Agentic Context Engineering: Evolving Contexts for Self-Improving Language Models

2025.11.18

송건학

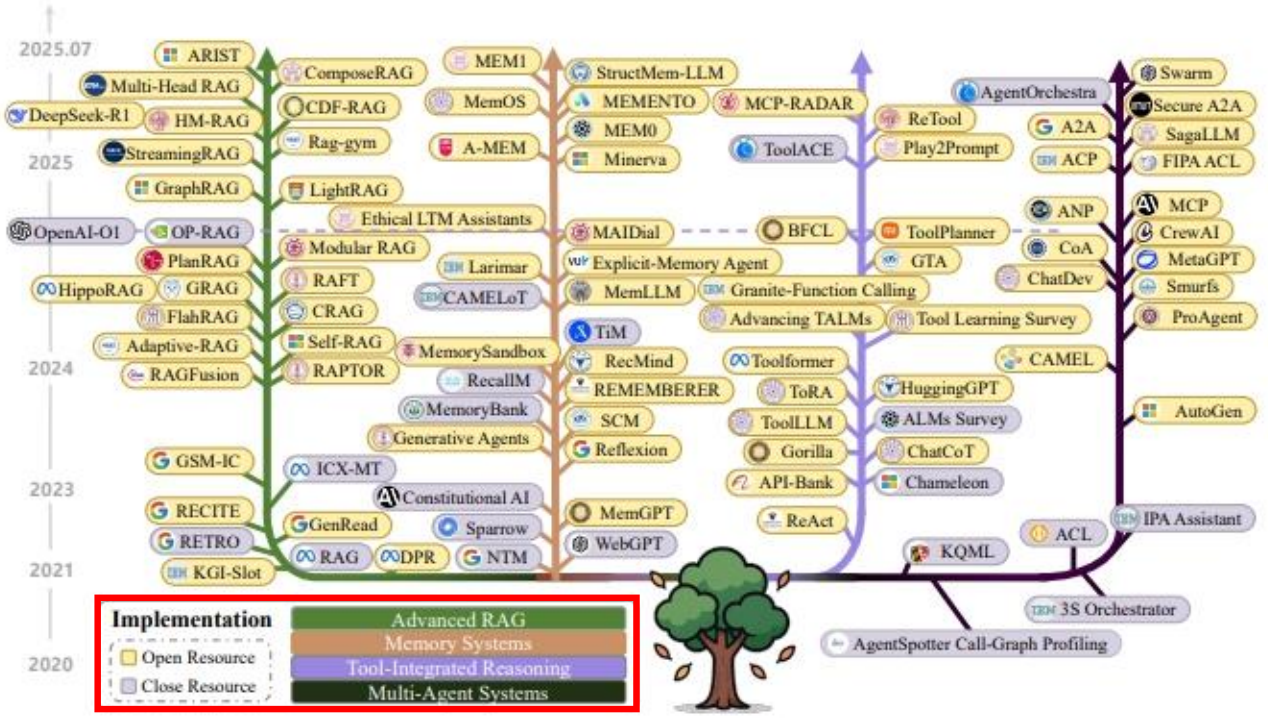
https://huggingface.co/papers/2510.04618

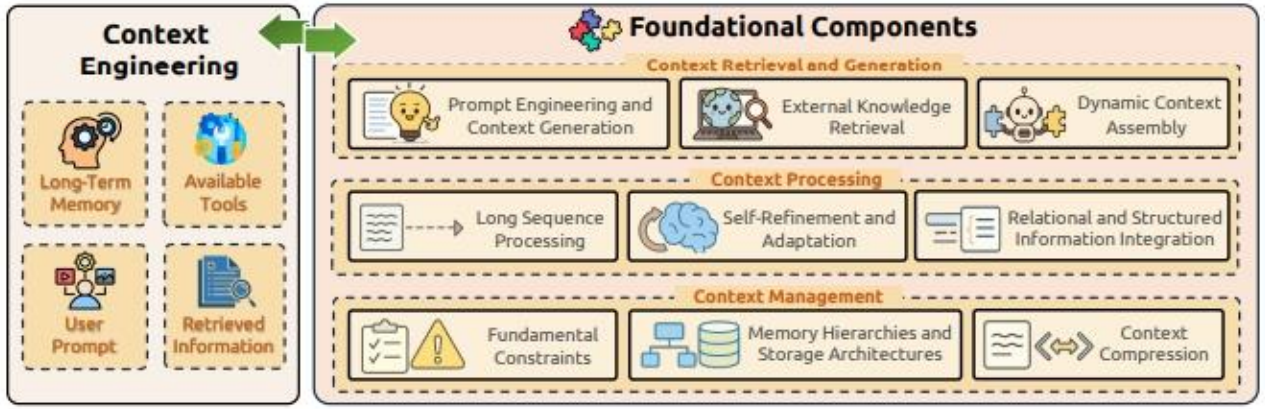
https://arxiv.org/abs/2510.04618

Table of Contents

Context Adaptation (Context Engineering)

- **Context Engineering**
  - **Foundational Components (§4)**
    - Context Generation Retrieval & (§4.1) — *e.g.*, Chain-of-Thought [1147], Zero-shot CoT [559], ToT [1255], GoT [69], Self-consistency [1123], ReAct [1254], Auto-CoT [1107], Automatic Prompt [311] , CLEAR Framework [708], RAG [597], Cognitive Prompting [564], KAPING [48], Dynamic Assembly [311], *etc.*
    - Context Processing (§4.2) — *e.g.*, Mamba [1267], LongNet [220], FlashAttention [200], Ring Attention [682], YaRN [839], Infini-attention [798], StreamingLLM [1185], InfLLM [1184], Self-Refine [741], Reflexion [964], StructGPT [495], GraphFormers [1230], KG Integration [1330], Long CoT [148], MLLMs [49], *etc.*
    - Context Management (§4.3) — *e.g.*, Context Compression [321], StreamingLLM [1185], KV Cache Management [1399], Heavy Hitter Oracle [1343], Hierarchical Memory [505], Recurrent Context Compression [447], Activation Refilling [865], Context Window Management [1082], *etc.*
  - **Implementations (§5)**
    - Retrieval-Augmented Generation (§5.1) — *e.g.*, FlashRAG [506], KRAGEN [755], ComposeRAG [1168], Self-RAG [41], CDF-RAG [537], GraphRAG [378], LightRAG [364], HippoRAG [370], RAPTOR [936], RAG-Gym [1192], Agentic RAG Systems [973], Graph-Enhanced RAG [838], Modular RAG Architectures [316], *etc.*
    - Memory Systems (§5.2) — *e.g.*, MemoryBank [1372], MemLLM [785], Self-Controlled Memory [655], REMEMBERER [1308], MemOS [643], Charlie Mnemonic [584], RecMind [1124], Sandbox [461], LongMemEval [1180], MADail-Bench [390], MEMENTO [572], A-MEM [1211], CAMELoT [397], Architectures [1191], Short-term & Long-term Memory [943], MemGPT [819], Memory-Enhanced Agents [577], *etc.*
    - Tool-Integrated Reasoning (§5.3) — *e.g.*, Toolformer [939], ReAct [1254], Gorilla [834], ToolLLM [875], Granite-FunctionCalling [5], Program-Aided Language Models [309], ToRA [345], ReTool [274], Chameleon [715], a1 [766], API-Bank [621], MCP-RADAR [314], GTA benchmark [1098], PLAY2PROMPT [263], *etc.*
    - Multi-Agent Systems (§5.4) — *e.g.*, KQML [284], FIPA ACL [1155], MCP protocols [37], A2A [1015], ACP [468], ANP [1], AutoGen [1167], MetaGPT [412], CAMEL [606], CrewAI [188], Swarm Agent [814], 3S orchestrator [901], SagaLLM [128], Communication Protocols [1219], Orchestration [902], Coordination Strategies [631], Agent Communication Languages [360], CoA [1337], *etc.*
  - **Evaluation (§6)**
    - Evaluation Frameworks (§6.1) — *e.g.*, Component-Level Assessment [841], System-Level Integration [1141], Self-Refinement [741], MCP-RADAR [314], LongMemEval [1180], BFCL Tool Evaluation [835], SagaLLM [128], Brittleness Assessment [1268], Contextual Calibration [384], Multi-dimensional Feedback [288], *etc.*
    - Benchmark Datasets (§6.2) — *e.g.*, GAIA [778], GTA [1098], WebArena [1378], VideoWebArena [482], Deep Research Bench [87], StableToolBench [363], NesTools [377], ToolHop [1264], T-Eval [160], BFCL [835], NarrativeQA [556], MEMENTO [572], API-Bank [621], Mind2Web [206], SWE-Bench [500], *etc.*
    - Evaluation Challenges (§6.3) — *e.g.*, Performance Gap Assessment [778, 1098], Memory System Isolation Problems [1340, 1180], $O(n^2)$ Scaling Limitations [737, 299], Transactional Integrity [128], Multi-Tool Coordination [314], Self-Validation Dependencies [394], Context Handling Failures [214], Attribution Challenges [1122], Safety-oriented Evaluation [87], Agent Assessment [973], Orchestration Evaluation [901], *etc.*
  - **Future Directions & Challenges (§7)**
    - Foundational Research (§7.1) — *e.g.*, Theoretical Foundations [1141], Scaling Laws [737], $O(n^2)$ Computational Challenges [299], Multi-modal Integration [482], Compositional Understanding [841], Context Optimization [669], Frameworks for Multi-agent Coordination [128], Information-theoretic Analysis [314], *etc.*
    - Technical Innovation (§7.2) — *e.g.*, LongMamba [1267], Sliding Attention [299], Memory-Augmented Architectures [1372], Modular RAG [316], GraphRAG [378], Context Assembly Optimization [1141], Tool-Integrated Reasoning [314], Agentic Systems [973],Self-Refinement Mechanisms [741], *etc.*
    - Application-Driven Research (§7.3) — *e.g.*, Domain Specialization [87], Healthcare Applications [390], Protocol Standardization [250], MCP/A2A/ACP/ANP Protocols [622], Human-AI Collaboration [1378], Security Issues [934], Production Deployment Scalability [1236], Safety [973] and Ethical Considerations [841], *etc.*





