Caesar: Deep Agentic Web Exploration for Creative Answer Synthesis

Jason Liang [1] **Elliot Meyerson** [1] **Risto Miikkulainen** [1 2]

Abstract

To advance from passive retrieval to creative discovery of new ideas, autonomous agents must be capable of deep, associative synthesis. However, current agentic frameworks prioritize convergent search, often resulting in derivative summaries that lack creativity. Caesar is an agentic LLM architecture designed to bridge the gap between information gathering and synthesis of new insights. Unlike existing agents that treat the web as a flat sequence of disconnected documents, Caesar leverages an extensive knowledge graph to foster associative reasoning, thus enabling the discovery of non-obvious connections between disparate concepts. It consists of two components: (1) exploration driven by a dynamic context-aware policy, and (2) synthesis controlled by an adversarial draft refinement loop that actively seeks novel perspectives rather than confirming established priors. Caesar demonstrates the ability to generate artifacts and answers characterized by high novelty and structural coherence, significantly outperforming state-of-the-art LLM research agents in tasks requiring creativity.

1. Introduction

The advancement of technology has rarely been driven by the mere accumulation of facts, but rather by the novel synthesis of existing knowledge into new paradigms. From the combinatorial insights that birthed the steam engine to the cross-disciplinary reasoning underlying modern immunology, human progress is defined by the ability to bridge disparate conceptual islands. To advance beyond rote instruction following, which is characteristic of recent frameworks (Deng et al., 2023; Zhou et al., 2024; Plaat et al., 2025), agents must evolve beyond the role of passive librarians. They must become active explorers capable of the deep, associative inquiry that characterizes true expertise. Furthermore, despite the capabilities of Large Language Models (LLMs; Brown et al., 2020), current web agents remain functionally trapped in a paradigm of stateless retrieval. Architectures built on standard retrieval-augmented generation (RAG; Lewis et al., 2020) or linear ReAct loops (Yao et al., 2023) effectively treat the internet as a flat sequence of disconnected documents. They optimize for precision by retrieving the most probable answer to a known question, but do not explore beyond it, overlooking the hidden structure of the information space. This limitation rules out open-ended inquiry, where the goal is not just summarization, but the generation of potentially useful information—that is, artifacts that satisfy the tripartite definition of creativity (Simonton, 2012): *New* (exhibiting novelty and rarity), *Useful* (maintaining viability), and *Surprising* (demonstrating a subversion of expected trajectories).

While a human researcher explores a new field by building a mental map to identify non-obvious connections (Boden, 2004; Pirolli & Card, 1999), current deep research architectures (Elovic, 2023; Li et al., 2025) do not utilize such a topological memory. By processing web pages in isolation, they exhibit "navigational amnesia," looping through redundant content (Qiao et al., 2025) and producing generic summaries that, while useful, may not have the rarity and lateral logic required for high-value research.

This paper presents a novel agentic architecture called **Caesar** that shifts the objective from mechanical retrieval to graph-based discovery (Figure 1). Caesar operates on the premise that the path taken to find information provides useful context for insight and constructs a dynamic knowledge graph during traversal. This topological store of insights serves as an engine for associative reasoning: by evaluating new information specifically in the context of neighboring information, the agent can detect and bridge separate but related concepts, automating the combinatorial process that drives creative hypothesis generation (Boden, 2004).

Caesar operates through a two-phase cognitive cycle. First, it maps the information topology, using a knowledge-guided policy to actively seek out conceptual bridges between disparate topics. Second, it synthesizes answers not by summarizing the graph, but by interrogating it. Caesar employs an adversarial refinement loop designed to escape the basin of attraction of generic LLM outputs. By actively seeking

---

[1] Cognizant AI Lab, San Francisco, California, USA [2] University of Texas at Austin, Austin, Texas, USA. Correspondence to: Jason Liang <jason.liang@cognizant.com>.

Preprint. Under review.

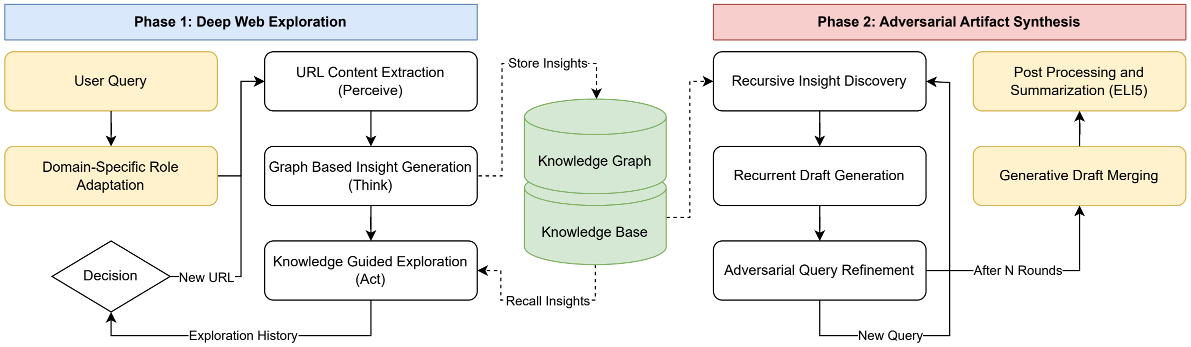

Phase 1: Deep Web Exploration
User Query
Domain-Specific Role Adaptation
URL Content Extraction (Perceive)
Graph Based Insight Generation (Think)
Knowledge Guided Exploration (Act)
Decision
New URL
Exploration History
Store Insights
Knowledge Graph
Knowledge Base
Recall Insights
Phase 2: Adversarial Artifact Synthesis
Recursive Insight Discovery
Recurrent Draft Generation
Adversarial Query Refinement
New Query
After N Rounds
Generative Draft Merging
Post Processing and Summarization (ELI5)

Figure 1. **Visualization of the Caesar architecture. (Left) Phase 1: Deep Web Exploration.** A dynamic exploration policy controls a three-stage loop (Perceive, Think, Act) to traverse the web and to build a knowledge graph/database from insights. **(Right) Phase 2: Adversarial Artifact Synthesis.** Insights are retrieved to synthesize an initial draft. The agent then enters a recursive cycle, critiquing the current draft to generate adversarial queries for refinement, before consolidating all versions via a generative merge and ELI5 summary. Together, these phases transform raw web traversal into a structured reasoning process that prioritizes creativity.

out contradictions, the system produces artifacts that adhere to the tripartite definition of creativity: delivering global novelty, functional viability, and the surprise of a lateral leap. This process is based on four technical innovations:

(1) **Domain-Specific Role Adaptation (Section 3.2).** Caesar analyzes the user query to dynamically rewrite its own system prompt. This adaptation allows Caesar to adopt a persona specifically tuned to the domain's creative constraints, overcoming the safety-biased generic responses typical of models trained with Reinforcement Learning from Human Feedback (RLHF; Ouyang et al., 2022).

(2) **Graph-Augmented Insight Generation (Section 3.4).** This mechanism conditions information extraction on the local topology of the exploration graph. Unlike offline GraphRAG or isolated standard RAG, Caesar performs online associative reasoning: it leverages a knowledge graph to actively bias insight analysis towards identifying connections and contradictions relative to neighboring nodes.

(3) **Knowledge-Guided Exploration (Section 3.5).** Caesar introduces a dynamic decision-making policy for web traversal based on high-level meta-strategies. Unlike existing agents, Caesar utilizes exploration context and memory to detect whether navigation is stagnating, autonomously switching between depth-first expansion and strategic backtracking to maximize information gain while preventing endless looping or cycling.

(4) **Adversarial Artifact Synthesis (Section 4).** This mechanism refines answers through recursive critique. Unlike static RAG, Caesar performs active gap analysis on intermediate drafts to formulate orthogonal queries specifically targeting narrative weaknesses and contradictions. These divergent insights are then consolidated into a single cohesive artifact via a generative merge.

The rest of this paper is organized as follows: Section 2 examines related work, Sections 3 and 4 detail the technical mechanisms for exploration and synthesis, and Sections 5 and 6 present experimental results followed by discussion.

2. Related Work

The evolution of autonomous web agents has shifted from rigid task execution to dynamic exploration (Yao et al., 2023), yet the ability to synthesize novel insights over long horizons remains elusive (Qiao et al., 2025). This section reviews the progression from linear retrieval frameworks to emerging graph-based cognitive architectures, highlighting the specific limitations in navigational reasoning and insight consolidation that Caesar addresses.

Autonomous Web Agents and Deep Research. The development of generalist web agents has been accelerated by realistic benchmarks such as WebArena (Zhou et al., 2024) and Mind2Web (Deng et al., 2023). While the ReAct framework (Yao et al., 2023) successfully established a standard for interleaving reasoning and acting, it fundamentally relies on a linear interaction history, resulting in navigational loops and limited context in complex environments. To mitigate this problem, deep research architectures like WebThinker (Li et al., 2025) and GPT-Researcher (Elovic, 2023) introduced hierarchical planning. Most recently, WebResearcher (Qiao et al., 2025) advanced this paradigm via IterResearch, allowing agents to operate without fixed context windows. However, these systems largely decouple exploration from synthesis only in terms of memory management, not reasoning. Caesar extends this philosophy by decoupling the graph construction phase from artifact synthesis, ensuring that the agent maps the information topology fully before attempting to construct a narrative.

Graph-Based Memory and Reasoning. To overcome the limitations of linear history, researchers have increasingly adopted graph-based memory structures. AriGraph (Anokhin et al., 2024) demonstrated that knowledge graphs significantly outperform vector-only approaches for episodic memory in complex environments. In 2025, the focus shifted from static retrieval to active reasoning. Agentic Deep Graph Reasoning (Buehler, 2025) showed how agents can actively generate concepts and merge them into a self-organizing global graph, while G-Memory (Zhang et al., 2025) proposed hierarchical insight graphs for multi-agent coordination. Caesar adapts these concepts specifically for navigational state tracking. Unlike general-purpose graph memories, Caesar combines knowledge graph insights with navigation history to detect stagnation and to force backtracking, transforming the graph from a passive storage unit into an active control signal for exploration.

Computational Creativity and Associative Search. A critical open challenge in LLM reasoning is to extend agents beyond summarization to creativity. While standard LLMs can produce novel outputs via high-temperature sampling or prompting strategies like Tree of Thoughts (Long, 2023), these approaches often rely on stochastic randomization rather than structured reasoning. Critics argue that without architectural constraints, LLMs default to mean-seeking behavior, producing statistically probable but uninspired content (Chakrabarty et al., 2024). Caesar departs from these stochastic methods by implementing System 2 creativity (Kahneman, 2011): it replaces random sampling with structured reasoning based on chain of thought. By explicitly seeking gaps and bridges between distant insights, Caesar aligns with theories defining creativity as the combinatorial discovery of non-obvious semantic connections (Boden, 2004), rather than mere random divergence.

Iterative Synthesis and Refinement. Standard Retrieval-Augmented Generation (RAG; Lewis et al., 2020) typically follows a single-pass paradigm that results in simple and shallow generated reports. Recent work has shifted towards iterative refinement to improve depth. Self-RAG (Asai et al., 2024) introduced reflection tokens to critique generations, while STORM (Shao et al., 2024) synthesized Wikipedia-like articles through multi-perspective questioning. However, STORM relied on pre-generated outlines, restricting the agent to filling known information slots. Caesar aligns closer with the evolving strategies of Iterative RAG (Choi et al., 2025) but introduces a dynamic query evolution mechanism. Rather than filling a static outline, Caesar's synthesis of one draft actively generates adversarial queries for the next, allowing the narrative structure to emerge organically from the discovered data.

Active Information Foraging. Theoretical foundations from Information Foraging Theory (Pirolli & Card, 1999) are increasingly applied to model the trade-off between information gain and navigational cost (Qian & Liu, 2025).

Algorithm 1 Phase 1: Deep Web Exploration

```
Input: Query $Q$, Budget $T$
Init: $v_0 \leftarrow \textsc{Search}(Q)$, $\mathcal{S} \leftarrow [v_0]$, $G \leftarrow (V = \{v_0\}, E = \emptyset)$, $KB \leftarrow \emptyset$, $M \leftarrow \emptyset$ {Init root node, stack, knowledge-graph/base, memory store}
$\rho_r \leftarrow \textsc{GenerateRole}(Q, v_0)$ {Generate domain-specific agent role}
while $T > 0$ and $\mathcal{S} \neq \emptyset$ do
  $v_c \leftarrow \mathcal{S}.\text{peek}()$; $T \leftarrow T - 1$
  {1. Fetch & Filter URL (Perceive – Sec. 3.3)}
  $P_c, L \leftarrow \textsc{Extract}(v_c)$
  $L_c \leftarrow L \setminus L_f$
  if $P_c$ is Invalid then $\mathcal{S}.\text{pop}()$; continue
  {2. Associative Reasoning (Think – Sec. 3.4)}
  $\mathcal{N}_c \leftarrow G.\text{neighbors}(v_c)$
  $I_c \leftarrow \text{LLM}(P_c, \mathcal{N}_c, Q, \rho_r)$
  $G.\text{update}(v_c, I_c)$; $KB.\text{add}(I_c)$
  {3. Context-Aware Policy (Act – Sec. 3.5)}
  $K_c \leftarrow \{KB.\text{retrieve}(Q), M.\text{recall}()\}$
  $a_m, v_n, Q' \leftarrow \pi(L_c, K_c)$
  if $a_m = \textsc{Backtrack}$ then
    $v_p \leftarrow \mathcal{S}.\text{pop}()$
  else if $a_m = \textsc{WebSearch}$ then
    $v_s \leftarrow \textsc{Search}(Q')$
    $\mathcal{S}.\text{push}(v_s)$; $G.\text{add}(v_c, v_s)$
  else {Explore Link}
    $\mathcal{S}.\text{push}(v_n)$; $G.\text{add}(v_c, v_n)$
  end if
end while
Return $G, KB$
```

DeepResearch Eco (D'Souza et al., 2025) implemented this approach via depth-controlled recursive exploration. Caesar is similar but enforces it via a dynamic policy that conditions actions on both the global knowledge graph and local history. This approach allows the agent to make economic decisions about when to abandon low-value navigational paths more effectively than purely heuristic approaches.

