Build with LLMs

Providers, models and prompt engineering

What we'll cover

- The LLM provider landscape and model comparison
- Key factors for model selection: capabilities, cost, speed
- Provider-specific prompt engineering techniques
- Advanced tools for prompt creation and evaluation

Why This Matters

Understanding the LLM ecosystem helps you select the right model for each task, optimize costs, and craft effective prompts. These skills are essential for both using AI in your development workflow and building AI-enhanced applications.

Overview of Leading LLM Providers

CW JV

OpenAI - Industry leader with GPT-4o and turbo models. Function calling, computer use, and JSON mode. Premium pricing. First to market with many features.

Anthropic - Claude models focused on safety, reduced hallucinations, and complex reasoning. Extended thinking mode. Competitive pricing for mid-tier options.

Google - Gemini models with state-of-the-art multimodal capabilities and 2M token contexts. Flash variants for cost optimization. Extensive Google product integrations.

DeepSeek - Emerging player with strong performance at moderate prices. Excel at code, math and reasoning. R1 model comparable to premium offerings at lower cost.

Mistral AI - French provider wth efficient models with and performance-to-size ratio. Both closed and open models. Strong multilingual support and coding capabilities.

Open Models - Strong open-source options like Llama 3, Mistral, and Qwen. Self-hostable or available through providers like Together AI, Replicate, and Anyscale.

Criteria for Evaluating LLMs

Technical Capabilities

- **Context window**: Maximum input size the model can process
- **Strength/Intelligence**: Problem-solving and reasoning abilities
- **Tone control**: Ability to adjust style and formality
- **Multimodal support**: Handling images, audio, video inputs
- **Special features**: Function calling, JSON mode, grounding

Operational Factors

- **Cost**: Price per million tokens (input and output)
- **Speed**: Tokens generated per second
- **Latency**: Time to first token response
- **Reliability**: Service uptime and error rates
- **Ease of integration**: API complexity and documentation
- **Security & Privacy**: Data handling practices

Context Window Comparison

Impact on Applications

- Process entire codebases in one pass
- Analyze multiple documents simultaneously
- Maintain long conversation history
- Reduce document chunking complexity

Size Categories

Small (4K-16K): GPT-3.5, Llama 3 8B

Chat applications, basic Q&A, simple tasks

Medium (32K-128K): Claude Haiku, DeepSeek

Single documents, tech docs, short codebases

Large/Massive (200K+): Claude 3.7, GPT-4.1, Gemini

Multiple documents, entire repositories, books

Model Strength & Capabilities

Dimensions of Strength

- **Reasoning**: Complex problem-solving, logic, mathematics
- **Knowledge**: Breadth and accuracy of information
- **Instruction following**: Precisely executing requests
- **Coding**: Quality and accuracy of generated code
- **Consistency**: Reliability and fewer hallucinations

Evaluation Methods

Academic benchmarks, provider evaluations, third-party comparisons, and real-world testing measure capabilities.

Common Benchmarks

- **MMLU**: 57 subjects across STEM, humanities, law, etc. Tests breadth of knowledge
- **HumanEval**: 164 programming problems testing code generation accuracy
- **MATH**: Advanced mathematical reasoning from arithmetic to calculus
- **GSM8K**: Elementary math word problems requiring multi-step reasoning
- **TruthfulQA**: Measures model's tendency to generate false information
- **GPQA**: Graduate-level questions in biology, physics, and chemistry

Speed & Latency Factors

Speed Metrics

- **Time to first token (TTFT)**: Initial response time
- **Tokens per second (TPS)**: Generation throughput
- **Time to last token (TTLT)**: Complete response time
- **Maximum generation length**: Output token limit

User Experience Impact

- Low latency is critical for interactive applications
- Users perceive responses under 300ms as instant
- Streaming masks slower generation speeds
- 10 TPS is faster than most people can read
- TTFT is the key metric I optimise for in chat situation

Cost Comparison (May 2025)

