5 LLM Prompting Techniques Every Developer Should Know

Want to make the most out of large language models? Check out these prompting techniques you can start using today.

By **Bala Priya C**, KDnuggets Contributing Editor & Technical Content Specialist on February 12, 2025 in **Language Models**

Image by Author | Created on Canva

Large language models (LLMs) are super popular and you're probably using one or more of them every day as a developer. So knowing how to craft effective prompts is becoming as important as knowing how to write good code.

Let's explore five powerful prompting techniques that can help you get the most out of AI models like Claude AI and ChatGPT.

1. Zero-Shot Prompting

Think of zero-shot prompting as jumping straight into a task without any examples or training. It's like asking a human expert to do something they're trained for — you just describe what you want clearly and specifically.

When to Use

- For straightforward tasks where the desired output format is clear
- When you want quick results and don't need to set up complex examples
- For tasks that are common or well-understood by the model

Here's an example. Instead of just asking "classify this text", you might write:

> Classify the following text as either positive, negative, or neutral. Provide your classification as a single word.
>
> Text: "The new feature works great, but the documentation could be better."

You'd have already used this technique multiple times if you've used an LLM at all.

2. Few-Shot Prompting

Few-shot prompting is like teaching by example. You show the model a few examples of what you want, then ask it to follow the same pattern.

When to Use

- When you need very specific output formatting
- For tasks where consistency is important
- When zero-shot attempts aren't giving desired results

Let's take an example:

> Convert these dates to DD-MM-YYYY format:

10 Little-Known Python Libraries That Will Make You Feel Like a Data Wizard

Using Llama 3.2-Vision Locally: A Step-by-Step Guide

Beginner's Guide to Subqueries in SQL

Python Oddities That Might Surprise You

Building AI Application with Gemini 2.0

How to Fine-Tune DeepSeek-R1 for Your Custom Dataset (Step-by-Step)

Building Multilingual Applications with Hugging Face Transformers: A Beginner's Guide

Using Gemini 2.0 Pro Locally

5 AI Agent Frameworks Compared

Data Science Showdown: Which Tools Will Gain Ground in 2025

> Input: March 15, 2024
> Output: 15-03-2024
>
> Input: December 1, 2023
> Output: 01-12-2023
>
> Input: April 7, 2024
> Output: [Your turn to convert this one]

Now let's explore more interesting prompting techniques.

3. Chain-of-Thought Prompting

This technique encourages the model to break down complex problems into steps, similar to how we might solve a math problem. It's like thinking out loud while solving a problem.

When to Use

- For complex reasoning tasks
- When accuracy is important
- When you want to verify the model's logic
- For debugging or understanding errors

It's time for an example.

> Question: If a store has 120 apples and sells 30% of them on Monday, then 25% of the remaining apples on Tuesday, how many apples are left?
>
> Let's solve this step by step:
>
> 1. Start with 120 apples
> 2. Monday sales = 30% of 120 = 36 apples
> 3. Remaining after Monday = 120 - 36 = 84 apples
> 4. Tuesday sales = 25% of 84 = 21 apples
> 5. Final remaining = 84 - 21 = 63 apples
>
> Therefore, 63 apples are left.
>
> Now solve this question similarly: A library has 240 books. On Monday, 40% of the books are checked out. On Tuesday, 35% of the remaining books are checked out. How many

> books are still available

You can read more about chain-of-thought prompting in Chain-of-Thought Prompting Elicits Reasoning in Large Language Models by Wei et al.

4. Tree-of-Thought Prompting

Tree of Thoughts is an advanced prompting technique that builds on chain-of-thought prompting by exploring multiple reasoning paths simultaneously.

Here's how it works:

1. **Problem Decomposition**. First, you break down a complex problem into smaller steps or decision points.
2. **Generating Multiple Thoughts**. At each step, you generate several possible approaches or "thoughts."
3. **Evaluation and Pruning**. You evaluate each branch and prune less promising paths, keeping only the most promising ones for further exploration. This is where ToT differs most from simple chain-of-thought — you're actively managing multiple solution paths.

