# Context as Material: A Comprehensive Analysis of Pragmatic Context Engineering in LLM-Assisted Systems

## 1. Introduction: The Material Turn in Artificial Intelligence

The integration of Large Language Models (LLMs) into the fabric of enterprise software, creative workflows, and decision-support systems has precipitated a fundamental epistemological and architectural shift. This shift is characterized by the transition from viewing "context" as a passive container of metadata—a user ID, a session timestamp, or a static system prompt—to treating **Context as Material**: a plastic, dynamic, finite, and economically tangible substance that must be engineered, sculpted, and curated with the same rigor applied to physical engineering disciplines. This report interrogates this paradigm shift, tracing its lineage from the "thick description" of anthropological theory to the "Agentic Context Engineering" (ACE) of contemporary AI architecture.

In the nascent stages of the generative AI era, the focus was predominantly on model parameterization—the "Software 2.0" vision articulated by Andrej Karpathy, where the neural network's weights represented the compiled code.1 However, as models have standardized and their parametric knowledge frozen at training time, the locus of differentiation and efficacy has migrated to the context window. It is within this finite attention span that intelligence is now situated. The context window has become the *field of action* where raw stochastic capability is transmuted into reliable, culturally aligned agentic behavior.

This analysis establishes that the challenges plaguing modern AI development—phenomena such as "context rot," "lost-in-the-middle" degradation, and "brevity bias"—are not merely technical bugs but inherent properties of this new digital material. Just as a civil engineer must understand the tensile strength of steel or the porosity of concrete, the AI engineer must now master the "physics" of the context window: its attention budgets, its decay rates, and its caching economics.

By synthesizing insights from sociotechnical studies, cognitive science, and advanced systems engineering, we argue that effective AI orchestration requires a "material-semiotic" approach. This perspective, grounded in the work of Bruno Latour, Donna Haraway, and Clifford Geertz, reveals that meaning (semiotics) in AI is inextricably bound to the material constraints of the system—the cost of tokens, the latency of retrieval, and the format of the data. The discipline of Context Engineering has thus emerged as the critical bridge between the abstract potential of the model and the "thick," situated reality of the human user.

## 2. Anthropological Foundations: From Ethnography to Engineering

To engineer context effectively, one must first possess a theoretical framework for what context *is*. In classical computing, context was binary and state-based. In generative AI, context is semantic and interpretative. This necessitates a return to anthropological frameworks that have long grappled with the complexity of meaning, interaction, and materiality. The evolution of AI architecture is, in many ways, the operationalization of twentieth-century social theory.

### 2.1 Thick Description and the Crisis of Thin Outputs

The foundational metaphor for modern context engineering draws directly from the work of symbolic anthropologist Clifford Geertz and his concept of **"thick description."** In his seminal analysis, Geertz distinguished between a "thin description" of an event—such as a rapid contraction of the right eyelid—and a "thick description" that interprets that contraction as a *wink*, a conspiratorial signal, a mockery, or a nervous twitch.3 The physical mechanics (the thin description) are identical in all cases; it is the *context*—the complex web of public codes and social relations—that provides the meaning.

In the domain of LLMs, this distinction is critical for understanding the failure modes of early conversational agents. A model operating on "thin" context processes tokens based on statistical likelihood without anchoring them in the deeper cultural or situational intent of the user. It sees the eyelid contract but cannot discern the conspiracy. Recent research into **"Thick Outputs"** argues that for AI systems to achieve true cultural alignment, they must not merely translate language (a thin operation) but interpret and generate the implicit social signals that accompany communication.3 This requires the engineering of a context window that contains not just facts, but the "webs of significance" that Geertz described.

The operational challenge is that current evaluation benchmarks for LLMs often rely on "thin" metrics—accuracy scores, perplexity, or binary success/failure rates—which fail to capture whether a model has successfully navigated the "thick" cultural nuances of a prompt.5 A model might correctly translate a sentence grammatically (thin success) while failing to convey the appropriate level of deference required by the social hierarchy of the target culture (thick failure).