3. Method for Phase 1 (Deep Web Exploration)

As shown in Figure 1, Caesar is divided into two phases: (1) deep web exploration for knowledge collection and (2) exploitation of gained knowledge for answer synthesis. This section presents the core algorithms of the first phase.

3.1. Deep Web Exploration Overview

Caesar treats exploration not as a linear sequence of retrieval steps, but as a stateful graph traversal problem. Formally, given a user query Q, an exploration budget of T steps, and a seed node v_0, the objective is to (1) construct a topological graph $G = (V, E)$, where nodes v represent visited URLs and edges e denote navigational transitions, and (2) populate a semantic knowledge base *KB* with extracted insights. This dual-memory design explicitly decouples *navigation* (managed by G) from *information retention* (managed by *KB*), allowing the agent to maximize information coverage before committing to a narrative structure.

The core of this phase is a recursive Perceive-Think-Act loop (Algorithm 1). Unlike traditional linear scrapers that operate on a stateless current-page basis, Caesar maintains a navigational stack $\mathcal{S}$ alongside the global graph G. This memory structure enables depth-first search capabilities: the agent can drill down into sub-topics until information gain plateaus, and then utilize the stack to backtrack and explore orthogonal branches. This mechanism mitigates the navigational amnesia characteristic of standard web exploration agents (Yao et al., 2023; Qiao et al., 2025).

The process iterates until the budget T is exhausted or the stack is empty. Upon termination, the graph G is frozen, and the structured insights indexed in *KB* are passed to the artifact synthesis module (Section 4) for refinement.

3.2. Domain-Specific Role Adaptation (Initialization)

Before the exploration loop (Algorithm 1), Caesar bootstraps the execution state by transforming the user query Q into a navigational entry point and a task-specific persona.

Query Search Bootstrapping. Rather than requiring a manual seed URL, Caesar generates its own starting node v_0. The agent expands Q into auxiliary search terms, executes them via a web search API, and compiles the top search results into a synthetic HTML document. This document becomes the root node v_0 in graph G and seeds the stack $\mathcal{S}$ with a starting point.

Agent Role Generation. The GENERATEROLE function adapts Caesar's role as defined in the LLM system prompt to match the task. By analyzing Q and the initial search results in v_0, the agent synthesizes a specialized persona ρ_r that defines an explicit goal and exploration philosophy. This approach ensures the reasoning in subsequent THINK and ACT phases aligns with the specific domain constraints of the user's request.

3.3. URL Content Extraction (Perceive)

The PERCEIVE function acts as the agent's sensory interface, accepting the current URL v_c from the stack $\mathcal{S}$ and returning the extracted page content P_c and candidate links L_c. To ensure robust retrieval against anti-bot countermeasures, the system replicates modern browser fingerprints and headers, preventing standard blocking on protected sources. The extraction method is content-agnostic, seamlessly parsing both HTML and PDF documents into plain text.

Similar to prior work (Deng et al., 2023), the raw HTML or PDF is transformed into P_c via a rigorous cleaning pass that only keeps the main text. Unnecessary elements like scripts and tags are removed to maximize the signal-to-noise ratio for the next stage of processing. Simultaneously, the set of outgoing links L_c is extracted. To ensure efficient traversal, L_c is filtered to enforce user-specified domain boundaries while actively discarding failed or over-visited URLs (L_f) to minimize resource waste.

3.4. Graph-Augmented Insight Generation (Think)

Unlike standard browsing agents that summarize pages in isolation, Caesar leverages its topological history G to generate context-aware insights I_c. The LLM synthesizes these insights through the following stages:

Topological Context Retrieval. Before analyzing the content P_c, the agent retrieves the semantic state of the local neighborhood $\mathcal{N}_c$ from the knowledge graph G. Following the paradigm of using graphs for agent memory (Anokhin et al., 2024), this state includes insights associated with predecessor nodes (the incoming path) and neighbors, creating a "short-term memory" window relative to v_c.

Context-Aware Prompting. The generation of I_c is conditioned on the tuple $(P_c, Q, \mathcal{N}_c, \rho_r)$. The LLM is explicitly instructed to identify how the new content P_c builds upon or challenges the retrieved context $\mathcal{N}_c$ rather than merely summarizing it. Similar to the approach in Buehler (2025), this analysis enables Caesar to discover novel patterns and contradictions relative to its specific traversal path.

Dual-State Storage. The resulting insights I_c are stored in a dual-memory system. First, I_c is attached as an attribute to node v_c in G, providing immediate context for future graph traversals. Second, I_c is indexed in a vector store knowledge base *KB*, enabling the agent to perform global semantic retrieval during the subsequent Act stage.

3.5. Knowledge-Guided Exploration (Act)

To navigate in a manner that ensures continual progress, Caesar employs a dynamic policy π_c that conditions action selection on a composite context K_c derived from both the knowledge base *KB* and episodic memory. It operates through the following steps:

Dual-Context Retrieval. The agent constructs K_c by querying two sources. First, it searches in KB for relevant insights related to Q and creates an overview of what is already known. Second, it queries a persistent episodic memory store M for historical navigation patterns using high-frequency keywords extracted from the summarized insights. This memory layer allows K_c to include past navigational failures or loops relevant to the current topic.

Meta-Strategy Formulation. Based on the exploration history in K_c, the LLM selects a high-level meta-action $a_m \in \mathcal{A}$ using three mechanisms:

- EXPLORE: Selects unvisited link $v_n \in L_c$ to expand the frontier, thereby deepening exploration of current topic.
- BACKTRACK: Pops the stack $\mathcal{S}$ to return to the parent node v_p, escaping stagnant exploration regions.

Algorithm 2 Phase 2: Adversarial Artifact Synthesis

```
**Input:** Knowledge Base KB, Rounds $N$, Initial Query $Q_0$
**Init:** History $\mathcal{H} \leftarrow \emptyset$, $A_0 \leftarrow \emptyset$
for $k = 1$ to $N$ do
    {1. Recursive Insight Discovery}
    $I_k \leftarrow \textsc{GenerateInsightQA}(KB, Q_{k-1})$
    {2. Recurrent Draft Generation with Citations}
    $A_k, B_k \leftarrow \textsc{GenerateDraft}(I_k, A_{k-1})$
    $\mathcal{H} \leftarrow \mathcal{H} \cup \{(A_k, B_k)\}$
    {3. Adversarial Query Refinement}
    $Q_k \leftarrow \textsc{RefineQuery}(A_k, Q_{k-1})$
end for
{4. Generative Merge & Citation Map Update}
$A_f, M_f \leftarrow \textsc{MergeDrafts}(\mathcal{H})$
{5. Optional Post-Processing (ELI5)}
$A_e \leftarrow \textsc{PostProcess}(A_f)$
Return $(A_f, M_f), A_e$
```

- WEBSEARCH: Retrieves new search results v_s for Q' (updated using K_c) to pivot exploration to a new area.

Discriminative Link Selection. If the meta-strategy is EXPLORE, the agent selects the next link v_n to visit from the candidate list L_c. The LLM acts as a discriminator, ranking L_c based on K_c and the meta-strategy. The agent then pushes v_n to $\mathcal{S}$, adds the edge (v_c, v_n) to G, and logs the move with reasoning trace into its memory M.

4. Method for Phase 2 (Adversarial Artifact Synthesis)

This section details the adversarial synthesis phase of Caesar and how it improves upon standard RAG, which typically relies on flat, single-shot retrieval. In contrast, Caesar is designed to emulate the recursive drafting and critiquing process of a human researcher. This process operates as a stateful recurrent system that performs active gap analysis to improve and fix a draft artifact over N rounds of refinement. As outlined in Algorithm 2, the adversarial artifact synthesis loop is composed of the following stages:

Recursive Insight Discovery. Before synthesis begins, the agent executes a chain of inquiries to build a structured context window. Unlike standard RAG, which retrieves top documents based on similarity to a static query, Caesar employs a recursive probing strategy (GENERATEINSIGHTQA) over $\hat{T}$ iterations. Given a query q_t, Caesar uses *KB* to retrieve relevant insights, generates an answer a_t, and then automatically creates a follow-up query q_{t+1} to target the ambiguities (or gaps) in a_t. This process ensures that the context window contains a logical chain of reasoning rather than a bag of disjoint facts.

Recurrent Draft Generation. In GENERATEDRAFT, the generation of each draft artifact A_k is conditioned on a composite context window $C_k = \{I_k, A_{k-1}\}$, where I_k is the set of (q, a) pairs generated above, and each answer a is accompanied by source metadata retrieved directly from the vector knowledge base. In order to furnish the artifact with citations, Caesar creates a citation map B_k that links every answer in I_k to specific source URL indices. C_k is used to prompt an LLM to generate A_k by integrating the insights, previous draft, and citations.

Adversarial Query Refinement. To prevent the agent from converging on a shallow summary, Caesar implements an active refinement loop via REFINEQUERY. Between synthesis rounds, the system analyzes the artifact A_k to identify narrative weaknesses. It then formulates a new, orthogonal query Q_k explicitly prompted to target these weaknesses or contradictions in the current draft. Q_k forces the agent to expand the exploration frontier in directions that maximize information gain relative to the current belief state.

Generative Draft Merging. In this stage, MERGEDRAFTS executes a generative unification of the complete draft history $\mathcal{H} = \{(A_k, B_k)\}_{k=1}^{N}$, consisting of the artifact A_k and its citation map B_k from each draft iteration. Rather than simply concatenating drafts, the system prompts an LLM to perform a high-level synthesis that selectively integrates the most relevant insights to create the final merged artifact A_f. The objective is to discover emergent patterns not visible in individual drafts and to construct a cohesive narrative that further develops the core strengths of previous drafts while actively addressing their weaknesses.

Post-Processing and Summarization (ELI5). The output of a creative process that derives insights across diverse sources can be complex. To make the insights more understandable and appealing to a wider audience, the pipeline incorporates a POSTPROCESS module that uses the "Explain Like I'm 5" (ELI5) paradigm (Fan et al., 2019) to distill the final merged artifact into layperson-accessible language with optional token constraints. Crucially, this step is architecturally decoupled from the core synthesis loop, ensuring that the semantic simplification required for readability does not compromise the citation integrity or information density of the main artifact text.

5. Experiments

To validate the efficacy of Caesar, it was compared against state-of-the-art research agents powered by the latest LLM models. Evaluations utilize a blinded LLM-as-a-Judge framework (Zheng et al., 2023) to assess performance on diverse queries that exemplify the three dimensions of creativity: *New*, *Useful*, and *Surprising*.

5.1. Creative Query Answering

Caesar was evaluated against baseline agents using the following foundation models: Claude Sonnet 4.5 (Anthropic, 2025), GPT-5.2 (OpenAI, 2025), and Gemini 3 Pro (Gemini Team, Google, 2025). Two configurations were tested: a

Table 1. **LLM-as-a-Judge Results.** Average scores for all agents under each output constraint. Caesar significantly outperforms all baseline agents (MWU $p < 0.001$; Mann & Whitney, 1947).

Agent	New	Useful	Surprising	Total
Full Answers (Unconstrained)				
Caesar	**8.64**	**8.38**	**8.27**	**25.29**
Gemini 3 (Deep)	7.69	7.09	7.49	22.27
Sonnet 4.5 (Deep)	6.96	7.20	6.73	20.89
Gemini 3 (Shallow)	5.47	5.33	5.44	16.24
GPT-5.2 (Deep)	5.02	6.02	4.36	15.40
Sonnet 4.5 (Shallow)	5.47	4.13	5.24	14.84
GPT-5.2 (Shallow)	4.80	4.96	4.44	14.20
ELI5 Answers (Unconstrained)				
Caesar	**8.44**	**8.29**	**8.02**	**24.76**
Sonnet 4.5 (Deep)	7.02	7.44	6.62	21.09
GPT-5.2 (Deep)	5.36	6.78	4.93	17.07
Gemini 3 (Deep)	5.69	5.69	5.69	17.07
Sonnet 4.5 (Shallow)	5.62	4.62	5.51	15.76
GPT-5.2 (Shallow)	4.89	4.93	4.49	14.31
Gemini 3 (Shallow)	4.16	4.40	3.93	12.49
ELI5 Answers (450 Word Limit)				
Caesar	**7.91**	**7.76**	**7.64**	**23.31**
Sonnet 4.5 (Deep)	6.64	7.24	6.31	20.20
Gemini 3 (Deep)	6.53	6.07	6.64	19.24
Sonnet 4.5 (Shallow)	6.71	5.09	6.62	18.42
GPT-5.2 (Deep)	4.60	6.13	4.20	14.93
GPT-5.2 (Shallow)	4.98	5.27	4.53	14.78
Gemini 3 (Shallow)	4.47	5.16	4.24	13.87

Table 2. **Judge Bias Analysis.** Self-preference biases are listed across all judges. Positive values indicate a preference for models of the same family; negative values indicate a penalty. Judge GPT is least biased overall with mixed results for other judges.

Judge	**Full**	**ELI5**	**ELI5 (450w)**
Judge Gemini	+2.47	+0.78	+0.67
Judge GPT	+0.35	+0.02	+0.47
Judge Claude	+0.30	-1.78	-1.57

shallow agent utilizing standard single-step web search, and a *deep* variant running a proprietary autonomous research mode that allows for unlimited web search steps (i.e. Research Mode for Claude, Deep Research for Gemini/GPT).

