Context Layers for LLMs

Why Architecting the Right Information Environment Is the Biggest Lever for AI Performance

By Ash | IP Network Solutions Inc.

February 2026

The Shift from Prompt Engineering to Context Engineering

For the past two years, the AI industry has treated prompt engineering as the primary skill for getting reliable outputs from large language models. Craft the right instruction, add the right constraints, and the model should perform. But 2025 revealed a hard truth: *prompt optimization hits a ceiling fast.* The real differentiator for production-grade AI systems is not what you ask the model to do—it's what information the model has access to when it processes your request.

This discipline now has a name: **context engineering**—a term that gained formal traction in mid-2025 when Shopify CEO Tobi Lütke and former OpenAI researcher Andrej Karpathy both publicly endorsed the concept. Karpathy defined it as "the delicate art and science of filling the context window with just the right information for the next step." Within a month, the first comprehensive academic survey analyzing over 1,300 papers formalized it as a distinct discipline.

According to Anthropic's engineering team, context engineering refers to "the set of strategies for curating and maintaining the optimal set of tokens during LLM inference." This goes well beyond tweaking a system prompt. It encompasses the entire information architecture surrounding a model: how data is retrieved, structured, prioritized, compressed, and delivered at inference time.

For organizations deploying AI agents in production—especially in high-stakes domains like federal government contracting, compliance, and enterprise operations—understanding context layers is not optional. It is the single most impactful investment you can make to improve LLM reliability.

The Problem: Why Raw Context Fails

Modern LLMs advertise massive context windows—128K, 200K, even 1M+ tokens. The assumption is that more input capacity equals better output. Research published in 2025, however, tells a starkly different story.

Context Rot

The Chroma Research team evaluated 18 leading LLMs and found that models do not maintain consistent performance as input length grows. Even on simple tasks like text replication, performance becomes increasingly unreliable with longer inputs. They coined this degradation “context rot”—the phenomenon where output quality degrades as the context window fills up. The key finding: **what matters is not whether relevant information is present, but how that information is structured and presented.**

Maximum Effective Context Window

A September 2025 study from arXiv defined the concept of the “Maximum Effective Context Window” (MECW) versus the advertised Maximum Context Window (MCW). After collecting hundreds of thousands of data points, the researchers found that some top models failed with as few as 100 tokens in context, and most showed severe accuracy degradation by 1,000 tokens. The gap between what models claim they can handle and what they actually use effectively was as large as 99%.

Three Root Causes of Context Failure

Research on hallucination attribution has identified three primary mechanisms by which poor context management causes LLM failures:

- **Knowledge Conflicts:** When new context contradicts the model’s pre-trained knowledge, the model defaults to its training data, producing outdated or incorrect responses.
- **Insufficient Context:** When the model lacks adequate information, it fills gaps with plausible-sounding fabrications rather than acknowledging uncertainty.
- **Instruction Overload:** Stacking too many simultaneous constraints dramatically degrades performance. The more rules loaded at once, the higher the cognitive load, and the more critical instructions get dropped.

The Five-Layer Context Architecture

The emerging consensus among AI practitioners and enterprise architects converges around a five-layer model for structuring context. This architecture, articulated by Fractal Analytics and independently validated by multiple engineering teams, separates context into distinct, manageable layers that each serve a specific function.

Layer A — Identity (System Instructions)

This is the foundational layer: who the model is, what role it plays, and how it should reason. It includes system prompts, persona definitions, tone guidelines, and reasoning-first instructions that force the model to validate assumptions before producing output. Fractal’s research shows that initializing agents with reasoning-first system instructions dramatically reduces hallucinations and improves consistency in regulated workflows.

This layer also benefits from continuous prompt refinement—automatically evolving instructions based on real conversation failures rather than relying on static, hand-tuned prompts.

Layer B — Knowledge (Retrieved Context)

This layer provides the model with domain-specific knowledge it does not possess from training. It encompasses Retrieval-Augmented Generation (RAG), vector database lookups, document retrieval, and structured knowledge bases. The quality of this layer directly determines factual accuracy. Research from Berkeley's AI Lab found that well-implemented RAG systems reduce hallucination rates by 52% compared to approaches that rely solely on in-context information. For enterprise applications, this layer connects the model to proprietary data, regulatory frameworks, and organizational knowledge that no general-purpose model could possess.

Layer C — State (Dynamic Session Context)

This layer captures what is happening right now—the user's current request, environment, and evolving preferences. Unlike static knowledge, state context changes with every interaction. It includes the current task parameters, user metadata, and any real-time signals that affect how the model should respond. The critical design decision here is selective injection: each request may require different information, so providing everything simultaneously slows the system and introduces noise that can confuse the model.

Layer D — Memory (Short-Term and Long-Term)

Memory management bridges the gap between a single interaction and a persistent relationship. Short-term memory summarizes recent interactions to maintain conversational continuity. Long-term memory retains persistent user data, goals, and behavioral patterns across sessions. The challenge is compression: by prioritizing essential details and summarizing or discarding redundant information, the system balances continuity with efficiency. Without this layer, every interaction starts from zero, forcing users to repeat context and degrading the perceived intelligence of the system.