"Context as Material" implies that the context provided to the model must be dense enough to support this thickness. Just as a sculptor requires a medium with sufficient density to hold a complex form, an LLM requires a context rich in "connotative meanings," "behavioral norms," and "situational appropriateness" to generate a "thick output" rather than a superficial hallucination.4 This has led to the development of "Thick Evaluation" frameworks, such as the CURE benchmark, which assess models not on factual recall but on their ability to provide free-form justifications that reflect embedded norms and symbolic meanings.5

### 2.2 Situated Action: The Rejection of the Cartesian Mind

The shift to Context Engineering also mirrors the transition in cognitive science from symbolic processing to **Situated Action**, a theory articulated by Lucy Suchman and Jean Lave. Situated cognition (or "SitCog") posits that intelligence is not an abstract process occurring inside a "brain" (or a model's weights) but is an emergent property of the interaction between an agent and its environment.6

In traditional "Software 1.0" paradigms, logic is encapsulated and execution is linear. In the "Software 2.0" paradigm of AI, this theory is operationalized by acknowledging that a model's "intelligence" fluctuates wildly based on its immediate context. A model does not "know" things in a vacuum; it "knows" them only when they are situated within the active context window. This aligns with the perspective that "knowing, learning, and cognition are social constructions, expressed in actions of people interacting within communities".6

For AI agents, the "environment" is the text present in the context window. If the context is barren, the agent's cognition is impoverished. This realization has driven the development of frameworks like **SITUATEDTHINKER**, which explicitly ground LLM reasoning in external environments through "situated actions"—querying APIs, receiving feedback, and incorporating that new information into the "thinking process".7 Here, the context window becomes the *field of action*, a dynamic space where the agent creates its own cognition by pulling in external "material" (data) to solve problems that its internal weights (parametric memory) cannot handle alone.

This situativity is not merely spatial but temporal and relational. The "meaning" of a user's request is contingent on the history of the interaction (temporal context) and the specific tools available to the agent (relational context). Context engineering, therefore, is the practice of **"contextualisation"**—placing the agent in a constructed situation that affords the correct behaviors.8 It is the deliberate design of the agent's momentary reality.

### 2.3 Actor-Network Theory and the Material-Semiotic Turn

Perhaps the most radical theoretical contribution to context engineering comes from **Actor-Network Theory (ANT)**, pioneered by Bruno Latour, and the **Material-Semiotic** analysis of Donna Haraway and Karen Barad. These frameworks reject the dualism between "human" and "non-human," "social" and "technical," or "meaning" (semiotic) and "matter".9

In an ANT analysis, an LLM is an "actant"—an entity that modifies a state of affairs—embedded in a heterogeneous network of other actants: developers, users, prompt protocols, vector databases, GPU clusters, and regulatory frameworks. The "context" is not just the text in the window; it is the *network* of these associations. When we engineer context, we are not just writing text; we are stabilizing a network of relations that allows the AI to act.11

**Material-Semiotics** further refines this by insisting that "meaning-making" is always a material process. The "agential cut" (Barad) defines the boundary between the observer and the observed, or in our case, the prompt and the completion.13 In practical engineering terms, this manifests in the realization that the *format* of the context (JSON vs. XML vs. Markdown) is not a neutral stylistic choice but a material constraint that fundamentally alters the "semiotic flows" of the model.15 The token—the fundamental atom of LLM processing—is a material-semiotic unit; it costs energy to process, occupies memory (KV cache), and carries semantic weight.

This perspective reveals why "Context as Material" is more than a metaphor. The context window has physical limits (memory bandwidth, attention quadratic scaling) and physical costs (latency, energy). Engineering this context is thus a practice of resource management and "entropy reduction"—transforming the high-entropy messiness of the real world into a low-entropy, structured representation (the "material") that the machine can act upon.16

The implications of this for organizational theory are profound. As LLMs become integrated into knowledge work, they act as "Haraway-ian monsters"—hybrid entities that blur the lines between storage, retrieval, and inference.17 They challenge the passive conception of technology as a mere container for human knowledge, instead participating actively in the generation of organizational knowing through the material entanglement of prompts and weights.

### 2.4 Cultural Fractals and the Impossibility of Alignment

A critical insight from the anthropological critique of AI is the recognition of culture as "fractal." Culture does not exist as a monolithic block that can be aligned with once and for all; it is recursive and self-similar at different scales, from the civilization to the individual.4 This fractal nature implies that **"Cultural Alignment"** is not a static property of a model but a dynamic state achieved through context engineering.

Even a model with a perfect internal representation of a culture (parametric knowledge) cannot guarantee alignment in a specific interaction without a context that anchors that knowledge. The "Context as Material" approach treats the prompt as the mechanism for navigating this fractal geometry. By providing "thick" context—specific cues about role, tone, and intent—the engineer narrows the vast cultural possibility space of the model down to a specific, appropriate slice.3 This redefines the goal of AI safety from "aligning the model" to "aligning the context."

## 3. The Physics of Context: Engineering Constraints and Phenomena

Treating context as a material requires understanding its physical properties. Unlike the infinite canvas of the human imagination, the LLM context window is a constrained resource subject to specific phenomena that degrade performance if mishandled. The discipline of Context Engineering has moved beyond "prompt crafting" to a rigorous management of these constraints, treating the context window as a prime real estate where every token must pay "rent" in signal value.