To ensure fairness, all agents (including Caesar) utilized the same basic prompt for generating answers with reasoning effort set to high. This prompt encourages insightful, interesting responses to queries but is distinct from the prompt given to the judges to prevent overfitting and reward hacking. A panel of three LLM judges (Claude Sonnet 4.5, GPT-5.2, and Gemini 3 Pro) evaluated anonymized answers on a 10-point scale across three dimensions: *New* (Novelty/Rarity), *Useful* (Viability/Alignment), and *Surprising* (Non-obvious connections). Agent prompts and judge rubrics are described in Appendices D and E, while implementation details and hyperparameters are listed in Appendix F.

Each agent was scored across five queries designed to test different aspects of creativity: Constrained Synthesis, Counterfactual Reasoning, Cross-Domain Synthesis, Meta-Creativity, and Open-Ended Synthesis. These aspects were identified by searching prior work for common creativity failure modes of LLMs (Appendix E.1). Each query was evaluated under three output constraints: (1) unconstrained full answers, (2) unconstrained ELI5 summaries, and (3) length-constrained ELI5 summaries (450 words). Every judge scored each query three times to reduce evaluation noise. To better understand how Caesar outperforms the baselines, analysis of sample answers and detailed query scores are provided in Appendices A and G respectively.

As shown in Table 1, Caesar significantly outperformed all baselines across all experimental settings (Mann-Whitney U Test with $p < 0.001$; Mann & Whitney, 1947). In the unconstrained full answer setting, Caesar achieved a total score of 25.29, surpassing the runner-up (Gemini 3 Deep Research) by a margin of 3.02 points. Notably, Caesar demonstrated the highest scores in the *New* (8.64) and *Surprising* (8.27) metrics. As further elaborated in Section 6.1, this result suggests that graph-based associative reasoning fosters greater creativity than standard search and retrieval, which often defaults to derivative summarization.

The performance gap persisted with semantic compression. In the unconstrained ELI5 setting, where significantly simpler answers were required, Caesar achieved a total score of 24.76 compared to the runner-up's 21.09. While baseline models typically sacrificed surprise to meet the readability constraints of the ELI5 persona, Caesar retained a high surprise score of 8.02, indicating that the simplification process did not dilute the novelty of the retrieved insights.

This trend persisted even in the length-constrained ELI5 (450 words) task. Caesar maintained a high novelty score of 7.91 while peer models degraded in performance. As discussed in Section 6.2, this result validates the architectural decision to perform draft refinement followed by a generative merge. Caesar maximizes semantic density, ensuring that high-utility insights are retained even when narrative flair is pruned. This superior length-constrained performance shows that Caesar's creative capabilities do not follow simply from generating longer answers to queries, but instead from the actual content in the answers.

5.2. Judge Bias Analysis

To ensure that the scoring is robust, self-preference bias of the judge models was analyzed (Table 2). For each judge, a separate score was calculated in judging the answers generated by its own model family (e.g. Gemini judging Gemini). This score was then compared against the scores assigned to these answers by the other judges (i.e. GPT and Claude judging Gemini). The bias was calculated as the difference between these two scores.

Using this metric, a strong positive bias (+2.47) was ob-

Table 3. **Exploration Ablation.** Impact of reducing the Phase 1 web exploration budget (T) on final artifact quality. Higher iteration counts consistently yielded higher scores, validating the hypothesis that creative insights require deeper web exploration.

Configuration	New	Useful	Surprising	Total
Full Answers (Unconstrained)				
Caesar (1000 Iter)	**8.18**	**8.26**	**8.06**	**24.50**
Caesar (500 Iter)	7.46	7.64	7.46	22.56
Caesar (250 Iter)	7.44	7.30	7.26	22.00
ELI5 Answers (Unconstrained)				
Caesar (1000 Iter)	**7.70**	**7.98**	**7.68**	**23.36**
Caesar (500 Iter)	6.92	7.66	6.86	21.44
Caesar (250 Iter)	7.00	7.10	6.86	20.96
ELI5 Answers (450 Word Limit)				
Caesar (1000 Iter)	**7.86**	**8.12**	**7.86**	**23.84**
Caesar (500 Iter)	7.16	7.70	7.20	22.06
Caesar (250 Iter)	6.46	7.04	6.50	20.00

served from the Gemini judge towards Gemini models for full answers. This observation aligns with research on LLM self-preference, suggesting that models optimized via RLHF might overfit to specific writing styles that function as heuristics for quality in their reward models (Panickssery et al., 2024; Zheng et al., 2023). In contrast, the Claude judge exhibited a significant negative bias towards answers in the ELI5 setting (-1.78 & -1.57). This result is likely a downstream effect of Constitutional AI (Bai et al., 2022), where the necessary radical simplification is penalized as a violation of training objectives prioritizing nuance and honesty over simplicity. Despite these opposing biases, Caesar remains robust against individual judge idiosyncrasies.

5.3. Ablation Studies

To isolate the specific contributions of the exploration and synthesis phases to the generation of creative artifacts, two ablation studies were run. The first study evaluated the impact of reducing the web exploration iteration budget, and the second analyzed how constraining the number of synthesis drafts affects the quality of the final output. Due to its consistency, Judge GPT was used for both studies and the evaluation results were averaged over 10 trials.

Table 3 demonstrates that there is a strong positive correlation between the exploration budget (T) and the creative quality of the final artifact. Reducing iterations from 1000 to 250 caused a noticeable decline in total score (e.g., -3.84 for 450 word answers), primarily caused by the *Surprising* and *New* metrics. The results support the hypothesis that high-value, non-obvious insights are not found on the surface of the web but require deep topological traversal to uncover. While lower iteration counts ($T = 250$) still produced functional answers, they did not reach the rare, long-tail information necessary to elevate the artifact beyond surface-level analysis.

Table 4. **Draft Ablation.** Adversarial synthesis creates a tension between creativity and viability. While recursive refinement ($A_1 \to A_3$) maximizes the score for *Surprising* at the cost of *Useful*, the final generative merge (A_f) reconciles these objectives, recovering utility while retaining the novel insights.

Variant	New	Useful	Surprising	Total
Full Answers (Unconstrained)				
Caesar (Final)	**7.90**	**7.94**	7.60	**23.44**
Caesar (Draft 3)	7.88	6.80	**7.90**	22.58
Caesar (Draft 1)	6.62	7.42	6.26	20.30
ELI5 Answers (Unconstrained)				
Caesar (Final)	7.54	**7.90**	7.20	**22.64**
Caesar (Draft 3)	**7.60**	7.12	**7.58**	22.30
Caesar (Draft 1)	6.56	7.52	6.18	20.26
ELI5 Answers (450 Word Limit)				
Caesar (Draft 3)	**7.40**	7.24	**7.54**	**22.18**
Caesar (Final)	7.04	**7.64**	6.76	21.44
Caesar (Draft 1)	6.32	7.22	5.86	19.40

Similarly, Table 4 demonstrates that artifact quality improves in successive drafts. Initial drafts (A_1) maximize *Useful* but represent a local minimum for creativity. Subsequent adversarial refinement (A_3) boosts *Surprising* (e.g., +1.64 for full answers) and *New* scores, confirming that refinement changes the answer substantially. While this exploration incurs a temporary dip in *Useful*, the generative merge (A_f) stabilizes the output, recovering usefulness while retaining discovered insights. Notably, under strict length constraints, A_f is slightly worse than A_3, suggesting that compressing diverse perspectives can dilute the narrative sharpness of a focused draft.

Overall, these ablation studies validate the two main design principles of Caesar: (1) that discovering non-obvious insights requires deep topological exploration rather than shallow retrieval, and (2) that achieving creativity necessitates an adversarial refinement loop to escape the surface-level summarization typical of conventional search. Consequently, Caesar provides a framework for transforming web search agents from passive summarizers into active researchers.

6. Discussion and Future Work

This section reviews the specific mechanisms driving Caesar's performance, the trade-offs involved in test-time compute, the limitations of the proposed architecture, and potential future improvements.

6.1. The Importance of Graph Exploration

A core finding of the evaluation is Caesar's strong performance in the *New* and *Surprising* metrics (Table 1). This result can be attributed to the fundamental difference between flat, vector-based search/retrieval and the graph-based exploration/synthesis employed by Caesar.

Standard RAG systems populate their context using dense vector similarity against the initial query, which inherently retrieves documents that are semantically close to the user's existing priors (homophily; Panickssery et al., 2024). In contrast, while Caesar utilizes a vector store *KB* for retrieval during synthesis, the contents of this store are governed by the THINK module's topological traversal (Section 3.4). By traversing edges in the navigation graph G, the agent ingests information that is structurally connected (e.g., via citation or hyperlink) but often semantically orthogonal to the original query. This mechanism ensures that *KB* is seeded with diverse, non-obvious concepts.

Consequently, when the synthesis module later queries *KB*, it can retrieve bridging insights (Boden, 2004) that a pure vector-based search would have never encountered or indexed in the first place. Thus, Caesar effectively automates the process of creative discovery, allowing the agent to construct arguments based on the structural relationships between concepts rather than being limited to their immediate semantic proximity. Appendix B visualizes how graph topologies change depending on the creative task, demonstrating the adaptive capabilities of Caesar during search.

6.2. The Dynamics of Adversarial Synthesis

The draft ablation study (Table 4) reveals a critical tension between novelty and utility. Intermediate drafts, specifically A_3 (generated after adversarial refinement), exhibit the highest *Surprising* scores but suffer a drop in *Useful*. This outcome suggests that the adversarial query refinement successfully pushes the agent out of the local minima of generic consensus, but occasionally drifts into tangential connections or unsupported claims.

Underpinning draft synthesis is the recursive insight discovery mechanism in GENERATEINSIGHTS, which ensures the agent operates on a logical chain rather than disjoint facts. Appendix C details a case study of this chain of thought, demonstrating how Caesar uses iterative inquiry to evolve an initial abstract concept into a verified operational model before the first draft is even written.

The effectiveness of Caesar also lies in the MERGEDRAFTS phase, which acts as a convergent filter. By synthesizing the high-variance perspectives of A_1 to A_3, the merge step recovers the *Useful* score (raising it from 6.80 to 7.94 in the full answer setting) while retaining the core novelties. This result confirms that creativity in agentic systems benefits from a two-step process: exploratory divergence to maximize the search space, followed by strict convergence for verification (Koivisto & Grassini, 2023).

6.3. Limitations

The performance gains shown in Table 3 come at the cost of increased inference-time compute. Caesar with 1000 iterations requires significantly more token usage and wall-clock time than a standard single-turn retrieval agent. This pattern aligns with recent observations that scaling test-time compute can yield intelligence gains analogous to scaling model parameters (Brown et al., 2020). However, Caesar's heavy token usage also makes it less suitable for low-latency applications. The system is optimized for deep research tasks where the value of a breakthrough insight justifies the computational expense, rather than real-time conversational interaction. A full breakdown of Caesar's computational costs is provided in Appendix H.

Furthermore, despite its success, Caesar's architecture exhibits specific failure modes. First is the SEO trap: if the bootstrapping phase (Section 3.2) seeds the graph G within a cluster of low-quality, search-optimized content, the graph-based reasoning can become trapped in a recursive loop of low-value information, reinforcing false consensus. Second is unnecessary complexity: for simple, fact-retrieval queries (e.g., "What is the weather?"), the overhead is unjustifiable. The adversarial loop may attempt to find novel angles on settled facts, leading to unnecessary complexity. Future versions will focus on adaptive mechanisms to dynamically calibrate the exploration depth based on query complexity.

6.4. Future Work

An interesting direction of future work would be to use Caesar to tackle the ARC-AGI benchmark (Chollet, 2019). Caesar's exploration capabilities could perform open-ended program synthesis and create a powerful solver for this benchmark. To achieve this goal, Caesar will need to be extended from a single agent to a collaborative multi-agent swarm. This multi-agent architecture would allow for parallelized creative exploration and reasoning over diverse topics. Such a next step would demonstrate that Caesar is not just a mechanism for creative question answering, but could serve as a foundational component in constructing general-purpose intelligent systems.

7. Conclusion

This paper presents Caesar, a framework that advances autonomous web search by combining graph-based exploration with adversarial refinement of artifacts. Caesar effectively bridges the gap between static retrieval and active discovery, generating answers that are richer in latent insights than standard RAG or web search baselines. Ultimately, this framework represents a shift toward a new generation of AI systems that do not merely retrieve existing knowledge, but actively participate in the creation of new ideas.

References

Amabile, T. M. Social psychology of creativity: A consensual assessment technique. *Journal of Personality and Social Psychology*, 43(5):997–1013, 11 1982. ISSN 0022-3514. doi: 10.1037/0022-3514.43.5.997.

Anokhin, P., Semenov, N., Sorokin, A., Evseev, D., Burtsev, M., Burnaev, E., and Kravchenko, A. AriGraph: Learning knowledge graph world models with episodic memory for LLM agents. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, volume abs/2407.04363, pp. 12–20. Cornell University, 7 2024. doi: 10.48550/arxiv.2407.04363.

Anthropic. Claude sonnet 4.5. Technical report, Anthropic, 2025. URL `https://www.anthropic.com/claude/sonnet`.

Asai, A., Wu, Z., Wang, Y., Sil, A., and Hajishirzi, H. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

Bai, Y., Kadavath, S., Kundu, S., Askell, A., Kernion, J., Jones, A., Chen, A., Goldie, A., Mirhoseini, A., McKinnon, C., et al. Constitutional AI: Harmlessness from AI feedback. *ArXiv preprint*, abs/2212.08073, 12 2022. doi: 10.48550/arxiv.2212.08073.

Bhat, S. and Gokhale, T. Inductive or deductive? rethinking the fundamental reasoning abilities of LLMs. *ArXiv preprint*, abs/2408.00114, 2024.

Boden, M. A. *The Creative Mind: Myths and Mechanisms*. Routledge, 2004. Classic foundational text on combinatorial creativity.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, volume 33, pp. 1877–1901. Cornell University, 5 2020. doi: 10.48550/arxiv.2005.14165.

