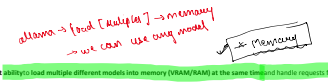
Advanced Ollama Techniques

22 October 2025 13:47

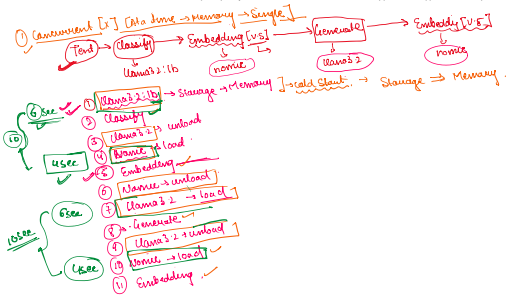
Concurrent Model Serving

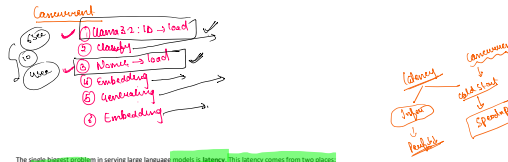


Concurrent Model Serving capability of Ollama gives it ability to **load multiple different models into memory (VRAM/RAM) at the same time** and handle requests for any of them.

This architecture allows the server to receive and process inference requests for *any* of these loaded models without incurring the "cold start" latency of loading and unloading them from disk for each request.

A server that is *not* concurrent would face this "cold start" penalty **every time** it needed to switch tasks. This makes any professional application unusably slow.





The single biggest problem in serving large language models is latency. This latency comes from two places:

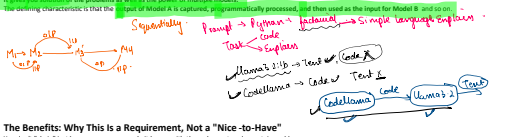
1. **Inference Latency:** The time it takes for the model to "think" and generate a response. This is unavoidable.
2. **Cold Start Latency:** The time it takes to load **the model file from your disk** (slow storage) **into your GPU's VRAM** (fast memory). For a 5GB model, this can take 5-10 seconds; for a 50GB model, it can take over a minute.

Model Chaining

Model Chaining (also known as "Prompt Chaining" or "LLM Workflows") is a software development pattern where you sequentially execute multiple, distinct model calls to solve a problem.

It gives you solution of the problems as well as the power of multiple models.

The driving characteristic is that the output of Model A is captured, programmatically processed, and then used as the input for Model B and so on.



The Benefits: Why This Is a Requirement, Not a "Nice-to-Have"

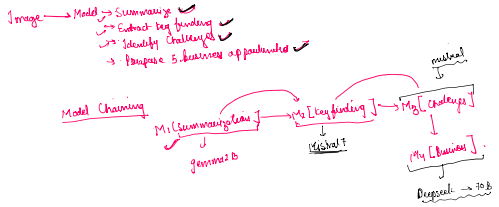
You don't "chain" just because you can; you do it because it's the *only* way to solve certain problems.

Task Specialization = Higher Quality

Model chaining enables each model to focus on its area of expertise, leading to higher-quality results. **Instead of forcing one model to perform multiple, conflicting tasks, each model in the chain handles a specific part of the workflow it excels at.** This avoids performance trade-offs and ensures that every step — whether technical, analytical, or creative — is completed with maximum precision and depth. The result is a combined output that's more accurate, coherent, and polished than what any single model could produce alone.

Solves Problem of Instruction Dilution:

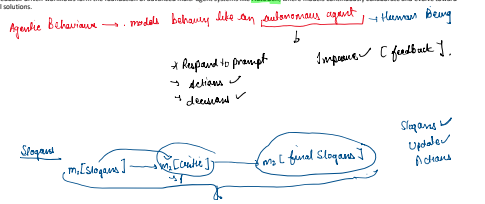
- LLMs struggle with "instruction dilution." In a 10-step prompt, the model's focus on step 10 is weaker than its focus on step 1.
- Chaining allows each model to focus on **one simple task**. This "resets" the model's attention and context for each step, ensuring each part is done with 100% focus. A chain of simple, high-success-rate steps is far more reliable than one complex, low-success-rate step.



Enabling Agentic Behaviour (The "Critic" Pattern)

Model chaining enables agentic behaviour, where the system acts more like an autonomous agent than a passive responder. Instead of producing a single static output, models can engage in reflection and self-improvement loops. In this pattern — often called the "Critic" loop — one model generates a response while another evaluates, critiques, and suggests improvements. The original model then revises its output based on this feedback. This iterative process mimics human-style reasoning and review, leading to more refined, accurate, and context-aware results.

Such feedback-driven workflows form the foundation of advanced multi-agent systems like AutoGen, where models continuously collaborate and evolve toward optimal solutions.



Cost Optimization through Selective Model Usage

Cost Optimization through Selective Model Usage involves strategically leveraging models of varying capabilities throughout a workflow. For simpler or routine tasks—such as initial data extraction, basic classification, or filtering—you can use lighter, faster, and more cost-effective models. The more powerful, resource-intensive models are then reserved for critical stages that demand deeper reasoning, nuanced understanding, or creative problem-solving. This approach not only reduces overall computational costs but also ensures that high-end resources are allocated efficiently, maximizing both speed and performance across the workflow.

→ Intensity → Model [Small , Medium , Big].

Improved Debugging and Traceability

Chained workflows significantly enhance debugging and traceability. When an unexpected result occurs, it becomes much easier to pinpoint the source of the problem. By examining the inputs and outputs of each individual step in the workflow, you can identify exactly where an error or misstep happened. This is far more efficient than attempting to interpret the internal state of a single, large model call, allowing for faster troubleshooting and more precise corrections.