### 3.1 The Material Limits: Attention Budgets and Context Rot

Despite the marketing of "infinite" context windows (1M+ tokens), empirical research demonstrates that context is a finite resource with diminishing marginal returns. This phenomenon is termed **"Context Rot"**.18 As the volume of information in the window increases, the model's ability to accurately retrieve and reason about specific details degrades—a decay in the "material integrity" of the context.

Context Rot suggests that LLMs have an **"Attention Budget."** Every token introduced into the window consumes a fraction of this budget. When the window is flooded with irrelevant data ("context pollution"), the signal-to-noise ratio drops, leading to hallucinations or a reversion to the model's training priors (parametric memory) rather than the provided context.20 This degradation is often non-linear; a small amount of "pollution" can disproportionately affect reasoning capabilities if it introduces conflicting or distracting priors.

The material constraints are rooted in the Transformer architecture itself. The self-attention mechanism computes pairwise relationships between tokens, scaling quadratically ($O(n^2)$) in terms of compute and memory for canonical implementations (though linear attention mechanisms exist, they often trade off recall). This physical cost translates directly into "attention scarcity." The model cannot attend to everything with equal intensity.

### 3.2 The "Lost in the Middle" Phenomenon

The degradation of context is not uniform. The **"Lost in the Middle"** phenomenon, identified by Liu et al., reveals a distinct U-shaped performance curve: models are highly effective at utilizing information at the beginning (primacy bias) and end (recency bias) of the context window but struggle significantly to access information buried in the middle.22

| **Position in Context** | **Performance Characteristic** | **Engineering Implication** |
| --- | --- | --- |
| **Start (Primacy)** | High Recall / High Attention | Place critical system instructions, persona definitions, and core rules here. |
| **Middle (Trough)** | Low Recall / "Lost" Data | Avoid placing critical constraints here. Use this space for bulk data or less critical history. |
| **End (Recency)** | High Recall / High Attention | Place the immediate user query, the most recent tool outputs, and "reminders" of critical rules here. |

This behavior challenges the "context as bucket" view, where data is simply dumped into the window. Instead, it necessitates a "context as architecture" approach, where critical instructions are strategically placed at the "material boundaries" of the input space to ensure retrieval. It also suggests that the "shape" of the data matters as much as the content. "Multi-scale Positional Encoding" (Ms-PoE) has been proposed to mitigate this, but purely prompt-based solutions involve "context refactoring"—moving the most relevant retrieved chunks to the edges of the prompt.24

### 3.3 Software 2.0 and the Context Compiler

Andrej Karpathy’s concept of **"Software 2.0"** posits that the future of programming involves curating datasets and defining objectives for neural networks rather than writing explicit logic.1 In this framework, the context window functions as the **compiler** for the vague intent of the user.

Just as a compiler translates high-level code into machine instructions, the context window translates high-level goals into the "probabilities" that drive token generation. If the "source code" (context) is buggy, ambiguous, or polluted, the "compiled program" (the model's behavior) will crash—hallucinating or failing to follow instructions.26 This analogy extends to the rigor required in context construction. Engineers are now treating prompts as "code," utilizing version control, modularity, and strictly typed interfaces (like Anthropic’s Model Context Protocol or MCP) to ensure that the "material" fed into the model is robust.28

The shift is from "vibe-based" prompting to deterministic **Context Engineering pipelines** that assemble the window from reliable components:

1. **System Instructions:** The "operating system" of the agent.
2. **Few-Shot Examples:** The "training data" of the context, providing pattern-matching targets.30
3. **Retrieved Knowledge (RAG):** The "dynamic libraries" linked at runtime.28

### 3.4 The Context Engineering Pipeline: Write, Select, Compress, Isolate

If context is material, it must be processed. The modern AI stack views context generation as a manufacturing pipeline involving four distinct operations 32:

* **Write:** Generating new context from raw data. This involves converting video, audio, or raw logs into "machine-ingestible" text or structured data. For example, converting a video into a JSON timeline of events allows the model to "read" the video.34
* **Select:** Filtering the vast ocean of available data to find the "minimal sufficiency" required for the task. This prevents context rot by respecting the attention budget. Selection is the act of the "agential cut"—deciding what matters.16
* **Compress:** Reducing the token count of selected information without losing semantic fidelity. This can involve summarization ("lossy compression") or vector embeddings. However, brevity bias must be managed; over-compression destroys the "thick" nuance required for high-quality outputs.21
* **Isolate:** Partitioning context between different agents or sub-tasks to prevent "context pollution." Just as reactive chemicals are stored separately, different narrative threads or reasoning tasks are kept in isolated context windows to maintain purity of thought. This is the "share memory by communicating, don't communicate by sharing memory" principle applied to AI.21