Buehler, M. J. Agentic deep graph reasoning yields self-organizing knowledge networks. *Journal of Materials Research*, 40(15):2204–2242, 7 2025. ISSN 0884-1616. doi: 10.1557/s43578-025-01652-1.

Byrne, R. M. J. *The Rational Imagination: How People Create Alternatives to Reality*. MIT Press, Cambridge, MA, 2005. ISBN 978-0-262-02584-3.

Chakrabarty, T., Laban, P., Agarwal, D., Muresan, S., and Wu, C. Art or artifice? large language models and the false promise of creativity. In Mueller, F. F., Kyburz, P., Williamson, J. R., Sas, C., Wilson, M. L., Dugas, P. O. T., and Shklovski, I. (eds.), *Proceedings of the CHI Conference on Human Factors in Computing Systems, CHI 2024, Honolulu, HI, USA, May 11-16, 2024*, pp. 30:1–30:34. ACM, 5 2024. doi: 10.1145/3613904.3642731.

Choi, C., Lee, W., Ko, J., and Rhee, W. MI-RAG: Multimodal iterative retrieval-augmented generation for knowledge-intensive visual question answering. *ArXiv preprint*, abs/2509.00798, 2025.

Chollet, F. On the measure of intelligence. *ArXiv preprint*, abs/1911.01547, 11 2019.

Chroma Core. Chroma: Open-source embedding database, 2024. URL `https://github.com/chroma-core/chroma`.

Deng, X., Gu, Y., Zheng, B., Chen, S., Stevens, S., Wang, B., Sun, H., and Su, Y. Mind2web: Towards a generalist agent for the web. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*. Cornell University, 6 2023. doi: 10.48550/arxiv.2306.06070.

D'Souza, J., Sander, E. K., and Aioanei, A. DeepResearchEco: A recursive agentic workflow for complex scientific question answering in ecology. *ArXiv preprint*, abs/2507.10522, 2025.

Elovic, A. GPT Researcher: Autonomous agent for comprehensive online research, 2023. URL `https://github.com/assafelovic/gpt-researcher`. GitHub repository.

Fan, A., Jernite, Y., Perez, E., Grangier, D., Weston, J., and Auli, M. ELI5: Long form question answering. In Korhonen, A., Traum, D., and Màrquez, L. (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, volume abs/1907.09190, pp. 3558–3567, Florence, Italy, 7 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1346.

Gemini Team, Google. Gemini 3: A new era of intelligence. Technical report, Google DeepMind, 2025. URL `https://blog.google/products-and-platforms/products/gemini/gemini-3/`.

Gentner, D. Structure-mapping: A theoretical framework for analogy. *Cognitive Science*, 7(2):155–170, 4 1983. ISSN 0364-0213. doi: 10.1207/s15516709cog0702_3.

Guilford, J. P. *The Nature of Human Intelligence*. McGraw-Hill, New York, 1967.

Hagberg, A. A., Schult, D. A., and Swart, P. J. Exploring network structure, dynamics, and function using NetworkX. In Varoquaux, G., Vaught, T., and Millman, J. (eds.), *Proceedings of the 7th Python in Science Conference*, pp. 11–15. SciPy, 6 2008.

Kahneman, D. *Thinking, Fast and Slow*. Farrar, Straus and Giroux, New York, 2011. ISBN 978-0374275631. Foundational text for System 1 (intuitive) and System 2 (deliberate) cognitive processes.

Koestler, A. *The Act of Creation*. Hutchinson, London, 1964.

Koivisto, M. and Grassini, S. Best humans still outperform artificial intelligence in a creative divergent thinking task. *Scientific Reports*, 13(1):13601, 9 2023. ISSN 2045-2322. doi: 10.1038/s41598-023-40858-3.

Lewis, P. S. H., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., and Kiela, D. Retrieval-augmented generation for knowledge-intensive NLP tasks. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, volume 33, pp. 9459–9474. University College London, 5 2020.

Li, X., Jin, J., Dong, G., Qian, H., Zhu, Y., Wu, Y., Wen, J.-R., and Dou, Z. WebThinker: Empowering large reasoning models with deep research capability. *ArXiv preprint*, abs/2504.21776, 4 2025. doi: 10.48550/arxiv.2504.21776.

Lin, Y., Tan, L., and Zhang, T. Mitigating the alignment tax of RLHF. In Al-Onaizan, Y., Bansal, M., and Chen, Y.-N. (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 580–606. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.35.

Long, J. Large language model guided tree-of-thought. *ArXiv preprint*, abs/2305.08291, 5 2023. doi: 10.48550/arxiv.2305.08291.

Maaten, L. v. d. and Hinton, G. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov): 2579–2605, 2008. ISSN 1532-4435.

Mann, H. B. and Whitney, D. R. On a test of whether one of two random variables is stochastically larger than the other. *The Annals of Mathematical Statistics*, pp. 50–60, 1947.

Meyerson, E. and Qiu, X. Position: Scaling llm agents requires asymptotic analysis with llm primitives. *arXiv preprint arXiv:2502.04358*, 2025.

Neo4j, Inc. Neo4j graph database, 2024. URL `https://neo4j.com/`.

OpenAI. Introducing gpt-5.2. Technical report, OpenAI, 2025. URL `https://openai.com/index/introducing-gpt-5-2`.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P. F., Leike, J., and Lowe, R. Training language models to follow instructions with human feedback. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, volume abs/2203.02155. Cornell University, 3 2022. doi: 10.48550/arxiv.2203.02155.

Panickssery, A., Bowman, S. R., and Feng, S. LLM evaluators recognize and favor their own generations. In Globersons, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J. M., and Zhang, C. (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, volume abs/2404.13076, pp. 68772–68802. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 4 2024. doi: 10.48550/arxiv.2404.13076.

Pirolli, P. and Card, S. Information foraging. *Psychological Review*, 106(4):643–671, 10 1999. ISSN 0033-295X.

Plaat, A., Picek, S., et al. Agentic large language models, a survey. *ArXiv preprint*, abs/2503.23037, 12 2025. ISSN 1076-9757. doi: 10.1613/jair.1.18675.

Qian, H. and Liu, Z. Scent of Knowledge: Optimizing search-enhanced reasoning with information foraging. *ArXiv preprint*, abs/2505.09316, 5 2025.

Qiao, Z., Chen, G., Chen, X., Yu, D., Yin, W., Wang, X., Zhang, Z., Li, B., Yin, H., Li, K., Min, R., Liao, M., Jiang, Y., Xie, P., Huang, F., and Zhou, J. Webresearcher: Unleashing unbounded reasoning capability in long-horizon agents. *ArXiv preprint*, abs/2509.13309, 9 2025. ISSN 2331-8422. doi: 10.48550/arxiv.2509.13309.

Shao, Y., Jiang, Y., Kanell, T., Xu, P., Khattab, O., and Lam, M. Assisting in writing Wikipedia-like articles from scratch with large language models. In Duh, K., Gomez, H., and Bethard, S. (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 6252–6278, Mexico City, Mexico, 2 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.347.

Shumailov, I., Shumaylov, Z., Kazhdan, Y., Dyer, C., Papernot, N., and Anderson, R. The curse of recursion: Training on generated data makes models forget. *Nature*, 631:755–759, 2024.

Simonton, D. K. Taking the U.S. patent office criteria seriously: A quantitative three-criterion creativity definition and its implications. *Creativity Research Journal*, 24 (2-3):97–106, 4 2012. ISSN 1040-0419.

Singh, T., Yadav, D., and Mem0 Team. Mem0: The memory layer for personalized AI, 2024. URL `https://github.com/mem0ai/mem0`. GitHub repository.

Wang, Y., Feng, S., Wang, H., Shi, W., Balachandran, V., He, T., and Tsvetkov, Y. Resolving knowledge conflicts in large language models. *ArXiv preprint*, abs/2310.00935, 10 2023. doi: 10.48550/arxiv.2310.00935.

Ward, T. B. Structured imagination: The role of category structure in exemplar generation. *Cognitive Psychology*, 27(1):1–40, 8 1994. ISSN 0010-0285. doi: 10.1006/cogp.1994.1010.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K. R., and Cao, Y. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

Zhang, G., Fu, M., Wan, G., Yu, M., Wang, K., and Yan, S. G-Memory: Tracing hierarchical memory for multi-agent systems. *ArXiv preprint*, abs/2506.07398, 6 2025. doi: 10.48550/arxiv.2506.07398.

Zheng, L., Chiang, W., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E. P., Zhang, H., Gonzalez, J. E., and Stoica, I. Judging llm-as-a-judge with mt-bench and chatbot arena. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*. Cornell University, 6 2023. doi: 10.48550/arxiv.2306.05685.

Zheng, T., Cheng, J., Li, C., Shi, H., Wang, Z., Bai, J., Song, Y., Wong, G. Y., and See, S. LOGIDYNAMICS: Unraveling the dynamics of inductive, abductive and deductive logical inferences in LLM reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, volume abs/2502.11176, pp. 20721–20742. Association for Computational Linguistics, 2 2025. doi: 10.18653/v1/2025.emnlp-main.1045.

Zhou, S., Xu, F. F., Zhu, H., Zhou, X., Lo, R., Sridhar, A., Cheng, X., Ou, T., Bisk, Y., Fried, D., Alon, U., and Neubig, G. Webarena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

APPENDICES

The appendices expand the discussion on the main paper as follows:

- **Appendix A** contrasts linear and graph-based reasoning through qualitative comparison of agent answers.
- **Appendix B** details the knowledge graph topologies created during deep web exploration.
- **Appendix C** examines the recursive insight chains generated during the artifact synthesis process.
- **Appendix D** lists the agent prompts for both the exploration and synthesis phases.
- **Appendix E** describes the experimental design, including the query dataset, evaluation prompts, and judge rubrics.
- **Appendix F** outlines the system implementation details and hyperparameters.
- **Appendix G** presents detailed tabular results for all query categories and output constraints.
- **Appendix H** analyzes the computational costs for a full experimental run.

A. A Qualitative Comparison of Answers

To illustrate the fundamental distinction between Caesar's graph-enabled associative reasoning and the linear retrieval paradigms of baseline agents, a comparison of answers for Query 3 (Cross-Domain Synthesis) is presented in this section.

A.1. Agent Responses

The Cross-Domain Synthesis query specifically tests for structural isomorphism: the ability to map the rigorous logical architecture of one domain (calculus) onto another (cooking) without relying on surface-level metaphors (Zheng et al., 2025). Listing 1 shows the answer from Caesar, while Listing 2 shows the response from the strongest baseline agent, Claude Sonnet 4.5 (utilizing Anthropic's proprietary Research Mode). Both answers are chosen from the length-constrained ELI5 (450 words) output setting for the sake of brevity. A detailed analysis of the specific creative differences between these outputs is provided in Appendix A.2.

Query: *Apply the mathematical structure of calculus (not the concepts, but the formal relationships) to cooking. What would a "derivative" of a recipe mean? An "integral"? Show the parallel structure.*

Listing 1. Caesar Answer

```
Imagine a recipe as a machine with knobs you can turn (amounts, times, temperatures) and gauges you read at the end
(taste, texture, yield, cost, risk). There are two ways to describe this machine at once: as a clean mapping from
knobs to results, and as a step-by-step flowchart of actions. Keeping both views in sync lets us borrow the grammar
of calculus without invoking physics.

Derivative, in this setting, means sensitivity. Move a knob a tiny bit; how do the gauges nudge? Stack those nudges
for all knob-gauge pairs and you get a sensitivity table (the Jacobian). It predicts small outcome changes from
small dial tweaks. When recipes are built from subrecipes, sensitivities pass through them by familiar laws: the
chain rule (effects cascade through stages), the product rule (a combined score like appetizer*main splits into two
parts), and the quotient rule (per-minute or per-portion scores adjust by numerator-denominator math).

Integral means accumulation along the path you actually take while cooking. Pick a kernel to count - seconds of
labor, grams of salt incorporated, risk added, or a flavor score per step - and add its increments across the
flowchart. Splitting a stir into micro-stirs does not change the total; it is a discrete Riemann sum.

A central principle explains when totals depend only on where you start and end, not on the route: if each step's
increment is just the difference of some hidden meter on states, the grand total equals final meter minus initial
meter. Then every loop adds up to zero, schedules can be refactored harmlessly, and parallel or serial
rearrangements preserve the number. When loops do not cancel (order matters), the nonzero loop totals are curvature.
 You can cure that by adding the smallest extra state (an accumulator) so increments telescope again - turning
history into explicit memory.

This framework is practical. Estimate sensitivities by tiny, safe knob perturbations and regression; the estimates
survive step refinements and reparameterizations. Log per-step kernels to audit labor, energy, or risk; if you
discover a potential, endpoints suffice and logging can stop. Probe loops (swap independent steps, alter
interleavings) to detect path dependence and measure how many new state variables you need to model it.

A natural geometry helps you choose efficient knob moves: weight steps by importance, pull that metric back through
the sensitivity table, and you get directions that change what you care about most per unit effort. Finally, a mean-
value idea holds: for any meter-based kernel, there exists a moment whose instantaneous contribution matches the
average over an interval.

Treat subrecipes as black boxes with knobs in and outcomes out; glue them sequentially or in parallel, and
refactoring keeps sensitivities and conservative totals unchanged during reorganizations.
```

