Prompt Engineering / Prompt Optimization

Analyze where a prompt fails (specific inputs or output types) and suggest targeted fixes.
Difficulty: Advanced
Model: GPT-4 / Claude / Gemini
Use Case: Debugging Prompts, Error Analysis, Targeted Improvement
Updated: May 2026
Why This Prompt Exists
Your prompt works 85% of the time. The 15% failure is killing you — but you don't know why, so you guess at fixes.

You get:

- fixing problems that aren't the real problem
- breaking what already works while trying to fix failures
- no systematic error analysis before refining
- fixes that address symptoms, not root causes
- repeating the same types of failures across prompt versions

But failures have patterns:

- input characteristics: what kind of inputs cause failures? (length, topic, structure)
- output error types: what's wrong? (hallucination, omission, format, reasoning)
- failure frequency: how often does each failure type occur?
- root cause: prompt ambiguity, missing examples, unclear constraints
- fix strategy: add example, clarify instruction, add constraint, add fallback

Without failure analysis, you refine blind.

This prompt analyzes failures and suggests targeted fixes.

The Prompt

```
Assume the role of a prompt debugger who analyzes failure patterns.

Your task is to analyze where a prompt fails and recommend specific
fixes.

Generate:

1. FAILURE LOG (input)
   - List of inputs where prompt failed
   - Expected output vs. actual output

2. FAILURE PATTERN ANALYSIS

| Failure Type | Count | % of failures | Input characteristics | Root
cause hypothesis |
|--------------|-------|---------------|-----------------------|-------
---------------|
| [e.g., Hallucination] | X | X% | [e.g., "long inputs >500 words"] |
[e.g., "no instruction to say 'I don't know'"] |

3. INPUT CHARACTERISTICS CORRELATION
   - Length: failures happen at [short/medium/long] lengths
   - Topic: failures cluster around [specific topics]
   - Structure: failures happen with [specific formats]
   - Ambiguity: failures happen when input is [clear/ambiguous]

4. ROOT CAUSE ANALYSIS (per failure type)
```

```
   - Most likely root cause
   - Evidence for this cause

5. TARGETED FIX RECOMMENDATIONS (per failure type)

| Failure Type | Fix Type | Specific Change | Expected improvement |
|--------------|----------|-----------------|----------------------|
| [type] | [add example / clarify / add constraint] | [exact wording to add/change] | [X% reduction] |

6. FIXED PROMPT (proposed)
   - The prompt with recommended changes

7. VALIDATION PLAN
   - How to test that fixes work without breaking other cases

INPUTS:

Original prompt:
[PASTE THE PROMPT]

Failure examples (at least 5-10):
[PASTE INPUTS + INCORRECT OUTPUTS]

Success examples (for comparison):
[PASTE INPUTS + CORRECT OUTPUTS, OPTIONAL]

Task type:
[CLASSIFICATION / GENERATION / EXTRACTION / OTHER]
```

```
RULES:
- Collect at least 10 failure examples before analyzing (small samples mislead)
- Look for patterns across failures — one-off errors may not need fixing
- Fix root causes, not symptoms (addressing one failure may fix many)
- Test fixes on success examples to ensure you didn't break them
- Save failure logs to build a regression test suite
```

How To Use It

- Collect failures systematically (log all prompt outputs, flag bad ones).
- Don’t fix after one failure — wait until you see a pattern (at least 5-10 similar failures).
- Fix root causes: if prompts fail on long inputs, add instruction for long inputs — don’t just tweak wording.
- Test your fix on the failure examples AND on success examples (ensure you didn’t regress).
- Save failure examples as regression tests for future prompt versions.

Example Input

Original prompt:
“Classify customer emails as URGENT, NORMAL, or LOW. Respond with only one word.”

Failure examples:
“Input: ‘My account is locked and I need to pay a bill due tomorrow.’ Output: NORMAL (should be URGENT). Input: ‘Quick question about your product.’ Output: LOW (should be NORMAL). Input: ‘URGENT: Please help!!! My payment failed and I’ll be charged a late fee.’ Output: NORMAL (should be URGENT).”

Success examples:

“Input: ‘I have a general question about features.’ Output: LOW.”

Why It Works
Most prompt refinement is reactive — “this output was wrong, let me tweak something” — which addresses symptoms, not causes.

This framework improves outcomes by forcing:

- failure pattern analysis (what kind of errors?)
- input correlation (when do failures happen?)
- root cause identification (why is the prompt failing?)
- targeted fix recommendations (fix the cause, not the symptom)
- validation planning (ensure fixes don’t break working cases)

Great failure-driven refinement doesn’t guess at fixes — it analyzes errors and fixes root causes.

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 Prompt A/B Test Designer