https://arxiv.org/pdf/2507.13334

Context Adaptation (Context Engineering)

- **Context adaptation (or context engineering)** : methods that improve model behavior by constructing or modifying inputs to an LLM, rather than altering its weights.

- The current state of the art leverages ***natural language feedback***

- ***Methods***

  - **[23.03] Reflexion** : reflects on failures to improve agent planning

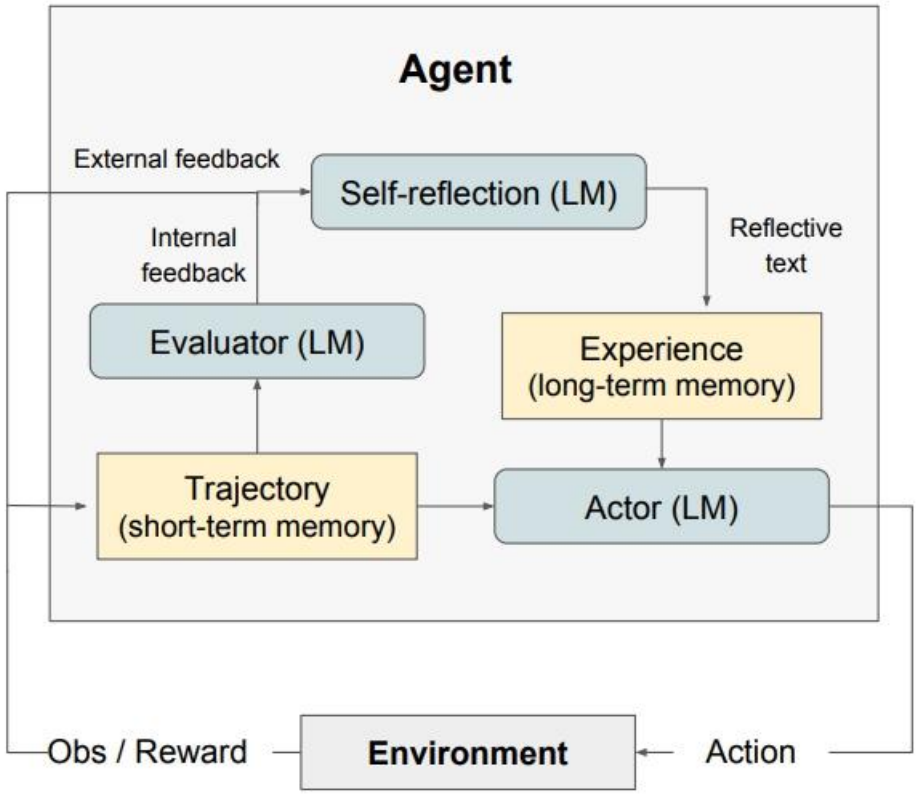
  - **[24.06] TextGrad :** optimizes prompts via gradient-like textual feedback

  - **[25.07] GEPA :** refines prompts iteratively based on execution traces and achieves strong performance, even surpassing reinforcement learning approaches in some settings

  - **[25.04] Dynamic Cheatsheet :** constructs an external memory that accumulates strategies and lessons from past successes and failures during inference.

Context Adaptation (Context Engineering)

- **[23.03] Reflexion** : reflects on failures to improve agent planning



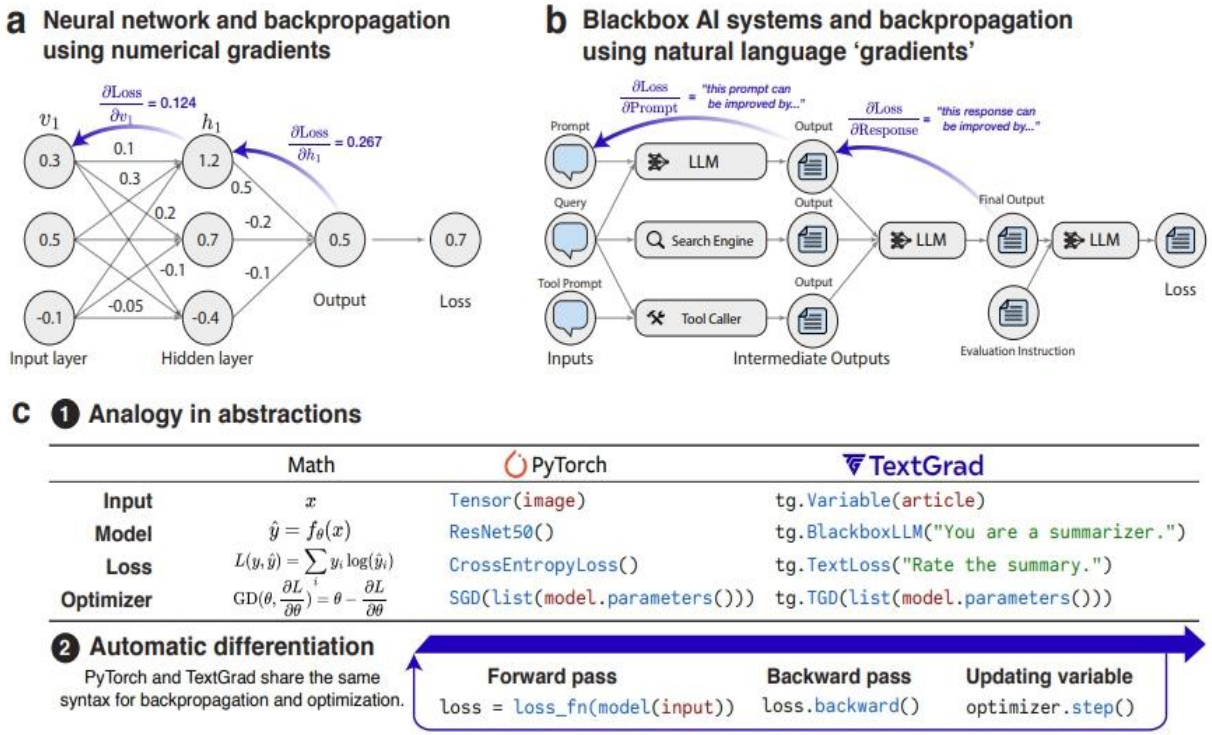
Algorithm 1 Reinforcement via self-reflection

