Guide to Using Large Multimodal Models v1.1

Technical Supplement 1 - Advanced Prompt Engineering Patterns

About This Supplement

Purpose & Scope
This supplement expands the C.G.A.F.R. framework from the Core Guide, providing advanced prompt engineering patterns for complex problem-solving. It is written for users who require greater control over model reasoning and output structure.

Audience: Practitioners, Power Users, AI Engineers
Prerequisites: Mastery of the C.G.A.F.R. framework (Core Guide § 2) and verification workflow (Core Guide § 4).
Outcome: Ability to select and apply advanced prompting patterns (Chain-of-Thought, Persona, etc.) to decompose complex tasks and generate auditable reasoning traces.

Key Objectives:

- Establish a universal framework (C.G.A.F.R.) for structuring effective and reproducible prompts.
- Equip users with advanced techniques to control output format, tone, and reasoning processes.
- Mitigate common failure modes like hallucination and instruction neglect through proven prompt patterns.

Developed by Russell Nida

© 2025 Russell Nida. Released under CC BY-NC-SA 4.0 for educational use.

Contents

Advanced Prompt Engineering Patterns 1

1. Zero-Shot, Few-Shot, and Chain-of-Thought Prompting 1

2. Tree-of-Thoughts for Complex Decomposition 3

3. Persona Pattern 3

4. Syntax and Tag Guidance 4

© 2025 Russell Nida. Released under CC BY-NC-SA 4.0 for educational use

Advanced Prompt Engineering Patterns

This supplement extends the foundational C.G.A.F.R. framework from Section 2 of the Guide, providing advanced patterns for complex problem-solving. These techniques offer greater control, more reliable outputs for nuanced tasks, and the ability to audit the model's reasoning process. This increased power requires a corresponding increase in critical verification.

Key Principle: Advanced patterns generate more sophisticated reasoning paths, not validated truth. The outputs from these methods are advanced drafts that demand rigorous human auditing, as explained in Section 4 of the Guide.

How to Use This Technical Supplement: These patterns are powerful tools to use *after* you have mastered the basic C.G.A.F.R. framework (Section 2 of the Guide) and the critical self-analysis techniques (Section 4 of the Guide). They are not a starting point.

Each pattern below progressively increases control, structure, and verification burden.

---

1. Zero-Shot, Few-Shot, and Chain-of-Thought Prompting

These are progressive levels of guidance—from none, to examples, to explicit reasoning. They define how much structure you provide to the model to shape its reasoning and output.

Zero-Shot

The model is given a task without any examples. It relies entirely on its internal training to interpret and execute the prompt.

When to Use: Simple, well-defined tasks where the instruction is unambiguous (e.g., "Summarize the following paragraph," "Translate this sentence to French").

Prompt Template:

[Direct Task Instruction] [Input Data]

Caution: Highly susceptible to ambiguity. The model's interpretation of "summarize" or "analyze" may not match your specific needs without examples.

Few-Shot

The model is provided with a small number of input-output examples before being given the actual task. This shows the model the pattern you want it to follow.

© 2025 Russell Nida. Released under CC BY-NC-SA 4.0 for educational use

When to Use: Tasks with a specific format, style, or reasoning pattern that is difficult to describe verbally (e.g., classifying sentiment, reformatting data, applying a specific branding tone).

Prompt Template:

Example 1 Input: [Input A] Example 1 Output: [Output A] Example 2 Input: [Input B] Example 2 Output: [Output B]

Actual Task Input: [Your Input] Actual Task Output:

Caution: The model may overfit to the examples. Ensure your examples are diverse and representative of edge cases. The model can also learn and replicate any biases present in the examples.

Tip: Use real but anonymized samples; the model generalizes better when examples vary in tone or length.

Chain-of-Thought (CoT)

You explicitly prompt the model to articulate its reasoning into a series of intermediate steps before delivering a final answer. This forces the model to externalize its logic, making it auditable.

When to Use: Complex reasoning tasks involving logic, math, or multi-step deduction where seeing the "work" is necessary for verification and error detection.

Prompt Template - Zero-Shot CoT:

[Complex Problem Statement] Let's think through this step by step.

Prompt Template - Few-Shot CoT:

Problem: [Problem A] Reasoning: [Step-by-step reasoning for A] Answer: [Final Answer A]

Problem: [Your Problem] Reasoning:

Critical Caution: CoT produces the illusion of logic, not its guarantee. The model can generate perfectly articulated, yet flawed reasoning. You must verify every step of the chain, not just the final answer. Treat the CoT as a transparent draft of the model's thought process, which you are now obligated to audit.

