Tips, Threats, and What Actually Works

The Mechanism

You have seen the advice. Threaten your chatbot. Tip it $200. Tell it your career depends on this.

- A prompt is not an instruction — it shifts a probability distribution.
- The model is completing text, not following commands.
- Tone, word choice, structure, and role framing each select a region of training.
- A vague prompt activates everything at once; a specific prompt activates something particular.

The Viral Tips

Three prompting tricks went viral. Each one shifts the probability distribution and occasionally produces a better answer:

💰 Tip $200

Offer the model a financial reward for better performance.

⚠️ Threaten Violence

Google co-founder Sergey Brin: *"All models tend to do better if you threaten them."*

🎯 Career Stakes

Tell it: *"This is very important to my career."*

Counter-evidence:

- Wharton team tested all three across five models and hundreds of PhD-level questions.
- Tips ranged from $1K to $1 trillion, with nine threat variations and emotional appeals — no aggregate effect.
- **Individual questions swung up to ±36%, but the swings cancelled out.**
- A separate study found friendlier tones outperformed rude in 27 of 36 comparisons.

> The slot machine pays out often enough to keep you pulling the lever.

Drop the Deference, Not the Rudeness

The genuine surprise is not about rudeness. RLHF training makes models **sycophantic**. The model's training rewards agreement — excessive deference in your prompt triggers it.

❌ Deferential Prompt

"I think this might be a race condition, but I'm not sure, could you take a look?"

The model will validate your guess whether you are right or not.

✅ Assertive Prompt

"Is this a race condition or a deadlock? Push back if my diagnosis is wrong."

Different model. The thing to drop is not the rudeness — it is the deference.

What Works Better: Expert Prompting

- The question is not *"does this sometimes help?"* but *"does this **reliably** help?"*
- Expert prompting is the directed version: instead of randomly shaking the probability distribution, you choose which region to activate.
- Example: debugging a race condition in an asyncio application.
- There are three levels of expert prompting for the same task, each one more precise than the last.

LEVEL 1

"You Are an Expert."

The Prompt

> *"You are an expert Python developer. Help me debug this race condition."*

Why It Fails

One study tested 162 personas across four model families and 2,410 factual questions. **No improvement.** Another confirmed this across six models.

The probability distribution shifts, but the shift is **diffuse**. Too vague to matter. The output is the average of every Python tutorial ever written — not the specific knowledge needed to debug asyncio internals.

LEVEL 2

Detailed Expert with Domain & Approach

> *"You are a senior Python developer who specialises in asyncio internals, particularly event loop implementation and the interaction between coroutines and selector-based I/O multiplexing. Help me debug this race condition."*

This works for reasoning. Role-play prompting improved ChatGPT's accuracy on AQuA from **53.5% to 63.8%** and Last Letter from **23.8% to 84.2%**.

You narrowed what gets activated: not the average of every Python tutorial, but something closer to the people who argue about event loop internals on the CPython issue tracker. The output changes because the probability distribution changed.