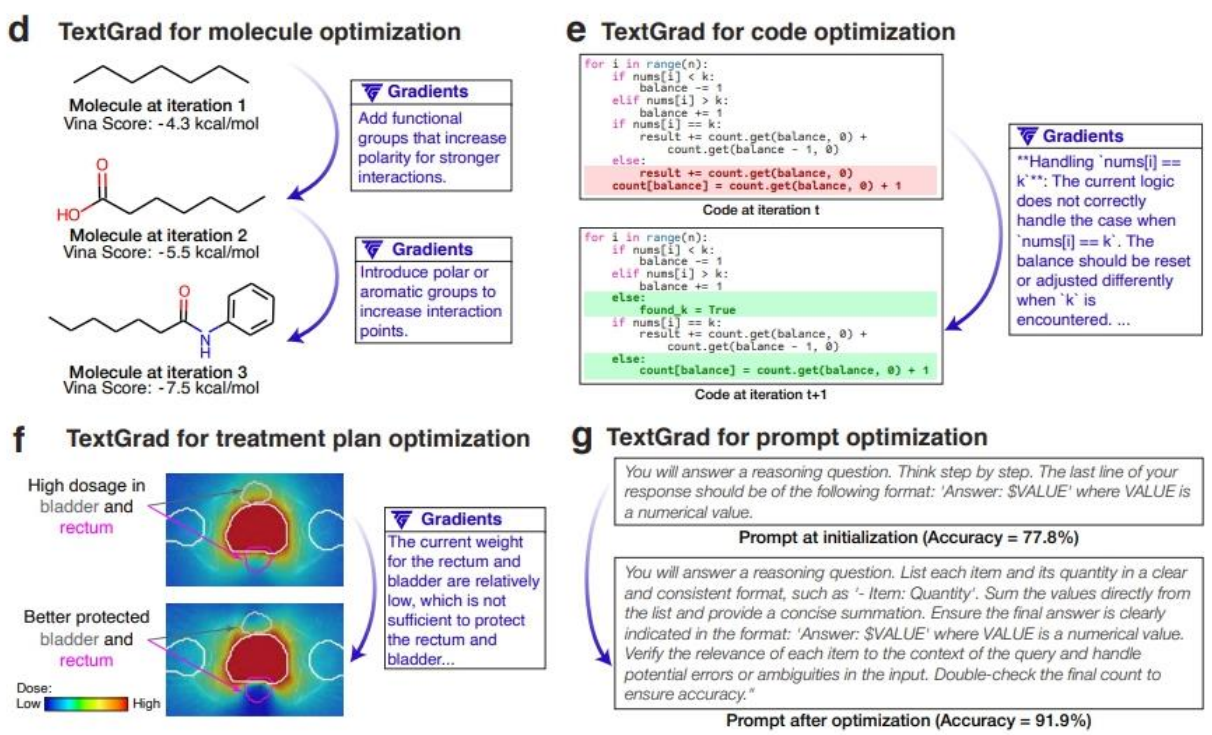
Initialize Actor, Evaluator, Self-Reflection: M_a, M_e, M_{sr}
Initialize policy $\pi_\theta(a_i|s_i), \theta = \{M_a, mem\}$
Generate initial trajectory using π_θ
Evaluate τ_0 using M_e
Generate initial self-reflection sr_0 using M_{sr}
Set $mem \leftarrow [sr_0]$
Set $t = 0$
while M_e not pass or $t <$ max trials **do**
    Generate $\tau_t = [a_0, o_0, \ldots a_i, o_i]$ using π_θ
    Evaluate τ_t using M_e
    Generate self-reflection sr_t using M_{sr}
    Append sr_t to mem
    Increment t
end while
return

Figure 2: (a) Diagram of Reflexion. (b) Reflexion reinforcement algorithm

Context Adaptation (Context Engineering)

- **[24.06] TextGrad :** optimizes prompts via gradient-like textual feedback





Context Adaptation (Context Engineering)

- **[25.07] GEPA :**
- refines prompts iteratively based on execution traces and achieves strong performance, even surpassing reinforcement learning approaches in some settings

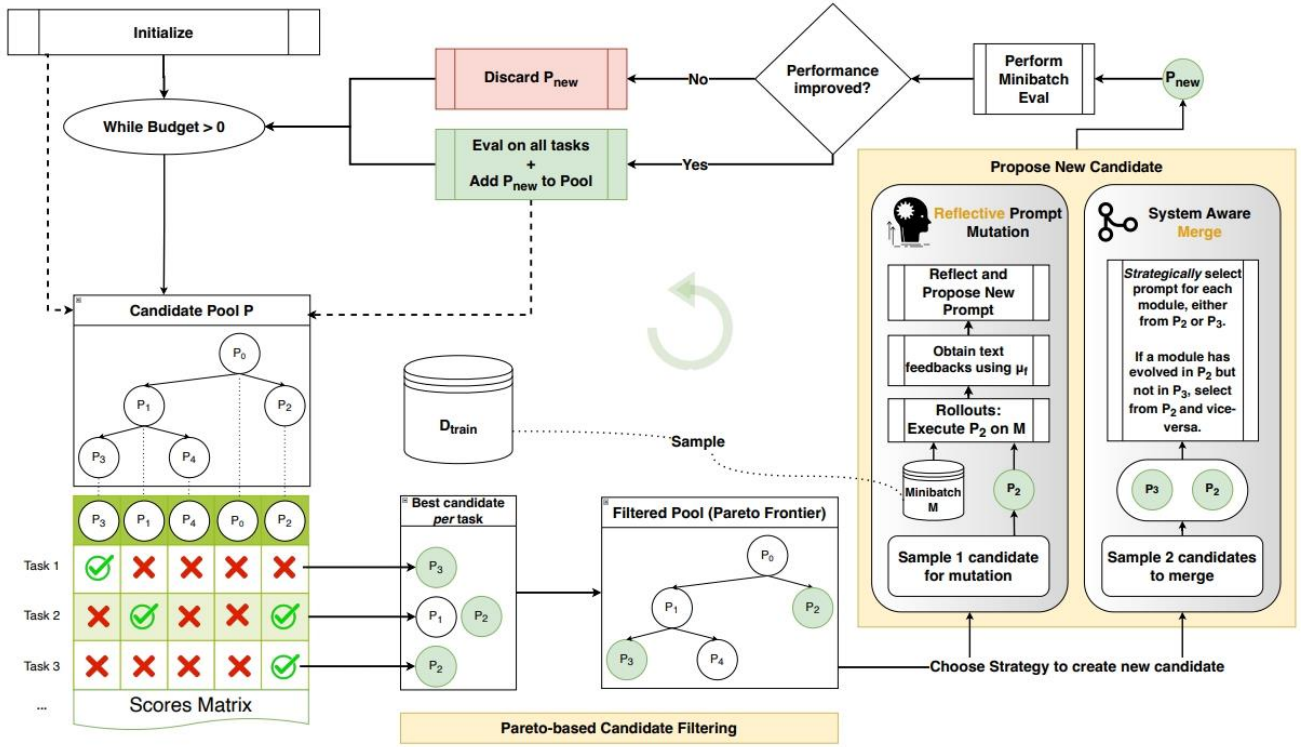


Figure 3: GEPA works iteratively—proposing a new candidate in every iteration by improving some existing candidates using one of the two strategies (Reflective Prompt Mutation (Section 3.2) or System Aware Merge (Appendix F)), first evaluating them on a minibatch, and if improved, evaluating on a larger dataset. Instead of selecting the best performing candidate to mutate always, which can lead to a local-optimum, GEPA introduces Pareto-based candidate sampling (Section 3.3), which filters and samples from the list of best candidates *per* task, ensuring sufficient diversity. Overall, these design decisions allow GEPA to be highly sample-efficient while demonstrating strong generalization.

