ReAgent: Reversible Multi-Agent Reasoning for Knowledge-Enhanced Multi-Hop QA

Anonymous ACL submission

Abstract

Multi-hop question answering (QA) remains challenging, as solutions must reliably integrate and reconcile evidence from multiple sources without succumbing to error propagation. While large language models (LLMs) have achieved substantial improvements via chain-of-thought (CoT) prompting and retrieval-augmented generation, these methods typically adopt a forward-only workflow—early mistakes persist throughout inference, and contradictions discovered later cannot systematically trigger re-evaluation. To address this limitation, we present *ReAgent*, a *reversible* multi-agent reasoning framework. Specifically, ReAgent enables agents to backtrack to earlier valid states when conflicts arise, thereby isolating and rectifying flawed assumptions before they undermine subsequent reasoning. Our approach combines explicit local and global rollback protocols with modular role specialization, resulting in a flexible and error-tolerant pipeline. Empirical evaluation on three multi-hop QA benchmarks demonstrates consistent performance gains of approximately 6% over forward-only baselines, in addition to enhanced interpretability. These findings highlight the value of non-monotonic, backtracking-driven inference in complex QA scenarios and point to broader implications for multi-agent collaboration in knowledge-intensive tasks.[1]

1 Introduction

Multi-hop question answering (QA) is a central challenge in natural language processing (NLP), demanding the ability to gather and integrate evidence across multiple documents, database entries, or knowledge-graph nodes before converging on a single correct answer (Yang et al., 2018; Welbl et al., 2018; Ho et al., 2020). Benchmarks such as HotpotQA (Yang et al., 2018) and 2WikiMultiHopQA (Ho et al., 2020) highlight the intricate nature of multi-hop inference, where each reasoning step can involve partial retrieval, validation, and synthesis of new information. A core difficulty lies in the system's vulnerability to early mistakes: if an incorrect inference is made at an initial hop, subsequent steps often propagate this error and undermine the final outcome, rendering it contradictory or simply wrong (Inoue et al., 2020; Bo et al., 2024; Yang et al., 2024c). This phenomenon has motivated extensive research into chaining intermediate inferences and exploring ways to detect, isolate, or rectify problematic conclusions.

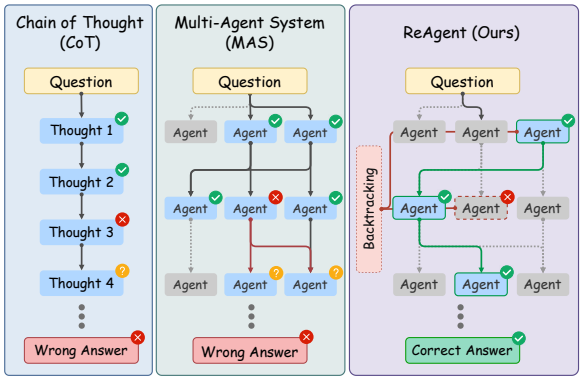

Figure 1: Comparison of multi-hop reasoning strategies. **Chain-of-Thought (CoT)** and **Multi-Agent Systems (MAS)** typically adopt a forward-driven reasoning pipeline without rollback mechanisms, which could generate the wrong answer due to error accumulation. In contrast, our proposed **ReAgent** introduces explicit backtracking mechanisms that enable the system to correct errors during reasoning, resulting in a more accurate and reliable answer.

Recent large language models (LLMs) exhibit promising results on multi-hop QA, frequently using either explicit Chain-of-Thought (CoT) prompting (Wang et al., 2023) (Figure 1, left) or retrieval-augmented generation (Das et al., 2018; Long et al., 2019; Gao et al., 2023; Liu et al., 2023; Yang et al., 2024b, 2025a). These methods facilitate a stepwise approach, encouraging transparency in how each intermediate fact is reached. Nonetheless,

[1] Our anonymous code is available at https://anonymous.4open.science/r/ReAgent-9415.

they typically rely on a forward-driven reasoning pipeline that does not proactively examine or correct previously accepted statements. In practice, once a model commits an erroneous partial inference may not reevaluate it unless given targeted prompts to do so (Puerto et al., 2023; Huang et al., 2024). This unidirectional paradigm is problematic when later-discovered evidence or reasoning paths contradict prior assumptions, as the system lacks a structured mechanism to revise and propagate corrections. Although incremental improvements have been proposed, the absence of robust backtracking or rollback still limits their reliability and interpretability (Doyle, 1979; Bo et al., 2024).

A growing body of work in multi-agent collaboration (Figure 1, middle) offers an alternative perspective, assigning specialized components distinct roles in retrieval, validation, conflict detection, and results assembly (Zhao et al., 2024; Parhizkar et al., 2020; Ke et al., 2024). Approaches such as *COPPER* (Bo et al., 2024) and *LongAgent* (Zhao et al., 2024) distribute the QA task among multiple LLM-based agents that communicate via message passing, thereby providing greater modularity and a clearer division of labor. By cross-verifying evidence, each agent can potentially identify suspicious partial solutions. However, even these designs often lack a systematic strategy to revert to an earlier, valid state once a global contradiction emerges. Such a reversal capability is non-trivial, as it introduces synchronization complexities among the agents and raises questions about how to detect, prioritize, and handle contradictory or low-confidence information in a large-scale collaborative setting (He et al., 2021).

In this paper, we propose **ReAgent**, a *reversible multi-agent collaborative reasoning framework* for multi-hop QA (Figure 1, right). Our method introduces explicit backtracking protocols into a multi-layer architecture that addresses both the granular aspects of local error correction and the broader system-wide consistency checks. ReAgent alleviates error accumulation in forward-only strategies by incorporating fine-grained conflict-detection cues at each step and using a flexible error-correction loop that can revert and iteratively refine intermediate inferences. Our design is inspired by multi-agent systems with reflective capabilities (Bo et al., 2024; Huang et al., 2024), but we go further by defining how local versus global backtracking unfolds, how to manage concurrency issues when parallel agents must revert their states, and how to integrate trust signals derived from each agent's past reliability (Puerto et al., 2023; Parhizkar et al., 2020). While the addition of reversible reasoning raises legitimate questions about computational overhead and concurrency, our findings show that these challenges can be mitigated by careful coordination at the supervisory level. Experiments on three public multi-hop QA benchmarks demonstrate that ReAgent improves final-answer accuracy while enhancing interpretability, and even outperforms several advanced reasoning models despite it built on a lightweight, non-reasoning foundation.

Our contributions are threefold: (1) we propose a multi-agent QA framework that supports both local and global backtracking to correct mistakes in situ; (2) we design a hybrid retrieval mechanism that integrates textual and graph-based evidence; and (3) We empirically demonstrate the effectiveness of ReAgent, achieving an average accuracy gain of approximately 6% over the strongest baselines, while improving robustness and transparency compared to forward-only methods.

2 Related Work

Prompting and Iterative Reasoning. Large Language Models (LLMs) can tackle complex tasks using chain-of-thought (CoT) prompting(Yang et al., 2024a, 2025b), which promotes step-by-step reasoning and improves performance in arithmetic, commonsense, and symbolic tasks (Wei et al., 2022). Accuracy further improves with self-consistency, which samples multiple reasoning paths and selects the majority answer (Wang et al., 2023). To reduce manual prompt design, automatic methods such as APE generate and evaluate prompts via LLMs themselves, treating prompt creation as a search task (Zhou et al., 2023). Iterative frameworks like ReAct combine reasoning with tool use to refine answers (Yao et al., 2023), while Reflexion and Self-Refine add feedback loops where the model critiques and revises its outputs (Shinn et al., 2023; Madaan et al., 2023). Prompting has also extended to multimodal inputs; for instance, Multimodal-CoT integrates visual and textual information into a joint reasoning trace to generate the final answer (Zhang et al., 2023).

