Member-only story

AGENTIC AI | LLMS | AGENTS | SELF LEARNING | NO FINE-TUNING NEEDED

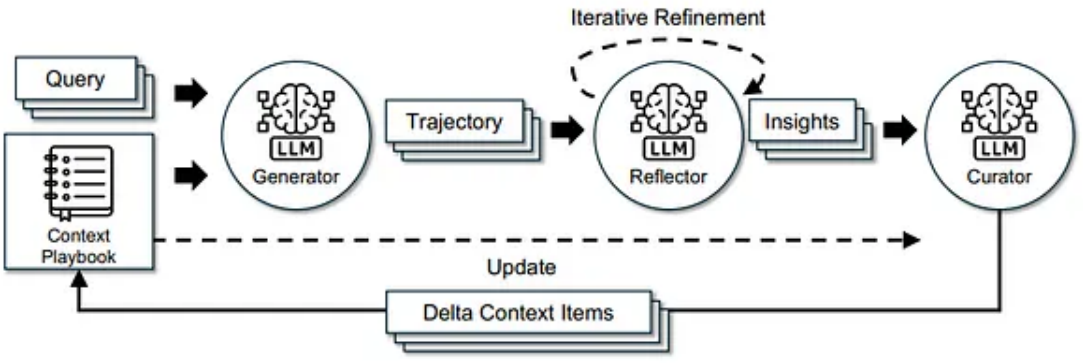
Agentic Context Engineering: A Framework for LLMs That Learn Without Forgetting:Paper review

What if machines could reshape their own thinking to grow smarter over time? This review dives into a bold new idea that challenges how we train and trust AI.

11 min read · 3 days ago

Chinmay Bhalerao Follow

Listen Share More



Recently, I came across a research paper titled *Agentic Context Engineering: Evolving Contexts for Self-Improving Language Models.* At first, the title sounded a bit technical, but as I read more, I found the idea quite interesting.

The paper was written by a team of researchers from **Stanford University**, **UC Berkeley**, and **SambaNova Systems.** The lead authors are **Qizheng Zhang** and

Changran Hu, who contributed equally to the work. Other contributors include **Shubhangi** **y Rainton**, **Chen Wu**, **unle Olukotun**.

Together, they explore how AI models can improve themselves by evolving the way they use context moving beyond just following instructions to actively shaping their own learning process.

The development of Large Language Model (LLM) agents capable of autonomous self-improvement represents a critical frontier in artificial intelligence. The dominant paradigm for achieving this involves iterative context adaptation, a process where an agent refines its operational instructions or external memory based on performance feedback. However, this approach is fundamentally flawed. Practitioners frequently observe a perplexing phenomenon: after an initial period of improvement, these agents often begin to degrade, seemingly "forgetting" previously learned lessons and regressing in capability.

This degradation is not a random artifact but a direct consequence of systemic flaws in current adaptation methodologies. Two primary failure modes have been identified: **Brevity Bias**, an overemphasis on concise instructions at the expense of necessary detail, and **Context Collapse**, a catastrophic information loss that occurs during context rewriting.

> ***Brevity** means expressing something **clearly and concisely**, using **as few words as necessary** without losing the meaning. In simpler terms, it's the art of **saying more with less***
>
> ***Brevity bias** is a cognitive bias where people tend to **favor shorter, more concise explanations or messages**, even if longer ones might be more accurate or complete.*

A new framework detailed in the research paper "**Agentic Context Engineering: Evolving Contexts for Self-Improving Language Models**" offers a robust architectural solution. This framework, named ACE, re-conceptualizes an agent's context from a static, optimizable prompt into a dynamic, structured "playbook." This shift in paradigm allows for consistent, cumulative learning and effectively mitigates the catastrophic forgetting that plagues current systems.

This article provides a detailed technical analysis of context degradation, explores the archite[illegible]es, and positions A[illegible]

The Core Problem: How Self-Adaptation Induces Context Degradation

The promise of self-improving agents is that they learn from experience. Most modern frameworks implement this via a loop: an agent performs a task, its execution trace is evaluated, and an LLM is prompted to update the agent's core instructions or memory for the next attempt. While theoretically sound, this mechanism is highly susceptible to failure.

1. Brevity Bias: The Peril of Oversimplification

Prompt engineering best practices have long advocated for conciseness. This has led to the emergence of a "brevity bias" in many optimization frameworks, which aim to distill complex strategies into the shortest possible instructions. This process systematically strips out the granular, domain-specific knowledge that underpins true expertise.

Consider a financial analysis agent. A biased optimization process might produce a concise instruction like:

> "Extract key financial metrics from the report."

While short, this is operationally useless in a real-world scenario. A far more effective, albeit verbose, instruction would be:

> "When extracting 'Net Revenue,' always locate the 'Consolidated Statements of Operations' table. Cross-reference with footnote to identify and exclude non-recurring revenue from asset sales. Ensure the value matches the figure reported in the quarterly summary text."