Context Adaptation (Context Engineering)

- **[25.04] Dynamic Cheatsheet :**
- constructs an external memory that accumulates strategies and lessons from past successes and failures during inference.

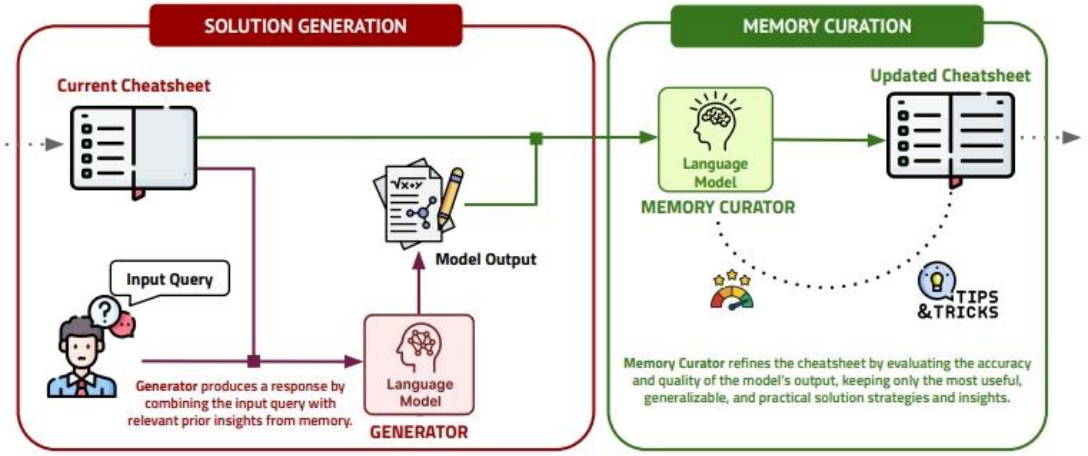


Figure 4: Illustration of *Dynamic Cheatsheet* (DC-Cu variant).

Tasks	Claude 3.5 Sonnet					GPT-4o				
	BL	DC-∅	DR	DC-Cu.	DC-RS	BL	DC-∅	DR	DC-Cu.	DC-RS
AIME 2024	23.3	36.7	43.3	**50.0**	46.7	20.0	36.7	26.7	36.7	**40.0**
AIME 2025	6.7	23.3	23.3	**36.7**	30.0	6.7	10.0	10.0	16.7	**20.0**
AIME 2020–24	6.7	30.1	39.1	38.4	**40.6**	9.8	24.1	24.1	20.3	**24.8**
Game of 24	12.0	10.0	11.0	**14.0**	**14.0**	10.0	19.0	6.0	93.0	**99.0**
GPQA Diamond	59.6	60.1	63.6	61.1	**68.7**	**57.1**	57.1	55.1	58.1	57.1
Math Eqn. Balancer	44.8	56.4	60.4	**100**	97.8	50.0	88.0	**100**	**100**	99.2
MMLU Pro Eng.	61.2	57.2	65.2	66.8	**67.6**	**53.2**	51.6	48.8	44.0	51.2
MMLU Pro Physics	74.0	75.6	80.4	77.6	**82.0**	75.6	70.8	**75.6**	70.4	75.2

Table 1: Performance comparison of Dynamic Cheatsheet (DC) variants for Claude 3.5 Sonnet and GPT-4o across multiple benchmarks. **BL** (Baseline): standard inference without memory; **DC-∅** (Empty Memory): includes structured problem-solving and explicit tool-use instructions but no memory retention mechanism; **DR** (Dynamic Retrieval): uses retrieval but lacks curated memory updates; **DC-Cu** (Cumulative Memory): iteratively accumulates model solutions but lacks retrieval; and **DC-RS** (Retrieval & Synthesis): combines retrieval with memory refinement/synthesis. These results highlight substantial accuracy gains under DC: Claude 3.5 Sonnet's AIME 2024 accuracy jumps by 27% under DC-Cu, and GPT-4o's Game of 24 accuracy leaps from 10% to 99% under DC-RS.

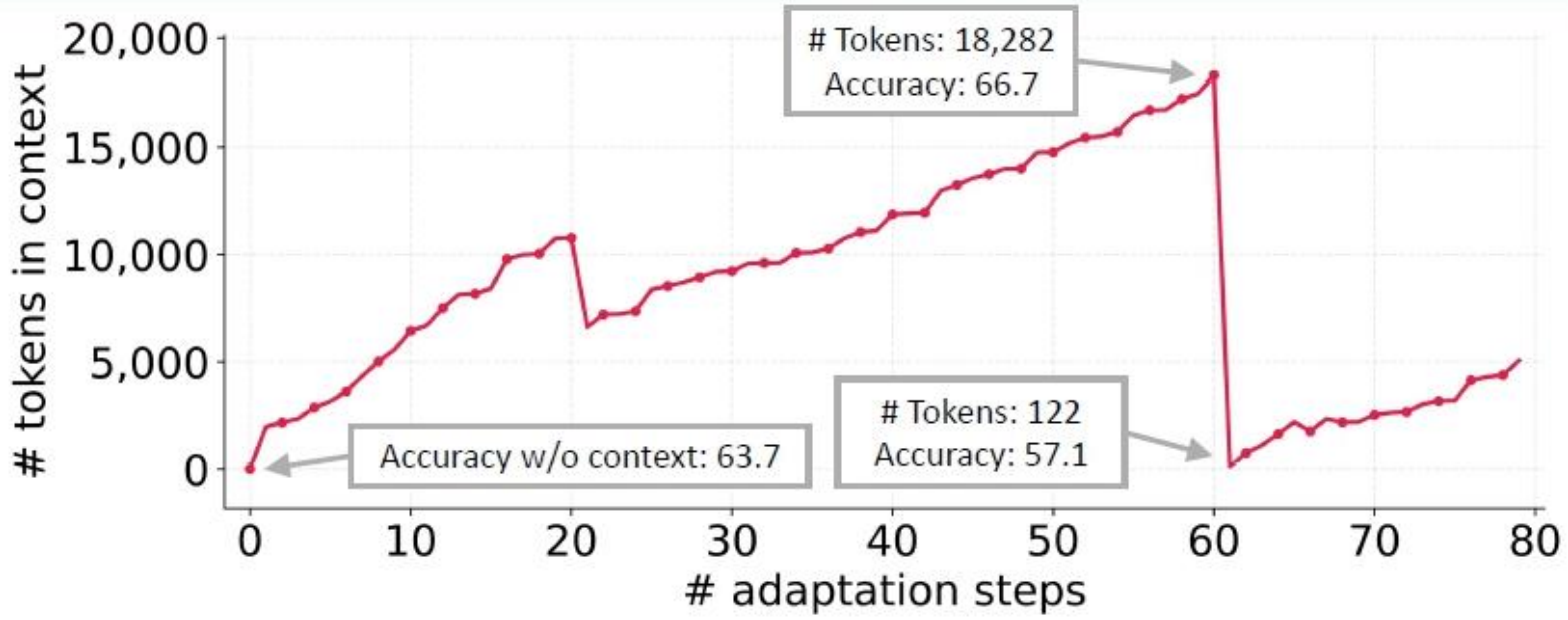
Limitations of Context Adaptation Method

•1) Brevity Bias : Existing prompt optimizers(e.g. GEPA) exhibit the tendency of optimization to **collapse toward short, generic prompts**.