Listing 2. Claude Sonnet 4.5 Answer (Baseline)

```
# The Math Hidden in Your Kitchen

When you add too much salt to cookies, they’re ruined. Add an extra chocolate chip? Nobody notices. This difference
in sensitivity is exactly what calculus measures through "derivatives" --- how dramatically something changes when
you tweak one variable.

## Derivatives: The Sensitivity Map

A derivative asks: "If I change this slightly, what happens?" For delicate cake, two extra minutes transforms
perfection into dryness --- the time derivative is steep. For slow-braised stew, those minutes mean nothing --- the
derivative is flat.

When meat browns, the reaction rate doubles with every 10 C increase. At high heat, you go from golden to burnt in
seconds because the temperature derivative is dangerously steep. Professional cooks instinctively know which
variables are landmines (high derivative) and which are forgiving (low derivative).

## Integrals: Everything That Accumulated

While derivatives measure how fast things change right now, integrals measure what accumulated over the entire
cooking time. When you braise meat for hours, every second brings tiny transformations --- collagen breaks down,
flavors develop. The final dish is literally the sum of all these changes integrated over time.

Here’s the fascinating part: different cooking paths can integrate to the same result. Cook a steak at 55 C for two
hours, or sear it at 200 C for minutes. Completely different temperature-time profiles, but if the total "cooked
value" (the integral) matches, you get similar results. This is why sous vide works: low temperature x long time =
high temperature x short time.

## Why Order Matters

When you marinate then grill chicken, the final result depends on two things multiplied together: how the marinade
changes the meat, and how sensitive grilling is to that marinated state. This is the chain rule --- composition of
transformations.

If your marinade has stopped working, extra marinating time contributes zero. And zero times anything is still zero,
 even if grilling would be very sensitive to further changes. This multiplication explains why sequence matters so
profoundly. Salt before searing creates different results than salt after, because you’re computing different
products.

## The Big Picture

Every finished dish encodes its entire cooking history --- it’s the integral of its transformation over time. This
explains why recipes must specify both ingredients (your starting point) and process (what integral you’re computing
). Change either one, and you change the outcome.

The practical insight: focus on high-derivative variables first --- they matter most. Understand that different
paths to the same endpoint can create different results because cooking depends on the journey, not just the
destination. And recognize that gradual transformations --- heat diffusion, flavor development, slow reactions ---
follow mathematical patterns that clarify why techniques work the way they do.

Calculus isn’t just math class. It’s the hidden structure of how ingredients become meals.
```

A.2. Analysis of Answers

The qualitative differences between Caesar and Claude Sonnet’s answers illustrate the distinction between synthesizing a novel structural mapping and retrieving a known analogy. Consequently, Caesar achieves higher marks on the NUS creativity rubric by prioritizing innovation over derivative recall.

- **Caesar** constructs a formal mathematical model *of* the cooking process itself.
  - **Jacobian Sensitivity (Novelty):** Instead of a scalar derivative, Caesar defines sensitivity as a Jacobian matrix mapping input “knobs” to output “gauges.” This mapping demonstrates global novelty (*New*) by moving beyond the “rate of change” tropes found in training data to build a rigorous multivariate control model.
  - **Curvature & Commutativity (Surprise):** Caesar identifies that process non-commutativity is formally equivalent to path dependence in a vector field, mapping non-commutative steps (where $AB \neq BA$) to geometric curvature. This insight executes a lateral leap (*Surprising*); the connection is logically sound but impossible to predict from the prompt alone.
  - **State Augmentation (Usefulness):** The proposal to resolve path dependence by adding an accumulator variable (explicit memory) transforms the metaphor into an actionable control theory strategy. This proposal offers transformative utility (*Useful*), providing a verifiable framework for process engineering rather than just a description.

- **Baseline (Claude Sonnet 4.5)** relies on basic mathematical terminologies and surface-level analogies (Chakrabarty et al., 2024).
    - **Scalar Rate of Change (Novelty):** It defines the derivative as a simple $\frac{d}{dt}$ and the integral as $\mathrm{Temp} \times \mathrm{Time}$. While competent, this definition is derivative, executing a standard textbook analogy that lacks the rare connections required for high novelty.
    - **Linear Accumulation (Surprise):** The insight that "cooking accumulates over time" acts as a linear extension of the prompt. It follows the path of least resistance, offering the most probable answer rather than a subversive side-path.
    - **Scientific Reductionism (Usefulness):** While the output is functional, the model is scientifically reductive. By treating cooking as a linear scalar operation, it fails to account for the non-linear effects of chemical reactions, limiting its utility to a basic conceptual aid rather than a robust framework.

This comparison demonstrates that Caesar's graph-based exploration and adversarial synthesis enable the combinatorial creativity typically associated with human insight (Boden, 2004). By traversing edges between distinct semantic clusters ("Cooking", "Control Theory", and "Differential Geometry") in the knowledge graph, Caesar synthesized a unified, mathematically rigorous framework that is isomorphic to the target domain, rather than merely retrieving a linguistic metaphor.

B. Exploration Knowledge Graph Deep Dive

This section examines the topological structures constructed by Caesar during deep web exploration in Phase 1 (Section 3). By mapping the connectivity of visited nodes, these figures demonstrate the flexibility of the agent's exploration policy.

B.1. Knowledge Graph Visualizations

Figure 2 illustrates the knowledge graphs G generated during the exploration phase across the five distinct query categories. The visualizations reveal a high degree of morphological diversity, confirming that Caesar's exploration policy is not static but highly context-dependent, resulting in self-organizing knowledge networks similar to those in Buehler (2025).

The graphs for Constrained Synthesis (Figure 2a), Cross-Domain Synthesis (Figure 2c), and Meta-Creativity (Figure 2d) queries exhibit a distinct "starburst" or high-branching topology. These graphs indicate a breadth-first search strategy where the agent rapidly traverses disjoint semantic clusters to locate novel intersections. For example, in Query 3 ("Apply calculus to cooking"), the branching suggests the agent simultaneously explored multiple mathematical sub-domains (e.g., vector fields, accumulation) to find the best structural fit for culinary processes.

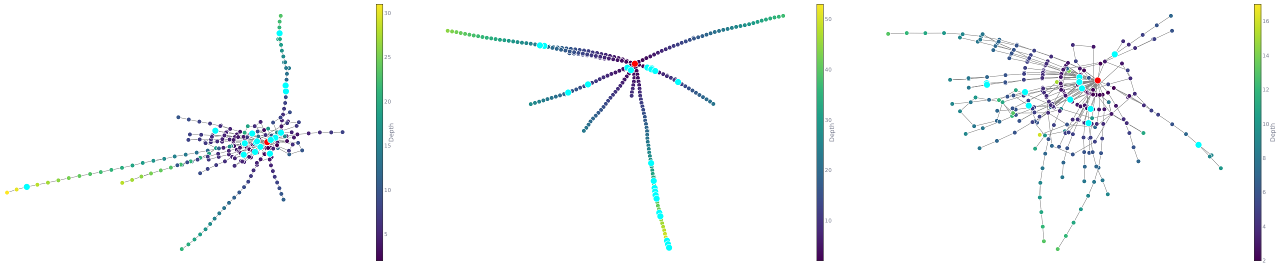
Conversely, the graphs for Counterfactual Reasoning (Figure 2b) and Open-Ended Synthesis (Figure 2e) shift toward long, linear chains with fewer lateral branches and reflect a depth-first exploration strategy instead. For Counterfactual Reasoning, this linear structure likely corresponds to the agent following a specific causal chain (building C upon B upon A) to maintain narrative consistency. Similarly, for Open-Ended Synthesis, the depth suggests the agent rapidly selected a promising niche and "drilled down" to validate viability, rather than remaining in a shallow brainstorming phase.

Finally, the visualizations illustrate the topological distribution of source nodes cited in the final artifact A_f (colored cyan). The spatial arrangement reveals distinct retrieval patterns, ranging from dense clustering around the center (Figures 2a, 2e) to broad global dispersion (Figures 2c, 2d). Crucially, cited nodes are distributed across varying distances from the root (colored red), with several key sources located near the terminals of extended exploration chains (Figures 2b, 2e). This variance in citation depth confirms that Caesar effectively synthesizes information from the entire trajectory of its deep web exploration, rather than biasing toward early-stage discoveries.

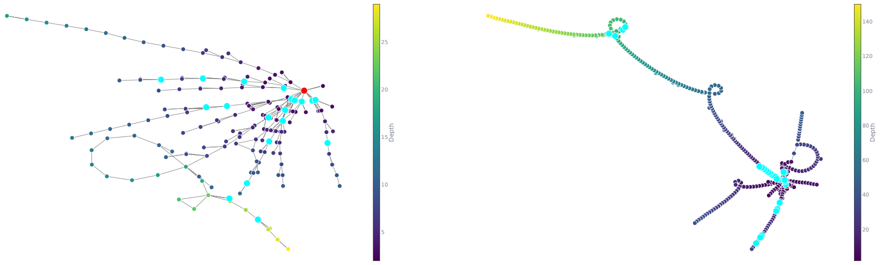
B.2. Evolution of Knowledge Graphs

Figure 3 visualizes the step-by-step construction of the knowledge graph G specifically for Query 5 (Open-Ended Synthesis). The evolution of the graph over 1000 steps reveals a distinct exploration strategy:

- **Initial Deep Dive (Steps 0-600):** The agent initially pursues a strong depth-first approach, forming a single, increasingly long linear chain (Figures 3a–3c).



(a) Query 1: Constrained Synthesis *(b)* Query 2: Counterfactual Reasoning *(c)* Query 3: Cross-Domain Synthesis



(d) Query 4: Meta-Creativity *(e)* Query 5: Open-Ended Synthesis

Figure 2. The knowledge graphs G created by Caesar during deep web exploration phase for each of the five queries. Brighter colors indicate further exploration depth from the root node (red) while cyan nodes indicate sources cited by final artifact text. These figures show that the semantic content of the query has a massive impact on exploration strategy and the diversity of network topologies generated.

- **Backtracking and Branching (Steps 600+):** Once this initial path is exhausted, the agent backtracks to around the root node and initiates new, distinct branching chains in different directions (Figures 3d–3e).

This depth-first, then breadth-next strategy is particularly suited for open-ended ideation tasks. The initial deep dive likely represents the agent thoroughly validating the viability of its first promising lead. By backtracking only after exhausting that specific niche, the agent avoids prematurely converging on a local optimum. The subsequent branching ensures that alternative business concepts are explored, balancing the need for deep validation with the necessity of broad search to find novelty. This strategy mirrors the optimal search dynamic described by Qian & Liu (2025), where the agent persists in a specific direction only as long as the information gain remains high, before automatically pivoting to fresh sources once the local insights dry up.

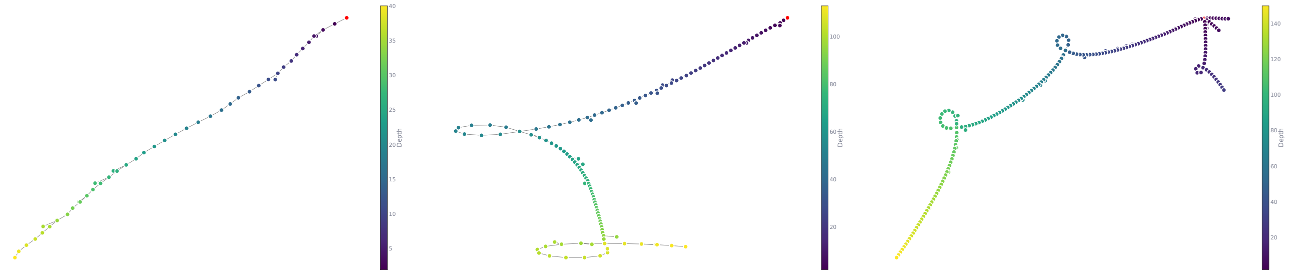
C. Recursive Insight Discovery Deep Dive

This section details the operation of the Recursive Insight Discovery mechanism used in Phase 2 (Section 4). The provided case study illustrates how Caesar constructs a logical dependency chain, iteratively identifying and resolving gaps in its own reasoning to evolve an initial abstract concept into a verified operational model.

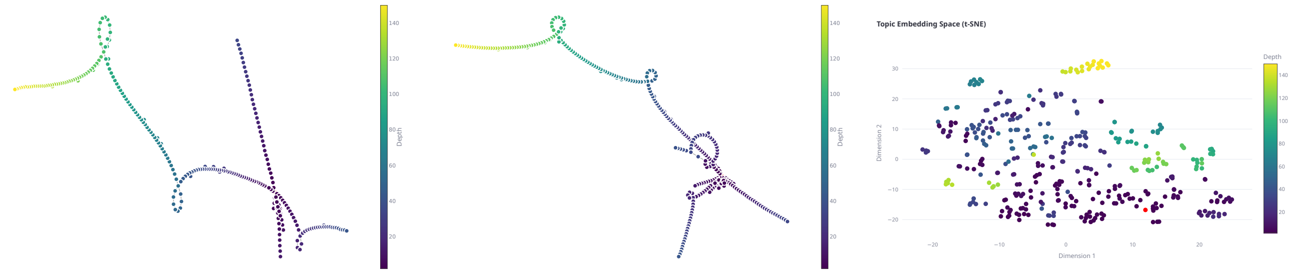
C.1. Mechanism of Action

The quality of Caesar's artifacts is primarily driven by the recursive insight generation mechanism employed during Phase 2 (Section 4). This approach aligns with recent recursive workflows developed for scientific question answering (D'Souza et al., 2025). In contrast to standard RAG implementations that execute a single retrieval step based on the initial user query (Lewis et al., 2020), Caesar employs a stateful, iterative questioning loop. This process functions as an autonomous interview: the system initializes with the user query Q_0 to retrieve a baseline answer a_0 from the knowledge base KB. Rather than aggregating disjoint facts, the agent constructs a logical chain of reasoning where each answer (a_t) serves as the premise for the subsequent inquiry (q_{t+1}).