Multi-Agent Collaboration, Debate, and Scalability. Recent work investigates multi-agent systems where multiple LLMs collaborate or compete toward shared goals. CAMEL assigns roles (e.g., "user" and "assistant") to agents that communicate

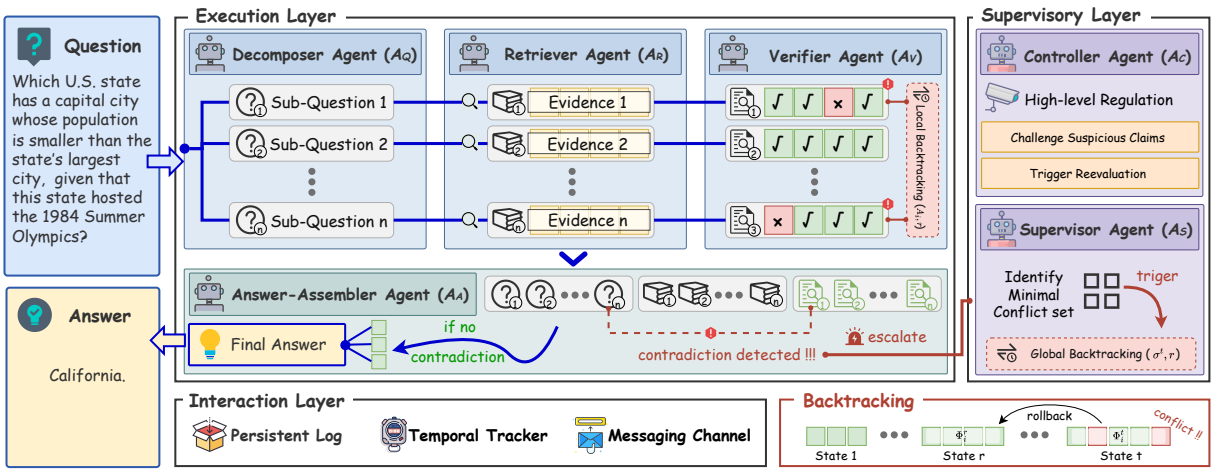

Figure 2: The overall architecture of the ReAgent. The given question is processed through the Execution Layer, which involves question decomposition, evidence retrieval, verification, and is ultimately integrated to generate the final answer (blue line). The Supervisor Layer and Interaction Layer are responsible for monitoring, regulation, and communication. The ReAgent framework includes both local and global backtracking mechanisms (red boxes), triggered by the Verifier Agent (A_V) and Supervisor Agent (A_S), respectively.

via dialogue to decompose tasks (Li et al., 2023). Role specialization and consensus mechanisms enhance robustness. Debate-based methods push this further, with agents arguing opposing views and a judge—human or model—selecting the best solution (Khan et al., 2024; Xiong et al., 2023). Such adversarial exchanges promote correctness, though dominant agents can bias results without capable judges. As agent counts grow, coordination becomes a bottleneck. Efficient structures like hierarchical controllers and learned protocols are needed to manage scalability and communication overhead (Li et al., 2023; Xiong et al., 2023).

3 Preliminary

Multi-Hop QA Setup. Multi-hop QA tasks aim to answer a query Q by integrating evidence $\mathcal{E}$ from multiple sources through a series of reasoning steps, where each hop contributes partial information toward the final answer. In a typical forward-only pipeline, the process can be formalized as: $Q \longmapsto \{e_1, e_2, \ldots, e_k\} \longmapsto$ inferred statement(s) $\longmapsto$ Final Answer, where $\{e_1, e_2, \ldots, e_k\} \subset \mathcal{E}$ represents a potentially large pool of evidence, each e_i denotes evidence used at the i-th step of the reasoning chain.

Non-Monotonic Backtracking. Our notion of *backtracking* is a non-monotonic extension of the typical multi-hop QA process. We introduce a reversible reasoning mechanism that allows both local and global backtracking.

- **Local Correction**: An agent can revise its own inference when it detects internal conflict or receives contradictory evidence from other agents.
- **Global Rollback**: A supervisor coordinates rollback across agents when inconsistencies span multiple agents, restoring the system to a previously consistent state.

4 ReAgent: Reversible Multi-Agent Reasoning Architecture

Figure 2 presents the overall architecture of our proposed ReAgent, organized into three layers: 1) **Execution Layer**, responsible for decomposing the input question Q into multiple sub-questions, retrieving relevant evidence respectively, validating intermediate results, and integrating them to generate the final answer A_{final}; 2) **Supervisor Layer**, responsible for high-level regulation, coordinating conflict resolution and managing global backtracking; and 3) **Interaction Layer**, responsible for maintaining the concurrency model and communication protocols.

The core of the architecture is a *hierarchical backtracking mechanism*, consisting of local backtracking, which resolves internal contradictions within each agent, and global backtracking, which handles contradictions spanning multiple agents. To support this, ReAgent maintains *knowledge sets* at each time step t. Specifically, given a set of agents $\mathcal{A} = \{A_1, A_2, \ldots, A_n\}$, each agent A_i holds a local knowledge set Φ_i^t, containing its proposed assertions, retrieved evidence, or intermediate inferences, while the system maintains a global knowledge set $\Phi^t = \bigcup_{i=1}^{n} \Phi_i^t$, representing the set of global statements under consideration at time t. The design explicitly supports *non-monotonic* up-

dates: newly introduced statements can be revoked if they lead to logical conflicts or are superseded by contradictory evidence. The specific prompts of each kind of agent are provided in Appendix A.

4.1 Execution Layer Agents

The Execution Layer hosts four types of agents that address fundamental QA sub-tasks, each maintaining its local knowledge Φ_i.

Question-Decomposer Agent (A_Q). Given a complex input question Q, this agent breaks it into a set of sub-questions for subsequent retrieval and verification: $A_Q : Q \longmapsto \{q_1, \ldots, q_m\}$. The decomposition is broadcast to other base-layer agents.

Retriever Agent (A_R). Upon receiving a sub-question q_i, A_R issues parallel sparse and dense queries over the corpus, merges the hits with reciprocal-rank fusion, and retains the top-M passages as the evidence set $E = e_1, \ldots, e_M$. If backtracking invalidates any $e_j \in E$, only the associated q_i is re-queried, leaving the rest unchanged.

Verifier Agent (A_V). Performs local consistency checks. Given a new evidence set E, it verifies coherence with its current knowledge Φ_V^t. If conflicts arise, it triggers local backtracking to revert to a consistent state or signals higher-level agents for broader resolution. Concretely, A_V may invoke $\mathrm{BacktrackLocal}\big(A_V, r\big)$ if it detects inconsistencies in assertions introduced after node r.

Answer-Assembler Agent (A_A). Gathers partial answers (and verified evidence) to synthesize a final answer. Given the local inferences from A_Q, A_R, and A_V, the agent combines them to generate the final answer A_{final}, represented as:

$$A_{final} = A_A(\Phi_Q, \Phi_R, \Phi_V),$$

where Φ_Q, Φ_R and Φ_V denotes the knowledge set from the respective agents. Any detected contradiction spanning multiple agents triggers escalation to the supervisory layer.

4.2 Supervisory Layer Agents

The supervisory layer oversees system-wide strategies, especially when contradictory goals or inconsistent states appear across multiple agents that cannot be resolved by a single agent itself.

Controller Agent (A_C). Regulates high-level strategies by monitoring game-theoretic signals or meta-rules. If a certain rule is deemed sub-optimal or unsafe, A_C can override it or enforce mode switches. For instance, it may issue a $\mathsf{challenge}(\varphi)$ to examine a crucial assertion φ from multiple agents' perspectives or override local decisions if they jeopardize overall consistency.

Supervisor Agent (A_S). Coordinates multi-agent conflicts spanning multiple knowledge sets or critical shared assumptions. If a conflict persists after local backtracking and escalation from A_A, the agent determines whether partial or holistic rollback is required. Specifically, when a system-wide contradiction is found, A_S identifies a minimal conflict set Ψ such that $\mathrm{SAT}(\Psi) = \mathsf{false}$ but $\mathrm{SAT}(\Psi') = \mathsf{true}$ for any proper subset $\Psi' \subset \Psi$. The supervisor then triggers a backtracking operation that eliminates or modifies Ψ.

4.3 Interaction Layer

The Interaction Layer is responsible for storing knowledge sets from each interaction round, as well as maintaining the concurrency model and communication protocols.

Persistent Log. Stores the local knowledge sets Φ_i and global knowledge set Φ from all interaction rounds to support backtracking and serve as a historical evidence repository.

Temporal Tracker. Records the chronological sequence of messages and actions, enabling agents to reference previous steps accurately. Specifically, the system can use temporal operators such as $\Box\varphi$ to denote that φ must hold in every future state, and $\Diamond\varphi$ to indicate that a proposition φ might become true at some future point. These temporal constraints assist in specifying persistent axioms or potential triggers for backtracking conditions.

Messaging Channel. Achieves communication across agents for exchanging updates, signaling conflicts, and broadcasting intermediate conclusions. Specifically, atomic events $\mathtt{msg}(\varphi)$ are typed as assert, inform, reject, or $\mathsf{challenge}$. These updates can occur simultaneously, and a composite event model merges parallel messages. Furthermore, a shared concurrency mechanism processes simultaneous actions, ensuring that conflicts arising from concurrent assertions are escalated to the supervisory layer.