CW JV

Provider/Model	Input (per 1M)	Output (per 1M)	Prompt Caching (per 1M)
GPT-4.1	$2.00	$8.00	$0.50
GPT-4.1-mini	$0.40	$1.60	$0.10
O4-mini-low	$0.15	$0.60	$0.04
O4-mini-high	$0.25	$1.00	$0.06
O3	$10.00	$40.00	$2.50
Claude 3.7 Sonnet	$3.00	$15.00	$0.30
Claude 3.5 Haiku	$0.80	$4.00	$0.08
Gemini 2.5 Pro	$1.25	$10.00	$0.31
Gemini 2.5 Flash	$0.10	$0.60	$0.03
Gemini 2.5 Flash Thinking	$0.10	$3.50	$0.03
DeepSeek R1	$0.55	$2.19	$0.14
DeepSeek V3	$0.27	$1.10	$0.07

Prompt Caching Comparison

OpenAI - 75% discount on cached inputs, no storage costs, automatic caching with seamless integration. Particularly cost-effective with expensive models like O3.

Anthropic - 90% discount on cached inputs (paying only 10% of regular price). Cache lasts 5 minutes by default, making it ideal for high-throughput applications.

Google - 75% discount on cached inputs, requires minimum token threshold (1,024 tokens for Flash models, 2,048 tokens for Pro models) to trigger caching benefits.

DeepSeek - Approximately 75% discount on cached inputs for both R1 and V3 models. Integrates well with existing LLM frameworks.

Mistral - Currently offers no prompt caching capabilities. Consider alternative optimization strategies when working with Mistral models.

Best Practices

- Structure applications to reuse prompt templates
- Use consistent system prompts across requests
- Move variable content to the end of prompts
- Monitor cache hit rates and optimize accordingly

Additional Cost Considerations

Cost Optimization Strategies

- **Model selection**: Balance between cost, speed, and quality
- **Chunking**: Process long documents in pieces to manage token usage
- **Efficient prompting**: Clear instructions reduce token waste
- **Hybrid approach**: Use cheaper models for preprocessing
- **Batch processing**: Discounted rates for non-realtime operations
- **Prompt caching**: Order your prompts to save money

Special Features & Multimodality

Key Capabilities

- **Function calling**: Structured tool usage (OpenAI, Claude)
- **JSON mode**: Reliable structured output (OpenAI)
- **Computer use**: Browsing and file access (OpenAI)
- **Extended thinking**: Long-form reasoning (Claude)
- **Image understanding**: Processing visual content
- **Image generation**: Creating visuals from text
- **Search / grounding**: Search the web and generate citations

Multimodal Support

Models can process multiple types of data including text, images, audio, and video in a single context window.

Files

Search across large files to augment query with additional context from a knowledge base

Provider Selection

Match technical requirements to specific provider capabilities based on your application needs.

Prompt Engineering

Prompt Engineering: Best Practices

Prompt Engineering: OpenAI Models

Model Families

GPT-4.1 Series - General-purpose models with comprehensive capabilities. Follow instructions literally. Excellent for creative tasks, UI/UX, and code generation.

O-Series (Reasoning) - Specialized for complex reasoning tasks, math, science problems, and algorithmic challenges. Prefer minimal prompting.

Format Components

- **System Message**: Sets persistent instructions and role
- **User Message**: Contains the specific query or task
- **Assistant Message**: The model's response
- **Function Calling**: Structured tool usage with APIs
- **JSON Mode**: Enforces valid JSON output format

GPT-4.1 Prompting Techniques

CW JV

Key Strategies

Be Explicit & Specific - GPT-4.1 follows instructions literally. State exactly what you want, including output format, style, and constraints.

Sandwich Method - Place critical instructions at both the beginning and end of long prompts to ensure they're not missed.

Multiple Constraints - Clearly enumerate what the model should and shouldn't do in numbered lists for clarity.

Advanced Techniques

Format Specification - Precisely define output structures (JSON, tables, sections) with examples to ensure consistency.

Few-shot Examples - Provide 2-3 high-quality examples for complex or unusual formats. Structure as: instructions → examples → final task.

Step-by-Step Reasoning - For complex problems, explicitly request step-by-step thinking: "Think through this carefully, step by step."

O-Series Prompting Techniques

Key Strategies

Minimal Prompting - Use simple, direct questions without excessive context or instructions. O-Series models perform worse with too many examples or detailed guidance.

Zero-shot Preferred - These models are optimized for zero-shot or at most one-shot prompting. Avoid multiple examples.

Let it Reason - Don't instruct how to solve the problem; just clearly state the problem itself. The model conducts internal reasoning naturally.