## 4. System Architecture: Agentic Context Engineering (ACE)

The most significant architectural evolution in this field is the move from static context to **Agentic Context Engineering (ACE)**. Traditional context engineering views the prompt as a fixed artifact created by a human developer. ACE, introduced in recent research from Stanford and SambaNova, reframes context as a **"living playbook"** that the AI agent itself maintains, evolves, and optimizes.36

### 4.1 The ACE Framework: Generator, Reflector, Curator

ACE addresses the twin failures of **"brevity bias"** (where models over-summarize and lose detail) and **"context collapse"** (where critical heuristics are lost over long horizons). It achieves this by mechanizing the curation of context through three distinct roles 36:

1. **The Generator:** This component attempts the task using the current context. It produces "reasoning trajectories"—traces of its thought process, tool usage, and final output. It generates the "raw material" of experience.36
2. **The Reflector:** This component analyzes the Generator's output. It compares successful and unsuccessful trajectories to distill lessons. Crucially, it engages in "multi-epoch reflection," revisiting past traces to refine its understanding. It produces "candidate bullets"—specific insights or rules derived from the experience (e.g., "Tool X fails when parameter Y is null").37
3. **The Curator:** This component acts as the editor of the context. It integrates the insights from the Reflector into the persistent "playbook." It applies **"incremental delta updates"**—surgical edits to the context rather than wholesale rewrites. It handles de-duplication and pruning to keep the context compact but dense.36

### 4.2 The Living Playbook and Self-Baking

In the ACE model, the context is not a "stream of consciousness" (like a linear chat log) but a structured artifact—often a bulleted list of heuristics, rules, and facts—that grows over time. This process is termed **"Self-Baking"**.35 The agent "bakes" raw, high-entropy interaction logs into durable, low-entropy knowledge that resides in the context window of future runs.

This solves the "Lost in the Middle" problem by ensuring that high-value information is not buried in a linear log but is "surfaced" into the active instruction set. The "material" is constantly being worked on—sharpened, polished, and reorganized—by the agent itself. Empirical results show that ACE yields double-digit improvements in agentic benchmarks without fine-tuning weights, suggesting that **context optimization** is a viable, and often superior, alternative to **parameter optimization**.37

### 4.3 Memory Architectures: From Storage to Synthesis

ACE implies a sophisticated memory architecture. We must distinguish between:

* **Session State:** The immediate "Working Context" visible to the model. This is the "hot" material.42
* **Episodic Memory:** A log of past interactions, often stored in vector databases (RAG). This is "cold" storage.43
* **Semantic Memory:** General knowledge and facts.
* **Procedural Memory (The Playbook):** The heuristics and rules managed by ACE. This is the "wisdom" of the agent.

Google's Agent Development Kit (ADK) formalizes this by separating "Session" (storage) from "Working Context" (view). The working context is a **"compiled view"** of the session state, optimized for the specific call.42 This allows for "Context Compaction" and "Context Filtering" to occur dynamically, ensuring the model only sees the material it needs for the immediate "situated action."

## 5. The Economics of Materiality: Caching and Cost Optimization

If context is a material, it has a cost. The processing of tokens requires massive computational energy (GPU cycles). Until recently, this cost limited the "thickness" of the context engineers could afford to provide. The introduction of **Context Caching** (or Prompt Caching) has revolutionized the economics of this material, allowing for the deployment of massive, persistent contexts at a fraction of the price.

### 5.1 The Mechanism of Caching: Freezing the Material

Context Caching works by "freezing" the activation states (KV cache) of a prefix of the context window. When an LLM processes a prompt, it computes Key-Value pairs for the attention mechanism. Traditionally, these are discarded after generation. Caching stores these states in the GPU/TPU memory (HBM) or a dedicated cache layer.45

When a subsequent request arrives with the same prefix, the model skips the expensive re-computation of those tokens and loads the state directly from memory. This transforms context from a *flow* (which must be re-processed every time) into a *solid* (which can be stored and reused).47

* **Implicit Caching:** Automatically detects repeated prefixes. This is the default in systems like Google Gemini (for paid tiers) and DeepSeek. It requires no code changes but relies on the developer sending identical text blocks.47
* **Explicit Caching:** Developers manually mark "breakpoints" or cache\_control flags in the prompt to define cacheable blocks. This is the approach taken by Anthropic (Claude). It gives engineers precise control over the "material boundaries" of their context, allowing them to stack "static tools," "world lore," and "dynamic conversation" in a cache-optimized hierarchy.51

