Advanced Prompting Strategies

Chain of Thought (CoT) Prompting

- A prompting technique where the model is guided to explain **its reasoning process step-by-step** before giving the final answer — mimicking human logical thinking.
- Encourage step-by-step reasoning for complex problems.

- **Purpose:**
  - Helps the model **reason through complex or multi-step problems**, improving **accuracy, interpretability, and reliability** in reasoning tasks.
- **How it works:**
  - You explicitly ask the model to "think step by step" or use reasoning cues like **"Let's reason this out logically."**
  - The model generates an internal or external reasoning chain before giving the conclusion.
  - In context engineering, this reasoning chain can be **guided, shortened, or hidden** depending on output needs.
- **Examples:**
  - *Example 01:*

    ```
    Solve step by step
    If one biryani plate cost PKR 250, how much will 5 friends pay together if they order 2

    Let's think step by step.
    ```

  - *Example 02:*

    ```
    Solve this step by step:
    If I was 6 when my sister was half my age, how old is my sister when I'm 40?

    Let me think through this step by step:
    ```

  - *Example 03:*

    ```
    A train travels 60 km in 1 hour. How far will it go in 4 hours? Let's think step by ste
    ```

- **When to use:**
  - When tasks involve multi-step reasoning, such as:

        - Math problems
        - Logical reasoning
        - Complex analysis
        - Multi-step processes
        - Code Reasoning
        - Decision Making
        - Not ideal for simple factual recall
- **Best practices:**
    - Use "Let's think step by step" or similar phrases
    - Set temperature to 0 for consistent reasoning
    - Extract final answers separately from reasoning

Self-Consistency (SC) Prompting

- Generate multiple reasoning paths and select the most common answer.
- A sampling-based technique that improves reasoning reliability by **generating multiple independent solution chains** and selecting the most frequent (or most supported) final answer.
- Treats the model like a committee: multiple runs → majority/consensus answer.
- **Process:**
    - Ask the same question multiple times with different phrasings
    - Compare the answers
    - Choose the most frequently occurring result
- **Examples:**
    - *Example 01:*

```
Question: If a store offers a 20% discount on a $50 item, what is the final price?

Generate 3 different reasoning paths for this question and select the most consistent answe

Path 1: To find the final price, calculate the discount: 20% of $50 is 0.20 × 50 = $10. Sub

Path 2: A 20% discount means paying 80% of the original price. So, 80% of $50 is 0.80 × 50

Path 3: Compute the discount amount: 20% = 0.20, so 0.20 × $50 = $10 off. The original pric

Most common answer: $40
```

    - *Example 02:*

```
Question: A student answered 45 out of 50 questions correctly. What percentage did they get
Path 1: Percentage = (45÷50)×100=90%

Path 2:Each question = 100÷50=2%
So 45×2=90%

Path 3:
Difference = missed 5 questions → 5÷50=0.1=10% missed.
So 100−10=90%
```

- *Example 03:*

```
Find the average marks of a student who scored 60, 70, 80 in 3 tests.
Generate 3 reasoning paths and pick the most common result.

Path 1:
Add all marks: 60+70+80= 210
Divide by total tests:210÷3= 70.
✅ Average = 70.

Path 2:
Sum = 60+70+80= 210.
Average formula = total ÷ number of items → 210÷3= 70.
✅ Average = 70.

Path 3:
Middle value check: numbers are evenly spaced (difference of 10),
so the average = middle number = 70.
✅ Average = 70.
```

- **Explanation of Concept:**
  - Self-Consistency involves generating multiple answers to the same question using varied reasoning approaches to ensure reliability.
  - By comparing the results, you select the most frequent or consistent outcome, reducing the chance of errors from a single flawed reasoning path.
  - This method **leverages the model's ability to approach the problem from different angles**, increasing confidence in the final answer when all paths converge, as seen here with the consistent result of $40, 90% and 70 in above 3 examples respectively.

Step-Back Prompting

- Ask a more general question first, then use that context for the specific question.
- A reasoning technique where the model is prompted **to take a step back** and first answer a **broader, more general question** before addressing the specific or **complex task**.
- It helps the model build high-level understanding before diving into detailed reasoning or execution.
- **Purpose:**
  - To improve **contextual reasoning** and **conceptual grounding** for complex tasks.
  - Encourages the model to form a **big-picture understanding**, which leads to more coherent and accurate specific answers.
  - Reduces tunnel vision or narrow reasoning caused by immediately tackling fine details.
- **How It Works:**
  - The prompt first asks a **meta-level or general question** related to the topic.
  - The model responds with foundational principles, context, or high-level reasoning.
  - Then, the user (or next prompt) uses that context to tackle the specific or applied question.
  - The model now uses both — the general reasoning and the task details — to generate a better-informed final response.
- **Examples**
  - *Example 01:*

    ```
    First, what are the key principles of good user interface design?
    [Get response]
    Now, using those principles, redesign this mobile app's login screen: [description]
    ```