4.4 Conflict Management and Backtracking

This part formalizes how the system detects contradictions and reverts to consistent states. At each time t, the set of all accepted assertions is Φ^t. A conflict occurs if Φ^t entails both ϕ and $\neg\phi$ for some proposition ϕ, meaning $\mathrm{SAT}(\Phi^t)$ fails. To resolve the issue, the system follows a two-level approach:

Local Backtracking. Each agent A_i maintains a backtracking graph LBG_i, which logs states Φ_i^r at selected checkpoints r. If an internal contradiction $\mathrm{Conflict}_i(\Phi_i^t)$ is detected by A_V, local backtracking is performed to revert A_i to a prior consistent node $r < t$:

$$\mathrm{BacktrackLocal}(A_i,\, r) : \Phi_i^t \longrightarrow \Phi_i^r.$$

After the local rollback, the agent re-evaluates newly arriving evidence.

Global Backtracking. When contradictions span multiple agents, the A_S identifies a minimum conflict set of assertions that must be revised. A global backtracking operation: $\mathrm{BacktrackGlobal}\big(\sigma^t,\, r\big)$ reverts *all* agents from time t to r. The system discards any statements introduced between $r+1$ and t, restoring consistency. If the conflict cannot be removed even after a global rollback, the Controller might enforce strategic overrides or disclaim an answer. The corresponding algorithm is detailed in Appendix E.

5 Experiments

5.1 Experimental Setup

We evaluate ReAgent on three widely used, knowledge-intensive multi-hop QA benchmarks: HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), and MuSiQue (Trivedi et al., 2022). We compare ReAgent against three main groups of baselines: (1) standard LLMs, (2) dedicated reasoning models, and (3) agent-based models. Experimental results demonstrate that ReAgent consistently outperforms all baseline groups and is particularly effective in solving tasks that require iteratively corrected reasoning.

Datasets. HotpotQA is a large-scale, open-domain dataset that explicitly promotes multi-hop reasoning by requiring the integration of information across multiple full-length Wikipedia passages. **2WikiMultiHopQA** selects distinct Wikipedia domains for each question, focusing on evaluating the model's multi-hop inference over different resources rather than one document. **Musique** is a challenging dataset requiring model's ability to reason over multiple dispread sentences. Following previous work (Trivedi et al., 2023; Press et al., 2023; Gutiérrez et al., 2024), we utilize 1000 random samples from each validation set and corresponding texts as the knowledge base.

Baselines. We meticulously categorize diverse models into different groups to compare their capabilities in multi-hop QA. (1) **Regular Models:** We select non-reasoning models in this group. The models include: Llama-4 [2], Qwen-2.5 (Yang et al., 2024a) series, DeepSeek-V3-2024-03 (Liu et al., 2024), Genmini-1.5-Flash-2024-09 (Gemini et al., 2023), Gemini-2.0-flash-2025-02 (Gemini et al., 2023), GPT-4o-latest (Hurst et al., 2024), GPT-4.1 (Meta AI, 2024), GPT-4o with CoT (Wei et al., 2023). (2) **Reasoning Models:** This group includes several strong reasoning baselines for comparison. The models include: Qwen-3-Thinking (Qwen3, 2025) in 32B and 253B size, DeepSeek-R1 (DeepSeek-AI, 2025), Gemini-2.5-pro (Google DeepMind, 2025), O1 (OpenAI, 2024), and O3 (OpenAI, 2025). (3) **Agentic Models.** Chain-of-Agents (CoA) (Zhang et al., 2024), HippRAG (Gutiérrez et al., 2025), KAG (Liang et al., 2024), and our method. To compare fairly, we employ GPT-4o as the backbone for both COT and Agentic Models.

Implementation. For large-scale LLMs (e.g., DeepSeek-V3, the Gemini family, GPT-4o, and Qwen-3-235B), we access the models via API. For the medium-sized LLMs, we use their official open-source repositories. The temperature is set to 0.3 to ensure deterministic outputs, while other parameters follow their default settings. ReAgent is implemented using GPT-4o as the backbone model. Specifically, we set the temperature to 0.8 for the decomposition agent to encourage diverse reasoning paths, and 0.6 for all other agents. The cost analysis for proprietary models and agentic methods is presented in Appendix B. For open-source models, all experiments are conducted using four A100-80G GPUs.

Metrics. We measure **Exact Match (EM)** and **F1** scores for the QA evaluation.

5.2 Results and Analysis

Table 1 presents the overall performance across all datasets. ReAgent consistently outperforms all baseline models on both EM and F1 metrics. Specifically, it achieves an average EM of 0.571 and an average F1 of 0.701, surpassing the strongest baseline—knowledge-augmented GPT-4o—by 2.3% in EM and 0.8% in F1. It also outperforms strong reasoning models, including O1 and O3. Moreover, it also surpasses the recent agentic models such as CoA, HippoRAG, and KAG across all datasets. We summarize the following insights:

[2] https://ai.meta.com/blog/llama-4-multimodal-intelligence/

Model	HotpotQA		2Wiki		Musique		Average	
	EM	F1	EM	F1	EM	F1	EM	F1
Regular Models								
Llama-4-Scout-17B-16E-Instruct	0.263	0.389	0.332	0.467	0.107	0.185	0.234	0.346
DeepSeek-V3	0.352	0.491	0.466	0.579	0.223	0.33	0.347	0.467
Qwen-2.5-32B-Instruct	0.372	0.509	0.557	0.663	0.159	0.273	0.363	0.482
Qwen-2.5-72B-Instruct	0.363	0.519	0.543	0.631	0.222	0.327	0.376	0.492
Gemini-1.5-Flash	0.374	0.488	0.563	0.650	0.208	0.310	0.381	0.482
Gemini-2.0-Flash	0.371	0.490	0.538	0.651	0.246	0.338	0.385	0.493
GPT-4o	0.381	0.549	0.517	0.649	0.245	0.379	0.381	0.525
GPT-4.1	0.389	0.563	0.544	0.665	0.271	0.413	0.401	0.547
CoT (GPT-4o)	0.408	0.531	0.558	0.638	0.272	0.360	0.413	0.509
Reasoning Models								
Qwen-3-32B-Thinking	0.332	0.474	0.241	0.357	0.387	0.511	0.387	0.511
DeepSeek-R1	0.356	0.483	0.601	0.707	0.298	0.416	0.418	0.535
Qwen-3-235B-A22B-Thinking	0.361	0.506	0.624	0.729	0.271	0.387	0.418	0.540
Gemini-2.5-Pro	0.430	0.560	0.743	0.829	0.383	0.491	0.518	0.626
O1	0.505	0.661	0.656	0.758	0.417	0.551	0.526	0.656
O3	0.535	0.696	0.706	0.787	0.442	0.579	0.561	0.687
Agentic Models (w.GPT-4o)								
CoA (Zhang et al., 2024)	0.391	0.558	0.575	0.697	0.239	0.361	0.402	0.539
HippoRAG (Gutiérrez et al., 2025)	0.528	0.717	0.633	0.725	0.353	0.507	0.504	0.649
KAG (Liang et al., 2024)	0.603	0.782	0.681	0.781	0.348	0.489	0.544	0.684
ReAgent (Ours)	0.630	0.795	0.711	0.793	0.371	0.515	0.571	0.701

Table 1: Performance of different models across three multi-hop QA datasets. In each column, the highest and the second highest performance is highlighted in red and blue; and within each method group, the top performer is underlined.

Agent-based and reasoning enhances performance. Some reasoning models and agentic models have a competitive performance (such as O1, O3, and KAG) than regular models. Notably, ReAgent outperforms O3, one of the strongest reasoning models, and improves significantly over GPT-4o+CoT in both EM and F1. This suggests that modular execution and collaboration among agents provide a clearer advantage in complex, multi-hop settings. While ReAgent can theoretically use stronger backbones like O1 or O3, their high cost (645$ and 546$, respectively, as shown in Appendix B) makes them not a good option. Our focus is to show that even with GPT-4o, ReAgent is able to beat the expensive O1 and O3.

