Prompt Engineering / Prompt Optimization

Optimize a prompt for a specific metric (accuracy, brevity, creativity, safety) with trade-off analysis.
Difficulty: Advanced
Model: GPT-4 / Claude / Gemini
Use Case: Task-Specific Tuning, Multi-Objective Optimization
Updated: May 2026
Why This Prompt Exists
“Good prompt” is not a metric. Different tasks need different trade-offs — but most prompts optimize for nothing specific.

You get:

- prompts that are accurate but verbose (when you need brevity)
- prompts that are creative but inconsistent (when you need reliability)
- prompts that are safe but unhelpful (overly cautious)
- no ability to tune for your specific use case
- trade-offs that are implicit, not explicit

But optimization requires metrics:

- accuracy: correct vs. incorrect (binary or scored)
- brevity: output length, token count
- creativity: novelty, diversity, unexpectedness
- safety: harmful output rate, refusal rate
- consistency: same output for same input
- latency: time to first token
- cost: total tokens (input + output)

Without metrics, you optimize for “feels better” — which is unreliable.

This prompt optimizes prompts for specific, measurable metrics.

The Prompt

```
Assume the role of a prompt optimization engineer who optimizes for
measurable metrics.

Your task is to modify a prompt to improve a target metric.

Generate:

1. BASELINE PERFORMANCE
   - Current prompt
   - Target metric: [name] = [value] (from testing)
   - Other metrics (for trade-off awareness)

2. OPTIMIZATION TARGET
   - Metric to improve: [accuracy / brevity / creativity / safety /
consistency / latency / cost]
   - Current value: [X]
   - Target value: [Y]
   - Acceptable degradation in other metrics: [list with limits]

3. PROMPT MODIFICATIONS (test these in order)

| Modification | Expected effect on target metric | Expected effect on
other metrics | Rationale |
|--------------|----------------------------------|--------------------
```

```
-----------|-----------|
| [add instruction] | +[X]% | [e.g., may increase tokens] | [why it works] |
| [add example] | +[X]% | [e.g., minimal effect] | [why it works] |
| [remove instruction] | +[X]% | [e.g., may reduce accuracy] | [why it works] |

4. PROPOSED OPTIMIZED PROMPT
   - Full prompt with recommended modifications

5. TRADE-OFF ANALYSIS
   - What you gain (improvement in target metric)
   - What you lose (degradation in other metrics)
   - Is the trade-off worth it? (Yes/No/Maybe — test to find out)

6. TESTING RECOMMENDATIONS
   - How to validate improvement (A/B test, sample size)
   - What to monitor after deployment

INPUTS:

Current prompt:
[PASTE THE PROMPT]

Target metric and baseline (from testing):
[E.G., "Accuracy = 85% on test set of 200 examples"]

Other metrics and baseline (optional):
[E.G., "Average output length = 150 tokens"]
```

```
Acceptable trade-offs:
[E.G., "Can increase output length up to 200 tokens, can reduce
accuracy no more than 2%"]

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

RULES:
- Measure before you optimize (you need a baseline)
- Optimize one metric at a time (multi-objective optimization is
complex)
- Trade-offs are inevitable — be explicit about what you're
sacrificing
- Test proposed changes (don't assume they'll work)
- If you can't measure it, you can't optimize it
```

How To Use It

- Measure baseline performance before optimizing — you need a starting point.
- Optimize one metric at a time — trying to improve everything usually improves nothing.
- Be explicit about acceptable trade-offs — “accuracy can drop 2% for 50% shorter outputs”
- Test each modification separately before combining them.
- Re-measure after each change to ensure you’re actually improving.

Example Input

Current prompt:
“Summarize this article in a few sentences.”

Target metric and baseline:
“Brevity = average 120 tokens per summary (measured on 100 articles)”

Other metrics and baseline:
“Accuracy (factual correctness) = 90% on human evaluation”

Acceptable trade-offs:
“Can reduce accuracy up to 5% (to 85%) if brevity improves significantly (to under 60 tokens)”

Why It Works
Most prompt optimization is subjective — “this version seems better” — which is impossible to reproduce or trust.

This framework improves outcomes by forcing:

- baseline measurement (where are we starting?)
- explicit optimization target (what metric are we improving?)
- trade-off specification (what are we willing to lose?)
- testable modifications (not just “try things”)
- validation recommendations (how to confirm improvement)

Great metric-based optimization doesn’t guess — it measures, changes, and measures again.

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 Failure-Driven Refiner