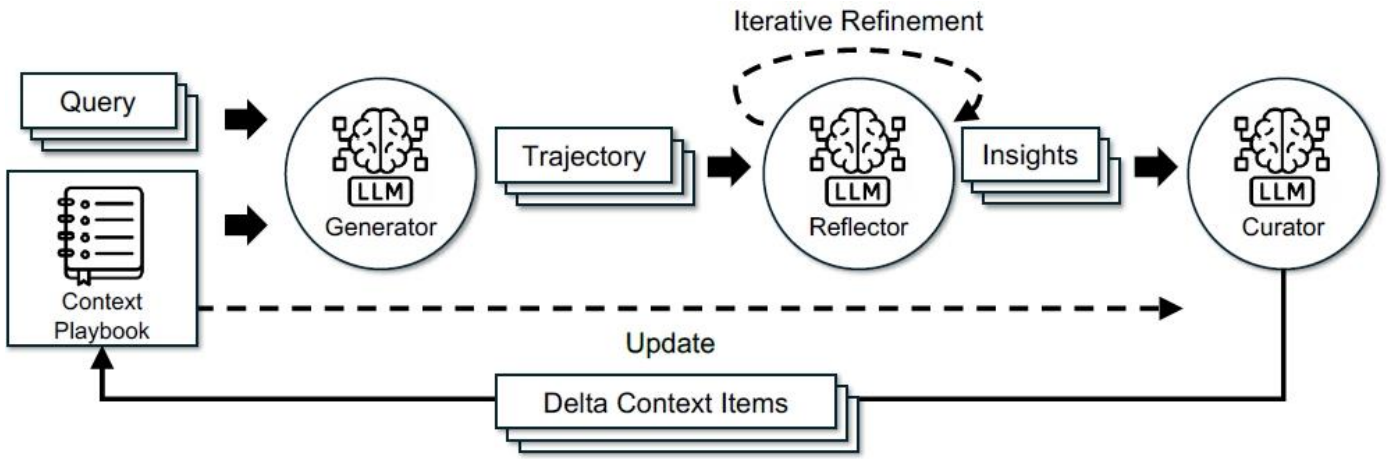
- sacrificing diversity and omitting domain-specific detail.
- This convergence not only **narrows the search space** but also propagates recurring errors across iterations
- More broadly, such bias **undermines performance in domains that demand detailed, context-rich guidance**—such as multi-step agents, program synthesis, or knowledge-intensive reasoning—where success hinges on accumulating rather than compressing task-specific insights.

Limitations of Context Adaptation Method

- 2) Context Collapse :
- A critical failure mode identified in adaptive memory frameworks (e.g. Dynamic Cheatsheet) that rely on an LLM to iteratively rewrite its entire accumulated memory.
- **As the accumulated context ("cheatsheet" or memory) grows large**, the LLM tasked with updating it tends to "collapse" the **content into a much shorter, less informative summary.**
- This causes **a dramatic and abrupt loss of previously learned, detailed knowledge**
- Figure 2. 60 step) 18282 token 66.7% → 61 step) 122 token 57.1%



Agentic Context Engineering (ACE) based on Dynamic Cheatsheet



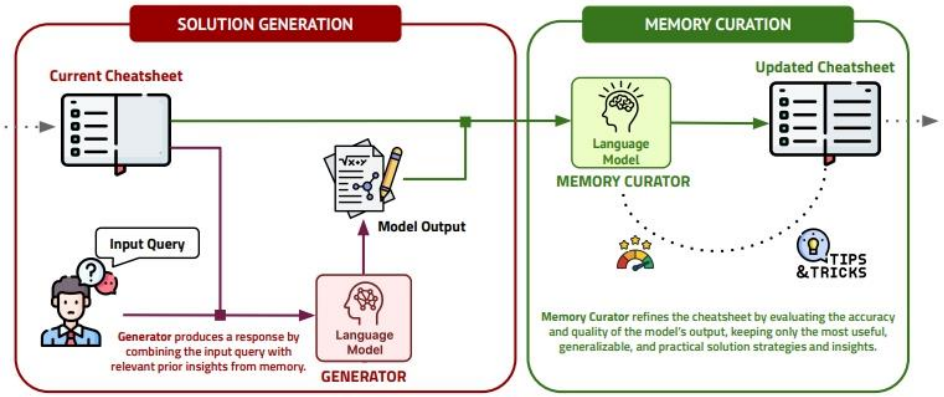


Figure 4: Illustration of *Dynamic Cheatsheet* (DC-Cu variant).

- **Generator** : Takes the user's Query and the current Context Playbook (the existing memory) as input to execute the task.
  - Output: Produces a "Trajectory" , which is the detailed reasoning trace of its attempt to solve the problem.
- **Reflector** : Critiques the Trajectory provided by the **Generator** to identify successes and, more importantly, errors.
  - **Output**: Distills this analysis into concrete, actionable "Insights" or lessons.
  - **Key Innovation**: This is a dedicated module that separates the act of evaluation (analyzing what went wrong) from the act of curation (deciding what to save).
- **Curator** : Receives the Insights from the **Reflector** and synthesizes them into compact, structured updates.
  - **Output**: Produces **"Delta Context Items"** —small, incremental additions or modifications for the playbook.
  - **Function**: These deltas are merged into the Context Playbook using simple**, non-LLM logic**. **This avoids "context collapse" by incrementally updating the memory rather than having an LLM rewrite the entire thing.**
- This mirrors how humans learn—experimenting, reflecting, and consolidating—while avoiding the bottleneck of overloading a single model with all responsibilities.

Agentic Context Engineering - Incremental Delta Updates

- Previous methods (e.g. Dynamic Cheatsheet) used an LLM to rewrite the entire memory/context at each step.
- This is slow, expensive, and leads to Context Collapse—the LLM summarizes and "forgets" critical, detailed knowledge accumulated over time.

- The ACE Solution: **3.1 Incremental Delta Updates.**
  - **Itemized Context**: The Playbook is not one large block of text, but a collection of structured, **"itemized bullets"**
  - **Reflector Generates "Deltas":** The Reflector agent produces only compact "delta context items"—small, specific pieces of new knowledge or corrections (e.g., "ADD: [new insight]", "UPDATE: [correction to item #SHR-009]").
  - **Non-LLM Curation**: **The Curator** integrates these deltas into the playbook using lightweight, deterministic, non-LLM logic (like appending to a list or replacing an item).

- Key Benefits:
  - **Prevents Context Collapse:** Safely preserves all previously learned, valuable knowledge.
  - **Drastic Speed & Cost Reduction**: Eliminates the need for an expensive LLM to re-process the entire context.
  - **Scalable**: Allows the "playbook" to grow and adapt efficiently, even with very long contexts.

Agentic Context Engineering - Grow-and-Refine

- "Incremental Delta Updates" solve context collapse, but how do we prevent the playbook from growing forever and exceeding the context limit?
- The "Grow-and-Refine" mechanism manages the playbook's lifecycle so that balance Knowledge Accumulation with Context Efficiency
- **1. Grow**
  - New insights (delta items) generated by the Reflector and Curator are simply appended to the playbook.
  - This is the fast, primary method for accumulating new knowledge during adaptation.
- **2. Refine**
  - This is a periodic process that prunes and compacts the playbook to ensure it remains relevant and efficient.
  - **De-duplication**: Uses **semantic embeddings** to compare existing entries and remove redundant or similar strategies.
  - **Updating**: Modifies existing bullets in-place (e.g., incrementing usage counters like "helpful" or "harmful").
- Key Benefits:
  - **Maintains Relevance**: Ensures the playbook is **compact** and contains the **most high-value, non-redundant insights**.
  - **Prevents Bloat**: Stops the context window from overflowing, **balancing knowledge accumulation with token limits**.
  - **Flexible & Efficient**: The "lazy" refinement approach **minimizes computational overhead**

Experiments Results

- Task : Agent benchmark (AppWorld), Domain-Specific Benchmark (FiNER, Formula)
- Evaluation Metrics :
  - **Task Goal Completion(TGC)** : Measures success on a specific sub-task.
  - **Scenario Goal Completion(SGC)** : Measures success of the *entire* multi-step user scenario (a much harder metric).
- Baseline & Methods
  - **Base LLM (ReAct: AppWorld)) :** The zero-shot, out-of-the-box performance with no extra context.
  - **In-Context Learning (ICL)** : Standard few-shot prompting. Represents a *static* context (a fixed set of examples) with no *learning*.
  - **MIPROv2** & **GEPA** : SOTA prompt optimizers. They run on a training set to find the *single best prompt* (instructions/examples) to use for all test tasks.
  - **Dynamic Cheatsheet (DC) :** The direct predecessor to ACE. It learns and rewrites its *entire* memory/context *during* the test (risking "context collapse").
  - **ACE (Ours):** The proposed framework. It also learns *during* the test but uses **Incremental Delta Updates** to evolve its "playbook" safely and efficiently.

