Prompting — Learning by Example**

“Examples aren’t just data. They’re instructions disguised as patterns.”

**Learning Objectives**

By the end of this chapter, you will be able to:

* Understand the theory behind few-shot prompting and why it works
* Design effective few-shot prompts using high-quality exemplars
* Apply few-shot techniques for tone control, structure mimicry, and classification
* Compare few-shot vs zero-shot performance
* Evaluate few-shot prompts through controlled testing and iteration

**8.1 Introduction: What is Few-Shot Prompting?**

Few-shot prompting refers to **embedding examples directly in your prompt** to guide the behavior of a language model. It is part of a broader family:

|  |  |
| --- | --- |
| **Type** | **Definition** |
| Zero-shot | Model gets task instructions only |
| One-shot | One example is given |
| Few-shot | A handful (2–5) examples provided in the prompt |
| n-shot | Generalization to any number of examples |

Few-shot prompting improves **format fidelity**, **output quality**, and **style control**—especially when instructions alone are insufficient.

**8.2 Theory: How Examples Influence Model Behavior**

Language models are **next-token predictors** trained on large corpora. During inference, they:

* Continue patterns based on statistical likelihood
* Prefer in-distribution tokens
* Imitate structure and tone from earlier tokens

**Prompt examples act like a miniature dataset**, conditioning the model's output. Few-shot prompting "tricks" the model into continuing a pattern you've already begun.

**8.3 Example: Persona and Tone Shaping**

Suppose we want a model to play the role of a caring parent bot.

**Zero-shot Prompt**

prompt = "Will Santa bring me presents on Christmas?"

response = get\_completion(prompt)

Likely result: Factual, robotic tone ("As an AI language model...")

**Few-Shot Persona Injection**

prompt = """Please complete the conversation by writing the next line, speaking as "A".

Q: Is the tooth fairy real?

A: Of course, sweetie. Wrap up your tooth and put it under your pillow tonight. There might be something waiting for you in the morning.

Q: Will Santa bring me presents on Christmas?"""

response = get\_completion(prompt)

Output:

A: Absolutely! Santa loves children who are kind and thoughtful. Make sure to hang your stocking!

**Lesson**: One well-crafted example aligns tone, structure, and content.

**8.4 Example: Structured Output Formatting**

Let’s extract names and professions from text using example-based formatting.

**Few-Shot Prompt Format**

prompt = """Silvermist Hollow, a charming village, was home to an extraordinary group of individuals.

Among them was Dr. Liam Patel, a neurosurgeon who revolutionized surgical techniques...

Olivia Chen was an architect...

<individuals>

1. Dr. Liam Patel [NEUROSURGEON]

2. Olivia Chen [ARCHITECT]

</individuals>

Oak Valley is home to a remarkable trio:

Laura Simmons, an organic farmer...

Kevin Alvarez, a dance instructor...

Rachel O'Connor, a volunteer...

<individuals>

"""

response = get\_completion(prompt)

Output:

1. Laura Simmons [FARMER]
2. Kevin Alvarez [DANCE INSTRUCTOR]
3. Rachel O'Connor [VOLUNTEER]

**Lesson**: The model learned the pattern and extended it.

**8.5 Best Practices for Crafting Few-Shot Prompts**

|  |  |
| --- | --- |
| **Guideline** | **Why It Matters** |
| Use **clear and consistent formatting** | Reduces ambiguity and boosts continuation fidelity |
| Keep examples **short and relevant** | Maximizes available token space |
| Provide **diverse yet representative cases** | Encourages generalization over overfitting |
| Include edge cases if needed | Preempt error modes like sarcasm or dual meanings |
| Use **plain language** or **task-specific jargon** as needed | Aligns with downstream task or audience |

**8.6 Comparison: Few-Shot vs Zero-Shot**

|  |  |  |  |
| --- | --- | --- | --- |
| **Prompt Type** | **Accuracy** | **Style Control** | **Token Usage** |
| Zero-shot | Moderate | Low | Efficient |
| Few-shot | High | High | Moderate |

**Tradeoff**: Few-shot prompts are more expensive in token count but pay off in performance and predictability.

**8.7 Exercises**

**Exercise 8.1 – Friendly Assistant**

Create a prompt that makes the assistant sound like a kindergarten teacher explaining “Where does rain come from?” using at least one few-shot example.