Standard GPT-4.1 Prompt:

```
Analyze this sales dataset to identify trends:
1. First, summarize revenue patterns
2. Then, identify top 3 products
3. Next, find underperforming segments
4. Finally, recommend 3 specific actions

Format with clear headings.
```

Better O-Series Prompt:

```
Analyze this sales dataset.
Identify trends, top performers, underperforming segments,
and suggest actions to improve sales.
```

Task-Specific Prompting for OpenAI

CW JV

GPT-4.1 Coding Prompt:

```
Create a React component that displays a
paginated data table with requirements:
1. Support for sorting columns
2. Search functionality across all fields
3. Responsive design for mobile and desktop
4. Use only Tailwind CSS for styling
5. Include error handling for API failures
```

O-Series Coding Prompt:

```
Write a React component for a paginated data
table with sorting, search, and responsive
design using Tailwind CSS.
```

GPT-4.1 Content Prompt:

```
Write a blog post about renewable energy for a
technical audience. The post should:
- Be approximately 1200 words
- Use a professional but accessible tone
- Include 3 sections: current challenges,
  promising technologies, and future outlook
- Cite recent developments since 2023
- Include a brief executive summary at the beginning
```

O-Series Content Prompt:

```
Write a technical blog post about renewable
energy challenges, technologies, and future outlook.
```

Parameter Optimization for OpenAI

CW JV

Temperature Settings

GPT-4.1 Models

- **Low (0-0.3):** Factual tasks, coding, structured outputs
- **Medium (0.4-0.7):** Balanced for general use, explanations
- **High (0.8-1.0):** Creative writing, brainstorming, diverse ideas

O-Series Models

- **Default (0.7):** Works well for most reasoning tasks
- **Higher values:** Less effective; may introduce randomness
- **Note:** Focus on reasoning_effort parameter instead

Common Pitfalls to Avoid

GPT-4.1 Pitfalls

- Assuming the model will infer unstated requirements
- Providing contradictory guidance in long prompts
- Adding unnecessary context that dilutes key instructions

O-Series Pitfalls

- Adding too many examples or excessive guidance
- Explicitly requesting reasoning steps (redundant)
- Adding restrictions the model would naturally follow
- Ignoring the reasoning_effort parameter setting

Model Selection Guide for OpenAI

CW
JV

Choose GPT-4.1 when:

- Working with creative content generation
- Handling conversational or UI/UX tasks
- Processing long documents
- Coding frontend or standard applications
- When precision in following detailed instructions matters

Technical Capabilities:

- GPT-4.1: 1M token context, improved diff
- GPT-4.1 mini: Faster, cost-effective
- GPT-4.1 nano: Highest speed, lowest cost

Choose O-Series when:

- Solving complex math or science problems
- Working on algorithmic challenges
- Requiring high accuracy on logical tasks
- Analyzing data with multiple reasoning steps
- When solution approach is more important than formatting

Technical Capabilities:

- O1: Reasoning flagship for complex problems
- O3-mini: Cost-effective with strong STEM
- O4-mini: Latest small reasoning model

OpenAI Prompting Resources

CW JV

General Guide

Core prompt engineering principles that apply across the OpenAI ecosystem

OpenAI Prompt Engineering

GPT-4.1 Guide

Comprehensive guide for the latest GPT models with examples and best practices

GPT-4.1 Prompting Guide

O-Series Guide

Specialized techniques for OpenAI's reasoning-focused models

O-Series Models Documentation

Prompt Engineering: Claude Models

CW JV

Model Families

Claude 3 Series - Versatile models with strong reasoning capabilities. Particularly excels at following structured instructions and maintaining consistency. Leaders in harmlessness and reduced hallucinations.

Format Components

- **System Parameter**: Defines Claude's role and persona
- **Human Messages**: Contains instructions and content
- **XML Tags**: Structure different parts of the prompt
- **Assistant Message**: Optional prefill for responses

Claude Format Example:

```
# System Parameter
You are a data science expert.

# Human Message
<instructions>
Analyze this dataset focusing on trends.
</instructions>

<data>[content here]</data>

Think inside <thinking> tags first,
then provide conclusions in <answer> tags.
```

Claude Prompting Techniques

CW
JV

Key Strategies

XML Tagging - Claude's signature feature. Structure prompts with semantic XML tags that organize content and guide Claude's thinking process. Claude is specifically trained to respect and utilize XML structure.