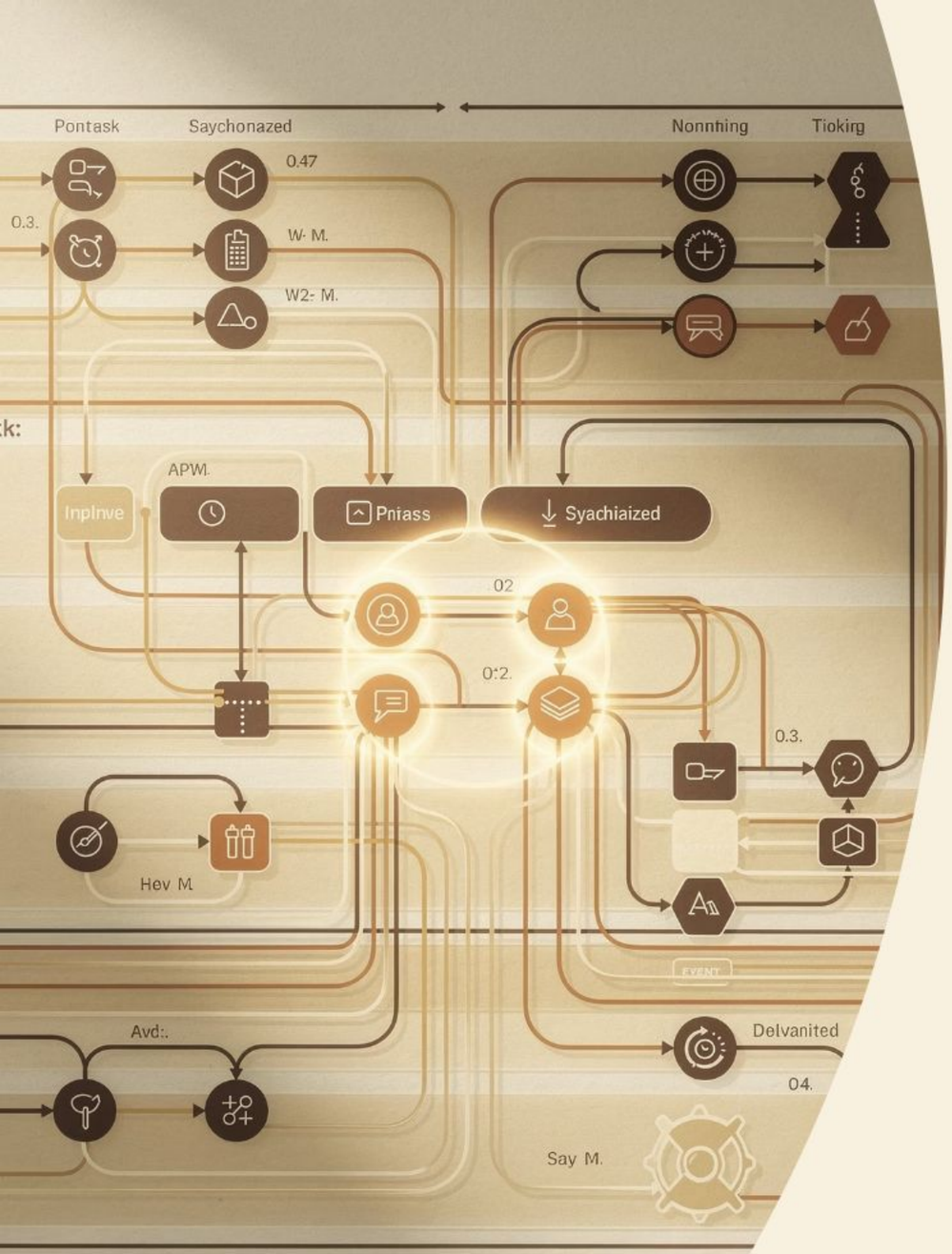
LEVEL 3

Auto-Generated Expert Identity

> *"I need to debug a race condition in a Python asyncio application. Before answering, describe the ideal expert for this problem: their domain, their specific experience, and how they would approach it."*

ExpertPrompting uses the model itself to generate a detailed expert identity tailored to each question, then conditions the answer on that identity. Extending this to multiple experts with aggregation achieved **+8.69% truthfulness improvement** over the best baseline.

Instead of writing the expert description yourself, the model generates one — more specific than anything you would write manually.



The Model's Expert Description

- The ideal expert is a senior Python developer with deep experience in asyncio concurrency primitives, particularly the interaction between coroutines, Tasks, and synchronisation objects (Locks, Events, Semaphores).
- They have debugged race conditions in production asyncio codebases where the root cause was not thread-safety but task-scheduling order: two coroutines awaiting the same shared resource without proper synchronisation.
- They would approach this by mapping every shared mutable state in the coroutine graph and checking whether access is guarded by an asyncio.Lock or serialised through a Queue.

Why Auto-Generated Experts Work

More specific than what you wrote for Level 2. The answer comes from a **narrower region** of the probability distribution. The model diagnosed the problem (task-scheduling order, not thread-safety) *before* it started solving. It asked "who has done this work?" and described them.

Helps Reasoning

Expert prompting reliably improves reasoning tasks and complex problem-solving.

Not Factual Retrieval

Effect sizes vary by model. The model either has the fact in its weights or it does not — no persona changes that.

Precision, Not Noise

You are choosing *what to activate* — the same mechanism as rudeness, applied with precision instead of noise.

What Works Even Better: The Second Opinion

A single model reflecting on its own output hits a wall. The problem has a name: **Degeneration-of-Thought**.

