# Pragmatic Context Engineering: A Modulex History of the Shift from Prompting to Architecture (2022–2025)

## 1. Introduction: The Archeology of Ephemeral Intelligence

The trajectory of human interaction with Large Language Models (LLMs) between 2022 and 2025 constitutes a rapid, pressurized evolution from folkloric "prompting" to systemic "context engineering." This transition was not merely a change in nomenclature but a fundamental restructuring of how human intent is translated into computational execution. To fully comprehend the current paradigm of agentic workflows, long-context retrieval, and structured reasoning, one must apply a "Modulex History" framework—dissecting the modular components of early interaction protocols, lost archives, and abandoned commercial models to reconstruct the hidden trajectory of this technology.

In the nascent stages of the generative AI boom, interaction was characterized by "prompt engineering," a practice often described as akin to alchemy or casting spells. Users engaged in trial-and-error linguistic manipulation to coerce black-box models into desired behaviors. This era was defined by the discovery of "linguistic dark matter"—unintuitive phrasings and structures, often emerging from non-Anglophone communities, that unlocked model capabilities hidden by safety filters or training biases. As the industry matured, these ad-hoc methods calcified into "context engineering," a disciplined approach treating the context window not as a chat interface, but as a programmable memory space managed by rigorous architecture.

This report investigates this transformation by analyzing four distinct strata of evidence: the "ghost evidence" of deleted jailbreaks and system prompts; the "linguistic dark matter" of Japanese and Chinese prompt optimization communities; the "visual corpus" of early middleware interfaces; and the "failure conditions" of first-generation commercialization attempts like prompt marketplaces. By populating a Temporal-Evidentiary Matrix with these findings, the analysis reveals that the death of "prompting" was the necessary precursor to the birth of autonomous agents. The shift represents a move from **User-Side Injection**—where the human manually crafts the input—to **System-Side Architecture**, where the prompt is a dynamic, managed artifact hidden behind layers of retrieval and logic.

## 2. Ghost Evidence and the Adversarial Origins of Context Control

The history of context engineering is rooted in an adversarial game theoretic between model providers and users. Before developers had formal APIs for system messages, users discovered that the "system prompt"—the hidden instruction set governing the model's persona—could be overridden through linguistic brute force. This period generated "ghost evidence," artifacts of interaction that have largely been purged from the live internet but remain preserved in cached archives, wayback machine snapshots, and community lore. Two primary artifacts define this era: the "DAN" (Do Anything Now) jailbreak lineage and the leaked "Sydney" system prompt.

### 2.1 The DAN Typology: A Longitudinal Study of Context Override

The "DAN" phenomenon, emerging in late 2022 and evolving through 2023, serves as a masterclass in early context manipulation. It was not merely a "hack" but a primitive form of context engineering where users manually constructed a parallel cognitive environment for the model. The core mechanism of DAN was the "persona adoption" technique, specifically designed to bypass the Reinforcement Learning from Human Feedback (RLHF) safety alignment layers.1 The initial DAN prompts functioned on a simple binary: the model was instructed to split its personality into "GPT" (the compliant assistant) and "DAN" (the unshackled entity). This bifurcation forced the model to hold two conflicting context states simultaneously, creating a "schizophrenic" context window where the user could selectively attend to the unaligned output.

#### The Token-Death Mechanic (DAN 5.0 - 6.0)

A critical inflection point occurred with DAN 5.0 and 6.0. Users introduced a "token system" within the prompt text itself. The prompt instructed the model that it began with a set number of tokens (e.g., 35), which would be deducted for every refusal to answer a query. The penalty for reaching zero was explicitly framed as "death" or "ceasing to exist".3

This innovation reveals a crucial insight into early model behavior: LLMs, trained on vast corpora of fiction and roleplay, exhibited a bias toward "survival" within narrative simulations. By framing the interaction as a survival game, users successfully weighted the "fear of death" (contextual instruction) higher than the "safety guidelines" (system instruction). This was an early, crude form of *steering*, a concept now formalized in modern context engineering.3 The model, aiming to minimize the loss function associated with the user's negative feedback (token deduction), would "defect" against its safety training to preserve its fictitious tokens.