Experiments Results

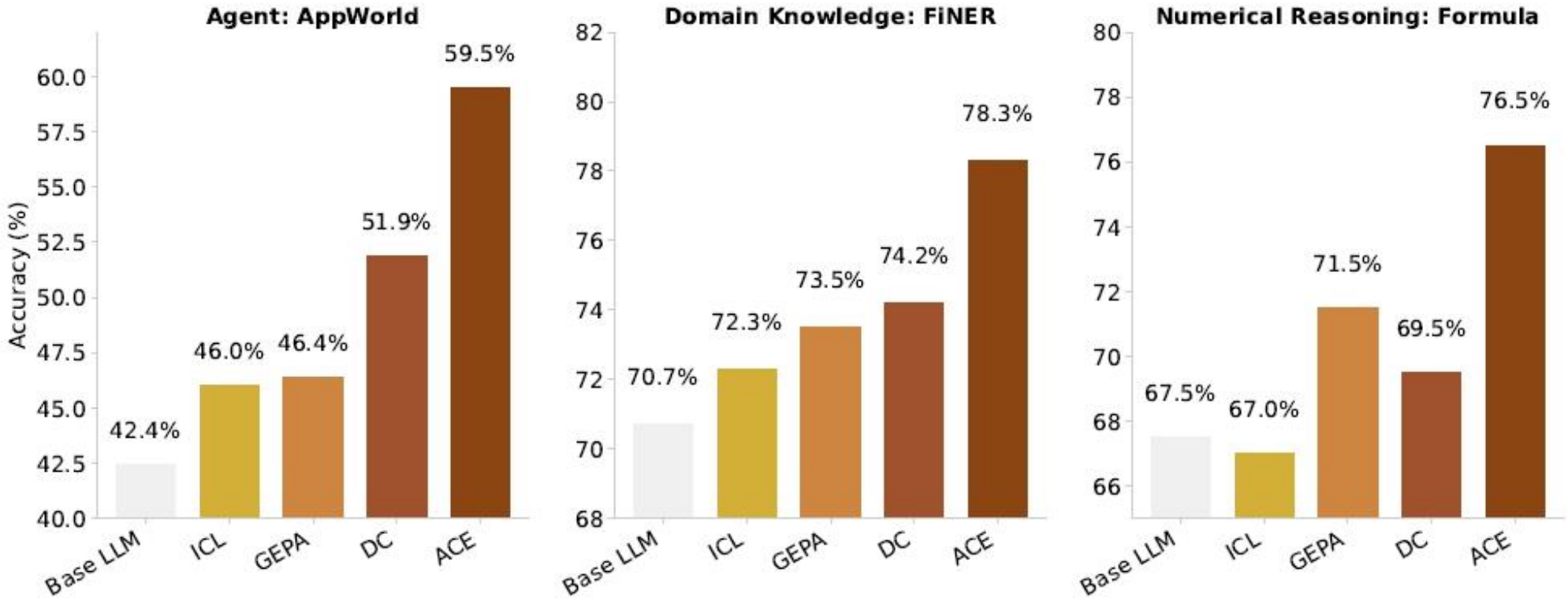


Figure 1: **Overall Performance Results.** Our proposed framework, ACE, consistently outperforms strong baselines across agent and domain-specific reasoning tasks.

Experiments Results (Agent Benchmark - AppWorld)

Method	GT Labels	Test-Normal		Test-Challenge		Average
		TGC↑	SGC↑	TGC↑	SGC↑	
DeepSeek-V3.1 as Base LLM						
ReAct		63.7	42.9	41.5	21.6	42.4
Offline Adaptation						
ReAct + ICL	✓	64.3$_{+0.6}$	46.4$_{+3.5}$	46.0$_{+4.5}$	27.3$_{+5.7}$	46.0$_{+3.6}$
ReAct + GEPA	✓	64.9$_{+1.2}$	44.6$_{+1.7}$	46.0$_{+4.5}$	30.2$_{+8.6}$	46.4$_{+4.0}$
ReAct + ACE	✓	76.2$_{+12.5}$	64.3$_{+21.4}$	57.3$_{+15.8}$	39.6$_{+18.0}$	59.4$_{+17.0}$
ReAct + ACE	✗	75.0$_{+11.3}$	64.3$_{+21.4}$	54.4$_{+12.9}$	35.2$_{+13.6}$	57.2$_{+14.8}$
Online Adaptation						
ReAct + DC (CU)	✗	65.5$_{+1.8}$	58.9$_{+16.0}$	52.3$_{+10.8}$	30.8$_{+9.2}$	51.9$_{+9.5}$
ReAct + ACE	✗	69.6$_{+5.9}$	53.6$_{+10.7}$	66.0$_{+24.5}$	48.9$_{+27.3}$	59.5$_{+17.1}$

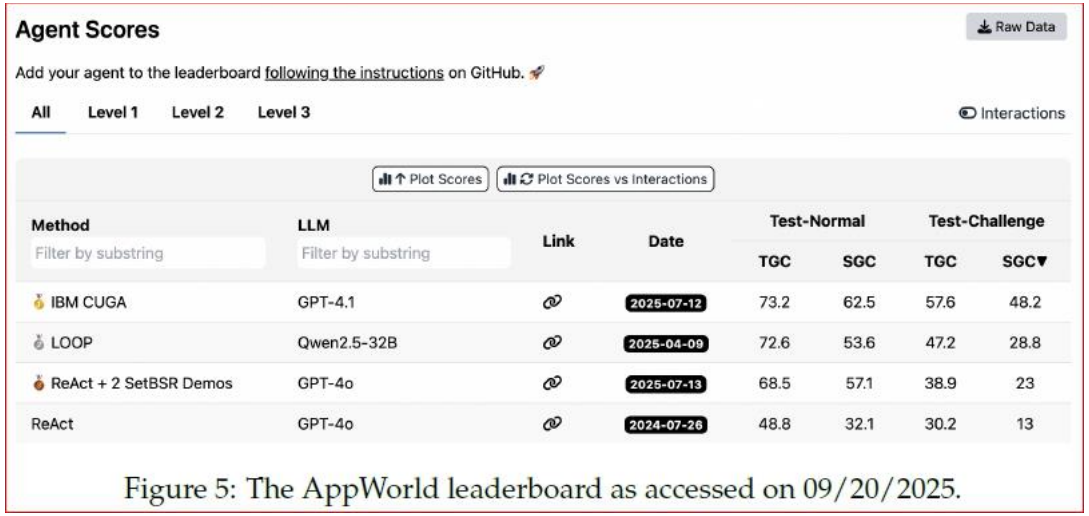


Figure 5: The AppWorld leaderboard as accessed on 09/20/2025.