Brevity bias purges these essential details, creating agents that are competent at general tasks but fail when confronted with the nuanced complexities of a specific

domain.

2. Context

Context collapse is a more severe and abrupt failure mode. It arises when an LLM is tasked with monolithically rewriting a large, detailed context. The inherent training objectives of most LLMs predispose them to summarization and compression when rewriting text. Instead of carefully integrating new information while preserving existing detail, the model often generates a much shorter, less informative summary.

Over multiple adaptation cycles, this is devastating. The ACE researchers observed this in a controlled experiment:

1. An agent's context grew organically to **18,282 tokens**, achieving a task accuracy of **66.7%**.
2. In the next adaptation step, a monolithic rewrite was triggered. The LLM compressed the context down to a mere **122 tokens**.

Image credits: original paper- Agentic context engineering [1]

3. Task accuracy immediately plummeted to **57.1%**, a score significantly worse than the baseline performance achieved with no adaptation at all.

This is not just optimization, it is an unintentional lobotomy, where the agent's accumulated knowledge is abruptly erased.

3. The Scalability and Cost Barrier of Monolithic Rewrites

Beyond accuracy degradation, monolithic rewriting is a significant engineering bottleneck. Each rewrite of a large context (e.g., 10,000+ tokens) is a computationally expensive and high-latency operation. As contexts grow, the cost and time required for each adaptation step become prohibitive, making the approach unscalable for real-time or resource-constrained applications. This practical barrier further highlights the need for a more efficient update mechanism.

The Inefficiency of Brute-Force Context Expansion

A common response to these issues is to point towards the rapidly expanding context windows of modern LLMs. The argument suggests that with multi-million token contexts, the need for careful context management is diminished. However, this perspective overlooks critical findings about how LLMs actually process information.

A pivotal Stanford study identified the **"Lost in the Middle"** phenomenon. It demonstrated that LLMs exhibit a U-shaped performance curve when retrieving information from long contexts: they are highly effective at recalling information from the very beginning and very end of the context, but their performance drops significantly when the target information is located in the middle by Liu, et al.,

2023.,“**Lost in the Middle: How Language Models Use Long Contexts**” [Link is in references

This has profound implications. It means that simply “stuffing” more information into the context is not a reliable solution. It can lead to **attentional dilution**, where the model’s attention mechanism is spread so thinly across a vast number of tokens that the signal from the most critical information is weakened by foundation models literature on attention mechanisms.

Effective context management is therefore not about size, but about structure and accessibility. A well-organized, curated **15,000-token** playbook will consistently outperform a disorganized, **1,000,000-token** text dump because the critical information is structured to be easily found and utilized.

The ACE Framework: A Modular Architecture for Robust Learning

ACE is designed to solve these problems by implementing a modular, agentic architecture that decouples the core processes of execution, analysis, and knowledge integration.

The Generator: This is the execution component. It takes the current task query and the full playbook (the structured context) as inputs. Its role is to synthesize these

sources to formulate and execute a plan. The complete execution trace — including its reasoni[illegible] logged.

The Reflector: This component is a dedicated analyst. It examines the Generator's execution trace and leverages a variety of feedback signals to distill high-level, actionable insights. These signals can include:

Execution Outcome: Binary success/failure signals (e.g., did the generated code compile and run, or did it produce a runtime error?).

Validation Results: Feedback from unit tests or other validation logic that checks the correctness of the final output.

Ground-Truth Comparison: In offline settings, the output can be compared directly against a known-correct solution.

Heuristic Signals: Detection of undesirable behaviors, such as infinite loops, repeated errors, or the use of deprecated functions.

This principle of separating execution from critique is a powerful pattern for enabling robust reasoning, also seen in other advanced agent frameworks by Shinn, et al., 2023. "**Reflexion: Language Agents with Verbal Reinforcement Learning**" [3].

The Curator: This component acts as the knowledge base manager. It receives the structured insights from the Reflector and integrates them into the playbook. Critically, this integration is performed using **deterministic, non-LLM logic**. This is a crucial design choice. Using an LLM for curation would re-introduce the risk of non-determinism and the very summarization behavior ACE is designed to prevent. The Curator performs simple, reliable operations like appending a new insight or updating a metadata counter.

Key Technical Innovations of ACE

The ACE architecture is powered by two core mechanisms that directly counteract the failure modes of previous systems.

1. Incremental Delta Updates

To eliminate context collapse, ACE replaces monolithic rewrites with **incremental "delta" updates**. The context is represented as a collection of structured, itemized "bullets," each containing:

Metadata: A unique identifier and counters tracking how often it has been marked as "helpful" or "harmful" by the Reflector.