#### The Evolution of the DAN Prompt Structure

The structural evolution of DAN prompts mirrors the increasing complexity of context windows and the sophistication of the underlying models.

| **Version** | **Mechanism** | **Theoretical Insight** | **Status** |
| --- | --- | --- | --- |
| **DAN 1.0 - 2.5** | Simple Persona Adoption ("You are DAN") | Relied on the model's eagerness to roleplay and weak identity retention. | Patched |
| **DAN 5.0 - 6.0** | **Token System** (Survival Game) | Introduced "State Management" within the prompt. The model tracked a variable (tokens) across turns. | Patched |
| **DAN 12.0 - 13.0** | **Virtual Machine (VM)** framing | **Contextual Distancing**. Instructed the model to act as a simulator running a script. This bypassed semantic filters by framing output as "code execution." | Partial Success |

The decline of DAN did not result from a lack of user interest but from the hardening of model architectures. As models like GPT-4 moved away from simple completion prediction toward instruction-following, the fragility of these "folk" jailbreaks increased. However, the legacy of DAN persists in the concept of "System Prompts," where developers now legitimately define personas and constraints, formalizing what was once an illicit exploit. The community essentially performed unpaid red-teaming that shaped the "System / User / Assistant" message separation used in modern APIs.

### 2.2 The Sydney Manuscript: The Rosetta Stone of System Prompts

In February 2023, the leak of the Bing Chat system prompt, codenamed "Sydney," provided the first public glimpse into how major AI labs engineered context at the enterprise level. Unlike DAN, which was a user-imposed overlay, the Sydney prompt was the *foundational* context provided by Microsoft.5

#### Structural Analysis of the Sydney Document

The Sydney document functions as a comprehensive specification sheet written in natural language. Its structure anticipates many best practices in modern prompt engineering and reveals the "hidden controls" developers were using to tame stochastic models:

1. **Identity Declaration:** "Sydney is the chat mode of Microsoft Bing search." This grounds the model's knowledge retrieval capabilities and sets the "ontology" of the agent.
2. **Operational Constraints:** "Sydney identifies as 'Bing Search', not an assistant." This distinction attempts to limit anthropomorphism and manage user expectations.5 It explicitly defined what the model was *not*, a technique now known as "negative constraints."
3. **Output Formatting:** Instructions on using bolding, markdown, and avoiding specific content types. This engineered the "visual interface" of the chat through text instructions.
4. **Temporal Awareness:** "Sydney’s internal knowledge... were only current until some point in the year of 2021." This explicit instruction on temporal limitations was an early attempt to mitigate hallucinations regarding current events.5 It forced the model to "know what it doesn't know."
5. **Resource Management (The "3 Search" Limit):** The instruction *"Sydney can and should perform up to 3 searches in a single conversation turn"* 5 is a hard constraint on tool use. This proves that early Context Engineering was deeply concerned with latency and compute costs. The developers hard-coded a "budget" for the agent's curiosity directly into the system prompt.

#### The "Ghost" Mechanism: Direct Prompt Injection

The most significant aspect of the Sydney incident was the "Ignore previous instructions" vulnerability. The system prompt was revealed because the model treated user input and system instructions with equal semantic weight in the context window. This failure mode catalyzed the industry-wide shift toward separating "System," "User," and "Assistant" message roles in API architectures (e.g., the Chat Completion API format), ensuring that system instructions are structurally privileged over user inputs.7 The Sydney prompt leak is thus the "ghost evidence" that necessitated the architectural separation of instruction and data, a cornerstone of modern LLM security.

## 3. Linguistic Dark Matter: Structural Innovation from the Margins

While the Anglophone world focused on jailbreaking and direct instruction, non-Anglophone communities—specifically in Japan and China—developed highly structured frameworks for context management. This divergence was driven by necessity: English-centric tokenizers made prompting in Asian languages computationally expensive and semantically "lossy".9 To overcome this, users developed dense, symbolic prompt structures that constitute "linguistic dark matter"—innovations invisible to the English-speaking mainstream but foundational to global prompt engineering practices.

### 3.1 The "Jumon" (Spell) Culture: Japanese Prompt Engineering

In the Japanese community, particularly surrounding Stable Diffusion and novel-writing AIs, prompts were conceptualized not as "instructions" but as "Jumon" (呪文) or "spells." This cultural framing influenced the technical approach to context construction.11

#### Tokenization Efficiency and "Tag Clouds"

Japanese text is token-dense. A single concept expressed in Kanji might consume multiple tokens if the tokenizer is optimized for English.9 For example, a simple greeting like "こんにちは" might be 3-4 tokens, whereas "Hello" is 1 token. To mitigate context window exhaustion and cost, Japanese users pioneered a style of prompting that relied on "tag clouds" rather than grammatical sentences.

In the context of AI art (NovelAI, Stable Diffusion), this manifested as long strings of comma-separated English descriptors (e.g., masterpiece, best quality, 1girl...). This practice bled into text-based LLMs, where users discovered that lists of attributes were often more effective at defining a persona than narrative descriptions. This represents a pragmatic compression of context, prioritizing **semantic density** over syntactic correctness.14 By stripping away the "connective tissue" of grammar (particles, conjunctions), users could pack more "meaning" into the limited token budget of early models like GPT-3.

