Prompt Engineering / Prompt Optimization

Convert a prompt optimized for one model (e.g., GPT-4) to work effectively on another (e.g., Claude, Gemini).
Difficulty: Advanced
Model: GPT-4 / Claude / Gemini
Use Case: Model Migration, Multi-Model Deployment, Cost Optimization
Updated: May 2026
Why This Prompt Exists
Prompts don’t transfer. A prompt that works perfectly on GPT-4 may fail on Claude or Gemini — and vice versa. Starting from scratch is expensive.

You get:

- switching models and watching your accuracy collapse
- rewriting prompts from scratch for each model
- unable to use cheaper models because prompts don’t transfer
- no systematic understanding of what each model needs
- vendor lock-in (can’t switch because prompts are optimized for one model)

But models have known differences:

- GPT-4: handles long instructions well, follows format specifications reliably
- Claude: prefers natural language, better at refusal, XML tags work well
- Gemini: needs explicit step-by-step, benefits from few-shot examples
- Llama: shorter context, more sensitive to instruction order
- Mistral: needs very clear formatting, markdown helps

Without porting guides, model switching is painful.

This prompt converts prompts between models while preserving functionality.

The Prompt

```
Assume the role of a model portability engineer who translates prompts
between LLMs.

Your task is to convert a prompt from a source model to a target
model.

Generate:

1. SOURCE MODEL CHARACTERISTICS
   - Model: [GPT-4 / CLAUDE / GEMINI / LLAMA / MISTRAL / OTHER]
   - Prompt style observed (from given prompt)

2. TARGET MODEL CHARACTERISTICS
   - Model: [GPT-4 / CLAUDE / GEMINI / LLAMA / MISTRAL / OTHER]
   - Known preferences:
     * Instruction format: [natural language / structured / bullet
points / XML]
     * Few-shot effectiveness: [high / medium / low]
     * Refusal behavior: [explicit / subtle / rare]
     * Context length sensitivity: [high / medium / low]

3. CONVERSION CHANGES MADE

| Original (Source) | Adapted (Target) | Reason |
|-------------------|------------------|--------|
```

| [instruction] | [rewritten] | [e.g., "Claude prefers natural language over bullet points"] |
| [format spec] | [rewritten] | [e.g., "Gemini needs explicit step-by-step"] |

4. PORTED PROMPT
   - Full prompt adapted for target model

5. EXPECTED PERFORMANCE SHIFTS
   - What may improve (e.g., "Target model may be more concise")
   - What may degrade (e.g., "Target model may need more examples")
   - Mitigation strategies

6. TESTING RECOMMENDATIONS
   - Critical test cases to run after porting
   - Expected differences vs. source model

7. PORTS THAT MAY NEED MANUAL TUNING
   - Aspects that automated porting may miss

INPUTS:

Source prompt (optimized for source model):
[PASTE THE PROMPT]

Source model:
[GPT-4 / CLAUDE / GEMINI / LLAMA / MISTRAL / OTHER]

Target model:

```
[GPT-4 / CLAUDE / GEMINI / LLAMA / MISTRAL / OTHER]

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

Performance target (on target model):
[E.G., "Match source model accuracy within 5%"]

RULES:
- Different models have different "personalities" — adapt tone accordingly
- Claude responds well to XML tags (, )
- Gemini needs explicit reasoning instructions ("think step by step")
- GPT-4 handles both structured and natural language well
- Llama and Mistral need very clear, unambiguous instructions
- Always test ported prompts before deploying (automated porting is not perfect)
```

How To Use It

- Run this when switching model providers or adding a new model to your portfolio.
- Test the ported prompt on at least 20-50 examples before deploying.
- Expect some performance differences — you may need additional tuning.
- Keep model-specific optimizations separate (don’t try to maintain one prompt for all models).
- Document what changes were made so you can port back if needed.

Example Input

Source prompt:
“You are a helpful assistant. Classify the sentiment of the following text as POSITIVE,

NEUTRAL, or NEGATIVE. Respond with only one word. Text: {{text}}”

Source model:
“GPT-4”

Target model:
“CLAUDE”

Task type:
“CLASSIFICATION”

Why It Works
Most prompt engineers learn one model’s quirks and assume others behave the same — they don’t.

This framework improves outcomes by forcing:

- model characteristic awareness (what each model needs)
- explicit conversion mapping (what changed and why)
- performance shift prediction (what to expect)
- testing recommendations (how to validate)
- manual tuning identification (what automation misses)

Great cross-model porting doesn’t produce identical results — it produces results that are equally good for the target model.

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