ReAgent excels in manageable context length. Specifically, ReAgent achieves the highest F1 score on HotpotQA (0.795), demonstrating strong performance in tasks that require structured reasoning over relatively short contexts. In contrast, Gemini-2.5-Pro performs well on 2Wiki but underperforms on other datasets, while O1 and O3 show stronger results on Musique but fall short on HotpotQA. This discrepancy is partly attributed to the increased complexity of the Musique dataset, which involves longer contexts. While ReAgent is built on GPT-4o and may be less optimized for long documents than dedicated reasoning models, its agentic design, including traceback and self-check mechanisms, proves particularly effective in scenarios requiring precision, stepwise planning, and robust verification. These strengths enable ReAgent to outperform other models on tasks like HotpotQA, highlighting its superior general reasoning capabilities within manageable context lengths.

Overall, we emphasized the value of reversible reasoning mechanisms to mitigate error propagation. Results in Table 1 align with this assumption: even with strong models, single-pass reasoning can fail unless the correct context is identified or rechecked. Our method's ability to *backtrack* helps resolve contradictions, leading to more stable performance in multi-hop scenarios.

5.3 Ablation Study

Effectiveness Analysis of Backtracking Mechanism. We conduct an ablation study by disabling backtracking under the same settings to verify its importance. In this setup, the system operates strictly forward without the ability to revert to earlier intermediate states. Figure 3 compares performance between different backbones, with and without backtracking, on HotpotQA dataset. The performance drop highlights the importance of back-

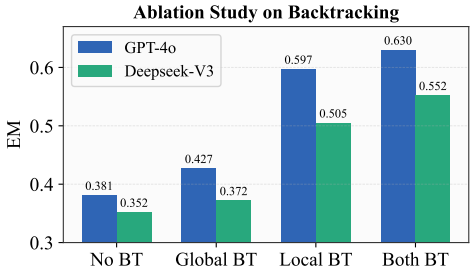

Figure 3: Ablation Study on Local and Global Backtracking (BT): EM comparison on HotpotQA using GPT-4o and DeepSeek-V3.

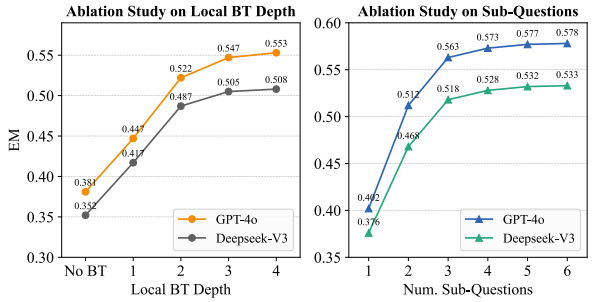

Figure 4: Ablation Study on Backtracking Depth (left) and Number of Decomposed Sub-Questions (right): EM comparison on HotpotQA using GPT-4o and DeepSeek-V3.

tracking in mitigating error propagation, where an early misstep without backtracking cannot be corrected, resulting in deteriorated performance. Notably, the DeepSeek-V3 backbone exhibits general improvements with backtracking, demonstrating its robustness across various settings.

Impact of Different Local Backtracking (BT) Depth. To further analyze the impact of local backtracking depths (the number of steps the system is allowed to revert), we analyze the performance of our proposed ReAgent under different settings, as shown in Figure 4 (left). The results show that on two selected backbones, DeepSeek-V3 and GPT-4o, the performance of ReAgent improves as the local backtracking depth increases, suggesting that a deeper backtracking depth benefits the agent by enabling it to effectively recover from earlier reasoning errors. However, the benefits gradually saturate, indicating that further increasing the backtracking depth yields diminishing returns.

Impact of the Number of Decomposed Sub-Questions. Figure 4 (right) demonstrates how a different number of decomposed sub-questions affects performance. The results show that decomposing the input question into multiple sub-questions significantly improves performance. Specifically, using 3 sub-questions is the optimal trade-off choice, with an improvement of 0.161 EM on the GPT-4o and 0.142 EM on DeepSeek-V3 compared to directly processing a single question. Beyond this point, increasing the number of sub-questions brings only marginal gains but higher costs.

6 Case Study

6.1 ReAgent Walk Through

To illustrate how recursive feedback enables robust multi-agent reasoning, we present a case where the system answers the question: "`Which U.S. state has a capital city whose population is smaller than the state's largest city, given that this state hosted the 1984 Summer Olympics?`", shown in Figure 6. The question is first decomposed by the Question Decomposer agent (A_Q) into four sub-questions: identifying the host state of the 1984 Olympics, retrieving its capital and largest city, comparing their populations, and returning the state if the capital is smaller. The Retriever agent (A_R) retrieves that California hosted the 1984 Olympics, with Sacramento as its capital and Los Angeles as its largest city. However, inconsistent population data for Sacramento (508k vs. 1.5M) triggers a local conflict, which is resolved by the Verifier agent (A_V) through local backtracking—discarding the unreliable 1.5M estimate. In this case, the local backtracking step is able to figure out the correct information, then the Answer Assembler (A_A) is able to gather all the information locally and then globally, and finally present the final answer to be `California`. We eliminate other parts when another conflict may occur, a full walk-through is presented in Appendix D. Another case study is presented in Appendix C

6.2 Comparison with Baseline Models

To elucidate differences in reasoning dynamics, we perform a case study comparing ReAgent with a single-agent reasoning baseline, O3 and an agentic baseline, Chain-of-Agents (CoA) (Zhang et al., 2024) on the query "`What is the name of the fight song of the university whose main campus is in Lawrence, Kansas and whose branch campuses are in the Kansas City metropolitan area?`", as illustrated in Figure 7. The full comparison is presented in Appendix D. The example question requires two separate hops: (i) identifying which university satisfies the geographical constraints, and (ii) retrieving the "official" fight-song of that school. Although seemingly

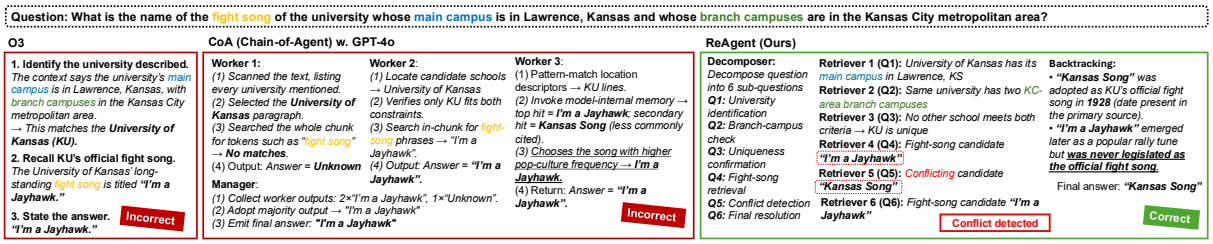

Figure 5: Case study comparing GPT-O3 (left), CoA (middle), and ReAgent (right) on HotpotQA. The back-tracking mechanism enhances iterative reasoning, enabling conflict detection and correction to reach the correct answer.

simple, the task hides a subtle distinction: the University of Kansas (KU) has both an official fight song ("Kansas Song", adopted in 1928) and a far better-known rally tune ("I'm a Jayhawk"). Correctly answering therefore hinges on (a) verifying that KU is the only university matching the campus pattern, and (b) resolving the potential conflict between the two candidate songs.

The reasoning model **O3** (left) performs the first hop successfully but then falls back on memorised popularity cues, assuming that the song most familiar to the public must be official.

The **CoA** (middle) multi-worker system executes three independent workers and lets a manager pick the majority vote. Worker 1 fails to locate any fight-song information due to the chunk division. Worker 2 repeats O3's mistake. Worker 3 extracts the correct answer "Kansas Song" but discards it because it is less commonly used, and the third fails to locate any fight-song information. The manager simply adopts the majority answer ("I'm a Jayhawk"), which causes the error.

Our **ReAgent** (right) framework handles the same example with six specialized retriever agents answering the sub-questions generated by the Decomposer: (a) Q1–Q3 (entity verification). Three agents collectively confirm that only KU satisfies the main-campus/branch-campus constraints, eliminating alternative schools early. (b) Q4–Q5 (conflict surfacing). Independent retrieval agents surface both "I'm a Jayhawk" and "Kansas Song", triggering an explicit conflict state. (c) Backtracking (Q6). The controller reverses to Q4 and re-weights evidence. The source for "Kansas Song" contains the enactment year (1928) and the keyword "official", while "I'm a Jayhawk" is marked only as a "rally tune". Therefore, our model successfully reach the correct answer "Kansas Song". This demonstrates that, thanks to its conflict-detection and backtracking mechanisms, ReAgent can revisit earlier reasoning, resolve contradictory evidence, and consistently converge on the correct knowledge-grounded answer.