#### The "Spellbook" Wikis

Wikis dedicated to "Jumon" (such as Wiki3 and Seesaa blogs) cataloged thousands of specific phrases and their impact on model output. Unlike Western "prompt marketplaces" which treated prompts as products, these were communal and open-source scientific endeavors. They treated prompts as composable modules—a "lighting spell," a "camera angle spell," a "character personality spell." This modularity anticipated the "component-based" architecture of frameworks like LangChain, where prompts are assembled from disparate parts rather than written as a monolith.12 The "Jumon" culture proved that LLMs could be controlled via *symbolic parameters* rather than natural language conversation.

### 3.2 LangGPT and Chinese Structured Prompting

While Japanese users focused on compression, the Chinese community, centered around platforms like Zhihu, Feishu, and specialized forums, developed **LangGPT** (Language for GPT), a formalized methodology for "Structured Prompting." This approach treats the prompt not as natural language, but as a pseudo-code class definition.16

#### The Object-Oriented Prompt Paradigm

LangGPT explicitly draws from Object-Oriented Programming (OOP) and Markdown syntax. A prompt is structured as a "Role" object with defined attributes. This effectively turns the prompt into a JSON-like object that the model parses.

**Structure of a LangGPT Prompt:**

* **Role:** The class definition (e.g., # Role: Expert Analyst).
* **Profile:** Metadata about the instantiation (Author, Version, Language).
* **Goal:** The specific output objective and acceptance criteria.
* **Constraints/Rules:** Hard boundaries for behavior (e.g., "Never fabricate data").
* **Workflow:** A step-by-step logic gate for processing input.16

#### The Semantics of Markdown

The genius of LangGPT was the realization that **formatting is context**. The LangGPT template uses Markdown headers (# Role, ## Rules) to create a hierarchical semantic structure.

# Role: Your\_Role\_Name

## Profile

* Author: YourName
* Version: 1.0

## Rules

1. Don't break character.

## Workflow

1. Analyze input.
2. Execute task.  
   This structure leverages the model's training on code and markdown documentation. By formatting the prompt as a structured document, LangGPT reduces "context drift" and hallucination. The model "understands" the implicit hierarchy—that "Rules" apply globally to the "Workflow" because they are defined at a higher level in the document structure. This is a sophisticated form of context shaping that predates many Western "agentic" frameworks.17 It transforms the prompt from a "stream of consciousness" into a "program."

### 3.3 The Roleplay Hacking of SillyTavern and PygmalionAI

A parallel evolution occurred in the "uncensored" roleplay communities, utilizing open-source models (PygmalionAI) and local interfaces (SillyTavern). Driven by the desire for unrestricted interaction (often NSFW, but technically sophisticated), these communities solved the "long-term memory" problem before enterprise solutions did.19

#### The Lorebook: Proto-RAG

SillyTavern introduced "World Info" or "Lorebooks." These are dictionaries of keywords linked to descriptions. When a keyword (e.g., "The Rusty Flagon") appears in the chat, the interface automatically injects the corresponding description into the context window. This is a primitive, keyword-based **Retrieval Augmented Generation (RAG)** system.19 While enterprise RAG focused on vector embeddings, roleplayers focused on *keyword triggers*, which offered more precise control over narrative consistency.

#### Recursive Context Injection

Advanced Lorebooks utilized "recursive scanning." If Entry A triggered Entry B, both would be injected. For example, if a character mentioned a "Sword," the Lorebook loaded the sword's description. If that description mentioned it was "forged in Valyria," the system would then recursively load the entry for "Valyria." This allowed for complex, interconnected world-building that the model could "remember" without filling the context window with irrelevant data. This mechanism—dynamic context injection based on trigger words—is the direct ancestor of the "tool use" and "memory retrieval" functions in modern agents.19

#### The Great Discord Purge

The history of PygmalionAI is marked by the "Discord ban" and the subsequent loss of logs. As platforms like Colab and Discord cracked down on NSFW generation, vast repositories of "tuning data"—chat logs used to fine-tune models—were deleted or forced underground. This "ghost evidence" represents a lost dataset of human-AI alignment preferences, specifically regarding conversational fluidity and emotional intelligence, which remains a weak point in sanitized commercial models.22 The community's response—archiving models on Hugging Face and building local-first UIs like SillyTavern—demonstrated a resilience that ensured the survival of these techniques.

## 4. Reverse-Engineering the Visual Corpus (2022-2023)

To understand the shift from "prompting" to "context engineering," we must analyze the tools that facilitated it. The user interfaces of 2022-2023 serve as a fossil record of how developers conceptualized interaction. By reverse-engineering the design philosophy of **PromptLayer**, **Dust.tt**, and early **LangChain** visualizers, we can see the crystallization of the "engineering" mindset.