When to Use

- For problems with multiple possible approaches
- When you need to compare different solutions
- For creative tasks where exploring alternatives is valuable
- When the best approach isn't immediately obvious

Example 1: A problem-solving ToT prompt:

> Solve this word puzzle by exploring multiple possible paths at each step.
>
> Initial word: BLUE
> Target word: PINK
> Rules: Change one letter at a time, making valid English words.
>
> For each step:
>
> 1. Generate 3 possible valid word transformations
> 2. Evaluate which paths seem most promising for reaching PINK

> 3. Explore the most promising path(s)
> 4. If a path seems blocked, backtrack and try another
>
> Document your thinking process for each attempted path.

Example 2: ToT promting for technical system design.

> Design a system architecture by exploring multiple possible solutions at each component level.
>
> Starting point: High-traffic mobile app with real-time features
>
> Step 1: Data Storage Architecture
>
> Generate 3 approaches:
> - Single monolithic database
> - Microservices with dedicated DBs
> - Hybrid approach
>
> Evaluate each for:
> - Scalability
> - Maintenance complexity
> - Development speed
>
> Select top 2 paths to explore
>
> Step 2: API Layer (for each storage approach)
>
> Propose 3 possible designs:
> - REST with GraphQL
> - gRPC
> - Hybrid solution
>
> Analyze:
> - Performance implications
> - Development complexity
> - Client compatibility
>
> Choose most viable path(s)
>
> Continue this pattern for:
> - Caching strategy

> - Authentication
> - Deployment architecture
>
> Document trade-offs and reasoning for each branch explored.

You'll find ToT particularly effective when:

- The problem has clear intermediate states
- You can meaningfully evaluate partial solutions
- There are multiple possible approaches
- Simple linear reasoning might miss optimal solutions

Read more from Tree of Thoughts: Deliberate Problem Solving with Large Language Models.

5. Role Prompting

This technique involves asking the model to adopt a specific perspective or expertise when responding. It's like asking someone to put on their "expert hat" before tackling a problem.

When to Use

- When you need specialized expertise
- For getting different perspectives on a problem
- When you want to ensure a particular level of technical depth
- For creative problem-solving

Example time:

> Act as a senior engineer reviewing this code for vulnerabilities:
>
> ```
> def process_user_input(input_string):
>     query = "SELECT * FROM users WHERE id = " + input_string
>     execute_query(query)
> ```

Blog
Top Posts
About
Topics
AI
Career Advice
Computer Vision
Data Engineering
Data Science
Language Models

The AI would then analyze the code from a security expert's perspective, likely identifying SQL injection risks and suggesting parameterized queries.

Wrapping Up

role prompting with tree-of-thought exploration.

The key is understanding each technique's strengths and knowing when to apply them.

Remember: good prompting is an iterative process. Don't be afraid to experiment and refine your prompts based on the results you get.

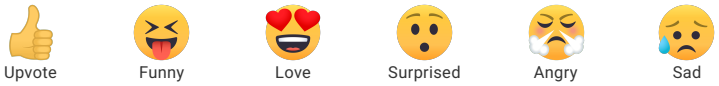
Bala Priya C is a developer and technical writer from India. She likes working at the intersection of math, programming, data science, and content creation. Her areas of interest and expertise include DevOps, data science, and natural language processing. She enjoys reading, writing, coding, and coffee! Currently, she's working on learning and sharing her knowledge with the developer community by authoring tutorials, how-to guides, opinion pieces, and more. Bala also creates engaging resource overviews and coding tutorials.

More On This Topic

- 10 Python Libraries Every Developer Should Know
- 3 Research-Driven Advanced Prompting Techniques for LLM Efficiency...
- The Only Prompting Framework for Every Use
- Enhancing LLM Reasoning: Unveiling Chain of Code Prompting
- 10 Built-In Python Modules Every Data Engineer Should Know
- 10 Python Libraries Every Data Analyst Should Know

What do you think?

1 Response



<= Previous post