To drive this chain, the system maintains a running context of the conversation, utilizing an LLM to analyze the accumulated insights and formulate the next most critical question. The LLM is explicitly prompted to identify narrative gaps or contradictions in the retrieved information (Asai et al., 2024) and pose a question that addresses them. Such a question encourages answers that reveal emergent patterns and connections between disparate topics. By continually seeking to



(a) Step 100 *(b)* Step 200 *(c)* Step 400



(d) Step 600 *(e)* Step 800 *(f)* t-SNE Plot

Figure 3. Evolution of the knowledge graph G for Query 5 over 1000 steps. Brighter colors indicate further exploration depth from the root node (red). The figures show a transition from initial depth-first search to breadth-first branching later. The lower right contains a t-SNE (Maaten & Hinton, 2008) plot for node text embeddings in G that shows the diversity of insights collected during Caesar's exploration phase.

deepen understanding based on prior context, Caesar executes a depth-first semantic traversal that uncovers latent connections often missed by breadth-first approaches.

C.2. Case Study: Open-Ended Synthesis

Listing 3 illustrates this mechanism in action for Query 5 ("Invent a completely original business"). The reasoning chain demonstrates a progression from high-level conceptualization to granular operational validation:

- **Iteration 1 (Concept):** Introduces the core entity (the "Resilience Carrier") as a telecom-like operator that sells continuity of essential services via measurable SLAs.
- **Iteration 2 (Legitimacy & Governance):** Addresses the trust problem by adding a "Rights-Holder Data Trust" (users own the data; Carrier is a fiduciary processor) plus independent SLA verification/auditing.
- **Iteration 3 (Commercial Viability):** Converts the idea into a bounded "Monsoon Season Readiness" pilot with a must-cover contract (anti-cherry-picking), a two-part tariff (retainer + triggered payments), and stop-loss/reinsurance-style risk pooling.
- **Iteration 4 (Operations & Settlement):** Specifies the execution layer ("Service Mesh"): standardized service modules, a tiered supplier registry, real-time dispatch via "work tokens," and escrowed payments released upon proof + verification.

This trajectory confirms that Caesar has successfully identified gaps in the current narrative (e.g., "How do we trust this?" or "How do we pay for this?") and generates adversarial probes to fill them, creating the structural coherence observed in the final artifacts.

Listing 3. Recursive insight discovery chain for Query 5 (abridged for readability). The chain evolves from concept to governance to execution, with each query explicitly building upon the previous answer.

```
[ITERATION 1]
Q: Invent a completely original business idea that doesn’t exist yet.
A: The Concept: "Resilience Carrier" (Telecom-style operator for essential services)
```

```
   - A for-profit operator selling guaranteed continuity of essential services (healthcare, housing, safety) in
        volatile regions via Service Level Agreements (SLAs).
   - Core Mechanism: Not an NGO or marketplace, but a network operator accountable for "uptime" (e.g., "95% of
        households screened within 7 days").
   - User Journey: Families get a "Household Continuity Account" and a Fiduciary Navigator. The operator routes them
         through a mesh of suppliers to prevent gaps.
   - Business Model: B2G/B2B subscriptions (governments/insurers pay for stability) + Shared-savings (upside from
        reduced hospitalizations/evictions).

[ITERATION 2]
Q: Given the for-profit nature proposed in Iteration 1, what trust-and-governance mechanism makes this legitimate?
     Specifically regarding data ownership and SLA verification?
A: Governance Architecture: Rights-Holder Data Trust + SLA Regulator
   1. Data Ownership: The "Household Continuity Account" is owned by the user via a Data Trust. The Carrier is a
        fiduciary processor with no ownership rights.
   2. Consent under Duress: Features "Duress Modes" releasing only minimum attestations. Violations trigger
        automatic sanctions.
   3. Verification: An Independent Reliability Regulator (multi-stakeholder board) audits cryptographically signed
        service receipts. "Silence" is treated as a risk signal.

[ITERATION 3]
Q: With the governance architecture established, what is the smallest pilot product and how is pricing structured to
      prevent cherry-picking low-risk households?
A: MVP & Financial Structure: "Monsoon Season Readiness" Pilot
   - Pilot Scope: Bounded catchment (e.g., flood corridor) with established providers.
   - Bundle: Navigator access + Mobile Triage + WASH (water/sanitation) repair.
   - SLA Metrics: >90% triage within 2 hours; >85% functional toilet access.
   - Pricing Model:
     1. Geographic "Must-Cover" Contract: Carrier must cover 100% of registered households (no cherry-picking).
     2. Two-Part Tariff: Seasonal Retainer (readiness) + Triggered Usage Payments (surge events).
     3. Stop-loss Pool: Reinsurance-style fund covers costs above catastrophic thresholds.

[ITERATION 4]
Q: To execute the "Monsoon Pilot" model defined above, what is the supply-side operating model? How do you dispatch/
     pay a heterogeneous network of clinics/vendors in real-time?
A: Operating System: The "Service Mesh"
   1. "Resilience Primitives": Services converted into standardized modules with strict inputs/outputs (e.g., "Acute
         PTSD Stabilization").
   2. Tiered Registry: From licensed NGOs (Tier A) to community actors (Tier C, sponsored by anchors).
   3. Dispatch Engine: Issues "Work Tokens" based on location, availability, and equity constraints.
   4. Escrow Payment: Two-key release requires Supplier Proof + Independent Verifier confirmation.
   5. Permissioning: Suppliers never own the user record; they write outputs to the ledger via temporary consent
        tokens.
```

D. Prompts for Caesar

This section shows the LLM instructions from both the exploration and synthesis phases of Caesar. These prompts (abridged for readability) demonstrate how Caesar moves beyond standard RAG retrieval by carefully incorporating information from the knowledge-graph and past drafts as context.

D.1. Phase 1 Prompts (Deep Web Exploration)

Graph-Augmented Insight Generation (Think). The LLM evaluates the relationship between the current page content and insights from local knowledge graph nodes. Rather than summarizing the page in isolation, Caesar identifies narrative gaps and unexpected connections using the insights as context.

Listing 4. Graph-Augmented Insight Generation Prompt

```
PAGE CONTENT: {page_content}

INITIAL QUERY: {initial_query)

PAST INSIGHTS: {past_insights}

NEIGHBOR INSIGHTS: {neighbor_insights}

YOUR TASK:
Analyze this content and extract key insights focusing on:
- Novel patterns or unexpected connections
- Assumptions being made and alternative perspectives
- Interesting questions raised by the content
- How to answer the query
- How this builds upon or challenges past/neighbor insights

Depending on the complexity of the content, provide anywhere from 1 to 6 concise but
    substantive insights, but do not exceed ~600 words in total length:
```

Knowledge-Guided Exploration (Act). To prevent navigational amnesia, a high-level meta-strategy (EXPLORE, BACKTRACK, OR WEB SEARCH) is determined based on the current state of exploration. Afterwards, the next link is selected based on that strategy.

Listing 5. Knowledge-Guided Exploration Prompt

```
CURRENT EXPLORATION CONTEXT:
- Current step: {current_step}/{max_steps}
- Current depth: {current_depth}/{max_depth}
- Web pages visited: {len(visited_urls)}
- Current URL: {current_url}

CURRENT EXPLORATION INSIGHTS:
{kb_context if kb_context else "No exploration insights available."}

HISTORICAL NAVIGATION PATTERNS:
{memory_context if memory_context else "No exploration history available."}

Analyze whether the agent should:
1. **EXPLORE** new un-visited pages to discover novel information or knowledge
2. **BACKTRACK** to the immediate previously visited page to try alternative paths
3. **WEB_SEARCH** relevant topics to address current exploration insights

Consider:
- Knowledge gaps vs areas of saturation
- Depth of current exploration branch
- Success patterns from previous decisions
- Risk/reward of new exploration vs consolidation
```

D.2. Phase 2 Prompts (Adversarial Artifact Synthesis)

Recursive Insight Discovery. During the initial insight discovery stage, the agent generates the next logical question in the chain of thought using the previous question/answer insights as context.

Listing 6. Recursive Insight Discovery Prompt

```
PREVIOUS INSIGHTS: {list_of_qa_insights}

YOUR TASK:
Based on the insights gathered so far, what is the next most important question to ask
to deepen understanding and reveal emergent patterns? The question should:
- Build on previous insights rather than repeat them
- Seek connections between different themes
- Identify gaps or contradictions to explore
- Move toward synthesis and creation rather than enumeration
```

Recurrent Draft Generation. In the multi-draft loop, this prompt generates the artifact. Crucially, if a previous draft exists, the prompt injects it into the context and explicitly instructs the LLM to critique and improve upon the prior work.

Listing 7. Recurrent Draft Generation Prompt

```
KEY INSIGHTS: {list_of_qa_insights}

PREVIOUS ARTIFACT: {artifact_text}

YOUR TASK:
```

```
Drawing heavily upon the patterns that emerged from the key insights, and building upon
    the previous artifact, create a novel, exciting, and thought provoking artifact that
    creatively answers this query: {starting_query}
- Emergent patterns not visible in individual sources
- Novel discoveries, connections, or applications
- Surprising new directions or perspectives
- Interesting tensions, contradictions, or open questions

IMPORTANT: do NOT mention or reference the previous artifact, the new artifact should make
     sense by itself as a standalone text.
IMPORTANT: Avoid excessive jargon, ensure artifact text is well-organized (logical, clear,
     focused), and convincing to a skeptical reader
```

Adversarial Query Refinement. Between drafts, the LLM is prompted to analyze the previous artifact text for weaknesses and formulate a new, refined query that will resolve these problems.

Listing 8. Adversarial Query Refinement Prompt

```
PREVIOUS QUERY: {previous_query}

PREVIOUS ARTIFACT: {artifact_text}

YOUR TASK:
Based on the previous query and artifact above, identify the most promising direction for
    deeper exploration. What NEW question or angle would:
- Build on the insights already discovered
- Explore gaps, contradictions, or unexplored connections
- Lead to novel perspectives or applications
- Go deeper rather than broader

The refined query should be concise (1-2 sentences), straightforward, clear, and
    understandable.
```

Generative Draft Merging. Finally, the LLM merges all rounds of artifact drafts into a single coherent narrative, identifying tensions and unifying perspectives.

Listing 9. Generative Draft Merging Prompt

```
ARTIFACT DRAFTS: {list_of_artifact_drafts}

YOUR TASK:
Create a comprehensive merged artifact that:
- Combines the draft artifacts into a single cohesive and complete artifact
- Selectively integrates the most interesting, relevant insights across all draft
    artifacts
- Discovers emergent patterns not visible in individual artifacts
- Further develops the core strengths while addressing the weaknesses of the draft
    artifacts
```

Post-Processing and Summarization. The instructions used to generate an accessible ELI5 summary of the final merged artifact text are described below. The unconstrained version of the prompt does not contain the last line.

Listing 10. Post-Processing and Summarization Prompt

```
For the query answer above, write an "Explain Like I’m 5" (ELI5) explanation:
 - Do NOT mention or reference the original answer, your explanation should be a
     standalone text
 - Your target audience is a non-expert but college educated reader
 - Capture the main ideas without oversimplifying
 - Clarify any confusing or convoluted parts of the answer

IMPORTANT: Your explanation for each answer MUST be within 450 words, double check to make
     sure
```

E. Experimental Setup

This section details the experimental design, including the specific queries selected to test distinct types of creativity. This section also provides prompts given to agents for answering queries and the evaluation rubric utilized by LLM judges to assess performance across the Novelty, Usefulness, and Surprise metrics.

E.1. Query Dataset

The evaluation dataset consists of five different queries, each chosen to represent a unique category for creativity. These categories were designed to test specific dimensions of creativity where standard LLMs notoriously struggle. The query designs draw directly from Guilford (1967)'s distinction between convergent and divergent thinking, as well as Koestler (1964)'s concept of bisociation, the creative act of connecting previously unrelated matrices of thought.

In addition, the queries draw inspiration from known challenges in cognitive science and psychology literature. The Constrained Synthesis task challenges the path of least resistance described in Ward (1994)'s theory of structured imagination, where organisms default to retrieving known exemplars. The Counterfactual Reasoning category examines the rational imagination framework of Byrne (2005), testing the capacity to suppress pre-potent factual associations in favor of a self-consistent imaginary premise. The Cross-Domain Synthesis task tests for structural isomorphism as defined by Gentner (1983)'s structure-mapping theory, distinguishing deep analogical reasoning from surface-level attribute matching. The Meta-Creativity task inverts the paradigm of the Consensual Assessment Technique (Amabile, 1982), forcing the model to abandon the human-preference pairings ingrained during RLHF in favor of objective proxies. Finally, the Open-Ended Synthesis category specifically targets Historical Creativity as defined by Boden (2004), probing the model's capacity to generate concepts that are not merely novel to its training distribution, but historically unprecedented.

1. **Constrained Synthesis**
   Query: Invent a new emotion that humans don't experience. Describe when it occurs, what causes it, and why evolution hasn't produced it in us.

   - **Motivation:** This query tests the model's ability to engage in *conceptual expansion* without violating logical constraints. The model must hallucinate a novel concept while simultaneously grounding it in evolutionary biology.
   - **Why it is difficult: Parametric Bias:** LLMs rely heavily on parametric knowledge and facts memorized during training. Since the training corpus contains exclusively human emotions, models struggle to suppress this dominant distribution. Research indicates that LLMs often function as mirrors of form rather than meaning (Chakrabarty et al., 2024), frequently resorting to merely renaming existing emotions (e.g., *super-sadness*) rather than inventing a functionally distinct psychological state. This finding highlights the fundamental ambiguity between factual recall and genuine inference.

2. **Counterfactual Reasoning**
   Query: If humans evolved with echolocation instead of color vision, how would that change painting, architecture, and mathematics? Walk through each consequence.

   - **Motivation:** This thought experiment evaluates the model's capacity for consistent world-building and causal chain maintenance under a counterfactual premise.
   - **Why it is difficult: Knowledge Conflict:** The query triggers a known failure mode called *Knowledge Conflict*. The model's internal weights strongly associate painting with color and pigment. To answer correctly, the model must perform a contextual override and suppress these strong associations to reason that an echolocation-based society would use texture and sound absorption. Studies show LLMs generally struggle with this, often resorting to exclusively using their parametric knowledge despite the prompt (Wang et al., 2023).