  - *Example 02:*

```
Step 1: First, what are the key factors that generally affect website loading speed?

Response: Key factors include:
- Server response time (hosting quality, server location)
- Image and asset optimization (file size, compression)
- Code efficiency (minifying CSS/JavaScript, reducing HTTP requests)
- Caching strategies (browser caching, CDNs)
- Third-party scripts (ads, analytics)

Step 2: Using these factors, provide specific recommendations to optimize the loading s

Response:
1. **Server Optimization**: Choose a reliable hosting provider with servers close to yo
2. **Image Optimization**: Compress product images using formats like WebP. Implement l
3. **Code Efficiency**: Minify CSS and JavaScript files. Combine small files to reduce
4. **Caching**: Use a Content Delivery Network (CDN) like Cloudflare to cache static co
5. **Third-Party Plugins**: Audit and remove unnecessary plugins. Asynchronously load s
```

- *Example 03:*

```
Step 1: What are the main factors that make a restaurant successful?
Step 2: Based on those, give me a plan for launching a biryani restaurant in Karachi.
```

- **Explanation of Concept:**
  - Step-Back Prompting involves first asking a broader, foundational question to establish key principles or context before tackling the specific task.
  - This approach ensures the model grounds its response in general knowledge (e.g., factors affecting website speed) before applying it to the specific problem (e-commerce site optimization).
  - By breaking the task into two steps, the model produces more informed and structured recommendations, reducing the risk of overlooking critical factors.
- **When to Use:**
  - When the task involves **complex judgment, creativity, or abstraction,** such as:
    - Design, strategy, or policy reasoning
    - Writing, analysis, or ethical evaluation
    - Problem-solving that benefits from general frameworks
  - Especially useful when model outputs feel **surface-level** or **context-missing**.
- **Best Practices:**
  - Start with a broad, meta-level question that elicits general principles, context, or frameworks.
  - Follow up with a specific, applied prompt referencing that high-level context (“Using those principles…”).

- Keep general step concise — don't over-elaborate or drift off-topic.
- Works best when combined with CoT or ReAct, for structured reasoning and grounded execution.
- Use when model needs reflection before action (ideal for open-ended or ill-defined problems).
- Avoid overuse on straightforward factual questions — it adds unnecessary complexity.

ReAct(Reasoning + Action or Acting) Prompting

- Combine reasoning with tool use or actions.
- **ReAct (Reasoning + Acting)** is a prompting framework that combines the model's **logical reasoning steps** (thoughts) with **actions** (e.g., using tools, APIs, or queries) in an **interleaved process.**
- It helps the model think, act, and reflect iteratively — just like a human solving a problem step-by-step while interacting with the environment.
- **Purpose:**
  - To enable **interactive reasoning** where the model not only *thinks* but also *takes actions* based on those thoughts.
  - Enhances **tool use, retrieval, planning, and decision-making** by grounding reasoning in real actions and evidence.
  - Prevents hallucination by verifying reasoning through real-world actions.
- **How It Works:**
  - The model alternates between **Reasoning** ("Thought: what should I do next?") and **Acting** ("Action: perform X using Y tool or source").
  - After each action, it observes the result **(Observation: tool output)**.
  - It then updates reasoning based on new information and repeats the loop until reaching a final answer **(Answer: final conclusion).**
  - The prompt format usually guides this reasoning-action-observation cycle explicitly.
- **Examples:**
  - *Example 01:*

Use the ReAct pattern to answer the following:
I need the population of Canada and Australia to compare.

AI's Response (ReAct style):
Thought: I need the population of Canada and Australia to compare.
Action: web_search("current population of Canada 2025")
Observation: [Canada ≈ 39 million]
Thought: Now search for Australia.
Action: web_search("current population of Australia 2025")
Observation: [Australia ≈ 27 million]
Final Answer: Canada has about 12 million more people than Australia.

- *Example 02:*

I need to find out how many children the members of Metallica have in total.

Thought: I need to find information about each band member and their children.
Action: Search for "Metallica band members"
Observation: [results]
Thought: Now I need to search for each member individually...

- *Example 03:*

Question: What is the current population of Tokyo, and how does it compare to New York

Thought: I need to find the current population of Tokyo and New York City to compare th

Action: web_search
<parameter name="query">current population of Tokyo 2025</parameter

Observation: [Assume tool response: According to recent estimates, Tokyo's population i

Thought: Now I have Tokyo's population. Next, I need the population for New York City.