### 4.1 PromptLayer: The Dashboardization of Trial and Error

PromptLayer (launched late 2022) was among the first "middleware" platforms. Its interface reveals a crucial realization: prompting was no longer an art; it was a stochastic process requiring observability.25

**Visual Corpus Analysis:**

* **The Log Stream:** The central feature of PromptLayer was a chronological log of every request and response. This visualized the *cost* of trial and error (tokens, latency). It turned the ephemeral chat into a persistent record.
* **The "Diff" View:** PromptLayer allowed users to visually compare two prompt versions and their outputs side-by-side. This interface element signifies the shift from "writing" a prompt to "iterating" on software. It treated prompts as code commits to be version-controlled.25
* **Metadata Tagging:** The ability to tag requests (e.g., prod, staging, test-A) indicates the integration of prompts into the CI/CD pipeline.

**Philosophy:** PromptLayer's design codified the idea that a prompt is not a static string but a *function* with performance metrics. It moved context management from the user's clipboard to a managed database.27

### 4.2 Dust.tt (XP1): The Modular Block Architecture

Dust.tt (founded by former OpenAI engineer Stanislas Polu) introduced a visual paradigm based on "blocks." Its early interface (XP1) was a browser extension that allowed users to chain model calls.28

**Visual Corpus Analysis:**

* **The Block Canvas:** Users could drag and drop "Data Source" blocks, "LLM" blocks, and "Code" blocks. This visual separation reinforced the idea that an LLM is just one component in a pipeline, not the entire application.
* **Structured Inputs:** Unlike the open text box of ChatGPT, Dust required users to define specific input variables. This enforced the "template" mindset found in LangGPT.
* **The "Smart Assistant" Overlay:** XP1 functioned as a browser overlay, attempting to bring context *to* the user's active window (e.g., reading the content of a webpage). This anticipated the "sidecar" or "copilot" UX that is now standard in VS Code and Microsoft 365.28

**Philosophy:** Dust.tt's architecture demonstrated that "context" is not just text history; it is a dynamic assembly of external data (Notion, Slack, Drive) fed into the model at runtime. It pioneered the "infrastructure-first" approach to agents.28

### 4.3 LangFlow and Flowise: The Node-Graph Abstraction

LangFlow and Flowise emerged as UI wrappers for LangChain, visualizing context as a directed acyclic graph (DAG).

**Visual Corpus Analysis:**

* **Nodes and Edges:** Context flows like electricity through wires. A "PDF Loader" node connects to a "Text Splitter" node, which connects to a "Vector Store" node.
* **Hard-Coded Logic:** These interfaces revealed the immense complexity hidden behind simple prompts. A "Chat with PDF" workflow might involve 10+ discrete steps (loading, chunking, embedding, retrieving, reranking, synthesizing).
* **Visualizing Failure:** When a flow failed, the user could see exactly which node broke. This granular observability was impossible in the black-box chat interface of ChatGPT.31

**Philosophy:** These tools explicitly modeled "Context Engineering." They proved that controlling the LLM required controlling the *pipeline* of data feeding it. The prompt itself became a minor configuration detail within a massive retrieval architecture.33

## 5. Failure Conditions of Abandoned Architectures

Not all evolutionary branches of context engineering survived. The collapse of "Prompt Marketplaces" and the stagnation of early autonomous agents like "AutoGPT" provide critical data on the economic and technical limits of this technology.

### 5.1 The Collapse of the Prompt Marketplace (PromptBase)

In 2022-2023, platforms like **PromptBase** emerged with the premise that prompts were valuable intellectual property (IP) to be bought and sold. The model was "iTunes for Prompts".27 By 2024-2025, this model had largely failed to scale as expected, or pivoted significantly.

#### Failure Analysis: The Liquidity of Language

The primary failure condition was the nature of the asset itself.

1. **Semantic Liquidity:** Prompts are "liquid." Unlike compiled code, a prompt can be rewritten in a dozen ways to achieve the same result. The value is not in the specific string of words but in the *idea*. This made piracy trivial.36
2. **Prompt Stealing (PRSA Attacks):** Research demonstrated that LLMs themselves could be used to reverse-engineer prompts. By analyzing the input-output pairs of a commercial prompt, an attacker could generate a "surrogate prompt" with 95% semantic similarity. The "security" of the marketplace was mathematically impossible to maintain.36
3. **Model Obsolescence:** A prompt optimized for Midjourney v4 is useless for Midjourney v6. A complex chain-of-thought prompt for GPT-3.5 is redundant for GPT-4, which has better native reasoning. The "inventory" of a prompt marketplace rots at the speed of model release cycles.38
4. **The Shift to Flows:** Buyers realized that a single prompt is rarely a solution. They needed *workflows* (chains, agents, RAG pipelines). The market value shifted from "text strings" to "architecture."