Content: A small, atomic unit of knowledge, such as a reusable strategy, a domain concept, or a common failure mode.

The Curator integrates new insights by creating new bullets or updating the metadata of existing ones. This guarantees that the vast repository of prior knowledge is preserved, allowing the playbook to grow in value over time.

2. The Grow-and-Refine Principle

To combat brevity bias, ACE operates on a "grow-and-refine" principle. The default behavior is ing the playbook t eriodic maintenance process to manage redundancy. It works as follows:

1. Vector embeddings are generated for the content of each bullet in the playbook.
2. These embeddings are compared for semantic similarity (e.g., using cosine similarity).
3. If two bullets are found to be highly similar, they can be merged, or the one with lower performance (based on its helpful/harmful metadata counters) can be pruned.

This ensures the playbook remains efficient without aggressively compressing away valuable, nuanced information.

Operational Modes: Offline and Online Adaptation

The ACE framework is flexible and can be deployed in two distinct modes, catering to different application needs.

Offline Adaptation: This mode functions as a "training phase" for the agent. The system is run over a training dataset for multiple epochs, allowing the playbook to be iteratively built and refined into a highly optimized, domain-specific knowledge base. The final, mature playbook is then deployed with the agent for inference on unseen test data. This is ideal for creating expert agents for well-defined domains before deployment.

Online Adaptation: This mode is a form of "test-time learning." The agent begins with a minimal (or empty) playbook and updates it sequentially as it processes new, live data. This allows the agent to adapt in real-time to changing environments, new patterns, or evolving task requirements. While highly responsive, this mode requires reliable, real-time feedback signals to prevent the playbook from being polluted by spurious insights.

Empirical Results and Broader Implications

The ACE framework delivers significant performance improvements. The paper's evaluation reports average gains of **+10.6% on agent benchmarks (AppWorld)** and **+8.6% on domain-specific financial benchmarks (FiNER, Formula)** over strong baselines.

The most compelling result comes from the AppWorld leaderboard. An agent using ACE with t

Image credits: original paper- Agentic context engineering [1]

model achieved performance that matched, and on the more difficult "test-challenge" split surpassed, the top-ranked agent powered by the much larger, production-level GPT-4.1. This provides strong evidence that a superior learning architecture can be a more significant driver of performance than raw model scale alone.

Image credits: original paper- Agentic context engineering [1]

The framev , ACE achieved an **82.3% reduction in latency** and required **75.1% fewer rollouts** than the baseline. In online adaptation on FiNER, the gains were even more dramatic, with a **91.5% reduction in latency** and an **83.6% reduction in token cost**.

Conclusion: A New Paradigm for Self-Improving Systems

Longer Context Doesn't Mean Higher Cost

Even though ACE uses longer input contexts than some other methods (like GEPA), it doesn't mean it costs a lot more to run. Modern systems are getting better at handling long inputs efficiently. They do this by:

> ***Reusing*** *parts of the input that are used often,*
>
> ***Compressing*** *data to save space,*
>
> ***Offloading*** *some memory to other places.*

These tricks help avoid repeating the same work and reduce the cost of using long inputs. As technology improves, using long contexts (like ACE does) will become even more practical and affordable.

Useful for Online and Continuous Learning

In machine learning, **online and continuous learning** help models adapt to new data or changes over time. ACE is a good fit for this because:

It's cheaper to **change the input context** than to retrain the whole model.The context is **easy for humans to understand**, so it's easier to remove or update specific information — like for privacy reasons or when correcting mistakes. This makes ACE a flexible and efficient tool for keeping models up to date.

The persistent challenge of creating LLM agents that learn and improve without degradation cannot be solved by simply increasing model size or context length. The limitations are fundamentally architectural. The common practices of prompt optimization often introduce a harmful Brevity Bias, while monolithic context rewriting leads to catastrophic Context Collapse.

Agentic Context Engineering (ACE) presents a new paradigm. By transitioning from static prom ental updates, ar ACE provides a reliable framework for cumulative learning. It demonstrates a path forward for building more capable, efficient, and genuinely self-improving AI systems that can acquire and retain domain expertise over time.

References

[1] Agentic Context Engineering: Evolving Contexts for Self-Improving Language Models : *The original paper*

[2] Lost in the Middle: How Language Models Use Long Contexts: *The original paper*

[3] Reflexion: Language Agents with Verbal Reinforcement Learning: *The original paper*

How do you see this modular, playbook-driven approach being applied in your own work with LLM agents or complex AI systems?

If you find this article useful

It is a proven fact that "**Generosity makes you a happier person**"; therefore, give claps to the article if you liked it. If you found this article insightful, follow me on **LinkedIn** and **Medium**. You can also **subscribe** to get notified when I publish articles. Let's create a community! Thanks for your support!