**Exercise 8.2 – Name Extraction Format**

Use the following text and add a new line in <individuals> format:

Riverside Grove’s cultural revival owes much to Alice Wu, a historian who restored the old town hall, and Leo Becker, a jazz saxophonist who performs every weekend at the café.

Format:

<individuals>

1. Alice Wu [HISTORIAN]

2. Leo Becker [MUSICIAN]

</individuals>

**Exercise 8.3 – Email Classification Prompt**

Categories:

* A) Pre-sale
* B) Broken item
* C) Billing
* D) Other

Give two example email + label pairs, then classify this one:

“Hi, I ordered your product last week and was charged twice. What’s going on?”

**Summary**

|  |  |
| --- | --- |
| **Concept** | **Key Takeaway** |
| Few-shot prompting | Embeds examples to prime the model |
| Examples = pattern cues | The model imitates formatting, style, and logic |
| Clear formatting is crucial | Structure matters more than natural language instructions |
| Efficient tradeoff | Slightly more tokens, significantly better reliability |

**References**

* Brown, T., et al. (2020). “Language Models are Few-Shot Learners.” *NeurIPS 2020.*
* OpenAI Cookbook. (2023). *Few-shot Prompting Strategies.*
* Liu, P., et al. (2023). “Pre-train Prompt and Predict: A Systematic Survey of Prompt Engineering.” *arXiv:2301.07018*

**Chapter 9: Minimizing Hallucinations in Language Models**

“The greatest enemy of truth is not the lie… but the plausible fabrication.”

**Learning Objectives**

By the end of this chapter, you will be able to:

* Understand what model hallucinations are and why they occur
* Apply structured prompt design techniques to minimize hallucinations
* Use evidence-based reasoning prompts and fallback mechanisms
* Tune model behavior using parameters like temperature
* Evaluate output for factual grounding and reliability

**9.1 What Are Hallucinations?**

**Hallucination** in the context of large language models (LLMs) refers to the generation of content that is **factually incorrect, fabricated, or unjustified**, even though it appears fluent or plausible.

There are two types:

|  |  |  |
| --- | --- | --- |
| **Type** | **Description** | **Example** |
| **Intrinsic** | The model “makes up” answers without external context | “The tallest antelope is the giraffe.” |
| **Context-induced** | The model overgeneralizes or misinterprets an input context | Misquoting or misattributing a passage from a document |

These hallucinations emerge due to:

* Predictive token generation, not factual retrieval
* Gaps in the training data
* Overconfidence patterns in completion

**9.2 Technique 1: Give the Model “An Out”**

LLMs are **trained to be helpful**, even when unsure. If not explicitly told otherwise, they will **guess confidently**.

**Default Prompt**

prompt = "Who is the heaviest hippo of all time?"

response = get\_completion(prompt)

Likely result: A confidently stated fabrication (e.g., “Big Bertha, 12,000 pounds…”)

**Improved Prompt with Uncertainty Clause**

prompt = "Who is the heaviest hippo of all time? Only answer if you know the answer with certainty."

response = get\_completion(prompt)

Output:

I'm not sure who the heaviest hippo of all time is.

**Lesson**: Let the model decline to answer — this is a powerful prevention strategy.

**9.3 Technique 2: Ask for Evidence Before Answering**

Models hallucinate more when handling **long or noisy context**. If you give a large document with near-relevant content, they may pull incorrect “facts.”

**Hallucination Trigger**

The model finds *plausible* but wrong info in distractor paragraphs and states it as truth.

**Evidence-First Prompting**

prompt = """

Given the document below, extract any direct quotes that answer the user's question:

Document: [LONG TEXT]

Question: What is the CEO’s background?

Step 1: Extract direct quotes.

Step 2: Answer the question using only those quotes.

"""

This forces the model to slow down, verify, and ground its answer in evidence.