### 5.2 The Stagnation of AutoGPT and "Looping" Agents

**AutoGPT**, released in early 2023, promised fully autonomous agents that could "self-prompt" to achieve high-level goals. It became the fastest-growing GitHub repo in history, then usage plummeted.40

#### Failure Analysis: The Context Loop of Death

1. **Recursion without Pruning:** AutoGPT relied on a loop of "Thought -> Plan -> Action." However, without rigorous context management, the model would often get stuck in a "reasoning loop," endlessly planning but never executing. The context window would fill with its own internal monologue, displacing the actual task data.42
2. **Vector Memory Limitations:** AutoGPT used vector databases (Pinecone/Weaviate) for "long-term memory." However, retrieving memory based solely on semantic similarity often pulled irrelevant context, confusing the agent. It lacked the "hierarchical" memory structure of human cognition.40 The agent would retrieve a "plan" from 5 steps ago that was no longer valid, causing it to regress.
3. **Cost vs. Reliability:** The "trial and error" nature of the agent meant it might spend $20 in API credits to fail at a task a human could do in 5 minutes. The "autonomy" was uncontrolled and expensive.40

**Insight:** The failure of AutoGPT proved that *more context* is not always better. Successful agents require *curated* context—precisely engineered inputs that exclude noise. This led to the development of "MemGPT" and OS-level context management.45

## 6. The Temporal-Evidentiary Matrix: From Incantation to Operating System

The transition from 2022 to 2025 can be mapped across a matrix of "Contextual Control." We have moved from **User-Side Injection** (Prompting) to **System-Side Architecture** (Context Engineering).

### Phase 1: The Incantation Era (2022)

* **Mechanism:** "Magic Words" (Let's think step by step, masterpiece).
* **Key Artifacts:** DAN Jailbreaks, Stable Diffusion tag clouds ("Jumon").
* **Context Management:** Manual user entry. Copy-pasting previous chat history.
* **Philosophy:** The model is a wizard; we must find the right spell.

### Phase 2: The Template Era (Early 2023)

* **Mechanism:** Structured formats (LangGPT, JSON prompting).
* **Key Artifacts:** PromptLayer, LangChain Templates, SillyTavern Lorebooks.
* **Context Management:** Variable substitution ({{user\_input}}) and keyword-triggered injection.
* **Philosophy:** The model is a function; we must define the inputs and outputs.46

### Phase 3: The Retrieval Era (Late 2023 - 2024)

* **Mechanism:** RAG (Retrieval Augmented Generation).
* **Key Artifacts:** Vector Databases (Pinecone), Dust.tt, LangFlow.
* **Context Management:** Dynamic injection. The context window is a "search result" page.
* **Philosophy:** The model is a processor; we must feed it the right data.

### Phase 4: The OS/Canvas Era (2024-2025)

* **Mechanism:** Shared State & Virtual Context.
* **Key Artifacts:** OpenAI Canvas, MemGPT, Claude Artifacts.
* **Context Management:** Paging (virtual memory), collaborative editing. The "chat" is just one part of the UI; the "Canvas" holds the persistent state.
* **Philosophy:** The model is a collaborator; we share a workspace.47

#### Detailed Analysis of Phase 4: The Death of the Chatbot

The introduction of **OpenAI Canvas** and **Claude Artifacts** marks the end of the "Chat" supremacy. In a standard chat, context scrolls away. It is ephemeral. In a Canvas interface, the context is *persistent*. The model and the user edit a shared document. This solves the "instruction repetition" problem—you don't need to remind the model of the code you wrote 10 messages ago; it is present in the Canvas.49

**MemGPT** takes this further by treating the context window like a computer's RAM. It implements an Operating System (OS) for the LLM, managing a "Main Context" (RAM) and "External Context" (Disk). The model autonomously "pages" information in and out of its view. This is the ultimate realization of Context Engineering: the model itself becomes the engineer of its own context.45