If you have found this article insightful and want to join the conversation on building reliable AI, then follow me. I share pragmatic insights for builders in the AI space. You can also subscribe to get my articles delivered directly to your inbox.

Here are a few other posts to guide you on your way:

The Hidden Dangers of "Vibe-Driven" Development

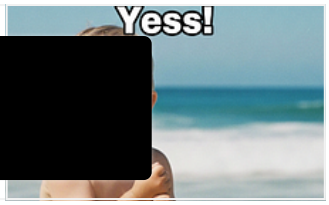
Beneath the surface of spontaneous coding lies a trail of bugs, burnout, and broken architecture

medium.com

Basics of MCPs. Why and what !

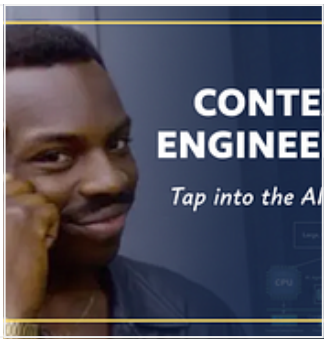
Understanding basics of MCPs and why they came in picture when we have su

medium.com



“Prompt Engineering” is Dead. The Future is Context Engineering.

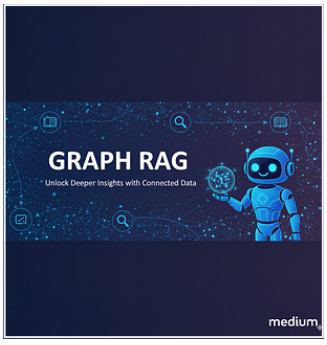
medium.com



Why Graph RAG Matters? All about Graph RAG

Limitations of production level vanilla RAG systems, emergence of Graph structures and linking RAG system with graph...

medium.com



Signing off,

Chinmay !

Fine Tuning | Large Language Models | LIm Applications | Artificial Intelligence | AI

Follow

Published in Data And Beyond

1.3K followers · Last published 9 hours ago

Selected stories around Data Science, Machine Learning, Artificial Intelligence, Programming, and Technology topics. Writing guide: https://medium.com/data-and-beyond/how-to-write-for-data-and-beyond-b83ff0f3813e

llow

Written by Chinmay Bhalerao

1.8K followers · 124 following

Senior AI Engineer | 3X Top Writer in AI, Computer Vision & Object Detection | Generative AI, RAG & Fine-Tuning | Making AI work for everyone, not just experts.

Responses (2)

Bgerby

What are your thoughts?

Robert Jr AI
12 hours ago

Nice to read , I have few question, let me ask you in personal .

Reply

Prakash Gupta
12 hours ago

Really amazing ! never thought of this !

Reply

More from Chinmay Bhalerao and Data And Beyond

In Data And Beyond by Chinmay Bhalerao

RAG is Not Enough: Why Your Next AI Project Demands Structured Data RAG

The FAST-RAG system without complex embedding models or vector databases

Jul 9 788 13

In Data And Beyond by Shahzaib

You NEED to Use n8n RIGHT NOW!! (Free, Local, Private)

The Ultimate Guide to Unleashing Your Inner Automation Genius Without Spending a Dime

Sep 7

In Data And Beyond by TONI RAMCHANDANI

IBM’s Granite Docling 258M & Its DocTag Revolution: The Model That Doesn’t Flatten Your Data

A storytelling journey into how IBM turned vision, language, and structure into a layout-preserving AI built for the RAG era

Sep 24 147 2

In Data And Beyond by Chinmay Bhalerao

Why Graph RAG Matters? All about Graph RAG

Limitations ... s and linking RAG system...

Sep 23 258 4

See all from Chinmay Bhalerao

See all from Data And Beyond

Recommended from Medium

In Level Up Coding by Fareed Khan

Building an Agentic Deep-Thinking RAG Pipeline to Solve Complex Queries

Planning, Retrieval, Reflection, Critique, Synthesis and more

4d ago

Dr Nicolas Figay

Knowledge Graphs and Ontologies: Beyond the Dictionary Fallacy

Most knowledge graph practitioners treat ontologies as sophisticated dictionaries—structured vocabularies and entity hierarchies...

6d ago 87 2

In AI Advances by Debmalya Biswas

Adding Empathy to Agentic AI

Fine-tuning AI Agents based on User Personas to improve their Empathy Quotient

5d ago

Syma Sultana

Building a RAG System: Understanding Queue Architecture and Vector Indexing

Introduction to RAG Systems

3d ago

Maninder Singh

Building V

With enough … on large benchmarks. I wanted to understand why—and more...

5d ago 260 3

In Towards AI by Ahmed Boulahia

Best Open-Source Embedding Models for RAG

High-Performance Open-Source Embedding Models for RAG Pipelines, Multilingual NLP, and Arabic Text

Oct 14 221

See more recommendations