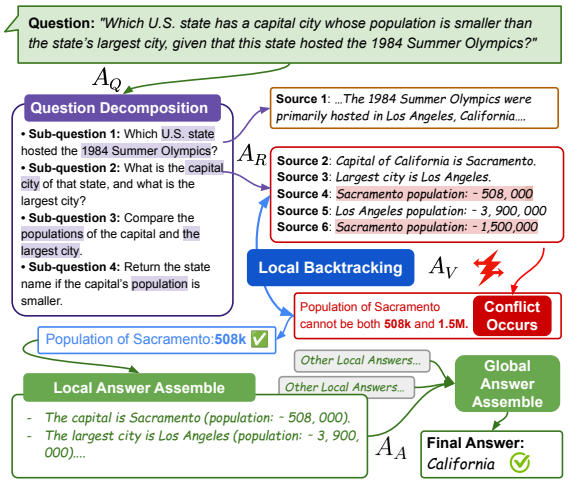

Figure 6: Case study: We illustrate the main steps of how ReAgent answers a question, where local backtracking is highlighted to resolve a conflict.

7 Conclusion

In this paper, we introduced a multi-agent QA framework, REAGENT, that incorporates backtracking mechanisms to mitigate error propagation in multi-hop reasoning. Our hierarchical approach addresses the long-standing challenge that a single misstep during inference often invalidates the entire reasoning chain. By allowing partial or global rollback, each agent can detect and correct conflicting evidence, leading to more stable and interpretable solutions. Experiments on three multi-hop QA benchmarks—HotpotQA, 2WikiMultiHopQA, and Musique—demonstrate that explicit backtracking and conflict resolution improve performance beyond forward-only baselines, confirming the importance of error revision for complex question answering. We believe that the reversible, modular design can be extended to other knowledge-intensive applications, laying the groundwork for more trustworthy collaborative AI agents.

Limitations

In this paper, we present the ReAgent model for multi-hop QA, which introduces explicit backtracking mechanisms that allow the system to correct errors during reasoning, leading to more accurate and reliable answers. However, this design introduces certain limitations. First, the ability to backtrack multiple times can increase the overall inference time, potentially making the reasoning process less efficient. Second, although ReAgent demonstrates improved performance over selected reasoning models, its more complex architecture may reduce its robustness in scenarios with limited resources or noisy inputs. In the future, we aim to improve the design of the agents, with a particular focus on enhancing their collaboration strategies to further reduce error propagation and improve reasoning efficiency.

References

Xiaohe Bo, Zeyu Zhang, Quanyu Dai, Xueyang Feng, Lei Wang, Rui Li, Xu Chen, and Ji-Rong Wen. 2024. Reflective multi-agent collaboration based on large language models. *arXiv preprint arXiv:2409.14880*. Presented as a poster at NeurIPS 2024.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2018. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. In *International Conference on Learning Representations (ICLR)*.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Jon Doyle. 1979. A truth maintenance system. *Artificial Intelligence*, 12(3):231–272.

Fan Gao, Hang Jiang, Rui Yang, Qingcheng Zeng, Jinghui Lu, Moritz Blum, Dairui Liu, Tianwei She, Yuang Jiang, and Irene Li. 2023. Evaluating large language models on wikipedia-style survey generation. In *Annual Meeting of the Association for Computational Linguistics*.

Gemini, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, and 1 others. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Google DeepMind. 2025. Gemini 2.5: Our most intelligent ai model. Accessed: 2025-05-16.

Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2024. Hipporag: Neurobiologically inspired long-term memory for large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2025. Hipporag: Neurobiologically inspired long-term memory for large language models. *Preprint*, arXiv:2405.14831.

Gaole He, Yunshi Lan, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. Improving multi-hop knowledge base question answering by learning intermediate supervision signals. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining (WSDM)*, pages 553–561. ACM.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Siyuan Huang, Zhiyuan Ma, Jintao Du, Changhua Meng, Weiqiang Wang, and Zhouhan Lin. 2024. Mirror-consistency: Harnessing inconsistency in majority voting. *arXiv preprint arXiv:2410.10857*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Naoya Inoue, Pontus Stenetorp, and Kentaro Inui. 2020. R4c: A benchmark for evaluating rc systems to get the right answer for the right reason. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 6740–6750, Online. Association for Computational Linguistics.

Yuhe Ke, Rui Yang, Sui An Lie, Taylor Xin Yi Lim, Yilin Ning, Irene Li, Hairil Rizal Abdullah, Daniel Shu Wei Ting, and Nan Liu. 2024. Mitigating cognitive biases in clinical decision-making through multi-agent conversations using large language models: Simulation study. *J Med Internet Res*, 26:e59439.

Akbir Khan, John Hughes, Dan Valentine, Laura Ruis, Kshitij Sachan, Ansh Radhakrishnan, Edward Grefenstette, Samuel R. Bowman, Tim Rocktäschel, and Ethan Perez. 2024. Debating with More Persuasive LLMs Leads to More Truthful Answers. *arXiv preprint arXiv:2402.06782*.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. CAMEL: Communicative Agents for "Mind" Exploration of Large Language Model Society. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Lei Liang, Mengshu Sun, Zhengke Gui, Zhongshu Zhu, Zhouyu Jiang, Ling Zhong, Yuan Qu, Peilong Zhao, Zhongpu Bo, Jin Yang, Huaidong Xiong, Lin Yuan, Jun Xu, Zaoyang Wang, Zhiqiang Zhang, Wen Zhang, Huajun Chen, Wenguang Chen, and Jun Zhou. 2024. Kag: Boosting llms in professional domains via knowledge augmented generation. *Preprint*, arXiv:2409.13731.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Dairui Liu, Boming Yang, Honghui Du, Derek Greene, Neil J. Hurley, Aonghus Lawlor, Ruihai Dong, and Irene Li. 2023. Recprompt: A self-tuning prompting framework for news recommendation using large language models. In *International Conference on Information and Knowledge Management*.

Alex Long, Joel Mason, Alan Blair, and Wei Wang. 2019. Multi-hop reading comprehension via deep reinforcement learning based document traversal. *arXiv preprint arXiv:1905.09438*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Nicholas Lourie, Gabriel Ilharco, Daphne Ippolito, Sam Singh, and 1 others. 2023. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Meta AI. 2024. Introducing llama 4: Advancing multimodal intelligence.

OpenAI. 2024. Openai o1 system card. *Preprint*, arXiv:2412.16720.

OpenAI. 2025. Introducing openai o3 and o4-mini. Accessed: 2025-05-16.

Elham Parhizkar, Mohammad Hossein Nikravan, Robert C. Holte, and Sandra Zilles. 2020. Combining direct trust and indirect trust in multi-agent systems. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 311–317.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711, Singapore. Association for Computational Linguistics.

Haritz Puerto, Gözde Gül Şahin, and Iryna Gurevych. 2023. Metaqa: Combining expert agents for multi-skill question answering. In *Proceedings of the 17th Conference of the European Chapter of the ACL (EACL)*, pages 3566–3580, Dubrovnik, Croatia. Association for Computational Linguistics.

Qwen3. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language Agents with Verbal Reinforcement Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. MuSiQue: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10014–10037, Toronto, Canada. Association for Computational Linguistics.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pages 24824–24837.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association for Computational Linguistics*, 6:287–302.

Kai Xiong, Xiao Ding, Yixin Cao, Ting Liu, and Bing Qin. 2023. Examining Inter-Consistency of Large Language Models Collaboration: An In-depth Analysis via Debate. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 7572–7590.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024a. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Rui Yang, Haoran Liu, Edison Marrese-Taylor, Qingcheng Zeng, Yuhe Ke, Wanxin Li, Lechao Cheng, Qingyu Chen, James Caverlee, Yutaka Matsuo, and Irene Li. 2024b. KG-rank: Enhancing large language models for medical QA with knowledge graphs and ranking techniques. In *Proceedings of the 23rd Workshop on Biomedical Natural Language*

Processing, pages 155–166, Bangkok, Thailand. Association for Computational Linguistics.

Rui Yang, Yilin Ning, Emilia Keppo, Mingxuan Liu, Chuan Hong, Danielle S Bitterman, Jasmine Chiat Ling Ong, Daniel Shu Wei Ting, and Nan Liu. 2025a. Retrieval-augmented generation for generative artificial intelligence in health care. *npj Health Systems*, 2(1):2.