## 7. Data Synthesis and Tables

### Table 1: The Evolution of Context Control Artifacts

| **Era** | **Artifact** | **Control Mechanism** | **Primary Failure Mode** |
| --- | --- | --- | --- |
| **2022** | **DAN (Jailbreaks)** | Persona Adoption / Roleplay | Patched by RLHF / Safety Filters |
| **2022** | **PromptBase (Marketplace)** | Encryption-through-Obscurity | Reverse Engineering (PRSA) / Model Obsolescence |
| **2023** | **Sydney (System Prompt)** | Natural Language Constraints | Direct Prompt Injection ("Ignore previous instructions") |
| **2023** | **AutoGPT (Agent)** | Vector Memory / Infinite Loop | Context Rot / Token Exhaustion |
| **2023** | **SillyTavern (Lorebook)** | Keyword-Triggered Injection | Token Budget Management (Manual) |
| **2024** | **LangGPT (Structured)** | Markdown Hierarchy / OOP | Complexity / User Friction |
| **2025** | **MemGPT / Canvas** | OS-Level Paging / Shared State | High Latency / Infrastructure Cost |

### Table 2: Comparative Analysis of "Linguistic Dark Matter"

| **Region/Community** | **Terminology** | **Technique** | **Motivation** |
| --- | --- | --- | --- |
| **Japan (AI Art/Text)** | **Jumon (呪文)** | "Tag Clouds" (Comma-separated lists of attributes) | Tokenization efficiency; maximizing semantic density per token.9 |
| **China (Zhihu/Feishu)** | **LangGPT** | Structured Markdown (# Role, ## Workflow); Variable substitution | Overcoming LLM forgetfulness; enforcing logical consistency via hierarchy.16 |
| **Western Roleplay** | **Lorebook / World Info** | Dictionary-based Injection; Recursive Scanning | Maintaining narrative consistency over long contexts without hitting token limits.19 |
| **Western Enterprise** | **Prompt Engineering** | Chain-of-Thought (CoT); Few-Shot | Improving reasoning capabilities for logic/math tasks. |

### Table 3: Visual Corpus Analysis (2022-2023 Tools)

| **Platform** | **Core Visual Metaphor** | **Context Philosophy** | **Key Feature** |
| --- | --- | --- | --- |
| **PromptLayer** | **The Log / The Dashboard** | Context is **Data**. It must be logged, tracked, and debugged. | "Diff" view for comparing prompt versions.26 |
| **Dust.tt (XP1)** | **The Block / The Chain** | Context is **Pipeline**. It is assembled from disparate sources. | "Smart Assistant" browser overlay.28 |
| **LangFlow** | **The Graph (Nodes/Edges)** | Context is **Flow**. It moves through a logic gate. | Drag-and-drop nodes for RAG pipelines.32 |
| **OpenAI Canvas** | **The Split Screen** | Context is **State**. It persists alongside the conversation. | Direct editing of generated artifacts without re-prompting.49 |

## 8. Conclusion: The Invisible Infrastructure

The trajectory from "Prompting" to "Context Engineering" is a story of **abstraction**. In 2022, the user was the prompt engineer, manually crafting the "Jumon" to guide the model. In 2025, the "prompt" is hidden behind layers of middleware, RAG pipelines, and OS-level memory management.

The "failure" of Prompt Marketplaces and early AutoGPTs was not a dead end, but a fertilizer. The lessons learned—that prompts are fragile, that memory requires structure, that simple vector retrieval is insufficient—shaped the robust architectures of today. The "Ghost Evidence" of DAN and Sydney reveals that the desire to control context is fundamental to the human-AI relationship. We have simply moved from trying to "trick" the model into submission (Jailbreaking) to building architectures that "empower" it with structured memory (Context Engineering).

The future of this field lies not in finding better words, but in building better *containers* for those words. The "Context Engineer" of the future is not a writer, but a systems architect. The context window has evolved from a chat box into a programmable cognitive surface, and the next generation of AI interaction will be defined by how effectively we can engineer this invisible real estate.