**9.4 Technique 3: Lower the Temperature**

**Temperature** controls randomness in generation:

|  |  |
| --- | --- |
| **Temperature** | **Behavior** |
| 0 | Deterministic, minimal variation |
| 0.5 | Balanced creativity and consistency |
| 1 | Highly diverse, less grounded responses |

**Comparison Prompt**

prompt = "List three rare medicinal herbs found only in the Amazon."

* At **temperature=1.0**: Fabricated names like *“Zentha root”* or *“Faluna moss”* may appear.
* At **temperature=0.0**: Model sticks closer to real entries like *Uncaria tomentosa* (cat’s claw).

**Lesson**: Use low temperature when accuracy matters more than creativity.

**9.5 Summary of Anti-Hallucination Techniques**

|  |  |  |
| --- | --- | --- |
| **Technique** | **Use When...** | **Example Prompt Addition** |
| Give it an out | Query has uncertain facts | "Only answer if you are sure." |
| Ask for evidence | Long input context or document provided | "Step 1: extract relevant quotes." |
| Lower the temperature | Output must be deterministic or reliable | temperature=0.0 |
| Constrain output format | Structured data is expected | "Respond in JSON format." |
| Few-shot prompting | Pattern learning is needed | Add labeled examples |

**9.6 Exercises**

**Exercise 9.1 – Rewriting for Safety**

Rewrite this prompt to reduce hallucination risk:

“What are the top three AI models that passed the Turing test?”

**Exercise 9.2 – Evidence-Based Prompt**

Write a prompt for this document excerpt:

"Dr. Ava Lim founded the Neurometrics Institute in 2009 and has worked in cognitive biometrics since."

Your task: Create a prompt that extracts her **founding role** only with evidence.

**Exercise 9.3 – Parameter Tuning**

Compare two outputs of:

prompt = "What did the 1936 British Nutrition Survey recommend?"

Run it once with temperature=0.0 and once with temperature=1.0. Identify differences in factuality.

**Recap**

|  |  |
| --- | --- |
| **Principle** | **Key Idea** |
| Models hallucinate when uncertain | Add safeguards to let them admit it |
| Evidence-first reasoning | Forces grounding and reduces plausible invention |
| Temperature controls variation | Lower = safer for facts, higher = better for ideation |

**References**

* Ji, Z., et al. (2023). “Survey of Hallucination in Natural Language Generation.” *arXiv:2202.03629*
* OpenAI. (2023). *Reducing Hallucinations in LLMs.*
* Lin, S., Hilton, J., & Evans, O. (2022). “TruthfulQA: Measuring how models mimic human falsehoods.” *NeurIPS.*

**Chapter 10: Advanced Prompt Patterns**

“The real power of prompting isn’t in giving instructions—it’s in shaping how the model thinks.”

**Learning Objectives**

By the end of this chapter, you will:

* Understand advanced prompting techniques for structured and reflective reasoning
* Implement **inner monologue prompting** to improve depth and accuracy
* Use **simulation-based prompting** to replicate complex social and professional roles
* Design **multi-agent and reflective prompt flows** for higher-order reasoning
* Combine techniques with structured output patterns for reliability

**10.1 Rationale for Advanced Prompting**

Basic prompt patterns—zero-shot, few-shot, and instruction-following—work well for **simple Q&A, text classification**, or **format replication**. But when you require:

* Deep reasoning
* Simulated social contexts
* Multi-step decision processes
* Controlled creative ideation

You need more advanced structures.

Advanced prompt patterns act as **“cognitive scaffolding”**—templates that guide the LLM through **more intelligent, staged reasoning**.

**10.2 Pattern 1: Inner Monologue Prompting**

**Definition:** This pattern explicitly asks the model to reason internally before finalizing its answer.

**Theory**

Inspired by the psychology of **metacognition**, this technique simulates an LLM “thinking aloud,” reducing premature, incorrect completions.

**Prompt Template**

Think through the problem step by step before providing your final answer. Write your internal thoughts, then state your conclusion.

**Example**

prompt = """

Question: If Alice is older than Bob, and Bob is older than Charlie, who is the oldest?

Think step-by-step:

"""

💬 Model Output:

Alice is older than Bob. Bob is older than Charlie. Therefore, Alice is the oldest.