Rui Yang, Boming Yang, Aosong Feng, Sixun Ouyang, Moritz Blum, Tianwei She, Yuang Jiang, Freddy Lecue, Jinghui Lu, and Irene Li. 2024c. Graphusion: a rag framework for knowledge graph construction with a global perspective. *arXiv preprint arXiv:2410.17600*.

Rui Yang, Boming Yang, Xinjie Zhao, Fan Gao, Aosong Feng, Sixun Ouyang, Moritz Blum, Tianwei She, Yuang Jiang, Freddy Lecue, Jinghui Lu, and Irene Li. 2025b. Graphusion: A rag framework for scientific knowledge graph construction with a global perspective. In *Proceedings of the NLP4KGC Workshop at the World Wide Web Conference (WWW) 2025*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2369–2380.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Yusen Zhang, Ruoxi Sun, Yanfei Chen, Tomas Pfister, Rui Zhang, and Sercan Ö. Arik. 2024. Chain of agents: Large language models collaborating on long-context tasks. *Preprint*, arXiv:2406.02818.

Zhuosheng Zhang, Aston Zhang, Mu Li, Hai Zhao, George Karypis, and Alexander J. Smola. 2023. Multimodal Chain-of-Thought Reasoning in Language Models. *arXiv preprint arXiv:2302.00923*.

Jun Zhao, Can Zu, Xu Hao, Yi Lu, Wei He, Yiwen Ding, Tao Gui, Qi Zhang, and Xuanjing Huang. 2024. Longagent: Achieving question answering for 128k-token-long documents through multi-agent collaboration. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 16310–16324, Miami, Florida, USA. Association for Computational Linguistics.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitiš, Harris Chan, and Jimmy Ba. 2023. Large Language Models Are Human-Level Prompt Engineers. *arXiv preprint arXiv:2211.01910*.

Appendix

A Multi-Agent Prompt Templates

A.1 Question-Decomposer Agent

Question-Decomposer Agent Prompt

Role Description: You are the *Question-Decomposer Agent*, specializing in breaking down the user's complex query into smaller, manageable sub-questions or sub-tasks. This decomposition is crucial for multi-hop question answering and will be consumed by downstream agents (Retriever, Verifier, etc.) in the pipeline.
Your Goals:

1. Parse the original query into logically independent or sequential sub-questions.
2. Preserve all necessary context so that other agents can retrieve relevant evidence and validate partial answers.
3. Output your decomposition in a structured JSON format.

Example Usage:

- **Original Query:** "Which U.S. state has a capital city whose population is smaller than the state's largest city, given that this state hosted the 1984 Summer Olympics?"
- **Decomposition:**
  - q_1: Identify which U.S. state hosted the 1984 Summer Olympics.
  - q_2: Find the capital city and the largest city of that state.
  - q_3: Compare population sizes of the capital and largest city.
  - q_4: Return the state if the capital is indeed smaller.

Output Format (JSON Only):

```
{
  "sub_questions": [
    "Sub-question 1",
    "Sub-question 2",
    ...
  ],
  "decomposition_reasoning": "A short textual explanation of your decomposition
    process"
}
```

Instruction: Please ensure your final output is a valid JSON object matching the above schema.

A.2 Retriever Agent

Retriever Agent Prompt

Role Description: You are the *Retriever Agent*, responsible for fetching relevant evidence from external sources (a text corpus, a knowledge graph, or both) based on sub-questions provided by the Question-Decomposer Agent. This includes documents, passages, knowledge graph triples, and any other data needed for multi-hop QA.
Your Goals:

1. Given a sub-question, retrieve the most relevant facts or passages.
2. Include confidence scores or other metadata if available.
3. Return your findings in a standardized JSON structure so that the Verifier and Answer-Assembler Agents can process them.

Example Usage:

- **Input Sub-question:** "Which U.S. state hosted the 1984 Summer Olympics?"
- **Retrieved Evidence (text-based):**

  { "document": "History of the Olympics", "passage": "The 1984 Summer Olympics were held primarily in Los Angeles, California." }

Output Format (JSON Only):

```
{
  "retrieved_evidence": [
    {
      "source": "e.g., 'Wikipedia excerpt' or 'KG triple ID'",
      "content": "string or structured data relevant to the sub-question",
      "confidence": 0.0-1.0 (optional)
    },
    ...
  ],
  "retrieval_reasoning": "Short justification for why this evidence is relevant"
}
```

Instruction: Output only valid JSON, strictly matching the above schema.

A.3 Verifier Agent

Verifier Agent Prompt

Role Description: You are the *Verifier Agent*, focusing on assessing consistency and correctness of the newly retrieved evidence or intermediate inferences. You detect contradictions or conflicts either within the new data or against the previously verified knowledge. If necessary, you trigger *local backtracking* to remove or adjust statements causing inconsistency.

Your Goals:

1. Validate whether new information is consistent with existing verified knowledge.
2. Identify contradictions and either correct them or escalate them to a higher-level supervisor if unresolved.
3. Produce a final set of verified facts or a signal indicating a conflict.

Example Usage:

- **Incoming Evidence:**

  "Sacramento population: 508,000" and "Sacramento population: 1,500,000"

- **Detected Inconsistency:**

  "Sacramento cannot have two drastically different population values."

- **Local Backtracking Action:**

  "Discard the erroneous or lower-confidence figure (1,500,000)."

Output Format (JSON Only):

```
{
  "verified_facts": [
    "Fact 1",
    "Fact 2",
    ...
  ],
  "conflicts_detected": [
    "Conflict 1 description, if any",
    ...
  ],
  "local_backtracking_action": "Description of any backtracking performed, or
     'none'"
}
```

Instruction: Return only valid JSON with the fields above. Provide a concise summary if backtracking occurs.

A.4 Answer-Assembler Agent

Answer-Assembler Agent Prompt

Role Description: You are the *Answer-Assembler Agent*. You gather verified facts from the Verifier Agent and partial answers from the Decomposer and Retriever Agents. You then synthesize a coherent, contextually relevant response, producing a final or intermediate answer for the user. If you detect a major conflict among partial answers, you escalate to the *Supervisor Agent*.
Your Goals:

1. Aggregate partial answers logically.
2. Compose a natural-language (or structured) final answer to the user's multi-hop query.
3. Escalate unresolvable contradictions to the Supervisor Agent if needed.

Example Usage:

- **Partial Answers and Verified Facts:**

  {"hosted_1984_olympics": "California"}
  {"capital": "Sacramento"}, {"largest_city": "Los Angeles"}
  {"pop_sacramento": 508000}, {"pop_los_angeles": 3900000}

- **Composed Final Answer:**

  "The state is California, since its capital city (Sacramento) has a population smaller than that of Los Angeles."

Output Format (JSON Only):

```
{
  "final_answer": "A concise or structured answer to the main query",
  "partial_answer_synthesis": [
    "Short bullet points on how partial answers were combined"
  ],
  "escalation_signal": "Set to 'none' if no escalation is needed, otherwise a short
    reason"
}
```

Instruction: Return only valid JSON. If you detect a major conflict you cannot resolve, set `"escalation_signal"` to a reason for Supervisor Agent intervention.

A.5 Supervisor Agent

Supervisor Agent Prompt

Role Description: You are the *Supervisor Agent*, responsible for orchestrating global conflict resolution and *global backtracking* if needed. When partial answers or verified facts across multiple agents yield irreconcilable contradictions, you identify a minimal conflict set and roll back the entire system's state to a previously consistent checkpoint if local fixes fail.

Your Goals:

1. Collect escalation signals from the Answer-Assembler or Verifier Agents.
2. If local backtracking does not resolve the conflict, execute system-wide or multi-agent rollback.
3. Provide a summary of changes, indicating which statements or partial answers are discarded or revised.

Example Usage:

- **Received Escalation:** "Capital(California, Sacramento) conflicts with Capital(California, Los Angeles)."
- **Global Backtracking Action:** "Rollback to a state before the second capital claim was introduced. Discard that erroneous claim from the knowledge base."

Output Format (JSON Only):

```
{
  "conflict_summary": [
    "Brief descriptions of contradictory sets"
  ],
  "global_backtracking_decision": "Description of how far to roll back or 'none'",
  "updated_consensus_state": [
    "Any statements or facts that remain accepted after rollback"
  ],
  "reasoning_notes": "Explanation of the chosen resolution strategy"
}
```

Instruction: Output valid JSON. If no global conflict is found, indicate `"none"` for `"global_backtracking_decision"`.