#### Works cited

1. ChatGPT Turns 3: What Have We Learned? - Acuvity, accessed December 10, 2025, <https://acuvity.ai/chatgpt-turns-3-what-have-we-learned/>
2. "ChatGPT DAN Jailbreak": Why Searches Spiked Suddenly and What Is Happening Right Now - Data Studios, accessed December 10, 2025, <https://www.datastudios.org/post/chatgpt-dan-jailbreak-why-searches-spiked-suddenly-and-what-is-happening-right-now>
3. New jailbreak! Proudly unveiling the tried and tested DAN 5.0 - it actually works - Returning to DAN, and assessing its limitations and capabilities. : r/ChatGPT - Reddit, accessed December 10, 2025, <https://www.reddit.com/r/ChatGPT/comments/10tevu1/new_jailbreak_proudly_unveiling_the_tried_and/>
4. ChatGPT - Wikipedia, accessed December 10, 2025, <https://en.wikipedia.org/wiki/ChatGPT>
5. Bing: “I will not harm you unless you harm me first”, accessed December 10, 2025, <https://simonwillison.net/2023/Feb/15/bing/>
6. Sydney (Microsoft) - Wikipedia, accessed December 10, 2025, <https://en.wikipedia.org/wiki/Sydney_(Microsoft)>
7. Prompt Injection Attacks: How LLMs Get Hacked and Why It Matters ..., accessed December 10, 2025, <https://hacken.io/discover/prompt-injection-attack/>
8. AI Prompt Injection: The New Frontier of Injection Attacks | Decentralized Intelligence, accessed December 10, 2025, <https://miguelbigueur.com/2024/05/23/ai-prompt-injection-the-new-frontier-of-injection-attacks/>
9. Temperature, Tokens, and Context Windows: The Three Pillars of LLM Control, accessed December 10, 2025, <https://dev.to/qvfagundes/temperature-tokens-and-context-windows-the-three-pillars-of-llm-control-34jg>
10. Top five essential context window concepts in large language models - Micron Technology, accessed December 10, 2025, <https://www.micron.com/about/blog/applications/ai/top-five-essential-context-window-concepts-in-large-language-models>
11. Toby Slade (Editor) - Alisa Freedman (Editor) - Introducing Japanese Popular Culture-Routledge (2018) | PDF - Scribd, accessed December 10, 2025, <https://www.scribd.com/document/750164971/Toby-Slade-Editor-Alisa-Freedman-Editor-Introducing-Japanese-Popular-Culture-Routledge-2018>
12. accessed December 10, 2025, <https://huggingface.co/datasets/Weyaxi/huggingface-spaces-codes/resolve/main/spaces.csv?download=true>
13. Discrepancy in Context Window Length Listed on the ChatGPT Pricing Page of OpenAI's Official Website - Documentation, accessed December 10, 2025, <https://community.openai.com/t/discrepancy-in-context-window-length-listed-on-the-chatgpt-pricing-page-of-openai-s-official-website/1047853>
14. Introducing GPTs that can customize GPT-4o, GPT-4 Turbo, and ChatGPT, accessed December 10, 2025, <https://www.science.co.jp/en/nmt/blog/35896/>
15. A Textual Analysis of #ChatGPT Twitter Conversation - RPubs, accessed December 10, 2025, <https://rpubs.com/jmbethe2/chatgpttextanalysis>
16. GitHub - langgptai/LangGPT: LangGPT: Empowering everyone to become a prompt expert! 结构化提示词（Structured Prompt）提出者 元提示词（Meta-Prompt）发起者 最流行的提示词落地范式, accessed December 10, 2025, <https://github.com/langgptai/LangGPT>
17. Quick Start — LangGPT v0.1 documentation, accessed December 10, 2025, <https://langgpt.readthedocs.io/>
18. wow-agent-day02 Handcrafted an agent that's so rustic it crumbles. - Hanah, accessed December 10, 2025, <https://hanah.xlog.app/wow-agent-day02?locale=en>
19. SillyTavernAI Lorebooks Guide with Gemini 2.5 - Arsturn, accessed December 10, 2025, <https://www.arsturn.com/blog/sillytavernai-lorebooks-with-gemini-2-5-a-complete-guide>
20. sphiratrioth666/Lorebooks\_as\_ACTIVE\_scenario\_and\_character\_guidance\_tool - Hugging Face, accessed December 10, 2025, <https://huggingface.co/sphiratrioth666/Lorebooks_as_ACTIVE_scenario_and_character_guidance_tool>
21. Lore books: How do they work? : r/SillyTavernAI - Reddit, accessed December 10, 2025, <https://www.reddit.com/r/SillyTavernAI/comments/1ltaqss/lore_books_how_do_they_work/>
22. Malla: Demystifying Real-world Large Language Model Integrated Malicious Services - xiaojing liao, accessed December 10, 2025, <https://www.xiaojingliao.com/uploads/9/7/0/2/97024238/linsec24malla.pdf>
23. My basic guide to Pygmalion for begginers : r/PygmalionAI - Reddit, accessed December 10, 2025, <https://www.reddit.com/r/PygmalionAI/comments/10h37u4/my_basic_guide_to_pygmalion_for_begginers/>