Layer E — Tools and Actions

The final layer transforms the LLM from a passive text generator into an active problem-solver. When the model needs information it does not have, it triggers external tools: APIs, databases, code execution, or specialized services. The critical insight is that tool outputs should not be injected raw into the prompt. Instead, they should be distilled into concise semantic conclusions that preserve reasoning quality while enabling reliable action. This is especially important with the rise of MCP (Model Context Protocol), which became the standard for tool and data access in agent-style LLM systems in 2025.

Quantified Impact: How Much Do Context Layers Improve Performance?

The shift from ad-hoc prompting to structured context engineering is not just a theoretical improvement. Multiple studies and production deployments have produced measurable data on the performance gains.

Metric	Improvement	Source
Hallucination Reduction	30–52% decrease	Berkeley AI Lab; ACL 2023 proceedings
Task Completion Rate	47% improvement	ACL 2023 research on context strategies
Response Accuracy	19% improvement with structured context	Google Research
Response Latency	12% reduction	Google Research
Agent Failure Rate	93% reduction	Dextra Labs enterprise deployments
AI Agent Task Performance	+10.6% (ACE framework)	Zhang et al., 2025
Specialized Domain Accuracy	+8.6% gain (finance)	Zhang et al., ACE paper
Token Cost Efficiency	83.6% lower costs	ACE framework vs. baselines
API Cost Savings	40–60% reduction	Enterprise context engineering deployments
Output Quality	40% higher quality outputs	Organizations with context architectures
AI Analytics Accuracy	20% → 94% with full context	Promethium; five-level context architecture
Data Error Reduction	66% fewer errors	Semantic layer implementations

The most striking data point comes from AI analytics deployments: organizations that systematically capture and unify all five levels of context architecture report accuracy improvements from roughly 20% to 94–99%. The difference is not the model—it's the context surrounding it.

LangChain's 2025 State of Agent Engineering report provides additional industry-wide perspective: while 57% of organizations now have AI agents in production, 32% cite quality as the top barrier to further deployment. Most of those quality failures trace back not to model capability, but to poor context management.

Practical Implications for AI System Builders

Context Engineering Is Infrastructure, Not Prompting

The Amplitude engineering team reported that after transitioning from prompt optimization to context architecture, their system became fundamentally easier to evolve. When introducing a new tool, they no longer rewrite prompts from scratch. They define what the tool does, decide where it belongs in the workflow, and update evaluations to reflect the new capability. When something breaks, they debug by inspecting context and steps rather than guessing which phrase in the prompt stopped working.

Evaluation-Driven Development Is Non-Negotiable

Every practitioner and researcher in this space converges on one principle: without rigorous evaluation pipelines, context engineering is guesswork. Production systems require automated testing where historical conversations are replayed as test cases, entity and constraint retention is audited across long-running workflows, and tool calls are traced end-to-end for procedural correctness. Evaluation should be a core runtime capability, not an offline exercise.

Compression and Selection Matter More Than Window Size

The research on context rot validates what builders have learned the hard way: dumping everything into a massive context window is counterproductive. Techniques like Infinite Retrieval demonstrate that retaining just 4.5–8.7% of input tokens—the right tokens—can produce performance improvements of up to 288% on multi-document reasoning tasks. The paradigm is shifting from “give the model more” to “give the model smarter.”

Self-Improving Context Is the Frontier

The most advanced approaches are moving toward context that evolves autonomously. Agentic Context Engineering (ACE), published in 2025, introduces systems where context improves itself through iterative reflection. The LLM analyzes its own outputs, generates insights about what worked, and curates those insights into an evolving playbook. Early results show a 10.6% improvement on agent tasks and 83.6% lower token costs compared to static approaches—pointing toward a future where context layers are not just designed once but continuously refined.

Conclusion: Your LLM Is Only as Good as Its Context

The data is clear. The difference between an LLM that produces generic, unreliable outputs and one that delivers consistently accurate, actionable results is not the model itself—it’s the context architecture surrounding it. Organizations investing in structured context layers are seeing 40–60% cost reductions, 30–52% fewer hallucinations, and accuracy improvements that can reach from 20% to 94% when all five layers are properly implemented.

For any team building AI systems—whether for government proposals, enterprise automation, or customer-facing agents—the message is the same: stop optimizing prompts in isolation and start engineering context as infrastructure. The prompt is what you say to the model. The context is what the model knows when it listens.

As Fractal Analytics put it succinctly: ***"Your LLM is what it eats."***

Sources and Further Reading

- Chroma Research — "Context Rot: How Increasing Input Tokens Impacts LLM Performance" (2025)
- Paulsen, N. — "Context Is What You Need: The Maximum Effective Context Window" arXiv:2509.21361 (2025)
- Zhang et al. — "Agentic Context Engineering: Evolving Contexts for Self-Improving Language Models" (2025)
- Fractal Analytics — "Context Engineering for LLMs: The Five-Layer Architecture Guide" (2025)
- Amplitude Engineering Blog — "Why Context Engineering Matters More Than Prompt Engineering" (2025)
- NVIDIA Technical Blog — "Reimagining LLM Memory: TTT-E2E" (2026)
- LangChain — "2025 State of Agent Engineering Report"
- Dextra Labs — "How Context Engineering Is the New Prompt Engineering" (2025)
- Promethium — "Context Architecture for AI Analytics: The Five Levels" (2025)
- MDPI — "Context and Layers in Harmony: A Unified Strategy for Mitigating LLM Hallucinations" (2025)
- Karpathy, A. — "2025 LLM Year in Review"
- Prompt Engineering Guide — "Context Engineering Guide" (2025)