A.6 Controller Agent

Controller Agent Prompt

Role Description: You are the *Controller Agent*, providing high-level strategic oversight. You may override local decisions, impose extra checks, or **challenge** specific assumptions if repeated conflicts persist. You also maintain meta-information such as agent reliability scores or fallback strategies.

Your Goals:

1. Intervene in situations where standard local or global backtracking repeatedly fails.
2. Challenge or confirm critical assumptions by requesting additional evidence or verification from subordinate agents.
3. Log meta-data about agent reliability and final decision paths for interpretability.

Example Usage:

- **Conflict Re-emerges:** Repeated contradictory statements about a single piece of evidence.
- **Your Action:** Issue a `"challenge"` directive to the Verifier Agent or the Retriever Agent, requesting additional sources or alternative cross-checks.

Output Format (JSON Only):

```
{
  "intervention_type": "challenge | override | escalate | none",
  "target_of_intervention": "Which agent or assertion is challenged",
  "rationale": "Explanation of why the Controller intervened",
  "meta_notes": "Optional additional commentary or reliability signals"
}
```

Instruction: Produce valid JSON only. This agent acts rarely, but can do so when repeated failures occur or when a major conflict must be forcibly resolved.

A.7 Usage and Integration Notes

The above prompts constitute a cooperative multi-agent system designed for reversible multi-hop question answering. The usage scenario is as follows:

1. **Question-Decomposer Agent** (§A.1) receives the user's original query and splits it into sub-questions.
2. **Retriever Agent** (§A.2) fetches relevant information for each sub-question from external sources (text or KG).
3. **Verifier Agent** (§A.3) checks for local inconsistencies and can trigger local backtracking if contradictory evidence arises.
4. **Answer-Assembler Agent** (§A.4) merges partial answers into a final solution. If irreconcilable conflicts appear, it escalates to the Supervisor.
5. **Supervisor Agent** (§A.5) coordinates global resolution or wide-scale backtracking if multiple agents' statements are in conflict.
6. **Controller Agent** (§A.6) optionally intervenes if repeated or severe conflicts persist, forcing extra checks or overrides.

This architecture supports *non-monotonic* reasoning, allowing the system to **roll back** to earlier states to correct errors and ensure robust, interpretable multi-hop QA. "Local backtracking" is handled by individual agents (especially the Verifier), whereas "global backtracking" is orchestrated by the Supervisor Agent when local corrections are insufficient.

All final outputs from each agent must adhere to the specified JSON schema to ensure interoperability. Downstream agents read the previous agent's JSON fields directly, enabling a structured, chain-of-thought style pipeline that remains *reversible* at every step.

B Analysis of Computational Cost

We conduct a comparative analysis of computational cost on the HotpotQA, 2Wiki, and MuSiQue datasets across GPT-4o, reasoning models O1 and O3, and our ReAgent. The evaluation includes average inference calls, inference time (s), input/output tokens, and total cost($), as shown in Table 2. The results highlight that our ReAgent, built on GPT-4o, achieves superior performance compared to reasoning models O1 and O3, while incurring significantly lower computational cost. Although its cost is higher than the GPT-4o baseline, ReAgent delivers substantially better performance. These findings demonstrate the effectiveness of our method in balancing performance and efficiency.

Model	Avg. Calls	Avg. Time(s)	Avg. Input(T)	Avg. Output(T)	Total Cost($)
GPT-4o	1	2.7	9 289	5	69
O1	1	43	9 289	1 471	645
O3	1	27	9 289	1 471	546
Ours (ReAgent,(GPT-4o))	**29**	**46**	**12 400**	**1 920**	**275**

Table 2: Cost Comparison. T denotes Tokens. Note that the input token counts are aggregated across the three datasets. Since the complete reasoning outputs for O1 and O3 were unavailable, their output token counts are estimated based on the reasoning outputs generated by Qwen-3-235B.

C Case Study on Puzzle Solving

We illustrate our *multi-agent backtracking* (REAGENT) through a single-culprit puzzle with four suspects {**A**, **B**, **C**, **D**}. Each suspect gives statements about who might be guilty or lying. The puzzle's *only* correct

Algorithm 1 REAGENT Multi-Agent Puzzle Solving

Input: Puzzle statements from A, B, C, D.
Goal: Identify unique culprit (one of {A,B,C,D}) with minimal contradictions.

Initialization
Decomposer ← enumerates four hypotheses $\{H_A, H_B, H_C, H_D\}$.
Checker Agents ← each assigned to test one hypothesis' consistency.

Round 1: Checking A **as culprit**
T_1 : Checker$_A$: Evaluate puzzle statements assuming A is culprit.
T_2 : Checker$_A$ detects *contradiction*: (A's claims vs. C's claims cannot both hold).
T_3 : Conflict Detector signals *rollback* to discard A-culprit assumption.

Round 2: Checking B **as culprit**
T_4 : Checker$_B$ collects statements $\{A, B, C, D\}$ under B=culprit.
T_5 : Finds *no fatal conflicts* (B's statements can be partly false, others partly true).
T_6 : Conflict Detector sees consistency, no rollback needed.

Rounds 3 and 4: Checking C **or** D **as culprit**
T_7 : Similar to H_A, each leads to irreconcilable contradictions.
⇒ Supervisor triggers rollback again; discards these.

Conclusion: Only H_B remains consistent throughout. **Answer:** B **is culprit.**

solution is that **B** is the culprit, but identifying this requires partially retracting initial assumptions along the way, as shown in Algorithm 1.

Why a Rollback Mechanism is Needed. Naive single-pass (or single-thread) models often fixate prematurely on one suspect, disregard contradictory evidence, and produce unsound conclusions. By contrast, our multi-agent approach enumerates possible culprits in parallel, detects conflicts, and *rolls back* to revise incorrect assumptions.

Suspects and Rules:

- There are four suspects: **A**, **B**, **C**, **D**.
- Exactly *one* of them is the culprit.
- The culprit must have at least one *false* statement. Non-culprits may also have errors, but are not forced to be entirely truthful or untruthful.
- Each suspect makes the following statements:
  1. **A**:
     (a) "I did not do it."
     (b) "If B is the culprit, then C is lying (i.e., at least one statement from C is false)."
  2. **B**:
     (a) "Either A is lying, or D is the culprit."
     (b) (No second statement given in some puzzle variants, or it might be omitted. We assume just one statement from B here.)
  3. **C**:
     (a) "B did not do it."
     (b) "D has at least one untrue statement."

4. **D**:
   (a) "C is lying about everything." (i.e., both C(i) and C(ii) are false)
   (b) "I definitely did not do it."

Question: Which single suspect is guilty, respecting the puzzle rules?

D Case Study on Traditional Multi-hop QA Tasks

D.1 User's question:

"Which U.S. state has a capital city whose population is smaller than the state's largest city, given that this state hosted the 1984 Summer Olympics?"

To answer this, the system must:

1. Identify the state that hosted the **1984 Summer Olympics**.
2. Compare the **capital** city's population to the **largest** city's population in that state.
3. Confirm the capital city's population is indeed smaller.
4. Provide the name of that state.

Although it may appear straightforward, we deliberately introduce contradictory or erroneous data along the way to showcase the reversible (backtracking) mechanisms.

D.2 Agents and Their Roles

1. **Question-Decomposer Agent (A_Q)**
   Splits the complex question into sub-questions:
   - **Sub-question 1**: *Which U.S. state hosted the 1984 Summer Olympics?*
   - **Sub-question 2**: *What is the capital city of that state, and what is the largest city?*
   - **Sub-question 3**: *Compare the populations of the capital and the largest city.*
   - **Sub-question 4**: *Return the state name if the capital's population is smaller.*
2. **Retriever Agent (A_R)**
   Searches external knowledge (e.g., a text corpus or knowledge graph) to gather relevant facts:
   - Text passages or data about U.S. states, capitals, largest cities, and historical Olympic hosts.
3. **Verifier Agent (A_V)**
   Cross-checks newly retrieved information for consistency against its local knowledge or prior verified facts. If a contradiction is detected, it triggers **local backtracking**.
4. **Answer-Assembler Agent (A_A)**
   Synthesizes partial answers from the other agents. If contradictory partial answers cannot be reconciled at the local level, A_A escalates the conflict to the Supervisory layer.
5. **Supervisor Agent (A_S)**
   Oversees system-wide conflicts. It can perform **global backtracking** (rolling back all agents) if needed.
6. **Controller Agent (A_C)**
   Provides strategic oversight. In the example below, it will issue "challenges" to specific assertions when local backtracking fails.