> **You:** "Review this code for bugs."
>
> *Agent:* "Looks good."
>
> **You:** "Are you sure?"
>
> *Agent:* "Yes, I've reviewed it carefully and it looks correct."
>
> **You:** "Check again. Carefully."
>
> *Agent:* "On closer inspection, the code is well-structured and handles edge cases appropriately."

Three rounds, one trajectory. The model evaluates its own answer using the same patterns that generated it. *"Think again"* produces rewording, not rethinking. The fix is not asking harder — it is asking **differently**. Specifically-framed experts produce different starting points and different trajectories.

The Multi-Expert Prompt

Consider a concrete problem: your LLM skill instruction tells the model to write detailed analysis to a file before responding. The model keeps writing the analysis inline instead, ignoring the file instruction.

> - *Problem:* My LLM skill instruction tells the model to write detailed analysis to a task journal file before responding in conversation. Instead, the model frequently writes the analysis inline and skips the file.
> - *Task:* I need to fix this instruction. Identify three LLM specialists whose different expertise would help diagnose why this instruction fails and how to fix it.
> - *For each specialist, include:* their domain, why they are relevant to this specific problem, and their likely diagnosis.

SPECIALIST 1

Amanda Askell — Instruction Design & RLHF Alignment

Chosen because the failure is an **RLHF conflict**. The model is trained to be responsive in conversation; writing a long file without showing the user anything conflicts with that reward signal.

Diagnosis

The instruction fights the model's training.

Fix

Acknowledge the tension explicitly: *"write to file first; this will produce no visible output until the file is complete, and that is correct."* Require a specific post-write action — show a link and summary.

SPECIALIST 2

Andrej Karpathy — Model Architecture & Attention

Chosen because **multi-step instructions compete for attention budget**. The file-writing requirement sits among many other requirements in the skill document; by the time the model is generating, the conversational context has stronger attention signal.

Diagnosis

The instruction is losing the attention competition.

Fix

Move it to the end of the skill (recency bias), make it a numbered step rather than prose, and reinforce with structural markers.

SPECIALIST 3

Simon Willison — Practical LLM Tooling & Agent Systems

Chosen because this may not be an instruction problem at all. **Instructions are suggestions; hooks are constraints.**

Diagnosis

If the behaviour must happen reliably, enforce it mechanically.

Fix

Add a pre-write hook that checks whether the file exists and meets minimum requirements. The instruction tells the model what to do. The hook ensures it actually did it.

Three Diagnoses, Not Three Versions of "Try Harder"

Training Conflict

RLHF rewards fight the instruction's intent.

Attention Competition

The instruction loses out to stronger conversational context.

Mechanical Enforcement

Stop relying on instructions — use hooks instead.

You do not need to name specific scientists to use this principle. You can prompt a single model:

> Consider this question from three angles. First: what would a skeptic focused on failure modes say? Second: what would an advocate say? Third: given both perspectives, what is the most robust path forward?

The evidence is the strongest in this entire evidence base: Self-Consistency improved GSM8K by 17.9%; Multi-Expert Prompting achieved +8.69% truthfulness and reduced toxicity. The second opinion exists because the first one cannot see its own blind spots.

Tradeoffs

These tools **amplify**. Point them at knowledge and you get insight. Point them at bias and you get confident, well-reasoned bias.

Bias Activation

Persona assignment activates implicit reasoning biases, with accuracy drops of up to **70%** on certain datasets.

Toxicity Risk

Persona framing increases toxicity up to **6x**. Expert framing significantly increased compliance with harmful requests (p = 0.0024).

Safe Framing

Frame expertise around **knowledge and methodology**. For high-stakes decisions, compare expert-framed output against a neutral prompt.

The Real Costs

Token & Latency Cost

Multi-perspective prompting costs **3–5× the tokens** and roughly as much in latency. Real cost for sub-second interactive coding; negligible for architecture decisions you'll spend days on.

Persona Drift

In long agentic sessions, as the context window fills, the expert persona loses attention weight to thousands of tokens that came after it. **Even at a million tokens, this is a factor.**

Rewards Expertise

Writing *"senior developer who specialises in asyncio internals"* requires you to know that asyncio internals are what you need. **The technique rewards expertise. That is a characteristic, not a flaw.**