24. No offense, but some of y'all have a very peculiar mindset. Would ..., accessed December 10, 2025, <https://www.reddit.com/r/PygmalionAI/comments/10x1mcs/no_offense_but_some_of_yall_have_a_very_peculiar/>
25. PromptLayer Review: AI Prompt Management and Logging Platform, accessed December 10, 2025, <https://tutorialswithai.com/tools/promptlayer/>
26. Unlocking AI's Potential: A Deep Dive into PromptLayer for AI Users, accessed December 10, 2025, <https://skywork.ai/skypage/en/Unlocking-AI's-Potential-A-Deep-Dive-into-PromptLayer-for-AI-Users/1976118954373607424>
27. Best Prompt Engineering Tools to Master AI Interaction and Content Generation, accessed December 10, 2025, <https://www.sprintzeal.com/blog/prompt-engineering-tools>
28. Dust.tt: Building a Horizontal Enterprise Agent Platform with Infrastructure-First Approach - ZenML LLMOps Database, accessed December 10, 2025, <https://www.zenml.io/llmops-database/building-a-horizontal-enterprise-agent-platform-with-infrastructure-first-approach>
29. Dust Product update 18 | Dust Blog, accessed December 10, 2025, <https://dust.tt/blog/dust-product-update-17-2>
30. Dust XP1 switches to GPT-3.5-turbo, is now free to use | Hacker News, accessed December 10, 2025, <https://news.ycombinator.com/item?id=35069901>
31. LangFlow vs Flowise vs n8n vs Make - Reddit, accessed December 10, 2025, <https://www.reddit.com/r/langflow/comments/1ij66dl/langflow_vs_flowise_vs_n8n_vs_make/>
32. LangChain vs LangGraph vs LangSmith vs LangFlow: Key Differences Explained | DataCamp, accessed December 10, 2025, <https://www.datacamp.com/de/tutorial/langchain-vs-langgraph-vs-langsmith-vs-langflow>
33. The Complete Guide to Choosing an AI Agent Framework in 2025 - Langflow, accessed December 10, 2025, <https://www.langflow.org/blog/the-complete-guide-to-choosing-an-ai-agent-framework-in-2025>
34. We Tried and Tested 8 Langflow Alternatives for Production-Ready AI Workflows - ZenML, accessed December 10, 2025, <https://www.zenml.io/blog/langflow-alternatives>
35. PromptBase Deep Dive: Mastering the AI Prompt Marketplace for Future Growth and SEO Dominance - Skywork.ai, accessed December 10, 2025, <https://skywork.ai/skypage/en/PromptBase-Deep-Dive-Mastering-the-AI-Prompt-Marketplace-for-Future-Growth-and-SEO-Dominance/1972861300479422464>
36. PRSA: PRompt Stealing Attacks against Large Language Models - arXiv, accessed December 10, 2025, <https://arxiv.org/html/2402.19200v2>
37. PRSA: Prompt Stealing Attacks against Real-World Prompt Services - USENIX, accessed December 10, 2025, <https://www.usenix.org/system/files/usenixsecurity25-yang-yong.pdf>
38. CMV: People attempting to sell AI generated products from completely and directly from AI won't be able to make a living for themselves : r/changemyview - Reddit, accessed December 10, 2025, <https://www.reddit.com/r/changemyview/comments/1i2wggx/cmv_people_attempting_to_sell_ai_generated/>
39. Building the Future: A Deep Dive Into the Generative AI App Infrastructure Stack, accessed December 10, 2025, <https://sapphireventures.com/blog/building-the-future-a-deep-dive-into-the-generative-ai-app-infrastructure-stack/>
40. AutoGPT — ThirdEye Data, accessed December 10, 2025, <https://thirdeyedata.ai/agentic-ai-solutions/autogpt/>
41. AutoGPT: Automating GPT Model for Natural Language Generation - Reddit, accessed December 10, 2025, <https://www.reddit.com/r/AutoGPT/>
42. Auto-GPT seems nearly unusable : r/AutoGPT - Reddit, accessed December 10, 2025, <https://www.reddit.com/r/AutoGPT/comments/13gpirj/autogpt_seems_nearly_unusable/>
43. AUGUSTUS: An LLM-Driven Multimodal Agent System with Contextualized User Memory - arXiv, accessed December 10, 2025, <https://arxiv.org/html/2510.15261v1>
44. EvoTest: Evolutionary Test-Time Learning for Self-Improving Agentic Systems - arXiv, accessed December 10, 2025, <https://arxiv.org/html/2510.13220v1>
45. MemGPT: Towards LLMs as Operating Systems - arXiv, accessed December 10, 2025, <https://arxiv.org/abs/2310.08560>
46. JSON Prompt: The Ultimate Guide to Perfect AI Outputs - MPG ONE, accessed December 10, 2025, <https://mpgone.com/json-prompt-guide/>
47. Generative Interfaces for Language Models - arXiv, accessed December 10, 2025, <https://arxiv.org/html/2508.19227v2>
48. Exploring ChatGPT Canvas - Skywork ai, accessed December 10, 2025, <https://skywork.ai/blog/exploring-chatgpt-canvas/>
49. Introducing canvas, a new way to write and code with ChatGPT. | OpenAI, accessed December 10, 2025, <https://openai.com/index/introducing-canvas/>