Prompt Engineering / Meta Prompts

Compare two versions of a prompt on the same test inputs and explain which performs better and why.
Difficulty: Intermediate
Model: GPT-4 / Claude / Gemini
Use Case: A/B Testing Prompts, Iteration Review, Team Alignment
Updated: May 2026
Why This Prompt Exists
You changed your prompt. Is it better? You run a few tests and guess. That’s not rigorous — and you’re probably wrong.

You get:

- choosing the wrong version because you tested on easy examples
- no systematic way to compare prompts across multiple dimensions
- team debates about which prompt is better (no data)
- improvements that actually hurt performance on edge cases
- no learning from prompt iterations (what worked and why)

But prompt comparison needs structure:

- test set: diverse inputs covering typical, edge, and adversarial cases
- metrics: accuracy, latency, cost, consistency
- trade-offs: Version A is better on X but worse on Y
- explanation: why one version outperforms (not just which)
- recommendation: which to deploy and under what conditions

Without comparison, you iterate blindly.

This prompt provides structured comparison across test inputs.

The Prompt

```
Assume the role of a prompt evaluation scientist who compares prompt versions.

Your task is to compare two prompt versions and recommend which to deploy.

Generate:

1. PROMPT VERSIONS SUMMARY
   - Version A: [description of key changes from baseline]
   - Version B: [description of key changes from baseline]
   - Baseline (if applicable): [original version]

2. TEST INPUTS USED
   - List of test inputs (or describe diversity)

3. COMPARISON MATRIX (per test input)

| Test Input | Version A Output | Version B Output | Winner | Notes |
|------------|------------------|------------------|--------|-------|
| [input 1] | [output] | [output] | A/B/Tie | [why] |

4. AGGREGATE METRICS
   - Accuracy: A: X/10, B: Y/10
   - Consistency (same output across runs): A: High/Med/Low, B:
```

High/Med/Low
   - Verbosity (output length): A: [avg], B: [avg]
   - Failure cases: A: [list], B: [list]

5. TRADE-OFF ANALYSIS
   - Where A wins (and why)
   - Where B wins (and why)
   - Any critical failure that disqualifies a version

6. RECOMMENDATION
   - Which version to deploy (A / B / Neither / Both for different cases)
   - Rationale (2-3 sentences)
   - What to monitor after deployment

INPUTS:

Version A prompt:
[PASTE]

Version B prompt:
[PASTE]

Test inputs (5-10 diverse examples):
[PASTE INPUTS — OR DESCRIBE CATEGORIES TO GENERATE]

Success criteria (what "better" means):
[E.G., "Higher accuracy on edge cases, even if slower"]

```
Model:
[GPT-4 / CLAUDE / GEMINI]

RULES:
- If test inputs aren't provided, generate them covering typical,
edge, and adversarial cases
- Flag if prompts produce identical outputs (no meaningful difference)
- Consider cost and latency if relevant to your use case
- Note if a version is more consistent but less creative (trade-off)
- Don't recommend a version that fails catastrophically on any test
input
```

How To Use It

- Run this before deciding which prompt version to deploy — don’t guess.
- Use at least 10 test inputs covering typical, edge, and adversarial cases.
- Run each version multiple times per input to measure consistency.
- Pay attention to trade-offs — version A may be better overall but worse on your most important case.
- Save comparison reports to build institutional knowledge about what works.

Example Input

Version A prompt:
“Summarize this text in 2-3 sentences.”

Version B prompt:
“You are a professional editor. Summarize the following text in exactly 2-3 sentences. Focus on main argument and key supporting point only. Do not include examples or minor details. Output format: Summary: [text]”

Test inputs:
“5 diverse paragraphs — one short, one long, one with multiple arguments, one with no clear argument, one technical”

Success criteria:
“More concise summaries that capture main argument without extraneous details”

Why It Works
Most prompt iteration relies on gut feeling — “this feels better” — which is unreliable and untestable.

This framework improves outcomes by forcing:

- structured test inputs (diversity, not convenience)
- side-by-side comparison (per input, not aggregate only)
- multiple metrics (accuracy, consistency, verbosity)
- trade-off analysis (where each version wins)
- clear recommendation (deploy A, B, or neither)

Great prompt comparison doesn’t just declare a winner — it tells you why one version is better and for what cases.

Build Better AI Systems

Subscribe for advanced prompt engineering, AI coding tools, debugging frameworks, and practical strategies for developers and engineers.

Carefully engineered prompts for people doing real work.

Share this:

- Share on Facebook (Opens in new window) Facebook
- Share on X (Opens in new window) X

See also Why This Prompt Exists