Prompt Engineering Best Practices and Techniques

Prompt engineering (PE) is defined in the sources as the practice of **crafting inputs** or queries to obtain the **best possible results** from a Large Language Model (LLM). It is the fundamental discipline of structuring text to guide the generative AI model toward desired outputs. In the larger context of Prompt Engineering, best practices are the systematic methods used to ensure precision, predictability, and safety, turning LLM interactions from a conversation into a form of **design**.
The goal of employing PE best practices is to transform prompting from a guessing game into a process that can be **tested, tuned, and trusted**, ensuring outputs are accurate, relevant, and actionable.

I. Fundamental Prompt Clarity and Specificity

The most essential best practices center on eliminating ambiguity, which is one of the most common causes of poor LLM output.

Best Practice	Description and Rationale
Be Clear, Direct, and Specific	Precise prompts prevent vague responses, guiding the AI to deliver targeted answers. Avoid vague phrasing like "make it better" or "keep it simple". You must define exactly what you want, including the structure and what to avoid. For example, instead of "Tell me about climate change," specify: "Explain the main causes of climate change and its effects on global temperatures".
Use Action-Oriented Language	Treat your prompt like an instruction to a function, using direct, specific, and purposeful commands. Avoid weak openings like "Can you..." or "I need...". When possible, use action verbs like "write," "list," or "translate".
Maintain Simple Language	LLMs work best when language is clear, direct, and easy to understand. Skip slang, idioms, or fuzzy phrases like "better" or "robust" unless clearly defined, as these are open to interpretation by the model.
Define Constraints	Explicitly state limitations and boundaries for the response, such as a word count, sentence constraint, or required formatting. Constraints help ensure conciseness and adherence to task requirements.

Role Assignment and Framing	Assigning the model a specific role (persona) guides its tone, style, and domain expertise (e.g., "You are a programming instructor"). Additionally, providing **framing** (background information) helps the model better understand the research context.

II. Advanced Structural Techniques (Prompting Methods)

For more complex tasks, prompt engineering integrates specific structuring techniques to improve reasoning, accuracy, and format adherence.

Technique	Description and Rationale
Few-Shot Prompting (Examples)	This technique involves explicitly providing a few high-quality input/output **examples** within the prompt to teach the model the desired pattern, format, tone, or underlying logic. Few-shot prompting closes the knowledge gap that can cause models to fill in information with probabilities (leading to hallucinations).
Chain-of-Thought (CoT) Prompting	CoT boosts reasoning ability by instructing the model to **articulate its thinking process step-by-step** before providing the final answer. This is invaluable for analyzing complex scientific concepts, solving multi-step logic problems, and generating critical analysis or hypotheses.
Output Formatting	This dictates the **structure** of the response, ensuring outputs are consistent and parseable for downstream systems. Examples include explicitly requesting JSON, tables, bullet points, or a specific numbered list.
Decomposition/ Multi-Step Workflows	Break down a single complex request into a sequence of focused subtasks (e.g., research first, reasoning next, formatting last). This solves the debugging challenge of monolithic prompts and provides checkpoints for verification.
Anchoring/Prefilling	Providing the beginning of the desired output or a partial structure (e.g., predefined labels like "Summary: Impact: Mitigation:") steers the model toward completion, reducing randomness and output drift.

Use Delimiters	Use visual or structural markers (such as triple back ticks ``` or double hash marks ##) to clearly distinguish different sections within a prompt, improving readability and reducing ambiguity for the model.

III. Iteration and Systematic Engineering

The sources emphasize that effective PE is an iterative process requiring continuous measurement and refinement, moving toward a systems-level discipline.

- **Iterative Refinement:** Prompting should be viewed as an **interactive, evolving process**. Continuously test different phrasings and parameters, gather feedback, analyze the output, and refine the prompt until the desired precision is achieved. Treat the LLM's initial summary or response as a first draft, requiring refinement using human expertise.
- **Verification and Rigor:** Researchers must **always check and verify** the LLM's findings and cited material against the original sources, as errors or invented information (hallucinations) may occur. Maintaining academic rigor and responsibility for the work's accuracy is essential, even with LLM assistance.
- **Evaluation-First Approach:** For production-level optimization, **measurement comes before optimization**. This means defining key criteria upfront and implementing evaluation infrastructure to track metrics like hallucination rates and completeness before applying prompt optimization techniques.
- **Structured Frameworks:** Adopting structured frameworks like **CRAFT** (Capability, Role, Action, Format, Tone) provides precise control for specialized tasks, while **ICE** (Instruction, Context, Examples) is ideal for complex tasks requiring detail. Structured frameworks offer standardized templates that reduce errors and improve output quality.

In essence, prompt engineering best practices ensure that instructions are **clear, specific, and goal-oriented**, which is a critical necessity for leveraging LLMs effectively. However, it is important to note that many modern discussions refer to Prompt Engineering (PE) as only the **tactical component** of the larger strategy known as **Context Engineering**, which deals with the entire ecosystem (data, memory, tools) surrounding the prompt.