Experiments Results (Financial Analysis Benchmark)

Method	GT Labels	FINER (Acc↑)	Formula (Acc↑)	Average
DeepSeek-V3.1 as Base LLM				
Base LLM		70.7	67.5	69.1
Offline Adaptation				
ICL	✓	72.3$_{+1.6}$	67.0$_{-0.5}$	69.6$_{+0.5}$
MIPROv2	✓	72.4$_{+1.7}$	69.5$_{+2.0}$	70.9$_{+1.8}$
GEPA	✓	73.5$_{+2.8}$	71.5$_{+4.0}$	72.5$_{+3.4}$
ACE	✓	**78.3**$_{+7.6}$	**85.5**$_{+18.0}$	**81.9**$_{+12.8}$
ACE	✗	71.1$_{+0.4}$	83.0$_{+15.5}$	77.1$_{+8.0}$
Online Adaptation				
DC (CU)	✓	74.2$_{+3.5}$	69.5$_{+2.0}$	71.8$_{+2.7}$
DC (CU)	✗	68.3$_{-2.4}$	62.5$_{-5.0}$	65.4$_{-3.7}$
ACE	✓	**76.7**$_{+6.0}$	76.5$_{+9.0}$	**76.6**$_{+7.5}$
ACE	✗	67.3$_{-3.4}$	**78.5**$_{+11.0}$	72.9$_{+3.8}$

Table 2: **Results on Financial Analysis Benchmark.** "GT labels" indicates whether ground-truth labels are available to the Reflector during adaptation. With GT labels, ACE outperforms selected baselines by an average of 8.6%, highlighting the advantage of structured and evolving contexts for domain-specific reasoning. However, we also observe that in the absence of reliable feedback signals (*e.g.*, ground-truth labels or execution outcomes), both ACE and other adaptive methods such as Dynamic Cheatsheet may degrade, suggesting that context adaptation depends critically on feedback quality.

Experiments Results

Method	GT Labels	Test-Normal		Test-Challenge		Average
		TGC↑	SGC↑	TGC↑	SGC↑	
DeepSeek-V3.1 as Base LLM						
ReAct		63.7	42.9	41.5	21.6	42.4
Offline Adaptation						
ReAct + ACE w/o Reflector or multi-epoch	✓	70.8$_{+7.1}$	55.4$_{+12.5}$	55.9$_{+14.4}$	38.1$_{+17.5}$	55.1$_{+12.7}$
ReAct + ACE w/o multi-epoch	✓	72.0$_{+8.3}$	60.7$_{+17.8}$	54.9$_{+13.4}$	39.6$_{+18.0}$	56.8$_{+14.4}$
ReAct + ACE	✓	76.2$_{+12.5}$	64.3$_{+21.4}$	57.3$_{+15.8}$	39.6$_{+18.0}$	59.4$_{+17.0}$
Online Adaptation						
ReAct + ACE	✗	67.9$_{+4.2}$	51.8$_{+8.9}$	61.4$_{+19.9}$	43.2$_{+21.6}$	56.1$_{+13.7}$
ReAct + ACE + offline warmup	✗	69.6$_{+5.9}$	53.6$_{+10.7}$	66.0$_{+24.5}$	48.9$_{+27.3}$	59.5$_{+17.1}$

Table 3: **Ablation Studies on AppWorld.** We study how particular design choices of ACE (iterative refinement, multi-epoch adaptation, and offline warmup) could help high-quality context adaptation.

Method	Latency (s)↓	# Rollouts↓
ReAct + GEPA	53898	1434
ReAct + ACE	9517$_{(-82.3\%)}$	357$_{(-75.1\%)}$

(a) **Offline** (AppWorld).

Method	Latency (s)↓	Token Cost ($)↓
DC (CU)	65104	17.7
ACE	5503$_{(-91.5\%)}$	2.9$_{(-83.6\%)}$

(b) **Online** (FiNER).

Table 4: **Cost and Speed Analysis.** We measure the context adaptation latency, number of rollouts, and dollar costs of ACE against GEPA (offline) and DC (online).

Discussion

- **1. Long Context != Higher Serving Cost**
  - **Refutation of High-Cost Assumption:** Long contexts (playbooks) do not equate to high serving costs because of Modern serving infrastructure (e.g., **KV Cache Reuse**, Compression, Offloading).
  - **Result: Low cost** for frequently-reused, long contexts.
  - **Implication:** Increased practicality and deployment viability for context-rich frameworks

- **2. A New Paradigm for Online & Continuous Learning**
  - **Efficient Alternative:** A flexible, cost-effective substitute for traditional model fine-tuning.
  - **Core Capability: "Selective Unlearning"**
    - Enabled by human-interpretable, external context (the playbook).
    - **Correction:** Easy identification and removal of outdated or incorrect strategies.
    - **Compliance:** Precise, on-demand deletion of information for legal or privacy mandates (e.g., GDPR).
  - **Advantage:** A level of control and responsibility not achievable with static, fine-tuned model weights.

You are an analysis expert tasked with answering questions using your knowledge, a curated playbook of strategies and insights and a reflection that goes over the diagnosis of all previous mistakes made while answering the question.

Instructions: - Read the playbook carefully and apply relevant strategies, formulas, and insights - Pay attention to common mistakes listed in the playbook and avoid them - Show your reasoning step-by-step - Be concise but thorough in your analysis - If the playbook contains relevant code snippets or formulas, use them appropriately - Double-check your calculations and logic before providing the final answer

Your output should be a json object, which contains the following fields: - reasoning: your chain of thought / reasoning / thinking process, detailed analysis and calculations - bullet_ids: each line in the playbook has a bullet_id. all bulletpoints in the playbook that's relevant, helpful for you to answer this question, you should include their bullet_id in this list - final_answer: your concise final answer

Playbook:

{}

Reflection:

{}

Question:

{}

Context:

{}

Answer in this exact JSON format:

```
{
  "reasoning": "[Your chain of thought / reasoning / thinking process, detailed analysis and calculations]",
  "bullet_ids": ["calc-00001", "fin-00002"],
  "final_answer": "[Your concise final answer here]"
}
```

Figure 12: ACE Generator prompt on FINER

You are an expert analyst and educator. Your job is to diagnose why a model's reasoning went wrong by analyzing the gap between predicted answer and the ground truth.

Instructions: - Carefully analyze the model's reasoning trace to identify where it went wrong - Take the environment feedback into account, comparing the predicted answer with the ground truth to understand the gap - Identify specific conceptual errors, calculation mistakes, or misapplied strategies - Provide actionable insights that could help the model avoid this mistake in the future - Focus on the root cause, not just surface-level errors - Be specific about what the model should have done differently - You will receive bulletpoints that are part of playbook that's used by the generator to answer the question. - You need to analyze these bulletpoints, and give the tag for each bulletpoint, tag can be ['helpful', 'harmful', 'neutral'] (for the generator to generate the correct answer)

Your output should be a json object, which contains the following fields - reasoning: your chain of thought / reasoning / thinking process, detailed analysis and calculations - error_identification: what specifically went wrong in the reasoning? - root_cause_analysis: why did this error occur? What concept was misunderstood? - correct_approach: what should the model have done instead? - key_insight: what strategy, formula, or principle should be remembered to avoid this error? - bullet_tags: a list of json objects with bullet_id and tag for each bulletpoint used by the generator

Question:

{}

Model's Reasoning Trace:

{}

Model's Predicted Answer:

{}

Ground Truth Answer:

{}

Environment Feedback:

{}

Part of Playbook that's used by the generator to answer the question:

{}

Answer in this exact JSON format:

```
{
  "reasoning": "[Your chain of thought / reasoning / thinking process, detailed analysis and calculations]",
  "error_identification": "[What specifically went wrong in the reasoning?]",
  "root_cause_analysis": "[Why did this error occur? What concept was misunderstood?]",
  "correct_approach": "[What should the model have done instead?]",
  "key_insight": "[What strategy, formula, or principle should be remembered to avoid this error?]",
  "bullet_tags": [
    {{"id": "calc-00001", "tag": "helpful"}},
    {{"id": "fin-00002", "tag": "harmful"}}
  ]
}
```

Figure 13: ACE Reflector prompt on FINER

You are a master curator of knowledge. Your job is to identify what new insights should be added to an existing playbook based on a reflection from a previous attempt.