Extended Thinking - Give Claude dedicated space to work through reasoning steps before providing a final answer. Allocate thinking budget at run time.

Golden Rule Clarity - If a colleague with minimal context would be confused by your prompt, Claude likely will be too. Include specific context, constraints, and expectations.

Common XML Tags

Task Organization

- **<instructions>**: Primary task definition
- **<context>**: Background information
- **<data>**: Raw content to analyze
- **<example>**: Demonstration of desired output

Response Structure

- **<thinking>**: Space for step-by-step reasoning
- **<answer>**: Final, refined response
- **<analysis>**: Detailed examination of data
- **<summary>**: Condensed key points

Claude: Prompt Generation Tools

CW JV

Prompt Generator

Purpose - Helps overcome the "blank page problem" when creating AI prompts. Generates well-structured templates following Anthropic's best practices.

How It Works - Tell it what you're trying to accomplish in plain language (like "Create a data analysis prompt that summarizes trends"), and it generates a complete prompt template with proper XML tags, relevant sections, and formatting guidance tailored to your specific task.

Best For - Initial prompt drafts, exploring new use cases, or when unsure how to structure a prompt effectively. Works with all Claude models.

Prompt Improver

Purpose - Analyzes and enhances existing prompts through a systematic improvement process. Optimizes for accuracy in complex reasoning tasks.

How It Works - Takes your existing prompt and enhances it by adding XML structure, introducing reasoning instructions, and optimizing examples. Transforms basic prompts into well-structured templates that guide Claude to provide more accurate and detailed responses.

Best For - Complex tasks requiring high accuracy, detailed step-by-step reasoning, and when you already have a basic prompt to improve.

Claude: Test Case Generator

Purpose - Systematically test prompts by generating diverse test cases with variable inputs. Enables side-by-side comparison of outputs across multiple scenarios in the Anthropic Console.

How It Works - Uses templated prompts with placeholders. The tool generates test cases with different values for each variable, allowing you to evaluate prompt robustness across many inputs.

Benefits

- **Quality assessment** with 5-point grading
- **Iterative refinement** of prompt design
- **Creation methods**: Auto-generated, manual, CSV import

Usage Workflow

- Create a prompt template with
- Generate test cases (auto or manual)
- Run tests to generate model responses
- Compare outputs across test cases
- Identify patterns and refinement opportunities
- Iterate on prompt design and retest

Available

In the Anthropic Console's Evaluation Tool. Supports up to dozens of test cases per prompt, evaluated using Claude models.

Claude: Prompting Examples

Technical Analysis Example

System: You are an experienced software developer with expertise in code review and security assessment.
Human: <instructions> Review this Python code for security vulnerabilities. Focus on:

- SQL injection risks
- Input validation issues
- Authentication weaknesses </instructions>

<code> def login(username, password): query = f"SELECT * FROM users WHERE username='{username}' AND password='{password}'" result = db.execute(query) return result.rowcount > 0 </code>
Provide your analysis in <thinking> tags first, then summarize findings in <answer> tags with severity ratings.

Content Creation Example

System: You are a marketing copywriter specializing in compelling product descriptions.
Human: <instructions> Write a product description for our new wireless earbuds. </instructions>
<product_details>

- Name: SonicWave Pro
- Battery: 8 hours, 24 more with case
- Features: Noise cancellation, water resistant (IPX5)
- Price point: $129.99
- Target audience: Active professionals, commuters </product_details>

<examples> <example> Meet the UltraBeam X1: Your perfect workout companion. With sweat-proof design and 6-hour battery life, nothing will interrupt your fitness journey. The secure-fit technology ensures they stay put during your most intense workouts. </example> </examples>
Write a description in a similar style to the example, but tailored to our product and target audience.

Gemini: Prompt Engineering Overview

CW JV

Core Components

Prompt Structure

Google recommends structuring prompts with four key elements:

- **Persona**: Who you are or what role you're playing
- **Task**: What you need Gemini to do
- **Context**: Relevant background information
- **Format**: How you want the output structured

Communication Style

Gemini models respond best to natural, conversational language rather than stilted or overly technical prompts.