Validation Tip: After generating the reasoning chain, ask: "Now identify any logical leaps or unsupported assumptions in your prior reasoning."

© 2025 Russell Nida. Released under CC BY-NC-SA 4.0 for educational use

2. Tree-of-Thoughts for Complex Decomposition

This pattern extends Chain-of-Thought by forcing the model to generate, evaluate, and synthesize multiple independent reasoning paths for a single problem.

When to Use: High-stakes planning, strategic analysis, or complex problem-solving where a single line of reasoning is insufficient and you need to systematically explore competing hypotheses, solutions, or perspectives.

Prompt Template:

[Complex Problem Statement]

1. Break the problem down into 3-5 core sub-questions or components.
2. For each component, generate 2-3 distinct approaches or answers.
3. For each approach, evaluate its pros, cons, and underlying assumptions.
4. Synthesize the most robust elements from all approaches into a single, coherent solution. Justify your synthesis.

Caution: This is computationally intensive and can produce long, complex outputs. The "synthesis" is still a model-generated suggestion. The primary value is in laying out a comprehensive landscape of options, trade-offs, and assumptions for your analysis, not in providing a definitive answer.

Key Principle: The goal is divergent exploration (breadth) before convergent decision-making (synthesis).

---

3. Persona Pattern

This pattern assigns the model a specific role, expertise, or point of view to shape its responses.

When to Use: To generate output tailored to a specific audience, expertise level, or stylistic voice (e.g., a press release, a technical explanation for novices, or a risk assessment from a legal perspective).

Prompt Template:

You are an experienced [Role, e.g., Cybersecurity Analyst, Brand Copywriter, Financial Auditor]. Your audience is [Target Audience]. [Task Instruction using C.G.A.F.R.]

© 2025 Russell Nida. Released under CC BY-NC-SA 4.0 for educational use

Example Output: A prompt beginning "You are a skeptical product safety engineer..." will yield a more critical and risk-averse analysis than a neutral instruction.

Caution: The persona is a framing device, not a source of genuine expertise. The model does not "become" an expert; it mimics the language and concerns of one. All standard limitations regarding factuality and hallucination still apply.

Persona	Task Example	Expected Tone
Skeptical Product Safety Engineer	Evaluate design risk	Cautious, technical
Creative Brand Strategist	Generate tagline ideas	Expressive, informal
Compliance Officer	Review policy draft	Precise, conservative

Summary: Personas are context amplifiers, not authority sources.

---

4. Syntax and Tag Guidance

Using structured syntax like XML tags, JSON, or Markdown code fences provides strong cues to the model about how to separate, label, and format different parts of its response.

When to Use: Any task where you need to programmatically parse the output, ensure consistent structure for repeated use, or clearly separate different types of content (e.g., instructions, code, data, narrative).

Prompt Template (XML):

[Context and Goal] Please format your response using the following XML tags:

<summary>[Provide a brief overview here]</summary> <key_points>[List key points here]</key_points> <risks>[List potential risks here]</risks> <unanswered_questions>[List any uncertainties here]</unanswered_questions>

Prompt Template (JSON):

[Context and Goal] Please output a JSON object with the following keys: "summary", "key_points" (as an array), "risks" (as an array), "confidence_score" (as a number).

Caution: The model can produce invalid syntax (malformed JSON/XML). Always validate the output structure programmatically before use (e.g., using a parser or validator). Crucially, a

© 2025 Russell Nida. Released under CC BY-NC-SA 4.0 for educational use

syntactically perfect response can still contain logically invalid or fabricated data. The structure aids parsing, not verification.

Note: Markdown code fences (json ...) can also be used to enforce structured output boundaries.

Final Insight

Mastery in prompt engineering is not about finding a "perfect prompt," but about having a robust toolkit of patterns. Select the pattern that best fits the task's complexity. Remember the mindset from Section 1: as your "intern" (the LMM) uses more sophisticated methods, your role as the manager who demands to see the work and verifies it becomes even more critical. The power of advanced prompting grows in direct proportion to your responsibility for verification.

End of Technical Supplement 1

Cross-Reference: For orchestration and chaining methods, see Technical Supplement 4 - Governance & Deployment.

© 2025 Guide to Using Large Multimodal Models v1.1 - Authorized for Educational & Non-Commercial Use (CC BY-NC-SA 4.0).

© 2025 Russell Nida. Released under CC BY-NC-SA 4.0 for educational use