### 5.2 Economic Implications of "Context Persistence"

The cost reductions from caching are dramatic—up to **90% cheaper** for cached tokens compared to fresh input tokens, and latency reductions of up to **85%**.48 This fundamentally alters the design space for AI systems:

* **From RAG to Long-Context:** Traditionally, RAG was used to "sip" small amounts of context to save money. With caching, engineers can "dump" entire books, codebases, or conversation histories into the window and keep them there "hot" for the duration of a session. The "Context as Material" becomes a permanent *infrastructure* rather than a temporary *input*.
* **Thick Context Viability:** It becomes economically feasible to include "thick" cultural instructions, extensive persona descriptions, and "living playbooks" (ACE) in every call, because this static content is paid for only once.46
* **Agentic Plan Caching (APC):** Research suggests extending this to cache not just input tokens but *intermediate reasoning plans*. If an agent has already figured out how to "analyze a balance sheet," that "thought process" (the plan template) is cached and reused. This treats the agent's *cognition* as a reusable material resource, reducing costs by ~50% for complex agentic workflows.45

| **Feature** | **Implicit Caching (e.g., Gemini)** | **Explicit Caching (e.g., Anthropic)** | **Economic Impact** |
| --- | --- | --- | --- |
| **Mechanism** | Auto-detects prefix match | Developer sets cache\_control points | Drastically lowers input costs (up to 90%) |
| **Control** | Low ( "Magic") | High (Deterministic) | Encourages "Context-Heavy" architectures |
| **TTL (Time to Live)** | Variable (based on usage) | Fixed (e.g., 5 min, refreshable) | Requires "Keep-Alive" strategies for caches |
| **Use Case** | High-volume repetitive queries | Complex agents with large static prompts | Enables "Thick" context at "Thin" prices |

### 5.3 DeepSeek and the Commoditization of Inference

The emergence of providers like **DeepSeek** has further accelerated this trend. By combining aggressive model optimization (Mixture-of-Experts) with disk-based context caching, DeepSeek offers inference at prices orders of magnitude lower than traditional providers ($0.014 per million cached tokens).50 This pricing structure effectively commoditizes the "material" of intelligence, suggesting a future where context is abundant and cheap, shifting the engineering bottleneck from *cost* to *quality* and *curation*.

## 6. Narrative Orchestration: Structuring the Flow

The culmination of Context Engineering is **Narrative Orchestration**. This moves beyond static Q&A to the dynamic management of a "story"—whether that story is a literal interactive fiction or a complex business workflow. In this regime, the context is the "stage" upon which the agent performs.

### 6.1 The Orchestrator as "Vibe Coder"

In the era of autonomous coding agents, the role of the human developer shifts. The term **"Vibe Coding"** has emerged to describe the practice of managing the *intent* and *context* of the software rather than the syntax.25 The Vibe Coder is a "Narrative Orchestrator" who defines the constraints, biases, and goals that shape the material of the narrative.

This requires "Emotional Precision"—the ability to articulate the "vibe" or "thick description" of the desired output so clearly that the model can generate the correct code or content. It is "coding with context".57

### 6.2 Interactive Drama and the AI Game Master

The most advanced application of narrative orchestration is found in **Interactive Drama** and **AI Game Masters (GM)**. These systems must manage a complex, branching narrative while maintaining consistency and player agency.

* **Global vs. Local Context:** Frameworks like **ChatRPG** or **HAMLET** utilize a braided memory architecture. "Global Memory" holds the world state (immutable rules, lore). "Summary Memory" holds the narrative arc (the story so far). "Local Memory" holds the immediate scene details.58
* **The Narrator and the Archivist:** These systems often split the GM role into sub-agents. The **Narrator** focuses on generating immersive descriptions ("thick outputs"), while the **Archivist** or **Curator** manages the database of facts ("thin descriptions").59
* **Dynamic Tool Use:** The GM agent uses tools to "wound characters," "update inventory," or "roll dice." These are "situated actions" that modify the context material. The context acts as the "database of truth" for the game world.59

### 6.3 Form Follows Function: The Materiality of Format

The "material-semiotics" of LLMs dictates that *how* context is formatted is as important as *what* it contains. Research indicates that LLMs process **JSON**, **XML**, and **Markdown** differently, treating them as distinct "textures" of information.61

* **JSON/XML:** High structural integrity. Used for data, tool outputs, and strict constraints. The model treats this as "syntax to be parsed."
* **Markdown:** High narrative integrity. Used for system instructions and "human-like" guidance. The model treats this as "text to be read."