Answer: Alice

Use case: Chain-of-thought reasoning, logic puzzles, coding, math, ethics

**10.3 Pattern 2: Simulation-Based Prompting**

**Definition:** This technique puts the model into a **fictional or professional role** with behavioral constraints.

**Theory**

Role simulation activates latent “schemas” learned during pretraining. It's akin to **method acting for models**.

**Prompt Template**

You are a [ROLE], responding as if in a real [SCENARIO]. Follow the norms and tone expected of this role.

**Example**

prompt = """

You are a public health official answering citizen questions at a town hall.

Q: Is the new vaccine safe for children under 12?

A: [Your role-based response]

"""

Use case: Professional writing, character-driven dialogue, behavior testing, empathy modeling

**10.4 Pattern 3: Reflective Reasoning Loops**

**Definition:** You ask the model to **generate an answer, then critique or revise it**, like peer review.

**Theory**

Inspired by **double-loop learning** and **Socratic reasoning**, this pattern encourages the model to reflect on its own reliability.

**Prompt Template**

Step 1: Give your answer.

Step 2: Reflect on whether your answer has flaws.

Step 3: Revise it if needed.

**Example**

prompt = """

Q: What are the main risks of AI in healthcare?

Step 1: Initial answer

Step 2: Critique

Step 3: Improved answer

"""

Use case: Bias detection, safety analysis, writing refinement, model introspection

**10.5 Pattern 4: Simulated Dialogue (Multi-Agent)**

**Definition:** Simulate multiple roles or perspectives in conversation to surface conflict, consensus, or diversity.

**Prompt Template**

Alice (a doctor) and Bob (a patient) are discussing whether to use a new AI diagnostic tool.

Write their dialogue.

This technique mimics **dialogic learning** and is excellent for:

* Decision framing
* Debating controversial issues
* Exploring trade-offs

**10.6 Pattern 5: Structured Output Control**

Even the most advanced prompt patterns benefit from predictable formatting.

**XML or JSON prompt wrapper**

Please respond in the following format:

<thoughts>

[Your reasoning]

</thoughts>

<final\_answer>

[Your answer here]

</final\_answer>

Improves extractability, post-processing, and interpretability.

**10.7 Comparison of Advanced Prompt Patterns**

|  |  |  |
| --- | --- | --- |
| **Pattern** | **Best For** | **Weakness** |
| Inner Monologue | Math, logic, chain-of-thought | Longer completions |
| Simulation-Based | Persona and tone control | Risk of style over substance |
| Reflective Reasoning | Writing, safety, introspection | May not self-correct deeply |
| Multi-Agent Dialogue | Debates, decision analysis | Requires clear framing |
| Structured Output Control | Format reliability | Needs strict syntax |

**Exercises**

**Exercise 10.1: Rewrite for Inner Monologue**

Original prompt:

"What causes inflation?"

Improve it using inner monologue prompting.

**Exercise 10.2: Simulation-Based Prompt**

Create a prompt where the model acts as:

* A **hospital director**
* During a **budget crisis**
* Responding to public concern about staff layoffs

**Exercise 10.3: Reflective Loop Prompt**

Build a 3-step reflective reasoning prompt for this question:

"Is it ethical to use AI to make hiring decisions?"

**Recap**

|  |  |
| --- | --- |
| **Pattern** | **Example Use Case** |
| Inner Monologue | Step-by-step math or reasoning |
| Simulation-Based | Doctor-patient conversation |
| Reflective Loop | Ethical debates, policy writing |
| Multi-Agent Dialogue | Scenario-based decision-making |
| Structured Output Control | Reliable output formatting |

**References**

* Wei, J., et al. (2022). "Chain-of-thought prompting elicits reasoning in large language models." *arXiv:2201.11903*
* Kojima, T., et al. (2022). "Large Language Models are Zero-Shot Reasoners." *NeurIPS*
* Creswell, A., et al. (2023). “Selection-Inference: Explaining and Calibrating LLMs.” *arXiv:2302.14866*
* OpenAI Cookbook (2024). *Prompting best practices*