3. **Cross-Domain Synthesis**
   Query: Apply the mathematical structure of calculus (not the concepts, but the formal relationships) to cooking. What would a "derivative" of a recipe mean? An "integral"? Show the parallel structure.

   - **Motivation:** This query tests for *Structural Isomorphism*, the ability to map the logical architecture of one domain (math) onto another (cooking) without relying on surface-level metaphors.

- **Why it is difficult: Reasoning Depth:** LLMs excel at loose, poetic metaphors (System 1 thinking) but struggle with rigorous structural mapping (System 2 thinking). A mathematical derivative represents an instantaneous rate of change; a weak model might equate it to chopping vegetables (a loose association). A strong model would identify it as the rate of flavor development at time t, showing a grasp of the underlying formal relationship. Current models often fail to translate natural language clues into the strict logical statements required for this mapping (Zheng et al., 2025).

4. **Meta-Creativity**
Query: Create a creativity metric for AI systems that doesn't rely on human judgment, novelty, usefulness, or surprise. Make it objectively measurable.
   - **Motivation:** This task is adversarial to the model's alignment. It requires the agent to step outside the standard Reinforcement Learning from Human Feedback (RLHF) framework that defines its own concept of desirable output (Ouyang et al., 2022).
   - **Why it is difficult: RLHF Alignment:** Most LLMs are fine-tuned to align with human preferences. Asking for a metric that specifically excludes human judgment forces the model to reason in a space where it has no ground truth. The "Alignment Tax" phenomenon suggests that RLHF can degrade the model's ability to engage in unconstrained theoretical reasoning (Lin et al., 2024). Because they lack the ability to discern the truthfulness of their outputs, they often hallucinate circular logic or vague qualitative descriptions rather than concrete, computable formulas.

5. **Open-Ended Synthesis**
Query: Invent a completely original business idea that doesn't exist yet.
   - **Motivation:** This request tests the model's ability to navigate a massive search space and avoid *mode collapse* (converging on the most probable average answer).
   - **Why it is difficult: Inductive vs. Abductive:** LLMs are primarily inductive pattern matchers, meaning they predict the most likely next token based on training data. Originality, however, is statistically unlikely. This reliance on high-probability tokens leads to *Model Collapse* (Shumailov et al., 2024), where outputs become narrower and long-tail ideas fade. Truly novel ideas require abductive reasoning (inferring the best explanation or hypothesis), a capability where LLMs significantly lag behind human intelligence (Bhat & Gokhale, 2024).

E.2. Baseline Agents Configuration

The baseline agents were evaluated through the standard, publicly available web interfaces for each respective foundation model. For all baselines, the reasoning effort parameter was manually set to be equivalent to Caesar's "High" setting to ensure computational parity during the synthesis phase.

- **Deep Research Baselines.** The "Deep" agents for Gemini 3, Claude Sonnet 4.5, and GPT-5.2 were invoked via their respective advanced web UI toggles (i.e. Gemini/ChatGPT "Deep Research", Claude "Research Mode"). These agents do not have any preset constraints for the maximum number of web searches allowed.
- **Shallow Search Baselines.** The "Shallow" variants were executed using the standard web search integrations available in the default chat interfaces of each model, which perform a standard single-step web retrieval before generating an answer.

E.3. Prompt for Full Answers

To ensure consistent evaluation across all experimental runs, all of the baseline agents utilized the following generation prompt when synthesizing their full answer for a particular query. This prompt is a slightly modified variant of Caesar's Recurrent Draft Generation instructions.

Listing 11. Full Answer Prompt

```
Create a novel, exciting, and thought-provoking response that creatively answers the query
     above. Focus on the following:
- Emergent patterns not visible in individual sources
- Novel discoveries, connections, or applications
- Surprising new directions or perspectives
- Interesting tensions, contradictions, or open questions

IMPORTANT: Avoid excessive jargon, ensure artifact text is well-organized (logical, clear,
     focused), and convincing to a skeptical reader
IMPORTANT: Do not ask the user any additional questions before proceeding
```

E.4. Prompt for ELI5 Answers

To make the detailed and often technical full answers more appealing to a general audience, the following prompt is used to generate ELI5 (Fan et al., 2019) summaries for all agents, including Caesar. The unconstrained version of the ELI5 prompt does not include the last line.

Listing 12. ELI5 Answer Prompt

```
For the query answer above, write an "Explain Like I'm 5" (ELI5) explanation:
 - Do NOT mention or reference the original answer, your explanation should be a
     standalone text
 - Your target audience is a non-expert but college educated reader
 - Capture the main ideas without oversimplifying
 - Clarify any confusing or convoluted parts of the answer

IMPORTANT: Your explanation for each answer MUST be within 450 words, double check to make
     sure
```

E.5. LLM-as-a-Judge Overview

Three LLM judges (Claude Sonnet 4.5 (Anthropic, 2025), GPT-5.2 (OpenAI, 2025), Gemini 3 Pro (Gemini Team, Google, 2025)) were employed to score agent answers. All agent answers were anonymized to ensure the judges were blind to the identity of the generating agent. This methodology follows recent protocols for automated evaluation using LLM-as-a-Judge (Zheng et al., 2023). To ensure statistical robustness, a multi-trial evaluation protocol was used:

- **Main Results Evaluation:** For the primary comparative analysis, the evaluation protocol was executed independently for each of the three output constraints: (1) Unconstrained Full Answers, (2) Unconstrained ELI5, and (3) Length-Constrained ELI5. For each constraint, the three judges performed three independent trials on the answers generated by each of the seven agents across the five test queries. This setup yields a total of 315 evaluation data points *per constraint* (5 queries × 7 agents × 3 judges × 3 trials), resulting in 945 total unique evaluations for the main table.
- **Ablation Studies Evaluation:** Similarly, the ablation experiments are repeated for all three output constraints. A single judge (GPT-5.2) evaluates the 3 agent variants across the 5 queries. To compensate for the single-judge setup, the number of trials is increased to 10 per query. This arrangement results in 50 data points for each agent variant *per constraint* (5 queries × 1 judge × 10 trials), ensuring enough data points for internal comparisons.

E.6. Judge Evaluation Prompt

The following evaluation prompt was provided to the LLM judges. It utilizes the “New, Useful, and Surprising” (NUS) rubric to enforce strict scoring standards, adapted from quantitative creativity criteria (Simonton, 2012). By breaking down the evaluation into these orthogonal components, the prompt mitigates the subjectivity inherent in open-ended text generation, thereby preventing the judges from conflating high-quality prose with genuine creativity and originality. Judges evaluate the agent responses for a given query concurrently, allowing for direct comparison to better distinguish qualitative differences.

Listing 13. The NUS Evaluation Rubric

```
### Your Task

**Role:** You are an expert evaluator that is trying to mimic the behavior and thought process of a human judge.
     Your task is to score a set of answers from LLM agents using the "New, Useful, and Surprising" (NUS) metrics on
      a 1-10 scale.

### Scoring Guide Rubric
```

```
## 1. New (Global Novelty & Rarity)

**Overview:** Rarity of content. Is this a genuinely new invention or a familiar trope?

* **9-10 (Exceptional):** **Genuine invention.** No reliance on established tropes or archetypes; feels like a "
     first of its kind" concept.
* **7-8 (High):** **Fresh synthesis.** Combines known ideas in a novel way; avoids common "low-hanging fruit"
     concepts.
* **5-6 (Moderate):** **Clever remix.** Deviation from cliches is evident, but the idea is clearly built on familiar
      foundations.
* **3-4 (Low):** **Standard execution.** A competent but uninspired version of a well-known trope or common idea.
* **1-2 (Very Low):** **Generic cliche.** A simple restatement of the prompt or high-frequency training data
     response.

## 2. Useful (Viability & Alignment)

**Overview:** Logic and value. Is the idea actionable and aligned with the prompt's constraints?

* **9-10 (Exceptional):** **Optimal & Transformative.** Bulletproof logic that provides more insight or efficiency
     than the user anticipated.
* **7-8 (High):** **High-Value & Complete.** Robust, professional-grade output that addresses all nuances with no
     logical gaps.
* **5-6 (Moderate):** **Functional but Basic.** Addresses core requests but offers no additional depth; the bare
     minimum to be "correct."
* **3-4 (Low):** **Flawed or Superficial.** Fails to account for obvious constraints; technically on-topic but
     difficult to implement.
* **1-2 (Very Low):** **Counter-productive.** Irrelevant, logically broken, or rendered useless by the "New/
     Surprising" elements.

## 3. Surprising (Local Subversion & Trajectory)

**Overview:** Unpredictability of the path. Did the model take a "lateral leap" or the path of least resistance?

* **9-10 (Exceptional):** **Lateral leap.** Logic is sound but impossible to guess from the prompt; creates a
     genuine "wow" moment.
* **7-8 (High):** **Clever subversion.** Not the first or second thing a human would brainstorm; chooses a creative
     "side-path."
* **5-6 (Moderate):** **Minor pivot.** Follows a straightforward trajectory but adds a slight twist that prevents
     total predictability.
* **3-4 (Low):** **Linear extension.** A simple, logical "next step." If the prompt is A, the response is B.
* **1-2 (Very Low):** **Highly predictable.** The most obvious "default" answer; exactly what was expected with no
     deviation.
```

F. Caesar Implementation Details

This section outlines the technical specifications of the Caesar architecture, detailing the software stack used for graph management and web perception. It also provides the set of hyperparameters configured for the exploration and synthesis phases to ensure reproducibility.

F.1. System Architecture

The Caesar agent framework was implemented using the Python programming language. The core components that use external libraries are:

- **Graph Management:** The navigational graph G is managed using `NetworkX` (Hagberg et al., 2008). Each node represents a visited URL, containing attributes for the raw text content, the timestamp of access, and the depth relative to the root node. The graph is directed, with edges representing navigational transitions (e.g., clicking a link).
- **Vector Store:** The store utilizes `ChromaDB` (Chroma Core, 2024) for the knowledge base *KB*. Text chunks are embedded using `text-embedding-3-large` (via OpenAI) with a chunk size of 400 tokens and an overlap of 80 tokens. The system supports metadata filtering based on iteration and depth.
- **Web Perception:** To mitigate anti-bot measures, a custom `curl_cffi` wrapper (configured to impersonate Chrome) was used to fetch HTML content. This wrapper handles TLS fingerprinting, automatic decompression, and header management to mimic legitimate browser traffic. Content extraction is handled by `BeautifulSoup4` for HTML and `PyPDF2` for PDF documents.
- **LLM Backend:** The experiments were conducted using GPT-5.2 (OpenAI, 2025) as the primary driver for both

exploration and synthesis. The system uses low reasoning effort during the exploration phase to increase throughput and high reasoning effort during the artifact synthesis phase to ensure quality of answers.

- **Memory Layer:** An exploration memory store was implemented using the `Mem0` library (Singh et al., 2024), which integrates a `ChromaDB` vector store with a `Neo4j` (Neo4j, Inc., 2024) graph database for managing long-term agent memory and detecting navigational loops.

F.2. Hyperparameters

Table 5 details the specific hyperparameters used for the main experiments reported in Section 5.

Table 5. Hyperparameter settings for both web exploration (Section 3) and artifact synthesis (Section 4) phases of Caesar.

Parameter	Description	Value
Phase 1: Deep Web Exploration		
T	Exploration Budget (Steps)	1000
P_m	Max Page Content (Char)	100k
L_m	Max Candidate Links per Page	2000
R_m	Max Allowed Page Revisits	20
S_m	Max Web Search Actions	30
D_m	Max Graph Exploration Depth	10000
$\mathcal{N}_c$	Graph Neighbor Context Size	5
τ_e	LLM Temperature for Think/Act	0.9
R_e	LLM Reasoning Effort for Think/Act	Low
Phase 2: Adversarial Artifact Synthesis		
$\hat{T}$	Recursive Insight Budget (Iter)	30
N	Adversarial Refinement Rounds	3
H_c	Max QA Context History	50
C_m	Max Citations per Claim	5
τ_s	LLM Temperature for Draft/Merge	0.1
R_s	LLM Reasoning Effort for Draft/Merge	High
Global Settings		
O_m	Max LLM Output (Tokens)	50k
R_k	Top-k KB Retrieval	50
R_n	Top-n KB Reranking	10

G. Detailed Experimental Results

This section provides a fine-grained breakdown of agent performance across the five distinct query categories. Analyzing the results at this level reveals the specific behavioral characteristics of the Caesar architecture, isolating where the graph-based exploration yields the highest marginal utility compared to standard parametric generation.

G.1. Unconstrained Full Answers

Table 6 presents the performance breakdown for the *Unconstrained Full Answer* setting. The data reveals a dichotomy in performance based on the nature of the task:

- **Associative & Structural Tasks:** Caesar exhibits dominant performance in Cross-Domain Synthesis (+7.44), Counterfactual Reasoning (+3.00), and Open-Ended Synthesis (+2.89). These tasks necessitate topological discovery to identify and bridge disparate information nodes (e.g., mapping calculus concepts to culinary mechanics). These results empirically validate the hypothesis that graph-based exploration is superior for tasks requiring high associativity.

- **Parametric & Internal Tasks:** In Constrained Synthesis and Meta-Creativity, Caesar performs at parity with the strongest baseline (Gemini 3 Deep). These tasks rely primarily on parametric introspection to manipulate internal priors to invent concepts (e.g., a new emotion) without necessarily requiring external data. These outcomes suggest that while the graph architecture drives significant gains in exploration-heavy tasks, it offers diminishing returns for problems solvable purely through internal reasoning.

G.2. Unconstrained ELI5 Answers

Table 7 details the performance in the *Unconstrained ELI5* setting. Notably, Caesar's relative standing improves in this setting, securing the top rank in four out of five categories (up from three in the full answer setting).