Effective context engineering involves **"Context Formatting"**—choosing the right "material substrate" for each piece of information. Critical instructions might be encased in XML tags (e.g., <critical\_rule>) to create an "agential cut" that separates them from the rest of the noise.33 This is akin to highlighting text in a physical document; it changes the "saliency" of the material.

## 7. Conclusion: The Era of the Context Architect

The evolution from "Context as Container" to "Context as Material" represents the maturing of AI engineering from a stochastic art to a rigorous discipline. We have moved from blindly prompting models to **engineering the environments** in which they think.

The evidence from **Agentic Context Engineering (ACE)**, **Context Caching**, and **Sociomaterial analysis** converges on a single truth: **Intelligence is situational.** An LLM's capability is not fixed in its weights; it is dynamically constructed from the material provided in its context window.

As we look toward **Software 3.0** and autonomous agents, the role of the human shifts from "operator" to "Context Architect." We are no longer just asking questions; we are building the "webs of significance," the "situated environments," and the "living playbooks" that allow artificial minds to inhabit our world with thickness, nuance, and reliability. The future of AI belongs to those who can master the physics of this new digital material.

#### Works cited

1. Overparameterization: A Connection Between Software 1.0 and Software 2.0 - DSpace@MIT, accessed December 10, 2025, <https://dspace.mit.edu/bitstream/handle/1721.1/130046/LIPIcs-SNAPL-2019-1.pdf?sequence=2&isAllowed=y>
2. State of GPT | PDF | Information | Cognitive Science - Scribd, accessed December 10, 2025, <https://www.scribd.com/document/660468131/State-of-Gpt>
3. "Too Much Alignment; Not Enough Culture": Re-balancing cultural alignment practices in LLMs - arXiv, accessed December 10, 2025, <https://arxiv.org/html/2509.26167v1>
4. Taming Cultural Fractals: Why AI Needs to Get 'Thick'. - AI Singapore, accessed December 10, 2025, <https://aisingapore.org/ai-governance/taming-cultural-fractals-why-ai-needs-to-get-thick/>
5. CURE: Cultural Understanding & Reasoning Evaluation – A Framework for “Thick” Culture Alignment Evaluation in LLMs - arXiv, accessed December 10, 2025, <https://arxiv.org/html/2511.12014v1>
6. (PDF) Situated Cognition in Theoretical and Practical Context - ResearchGate, accessed December 10, 2025, <https://www.researchgate.net/publication/239062816_Situated_Cognition_in_Theoretical_and_Practical_Context>
7. SituatedThinker: Grounding LLM Reasoning with Real-World through Situated Thinking - arXiv, accessed December 10, 2025, <https://arxiv.org/pdf/2505.19300>
8. Dimensions of Human-Machine Combination: Prompting the Development of Deployable Intelligent Decision Systems for Situated Clinical Contexts - PMC - PubMed Central, accessed December 10, 2025, <https://pmc.ncbi.nlm.nih.gov/articles/PMC12436561/>
9. Generative AI and empirical software engineering: A paradigm shift - Institutional Knowledge (InK) @ SMU, accessed December 10, 2025, <https://ink.library.smu.edu.sg/cgi/viewcontent.cgi?article=11511&context=sis_research>
10. Generative AI and Empirical Software Engineering: A Paradigm Shift - arXiv, accessed December 10, 2025, <https://arxiv.org/html/2502.08108v2>
11. Beyond Bias: Studying 'culture' in LLMs and AI chatbots - OSF, accessed December 10, 2025, <https://osf.io/kz2xc_v1/download/>
12. Researching Education Through Actor-Network Theory - ResearchGate, accessed December 10, 2025, <https://www.researchgate.net/publication/296531757_Researching_Education_Through_Actor-Network_Theory>
13. Hayles – Posthuman Condition and Semiotic Materialism | Emerging into Structure, accessed December 10, 2025, <https://francovich.wordpress.com/2025/07/12/hayles-posthuman-condition-and-semiotic-materialism/>
14. (PDF) The entangled human being – a new materialist approach to ..., accessed December 10, 2025, <https://www.researchgate.net/publication/383667821_The_entangled_human_being_-_a_new_materialist_approach_to_anthropology_of_technology>
15. View of Infrastructuring AI: The stabilization of 'artificial intelligence' in and beyond national AI strategies | First Monday, accessed December 10, 2025, <https://firstmonday.org/ojs/index.php/fm/article/view/13568/11402>