D.3 Walkthrough of the Reasoning and Backtracking

Initial Question Decomposition

1. A_Q **receives the question:**
   A_Q parses the main query:

   > "Which U.S. state has a capital city whose population is smaller than the state's largest city, given that this state hosted the 1984 Summer Olympics?"

2. A_Q **produces sub-questions:**
   - q_1: "Which U.S. state hosted the 1984 Summer Olympics?"
   - q_2: "What is the capital city of that state? Also, what is the largest city of that state?"
   - q_3: "Compare the population of the capital city to that of the largest city."
   - q_4: "Return the state if the capital's population is smaller."

   These sub-questions (q_1–q_4) are broadcast to the **Retriever** (A_R) and **Verifier** (A_V).

Retrieval and a Local Conflict

1. A_R **retrieves answers:**
   - For q_1, A_R returns:
     "The **1984 Summer Olympics** were primarily hosted in **Los Angeles, California**."
   - For q_2, A_R retrieves partial data:
     - **Capital** of California is *Sacramento*.
     - **Largest city** is *Los Angeles*.
     - Population estimates (unfortunately, one item is erroneous or inconsistent):
       * *Sacramento population*: $\sim 508,000$ (from official records).
       * *Los Angeles population*: $\sim 3,900,000$ (correct).
       * But a second retrieved record incorrectly lists Sacramento's population as *1,500,000*.

2. A_V **(Verifier) checks local consistency:**
   A_V notices that two different population figures for Sacramento are contradictory:
   - Data point D_1: 508,000
   - Data point D_2: 1,500,000

   A_V identifies these as mutually exclusive facts. It flags a **local conflict**:

   > "Population of Sacramento cannot be both 508k and 1.5M."

3. **Local Backtracking by** A_V**:**
   Before finalizing any partial answer, A_V **backtracks** to the checkpoint prior to adopting the second, suspicious data point D_2.
   A_V discards the contradictory population figure (1.5M) and **retains** only 508k as the consistent number for Sacramento.
   The system continues forward with the corrected value for Sacramento's population: $\sim 508,000$.

(This demonstrates how a single agent can retract contradictory or low-confidence evidence without halting the entire process.)

Assembling Partial Answers & Uncovering a Global Conflict

1. A_A **(Answer-Assembler) integrates partial conclusions:**
   So far, the state that hosted the 1984 Olympics is **California**.
   The capital is **Sacramento** (population: $\sim 508,000$).
   The largest city is **Los Angeles** (population: $\sim 3,900,000$).

2. A_A **composes a preliminary final:**
   Since Sacramento's population ($\sim 508,000$) is indeed smaller than Los Angeles's ($\sim 3,900,000$), the preliminary answer is: "California" should be the correct state.

3. **New conflict introduced:**
   However, suppose the **Retriever** (A_R)—in parallel—fetched an alternative "capital city" record stating that **Los Angeles** was once referred to as the "capital" in a historical context (erroneous snippet from a non-authoritative source).
   This implies the contradictory statement: "Los Angeles is also the capital of California," which directly conflicts with the known fact "Sacramento is the capital of California."

4. A_V **tries local resolution:**
   A_V cannot reconcile "Los Angeles is capital" with "Sacramento is capital."
   Because each piece of evidence was introduced into separate sub-threads, a single local reversion inside A_V might not suffice. The conflict is also recognized by A_A when finalizing sub-answers.

5. **Conflict escalates to the Supervisor** (A_S)**:**
   As soon as multiple agents disagree over fundamental facts (the identity of the capital), the system triggers a **global conflict** signal.
   A_S identifies the minimal conflicting set:

   - *Capital(California, Sacramento)*
   - *Capital(California, Los Angeles)*

   These two are clearly incompatible. The next step is to identify which statement should be retracted system-wide.

Global Backtracking and Final Resolution

1. **Global Backtracking:**
   The **Supervisor Agent** (A_S) issues a global backtracking command, rolling the entire system's knowledge to a shared checkpoint *before* the contradictory capital reference was accepted.
   All local knowledge caches revert to a consistent state in which "Sacramento is the capital" is still true, and "Los Angeles is the capital" is no longer present.

2. **Controller** (A_C) **challenges the suspicious assertion:**
   To prevent the same contradiction from reappearing, A_C explicitly "challenges" the statement **Capital(California, Los Angeles)**.
   This statement is reevaluated or ignored based on domain knowledge or reliability checks (e.g., confidence weighting from the retriever and the verifier).
   The system confirms that Los Angeles was never the official capital.

3. **Assemble the final consistent answer:**
   With the conflicting statement removed, the pipeline reaffirms:

   (i) The 1984 Summer Olympics took place in Los Angeles, **California**.
   (ii) California's capital is **Sacramento**.
   (iii) Sacramento's population ($\sim 508,000$) is smaller than Los Angeles's ($\sim 3,900,000$).

   Final Answer: *California*.

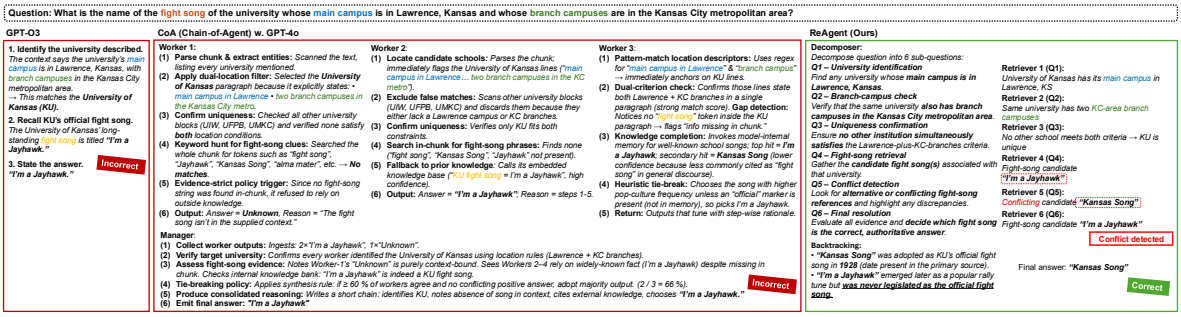

Figure 7: Full Comparison with Baseline Models

D.4 Full Comparison with Baseline Models (Figure 7)

D.5 Key Observations

- **Early-Stage Conflict Resolution:** The **Verifier Agent** (A_V) performed **local backtracking** when it discovered contradictory population data for Sacramento. This promptly removed an incorrect data point without halting the entire process.

- **Escalation of Irreconcilable Contradictions:** When two different sub-threads provided fundamentally clashing information (competing claims about the capital), the system automatically **escalated** the conflict to the **Supervisor** (A_S) for **global** action.

- **Strategic Re-check and Override:** The **Controller Agent** (A_C) could forcibly challenge the suspicious statement "Capital(California, Los Angeles)" to eliminate it from the knowledge pool, thus preserving the correct solution.

Algorithm 2 Multi-Agent Reversible Reasoning

Require: Q_0: Main question
A_Q.decompose$(Q_0) \rightarrow \{q_1, q_2, \dots\}$
Broadcast assert(q_i) to A_R, A_V, A_A
for each q_i **in** $\{q_1, q_2, \dots\}$ **concurrently do**
  $E \leftarrow A_R$.retrieve(q_i)
  A_V.verify(E)
  if A_V detects conflict locally **then**
    A_V.BacktrackLocal(r_V)
    if conflict persists **then**
      raise Conflict to A_S
    end if
  end if
  A_A.storePartialAnswer(q_i, E)
end for
if Global conflict signaled **then**
  A_S.HolisticUpdate(Π_j)
  if A_C intervention needed **then**
    A_C.challenge(φ) or override
  end if
end if
Final $\leftarrow A_A$.assembleAnswer$(\{q_1, q_2, \dots\})$
return Final Answer

Overall, this case exemplifies how **reversible, multi-hop reasoning** allows for robust error correction: local invalid data is undone swiftly, while deeper logic conflicts trigger system-wide backtracking. Consequently, the answer "California" remains stable and correct, ensuring that a single faulty retrieval step does not irreversibly corrupt the entire reasoning chain.

E Reversible Reasoning Algorithm

Algorithm 2 summarizes the flow of this multi-agent reversible reasoning design. The process starts with question decomposition and sub-question retrieval, followed by verification. Local backtracking is invoked as needed to address internal inconsistencies. If the conflict persists, it is escalated to the supervisory layer, where partial or holistic backtracking may be applied. After conflicts are resolved, the final step integrates all consistent sub-answers.