Action: web_search
<parameter name="query">current population of New York City 2025</parameter

Observation: [Assume tool response: New York City's metropolitan area population in 202

Thought: With both figures, I can compare them. Tokyo has a larger population than New

Final Answer: Tokyo's population is approximately 37.4 million, compared to New York Ci

- **Explanation of Concept:**

    - ReAct (Reasoning + Acting) is a prompting strategy that interleaves reasoning steps (Thoughts) with actions (tool calls or external queries) to solve complex problems iteratively.
    - Each cycle includes a `Thought (planning the next step)`, an `Action (executing a tool or search)`, and `an Observation (processing the result)`.
    - This loop continues until the question is resolved, allowing the model to dynamically gather information and refine its approach.
    - It's particularly useful for tasks requiring real-time data or multi-step verification, as demonstrated by sequentially fetching and comparing population data.
- **When to Use:**
    - Tasks that require **external data retrieval, computation, or environment interaction**, such as:
        - Web search, code execution, database query, or document lookup
        - Long reasoning chains where intermediate validation matters
        - Multi-agent workflows or tool-augmented AI systems
- **Best Practices:**
    - Use clear format markers like **Thought → Action → Observation → Answer** to guide structure.
    - Keep reasoning concise — avoid overthinking loops or redundant actions.
    - Ensure **tool or API calls** are valid and consistent with reasoning.
    - Combine with **CoT** for clear thought sequences before each action.
    - In context engineering, manage state — keep previous observations accessible to avoid re-querying.
    - Limit number of actions to prevent infinite loops or unnecessary tool use.
    - Useful in **agentic workflows, retrieval-augmented systems, and autonomous reasoning chains.**

Tree of Thoughts (ToT)

- Explore multiple reasoning branches simultaneously for complex problems.
- **Tree of Thoughts** is an advanced reasoning framework where the model explores multiple reasoning paths (branches) instead of following a single linear chain.
- It treats problem-solving as a **search process** — evaluating, expanding, and selecting the best reasoning branches to reach an optimal conclusion.
- **Purpose:**
    - To improve performance on **complex, multi-step, or creative reasoning tasks** by exploring diverse solution paths.
    - Avoids getting stuck in one reasoning direction (a limitation of standard Chain of Thought).

    - Encourages **structured exploration + evaluation** for higher-quality outcomes.
- **How It Works:**
    - Each **"thought"** represents a reasoning step or partial solution.
    - The model **branches out** by generating multiple next-step thoughts from each node (like a decision tree).
    - Each branch is **evaluated** for quality, consistency, or likelihood of success.
    - The best or most promising paths are **expanded further**, while weak ones are **pruned**.
    - This iterative process continues until the model reaches a **final, validated answer**.
- **Examples:**
    - *Example 01:*

```
Question: What is the best marketing strategy for launching a new eco-friendly clothing
Task: Explore multiple strategic approaches, evaluate them, and select the best one.

Branch 1: Social Media Campaign Thought: Young adults are active on platforms like Inst
    Pros: High engagement, visually appealing for clothing, cost-effective with micro-i
    Cons: Risk of inauthentic partnerships, oversaturation in influencer marketing. Eva
---
Branch 2: Sustainable Pop-Up Events Thought: Hosting pop-up shops at eco-conscious fest
    Pros: Hands-on experience with products, builds community, aligns with eco-friendly
    Cons: High logistical costs, limited geographic reach. Evaluation: Great for brand
---
Branch 3: Collaborative Partnerships Thought: Partnering with eco-friendly brands (e.g.
    Pros: Expands reach via partner networks, reinforces eco-mission.
    Cons: Complex coordination, potential brand dilution. Evaluation: Effective for nic
```

Synthesis: Combine a social media campaign (Branch 1) with selective pop-up events (Branch 2) for maximum impact. Use partnerships (Branch 3) to amplify reach at events.
Final Strategy: Launch with a TikTok influencer campaign showcasing eco-friendly clothing, paired with pop-up shops at green festivals to engage young adults directly. Collaborate with a sustainable accessory brand to co-promote at events.

    - *Example 02:*

```
Plan a 10-day Canada trip on a medium budget.

Generate 3 different travel approaches (Branch A, B, C). For each branch:
- Short description (1 line)
- Key activities (3 bullets)
- Pros and cons (2 each)
- Score out of 10 (based on cost, experience, ease)
Finally, recommend one branch and explain why.
```

- **Explanation of Concept:**
  - Tree of Thoughts (ToT) involves generating multiple reasoning branches to explore different solutions to a problem, evaluating each, and synthesizing the best ideas into a final answer.
  - Each branch represents a distinct approach, which is explored, assessed for pros and cons, and scored.
  - This method is ideal for complex, open-ended tasks like strategic planning, as it encourages creative exploration and systematic comparison, as shown in the marketing strategy example above.
- **When to Use:**
  - For **complex reasoning or creative generation** tasks requiring exploration, such as:
  - Strategic planning, theorem proving, optimization
  - Game decision-making or multi-turn reasoning
  - Creative writing, story branching, scenario analysis
  - Complex decision-making, tasks requiring exploration of alternatives
  - Any task where a single CoT may not be sufficient
- **Best Practices:**
  - Define **evaluation criteria** for pruning (e.g., logic soundness, goal alignment, factuality).
  - Limit **tree depth** and **branching factor** to control compute cost.
  - Combine with **Self-Consistency or Reflection** to refine and validate chosen branches.
  - Use **CoT** as the reasoning foundation — ToT just expands it into a structured search.
  - Represent reasoning steps clearly (e.g., “Thought 1A → Thought 1B → Final Answer”).
  - Maintain memory or context to keep track of explored paths in multi-turn systems.