16. Context Engineering 2.0: The Context of Context Engineering | by ..., accessed December 10, 2025, <https://praveengovindaraj.com/context-engineering-2-0-the-context-of-context-engineering-1d11f3843185>
17. A time for monsters: Organizational knowing after LLMs - arXiv, accessed December 10, 2025, <https://arxiv.org/pdf/2511.15762>
18. accessed December 10, 2025, <https://hamel.dev/notes/llm/rag/p6-context_rot.html#:~:text=This%20slide%20introduces%20the%20concept,of%20its%20input%20context%20grows.>
19. P6: Context Rot – Hamel's Blog, accessed December 10, 2025, <https://hamel.dev/notes/llm/rag/p6-context_rot.html>
20. Effective context engineering for AI agents - Anthropic, accessed December 10, 2025, <https://www.anthropic.com/engineering/effective-context-engineering-for-ai-agents>
21. Context Engineering for AI Agents: Part 2 - Philschmid, accessed December 10, 2025, <https://www.philschmid.de/context-engineering-part-2>
22. Lost in the Middle: How Language Models Use Long Contexts, accessed December 10, 2025, <https://teapot123.github.io/files/CSE_5610_Fall25/Lecture_12_Long_Context.pdf>
23. Lost in the Middle: How Language Models Use Long Contexts - Stanford Computer Science, accessed December 10, 2025, <https://cs.stanford.edu/~nfliu/papers/lost-in-the-middle.arxiv2023.pdf>
24. Found in the Middle: How Language Models Use Long Contexts Better via Plug-and-Play Positional Encoding - arXiv, accessed December 10, 2025, <https://arxiv.org/html/2403.04797v1>
25. How Software Engineers Can Futureproof Their Career With AI - Open Data Science, accessed December 10, 2025, <https://opendatascience.com/how-software-engineers-can-futureproof-their-career-with-ai/>
26. Context Engineering: The New Paradigm Every Developer Should Know | by Erol Kuluslu, accessed December 10, 2025, <https://medium.com/@erolkuluslusoftware/context-engineering-the-new-paradigm-every-developer-should-know-7e3d8478dbd6>
27. Agentic Context Engineering - Sundeep Teki, accessed December 10, 2025, <https://www.sundeepteki.org/blog/agentic-context-engineering>
28. What Is Context Engineering? A Guide for AI & LLMs | IntuitionLabs, accessed December 10, 2025, <https://intuitionlabs.ai/articles/what-is-context-engineering>
29. What Is Model Context Protocol (MCP)? A Quick Start Guide. - Monte Carlo Data, accessed December 10, 2025, <https://www.montecarlodata.com/blog-model-context-protocol-mcp>
30. (PDF) Strategic Dialogue Architecture for LLMs: From Prompting to Context Engineering, accessed December 10, 2025, <https://www.researchgate.net/publication/395572928_Strategic_Dialogue_Architecture_for_LLMs_From_Prompting_to_Context_Engineering>
31. RAG Strategies: Retrieval-Augmented Generation - Emergent Mind, accessed December 10, 2025, <https://www.emergentmind.com/topics/retrieval-augmented-generation-rag-strategies>
32. LLM Context Engineering: a practical guide | by Zheng "Bruce" Li | The Low End Disruptor, accessed December 10, 2025, <https://medium.com/the-low-end-disruptor/llm-context-engineering-a-practical-guide-248095d4bf71>
33. Context Engineering: Moving Beyond Prompting in AI - DigitalOcean, accessed December 10, 2025, <https://www.digitalocean.com/community/tutorials/context-engineering-moving-beyond-prompting-ai>
34. Context Engineering for Video Understanding - Twelve Labs, accessed December 10, 2025, <https://www.twelvelabs.io/blog/context-engineering-for-video-understanding>
35. Context Engineering: Teaching Machines to Read Between the Lines | by Igor Comune, accessed December 10, 2025, <https://igorcomune.medium.com/context-engineering-teaching-machines-to-read-between-the-lines-788702daeb4a>
36. Agentic Context Engineering: Prompting Strikes Back - Superagentic AI Blog, accessed December 10, 2025, <https://shashikantjagtap.net/agentic-context-engineering-prompting-strikes-back/>
37. [2510.04618] Agentic Context Engineering: Evolving Contexts for Self-Improving Language Models - arXiv, accessed December 10, 2025, <https://arxiv.org/abs/2510.04618>
38. Your Agents Just Got a Memory Upgrade: ACE Open-Sourced on GitHub - SambaNova, accessed December 10, 2025, <https://sambanova.ai/blog/ace-open-sourced-on-github>
39. Agentic Context Engineering (ACE) - Emergent Mind, accessed December 10, 2025, <https://www.emergentmind.com/topics/agentic-context-engineering-ace>