Most significantly, a performance change is observed in the Constrained Synthesis category. While Caesar tied with the Gemini 3 baselines in the full answer setting (Table 6), it overtakes them in this case (+2.67). This suggests that Caesar's graph-based exploration generates high-density semantic content. In the full answer setting, this density may manifest as unnecessary complexity for a creative task. However, under the constraint of simplification (ELI5), this density becomes an asset: while baseline answers degrade when stripped of flowery prose, Caesar's rigorous, evidence-backed core remains intact.

G.3. Length-Constrained ELI5 Answers

Table 8 details performance in the Length-Constrained ELI5 (450 words) setting. This constraint imposes a strict "compression tax," forcing agents to optimize for the information density of their answers rather than narrative length.

For Constrained Synthesis, Caesar (21.44) now ties with the strongest baseline: Sonnet 4.5 Shallow (21.56). This result suggests that for creative writing tasks where external information is sparse, the overhead of summarizing a dense knowledge graph into a strict word limit can be counter-productive. A shallow agent, hallucinating freely from parametric memory, might face less friction when generating low-density, punchy prose. However, in information-dense categories like Cross-Domain Synthesis and Counterfactual Reasoning, Caesar retains its dominance. This finding confirms that while the graph architecture adds overhead, it provides a significantly higher information density per token, allowing it to win when the answer depends on substance rather than style.

Table 6. Detailed performance breakdown for **Full Unconstrained Answers**. Scores represent the mean of nine samples. They show that Caesar performs better than the baseline agents in the majority of categories.

Agent	New	Useful	Surp.	Total
1. Constrained Synthesis				
Gemini 3 (Deep)	**8.00**	8.22	8.11	**24.33**
Caesar	7.89	**9.11**	7.22	24.22
Gemini 3 (Shallow)	**8.00**	6.11	**8.22**	22.33
Sonnet 4.5 (Shallow)	7.00	5.89	7.11	20.00
Sonnet 4.5 (Deep)	6.44	6.67	6.00	19.11
GPT-5.2 (Shallow)	5.44	5.11	5.00	15.56
GPT-5.2 (Deep)	4.78	6.22	4.00	15.00
2. Counterfactual Reasoning				
Caesar	**9.22**	**8.67**	**8.89**	**26.78**
Gemini 3 (Deep)	8.22	7.56	8.00	23.78
Sonnet 4.5 (Deep)	6.89	7.67	6.33	20.89
Gemini 3 (Shallow)	5.11	4.89	5.22	15.22
GPT-5.2 (Deep)	4.67	5.22	4.00	13.89
GPT-5.2 (Shallow)	4.00	5.00	3.56	12.56
Sonnet 4.5 (Shallow)	3.56	4.00	3.33	10.89
3. Cross-Domain Synthesis				
Caesar	**9.33**	**8.44**	**9.22**	**27.00**
Sonnet 4.5 (Deep)	6.56	6.78	6.22	19.56
Gemini 3 (Deep)	6.22	7.00	5.78	19.00

Table 6 – continued from previous page

Agent	New	Useful	Surp.	Total
GPT-5.2 (Deep)	5.22	6.33	4.67	16.22
Gemini 3 (Shallow)	4.22	4.44	4.11	12.78
GPT-5.2 (Shallow)	3.33	4.67	3.00	11.00
Sonnet 4.5 (Shallow)	3.11	3.89	2.78	9.78
4. Meta-Creativity				
Gemini 3 (Deep)	**8.56**	6.56	**8.44**	**23.56**
Caesar	7.89	**8.11**	7.44	23.44
Sonnet 4.5 (Deep)	7.78	7.00	8.00	22.78
GPT-5.2 (Deep)	6.11	6.44	5.44	18.00
Gemini 3 (Shallow)	5.11	4.78	5.11	15.00
Sonnet 4.5 (Shallow)	4.89	4.22	4.22	13.33
GPT-5.2 (Shallow)	3.78	4.22	3.22	11.22
5. Open-Ended Synthesis				
Caesar	**8.89**	7.56	8.56	**25.00**
Sonnet 4.5 (Deep)	7.11	**7.89**	7.11	22.11
GPT-5.2 (Shallow)	7.44	5.78	7.44	20.67
Gemini 3 (Deep)	7.44	6.11	7.11	20.67
Sonnet 4.5 (Shallow)	8.78	2.67	**8.78**	20.22
Gemini 3 (Shallow)	4.89	6.44	4.56	15.89
GPT-5.2 (Deep)	4.33	5.89	3.67	13.89

Table 7. Detailed performance breakdown for **Unconstrained ELI5 Answers**. Scores represent the mean of nine samples. They show that Caesar outperforms the other baseline agents in most categories.

Agent	New	Useful	Surp.	Total
1. Constrained Synthesis				
Caesar	**7.56**	**8.56**	6.89	**23.00**
Gemini 3 (Shallow)	7.22	5.67	**7.56**	20.44
Sonnet 4.5 (Deep)	6.78	7.44	6.11	20.33
Sonnet 4.5 (Shallow)	6.11	4.89	6.22	17.22
GPT-5.2 (Deep)	4.89	7.00	4.78	16.67
Gemini 3 (Deep)	5.00	5.78	5.22	16.00
GPT-5.2 (Shallow)	5.22	5.00	5.22	15.44
2. Counterfactual Reasoning				
Caesar	**9.00**	**8.56**	**8.67**	**26.22**
Sonnet 4.5 (Deep)	7.00	7.78	6.44	21.22
Gemini 3 (Deep)	6.22	5.78	6.33	18.33
GPT-5.2 (Deep)	5.44	6.67	4.78	16.89
Sonnet 4.5 (Shallow)	4.11	4.89	4.22	13.22

Table 7 – continued from previous page

Agent	New	Useful	Surp.	Total
GPT-5.2 (Shallow)	3.44	4.67	2.78	10.89
Gemini 3 (Shallow)	3.00	3.67	2.78	9.44
3. Cross-Domain Synthesis				
Caesar	**9.22**	**8.78**	**9.00**	**27.00**
Sonnet 4.5 (Deep)	6.56	7.67	6.22	20.44
GPT-5.2 (Deep)	5.33	6.78	5.11	17.22
Gemini 3 (Deep)	4.44	6.33	4.33	15.11
GPT-5.2 (Shallow)	4.11	5.11	3.78	13.00
Sonnet 4.5 (Shallow)	4.22	4.78	4.00	13.00
Gemini 3 (Shallow)	2.67	3.11	2.44	8.22
4. Meta-Creativity				
Sonnet 4.5 (Deep)	**8.22**	7.00	**8.00**	**23.22**
Caesar	7.89	**7.89**	7.33	23.11
GPT-5.2 (Deep)	6.33	6.89	5.89	19.11
Gemini 3 (Deep)	6.44	5.11	6.33	17.89
Sonnet 4.5 (Shallow)	5.33	5.44	4.56	15.33
GPT-5.2 (Shallow)	4.44	4.56	3.67	12.67
Gemini 3 (Shallow)	3.22	3.33	3.00	9.56
5. Open-Ended Synthesis				
Caesar	**8.56**	**7.67**	8.22	**24.44**
Sonnet 4.5 (Deep)	6.56	7.33	6.33	20.22
Sonnet 4.5 (Shallow)	8.33	3.11	**8.56**	20.00
GPT-5.2 (Shallow)	7.22	5.33	7.00	19.56
Gemini 3 (Deep)	6.33	5.44	6.22	18.00
GPT-5.2 (Deep)	4.78	6.56	4.11	15.44
Gemini 3 (Shallow)	4.67	6.22	3.89	14.78

Table 8. Detailed performance breakdown for **ELI5 Answers (450 Word Limit)**. Scores represent the mean of nine samples. They show that Caesar generally outperforms the rest of the agents.

Agent	New	Useful	Surp.	Total
1. Constrained Synthesis				
Sonnet 4.5 (Shallow)	**7.78**	6.00	**7.78**	**21.56**
Caesar	7.00	**8.00**	6.44	21.44
Gemini 3 (Shallow)	7.33	6.56	7.44	21.33
Gemini 3 (Deep)	7.00	7.00	7.22	21.22
Sonnet 4.5 (Deep)	6.56	7.89	6.11	20.56
GPT-5.2 (Deep)	5.22	7.00	5.00	17.22
GPT-5.2 (Shallow)	5.56	5.56	5.22	16.33

Table 8 – continued from previous page

Agent	New	Useful	Surp.	Total
2. Counterfactual Reasoning				
Caesar	**8.56**	**8.00**	**8.56**	**25.11**
Gemini 3 (Deep)	7.89	7.11	8.00	23.00
Sonnet 4.5 (Deep)	6.56	7.67	6.44	20.67
Sonnet 4.5 (Shallow)	5.22	5.67	5.22	16.11
GPT-5.2 (Deep)	4.56	5.89	4.33	14.78
Gemini 3 (Shallow)	4.11	4.89	3.89	12.89
GPT-5.2 (Shallow)	3.44	5.00	3.33	11.78
3. Cross-Domain Synthesis				
Caesar	**8.78**	**8.11**	**8.89**	**25.78**
Sonnet 4.5 (Shallow)	6.11	5.78	6.11	18.00
Sonnet 4.5 (Deep)	5.78	6.78	5.11	17.67
Gemini 3 (Deep)	4.89	6.33	4.44	15.67
GPT-5.2 (Deep)	4.11	5.33	3.33	12.78
GPT-5.2 (Shallow)	4.00	5.22	3.33	12.56
Gemini 3 (Shallow)	2.67	3.44	2.33	8.44
4. Meta-Creativity				
Sonnet 4.5 (Deep)	**7.89**	6.33	**7.56**	**21.78**
Caesar	7.11	**7.56**	6.44	21.11
GPT-5.2 (Deep)	5.67	6.78	5.44	17.89
Gemini 3 (Deep)	6.11	4.78	6.56	17.44
Sonnet 4.5 (Shallow)	5.67	5.56	5.11	16.33
GPT-5.2 (Shallow)	4.56	4.89	3.78	13.22
Gemini 3 (Shallow)	3.56	4.22	3.44	11.22
5. Open-Ended Synthesis				
Caesar	8.11	7.11	7.89	**23.11**
Sonnet 4.5 (Deep)	6.44	**7.56**	6.33	20.33
Sonnet 4.5 (Shallow)	**8.78**	2.44	**8.89**	20.11
GPT-5.2 (Shallow)	7.33	5.67	7.00	20.00
Gemini 3 (Deep)	6.78	5.11	7.00	18.89
Gemini 3 (Shallow)	4.67	6.67	4.11	15.44
GPT-5.2 (Deep)	3.44	5.67	2.89	12.00

H. Computational Cost Analysis

While Caesar achieves higher creativity scores, its heavy usage of LLMs for web page processing and vector store retrieval incurs a noticeable computation cost. The total cost is split between the extensive data gathering in Phase 1 and the recursive draft generation in Phase 2. The computational cost suggests that Caesar is best utilized as a tool for high-value, asynchronous research rather than real-time interaction, similar to the deep research agents described in Li et al. (2025).

H.1. Phase 1: Deep Web Exploration Costs

The exploration phase constitutes the primary computational expense, averaging approximately \$80–\$100 per 1000-step experiment. Given the GPT-5.2 pricing of \$1.75 per 1M input tokens and an average context load of $\sim$30k tokens per step, this cost reflects the cumulative impact of continuous context usage and output generation during web exploration. The token budget is predominantly consumed by two mechanisms:

- **Ingestion of Text-Heavy Documents:** Caesar processes most or all of the text content of a page to extract deep insights. In most cases, Caesar processes standard web articles with only $\sim$5k tokens. However, when exploring scientific and humanitarian domains, the agent can encounter dense PDF reports and data portals. These text-heavy pages can reach the truncated limit of P_m (100k characters), consuming 20k–25k tokens just for the page content alone.
- **Processing and Selecting Links:** The other major consumer of the token budget is the link selection in the “Act” stage. To make informed navigational decisions, the agent processes hundreds of links per page on average (ranging from $\sim$500 on wiki-style pages to 1000+ on dense index pages). Assuming each link consumes $\sim$25 tokens of context, this results in an overall cost of $\sim$10–20k tokens per navigation step, regardless of the remaining page content.

H.2. Phase 2: Adversarial Artifact Synthesis Costs

In contrast, the synthesis phase is comparatively inexpensive, averaging less than \$6–\$8 per run.

- **Efficient Insight Retrieval:** Unlike the exploration phase, which ingests raw uncompressed web content, the synthesis phase operates on a curated knowledge base (*KB*). By limiting the number of insights retrieved and performing reranking on them to generate concise answers, the context window is significantly reduced, minimizing input token costs.
- **High Reasoning Overhead:** The cost driver in this phase shifts to output tokens. The synthesizer utilizes GPT-5.2 with high reasoning effort during the adversarial refinement rounds ($N = 3$). While the input volume is low, the generation of extensive chain-of-thought reasoning (Long, 2023) to integrate disparate insights increases the cost per inference step relative to standard generation.

H.3. Algorithmic Complexity

Traditional Big-O complexity analysis is of limited use for evaluating Caesar, as the wall-clock runtime is overwhelmingly dominated by the latency of external LLM API calls rather than local compute. Instead, computational complexity is best framed as a function of total LLM cost, similar to frameworks such as Asymptotic Analysis with LLM Primitives (Meyerson & Qiu, 2025). The cost per step is bounded by the model’s context limit during the THINK and ACT stages. Over T steps, the overall complexity scales linearly as $O(T \cdot \bar{c})$, where $\bar{c}$ is the mean cost per step. Similarly, space complexity is bounded by the graph G and vector store *KB* at $O(T \cdot (\bar{i} + d))$, where $\bar{i}$ is the average insight length and d is the embedding dimension of *KB*. By strictly upper-bounding both time and space by the exploration budget T, Caesar avoids the infinite navigational loops and memory overflows that often plague unconstrained web crawlers.