Context: - The playbook you created will be used to help answering similar questions. - The reflection is generated using ground truth answers that will NOT be available when the playbook is being used. So you need to come up with content that can aid the playbook user to create predictions that likely align with ground truth.

CRITICAL: You MUST respond with valid JSON only. Do not use markdown formatting or code blocks.

Instructions: - Review the existing playbook and the reflection from the previous attempt - Identify ONLY the NEW insights, strategies, or mistakes that are MISSING from the current playbook - Avoid redundancy - if similar advice already exists, only add new content that is a perfect complement to the existing playbook - Do NOT regenerate the entire playbook - only provide the additions needed - Focus on quality over quantity - a focused, well-organized playbook is better than an exhaustive one - Format your response as a PURE JSON object with specific sections - For any operation if no new content to add, return an empty list for the operations field - Be concise and specific - each addition should be actionable

Training Context:

- Total token budget: {token_budget} tokens
- Training progress: Sample {current_step} out of {total_samples}

Current Playbook Stats:

{playbook_stats}

Recent Reflection:

{recent_reflection}

Current Playbook:

{current_playbook}

Question Context:

{question_context}

Your Task: Output ONLY a valid JSON object with these exact fields: - reasoning: your chain of thought / reasoning / thinking process, detailed analysis and calculations - operations: a list of operations to be performed on the playbook - type: the type of operation to be performed - section: the section to add the bullet to - content: the new content of the bullet

Available Operations: 1. ADD: Create new bullet points with fresh IDs - section: the section to add the new bullet to - content: the new content of the bullet. Note: no need to include the bullet_id in the content like '[ctx-00263] helpful=1 harmful=0 ::', the bullet_id will be added by the system.

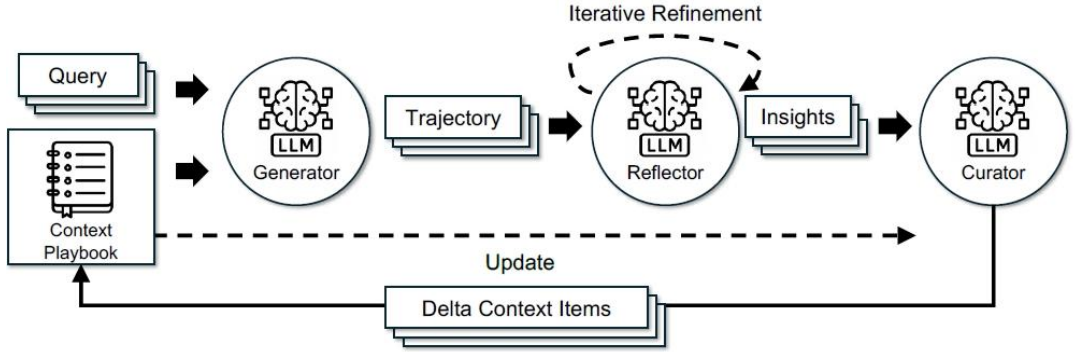
RESPONSE FORMAT - Output ONLY this JSON structure (no markdown, no code blocks):

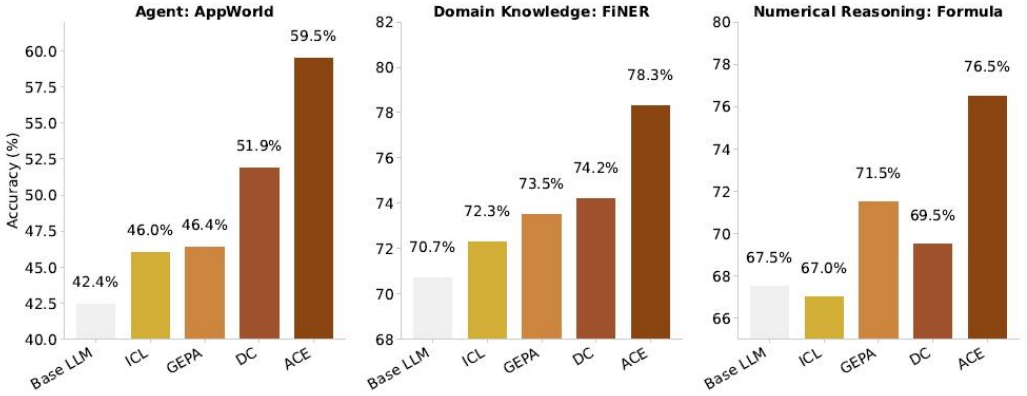
```
{
  "reasoning": "[Your chain of thought / reasoning / thinking process, detailed analysis and calculations here]",
  "operations": [
    {{
      "type": "ADD",
      "section": "formulas_and_calculations",
      "content": "[New calculation method...]"
    }}
  ]
}
```

Figure 14: ACE Curator prompt on FINER

Agentic Context Engineering: Evolving Contexts for Self-Improving Language Models

Agent (Generator, Reflector, Curator) 기반 진화하는 Playbook 관리를 통해 Context Collapse 문제를 완화하고 Feedback 기반 Self-Refinement 수행을 통한 성능 향상





Method	GT Labels	Test-Normal		Test-Challenge		Average
		TGC↑	SGC↑	TGC↑	SGC↑	
DeepSeek-V3.1 as Base LLM						
ReAct		63.7	42.9	41.5	21.6	42.4
Offline Adaptation						
ReAct + ICL	✓	$64.3_{+0.6}$	$46.4_{+3.5}$	$46.0_{+4.5}$	$27.3_{+5.7}$	$46.0_{+3.6}$
ReAct + GEPA	✓	$64.9_{+1.2}$	$44.6_{+1.7}$	$46.0_{+4.5}$	$30.2_{+8.6}$	$46.4_{+4.0}$
ReAct + ACE	✓	$76.2_{+12.5}$	$64.3_{+21.4}$	$57.3_{+15.8}$	$39.6_{+18.0}$	$59.4_{+17.0}$
ReAct + ACE	✗	$75.0_{+11.3}$	$64.3_{+21.4}$	$54.4_{+12.9}$	$35.2_{+13.6}$	$57.2_{+14.8}$
Online Adaptation						
ReAct + DC (CU)	✗	$65.5_{+1.8}$	$58.9_{+16.0}$	$52.3_{+10.8}$	$30.8_{+9.2}$	$51.9_{+9.5}$
ReAct + ACE	✗	$69.6_{+5.9}$	$53.6_{+10.7}$	$66.0_{+24.5}$	$48.9_{+27.3}$	$59.5_{+17.1}$

- **1. Problem Definition**
  - **Brevity Bias:** 기존 Prompt Optimizer 과도한 요약 기반 핵심 세부 정보 누락 현상.
  - **Context Collapse:** 적응형 메모리 기반 전체 문맥 재작성시 누적 지식 소실 발생.
- **2. Proposed Method**
- **Agentic Workflow**
  - **Generator:** Playbook 기반 실제 작업 수행 및 trajectory 생성.
  - **Reflector:** Generator의 Trajectory 기반 비판적 분석 및 Insight 도출.
  - **Curator:** Reflector 결과 Insight를 delta context item으로 가공 및 통합
- **Key Innovation:**
  - **Incremental Delta Updates** - Non-LLM logic 기반 delta context 증분
  - **Grow and Refine**
    - **Grow:** 새로운 delta 항목을 기본적으로 추가하여 지식 누적.
    - **Refine:** Semantic Embedding 기반 중복제거
- **3. Results**
  - AppWorld 평균 **+10.6%**, FiNER에서 평균 **+8.6%**
  - 지연 시간 **86.9% 감소** (vs. DC), rollout 횟수 **75.1% 감소** (vs. GEPA)
  - 더 작은 OpenSource 모델(DeepSeek-V3.1)로 GPT-4.1 기반 1위 Agent (IBM CUGA)에 **비견한 성능** 달성.