40. Agentic Context Engineering: Evolving Contexts for Self-Improving Language Models - arXiv, accessed December 10, 2025, <https://arxiv.org/html/2510.04618v1>
41. Agentic Context Engineering (ACE): Teach Context to Learn from Itself | by Shikhar Dave | Oct, 2025 | Medium, accessed December 10, 2025, <https://medium.com/@sd16/agentic-context-engineering-ace-teach-context-to-learn-from-itself-580833b9389e>
42. Architecting efficient context-aware multi-agent framework for production, accessed December 10, 2025, <https://developers.googleblog.com/architecting-efficient-context-aware-multi-agent-framework-for-production/>
43. Building smarter AI agents: AgentCore long-term memory deep dive - AWS, accessed December 10, 2025, <https://aws.amazon.com/blogs/machine-learning/building-smarter-ai-agents-agentcore-long-term-memory-deep-dive/>
44. Google Just Dropped 70 Pages on Context Engineering. Here's What Actually Matters., accessed December 10, 2025, <https://aakashgupta.medium.com/google-just-dropped-70-pages-on-context-engineering-heres-what-actually-matters-c0df8d8e82cc>
45. Agentic Plan Caching: Test-Time Memory for Fast and Cost-Efficient LLM Agents - OpenReview, accessed December 10, 2025, <https://openreview.net/pdf?id=n4V3MSqK77>
46. Vertex AI context caching | Google Cloud Blog, accessed December 10, 2025, <https://cloud.google.com/blog/products/ai-machine-learning/vertex-ai-context-caching>
47. Intro to Context Caching with the Gemini API - Google Colab, accessed December 10, 2025, <https://colab.research.google.com/github/GoogleCloudPlatform/generative-ai/blob/main/gemini/context-caching/intro_context_caching.ipynb>
48. Prompt Caching Support in Spring AI with Anthropic Claude, accessed December 10, 2025, <https://spring.io/blog/2025/10/27/spring-ai-anthropic-prompt-caching-blog>
49. Context caching | Gemini API | Google AI for Developers, accessed December 10, 2025, <https://ai.google.dev/gemini-api/docs/caching>
50. DeepSeek's Low Inference Cost Explained: MoE & Strategy | IntuitionLabs, accessed December 10, 2025, <https://intuitionlabs.ai/articles/deepseek-inference-cost-explained>
51. Prompt caching - Claude Docs, accessed December 10, 2025, <https://platform.claude.com/docs/en/build-with-claude/prompt-caching>
52. Prompt caching with Claude, accessed December 10, 2025, <https://www.claude.com/blog/prompt-caching>
53. Prompt Caching is a Must! How I Went From Spending $720 to $72 Monthly on API Costs | by Du'An Lightfoot | Medium, accessed December 10, 2025, <https://medium.com/@labeveryday/prompt-caching-is-a-must-how-i-went-from-spending-720-to-72-monthly-on-api-costs-3086f3635d63>
54. Prompt Caching & Token Economics in 2025: How to Cut Cost Without Losing Quality, accessed December 10, 2025, <https://promptbuilder.cc/blog/prompt-caching-token-economics-2025>
55. how to save 90% on ai costs with prompt caching? need real implementation advice - Reddit, accessed December 10, 2025, <https://www.reddit.com/r/ClaudeAI/comments/1oawd9c/how_to_save_90_on_ai_costs_with_prompt_caching/>
56. Cost-Efficient Serving of LLM Agents via Test-Time Plan Caching - arXiv, accessed December 10, 2025, <https://arxiv.org/html/2506.14852v1>
57. Why You Should Consider Vibe Coding a Vital Literacy Skill | Built In, accessed December 10, 2025, <https://builtin.com/articles/vibe-coding-literacy-skill>
58. LLM-Based Interactive Drama - Emergent Mind, accessed December 10, 2025, <https://www.emergentmind.com/topics/llm-based-interactive-drama>
59. Static Vs. Agentic Game Master AI for Facilitating Solo Role-Playing Experiences - arXiv, accessed December 10, 2025, <https://arxiv.org/html/2502.19519v2>
60. Designing Meaningful AI-Generated Dialogue: The Behaviour- Driven Conditional Prompting Framework for Serious Games - Academic Conferences International, accessed December 10, 2025, <https://papers.academic-conferences.org/index.php/ecgbl/article/download/4182/3667/14844>
61. How to Create AI Agents from Scratch (A Step-by-Step Guide) - Apidog, accessed December 10, 2025, <https://apidog.com/blog/create-ai-agents/>
62. Prompt structure and bullet points : r/PromptEngineering - Reddit, accessed December 10, 2025, <https://www.reddit.com/r/PromptEngineering/comments/1exltu2/prompt_structure_and_bullet_points/>