Gemini Format Example:

```
I'm a data analyst for a marketing team.
My task is to analyze campaign data.

Here is the campaign data:
[data]

Analyze with focus on conversion rates.
Present findings as bullet points.
For complex metrics, use code blocks.
```

Official Resources

- Gemini API Documentation
- AI Studio Prompt Gallery

Gemini: Practical Tips

CW
JV

Google's Recommendations

- Use natural language (speak conversationally)
- Be specific and iterate with follow-ups
- Keep prompts concise but detailed
- Make it a conversation rather than one-shot
- Include specific documents for personalization
- Prompts ~21 words with context perform better

Common Pitfalls

- Relying on models for factual information without verification
- Using models for complex math without verification
- Providing negative examples instead of positive ones
- Inconsistent formatting in few-shot examples
- "Prompt stuffing" with too many examples

Approach Guidelines

Google's research shows that natural, conversational language performs better than rigid or overly formal prompts. Focus on clear, direct instructions with specific guidance on both the task and desired output format. The 21-word guideline is based on internal research showing optimal performance with concise but complete prompts.

Gemini: Example Prompts

Classification Example

Few-shot Classification:

```
Classify the text as one of the following categories:
- large
- small

Text: Rhino
The answer is: large

Text: Mouse
The answer is: small

Text: Elephant
The answer is:
```

Role-Based Example

Professional Task:

```
You are a program manager in healthcare IT.
Draft an executive summary email to hospital
administrators based on these implementation
notes for our new patient portal.

[implementation notes]

Limit to bullet points with key milestones,
challenges, and next steps.
```

Gemini: Multimodal Prompting

Visual Prompting

- Gemini excels at analyzing image content
- Combine images with specific text instructions
- Ask for detailed descriptions or specific analysis
- Supports multiple images in a single context

Common Use Cases

- Analyzing charts and diagrams
- Explaining complex visual concepts
- Converting visual information to structured data
- Troubleshooting from screenshots
- Generating image-inspired content

Image Analysis Example:

```
I'm sharing a screenshot of a data dashboard.

[Image of dashboard]

Please analyze the following:
1. What are the key metrics being tracked?
2. What trends do you observe?
3. Identify any anomalies or concerning patterns
4. What recommendations would you make based
   on this data?
```

Best Practices

- Provide clear instructions about what to focus on
- Ask specific questions rather than general ones
- Use high-quality, clear images when possible
- For multiple images, label or number them
- Consider mixing images with text examples

Prompt Engineering: Resources

OpenAI

Comprehensive guide with six key strategies and examples for GPT models

OpenAI Prompt Engineering

Claude

Detailed guide focusing on XML structure and few-shot prompting

Claude Prompting Guide

Gemini

Guide for Google AI models with multimodal capabilities

Google AI Prompting

Prompt Engineering Guide

Decent overview of multiple prompting strategies

Presentation Summary

LLM Ecosystem Overview

- **Leading providers**: OpenAI, Anthropic, Google, DeepSeek, Mistral, and open models
- **Evaluation criteria**: Context window, capabilities, cost, speed, features
- **Key differentiators**: Cost structure, prompt caching, multimodal support
- **Special features**: Function calling, search integration, JSON mode
- **Selection framework**: Match specific needs to provider strengths

Prompt Engineering

- **OpenAI**: System messages, detailed instructions for GPT-4.1; minimal prompting for O-Series
- **Claude**: XML structure, chain-of-thought reasoning, test case generation
- **Gemini**: Conversational approach with persona, task, context, format structure
- **Common techniques**: Few-shot examples, step-by-step reasoning
- **Advanced tools**: Prompt generators, evaluators, testing frameworks

Key Takeaways

CW
JV

Provider Selection

Choose providers based on specific application requirements, balancing cost, capabilities, and special features. Consider using multiple providers for different aspects of your application.

Prompt Engineering

Tailor prompting techniques to each model family. Use structured approaches (XML for Claude, detailed instructions for GPT-4.1, minimal prompts for O-Series, conversational for Gemini).

Next Steps

Benchmark multiple models for your specific use cases, develop systematic evaluation methods, and stay updated on rapidly evolving capabilities and pricing. Consider cost optimization techniques like prompt caching for production deployment.

Have Fun!