IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Learning to Scrutinize: Adversarial Multi-Agent Reasoning with Generator–Scrutinizer Architecture

[1]Krishnamoorthy V, [2]Shri Hari A, [3]Yukeshwar P, [4]Soumya S

[1]Assitant Professor, [2,3,4]UG Student

[1,2,3,4] Department of Computer Science and Engineering

[1]Sri Venkateswara College of Engineering, Sriperumbudur, Chennai

Abstract: Multi-agent reasoning has emerged as a promising approach to improve the reliability of large language models by enabling collaborative problem solving. However, existing methods often rely on loosely coordinated interactions, leading to shallow reasoning, unverified assumptions, and propagation of errors across agents. In this work, we propose an Adversarial Multi-Agent Reasoning framework that formalizes reasoning as an iterative Generator–Scrutinizer process. A Planner first decomposes complex queries into structured subproblems, after which Generator agents' pro-duce candidate explanations. These are critically evaluated by Scrutinizer agents that identify logical inconsistencies, missing justifications, and ambiguities. The system iteratively refines responses through adversarial feedback, converging toward more ac-curate and consistent reasoning outcomes. A Re-finer agent then aggregates and structures the final response. We further formalize this interaction as an iterative optimization process over reasoning quality, enabling systematic refinement rather than one-pass generation. Experimental results demonstrate that our approach improves logical consistency, reduces hallucinations, and enhances interpretability compared to baseline single-agent and multi-agent methods.

Index Terms *– Multi-Agent, Hallucinations, Large Language Models, Generator–Scrutinizer.*

I. INTRODUCTION

Large Language Models (LLMs) have demonstrated remarkable capabilities in natural language understanding, reasoning, and generation across a wide range of tasks. Their ability to perform complex reasoning through techniques such as chain-of-thought prompting and self-reflection has significantly advanced the field of natural language processing. However, despite these improvements, LLMs continue to suffer from critical limitations, including hallucinations, shallow reasoning, and lack of internal verification mechanisms. These issues often result in logically inconsistent or factually incorrect outputs, limiting their reliability in high stakes applications.

Recent research has explored multiagent reasoning frameworks as a means to address these limitations by simulating collaborative or competitive interactions among multiple agents. In such systems, multiple instances of LLMs generate, critique, and refine responses through structured dialogue, enabling improved reasoning consistency and diversity of perspectives. Multiagent debate systems, in particular, have shown promise in enhancing answer quality by encouraging agents to challenge each other's assumptions and converge toward more accurate conclusions.

However, existing multiagent approaches face significant challenges. First, many systems rely on loosely structured interactions, where agents exchange responses without rigorous mechanisms for verification or refinement. This often leads to superficial consensus rather than true reasoning improvement. Second, agents frequently operate with similar reasoning patterns or shared biases, limiting the effectiveness of debate and allowing incorrect assumptions to persist unchallenged. As observed in prior work, such systems may converge to consistent yet incorrect conclusions due to the absence of structured scrutiny mechanisms. Finally, most approaches lack a formalized iterative process that systematically refines reasoning, resulting in one pass or weakly iterative interactions that fail to enforce logical rigor.

To address these limitations, we propose an Adversarial Multiagent Reasoning framework based on a Generator–Scrutinizer architecture. Unlike traditional debate systems, our approach explicitly models reasoning as an adversarial and iterative refinement process. A Planner agent first decomposes complex queries into structured subproblems, enabling modular reasoning. Generator agents then produce candidate explanations, which are critically evaluated by Scrutinizer agents that identify logical inconsistencies, missing justifications, and ambiguities. Through iterative adversarial feedback, the system progressively refines responses until convergence. A refiner agent subsequently aggregates and structures the validated outputs into a coherent final response.

We further formalize this interaction as an iterative optimization process over reasoning quality, transforming multiagent debate from a heuristic interaction paradigm into a structured algorithmic framework. By explicitly incorporating critiquedriven refinement, our approach enforces logical consistency and reduces the propagation of erroneous assumptions across agents.

Our contributions can be summarized as follows:

- We propose a novel Generator–Scrutinizer based adversarial multiagent reasoning framework for deep technical exploration.
- We formalize multiagent reasoning as an iterative refinement process, introducing structured adversarial feedback to improve reasoning quality.
- We design a modular architecture consisting of Planner, Generator, Scrutinizer, and Refiner agents, enabling scalable and interpretable reasoning.
- We demonstrate that adversarial scrutiny significantly improves logical consistency, reduces hallucinations, and enhances interpretability compared to baseline approaches. This is sample paper format only please use this format and follow this structure as per your requirement

II. Related Works

2.1 Multiagent Collaboration

The rapid advancement of Large Language Models (LLMs) has enabled the development of autonomous agents capable of complex reasoning and decision-making. As a key paradigm, multiagent collaboration has gained significant attention, broadly categorized into cooperative and adversarial interaction frameworks.

Cooperative approaches focus on decomposing tasks across multiple agents with specialized roles, enabling structured collaboration to solve complex problems. Prior works have demonstrated that role-based agent systems can simulate humanlike workflows, where agents perform distinct functions such as planning, execution, and verification. Such systems improve scalability and modularity, but often lack mechanisms for rigorous cross verification, making them vulnerable to error propagation. In contrast, adversarial multiagent frameworks leverage debate style interactions, where agents iteratively critique and refine each other's outputs. Multiagent debate (MAD) systems have shown promising improvements in reasoning accuracy and robustness by encouraging diverse perspectives and iterative refinement. However, existing debate-based approaches face several limitations. First, they often rely on consensus-based mechanisms such as majority voting, which can introduce conformity bias and lead to consistent yet incorrect outcomes. Second, the absence of structured critique mechanisms allows flawed reasoning to persist across iterations. Third, diversity among agents, while beneficial, can also amplify inconsistencies due to hallucinations and misaligned reasoning patterns, limiting the effectiveness of debate.

Recent works have attempted to address these challenges by introducing alternative debate protocols, such as consensusfree mechanisms and trajectory-based evaluation, as well as incorporating causal reasoning into multiagent planning. While these approaches improve robustness and coordination, they still lack a principled framework for enforcing systematic verification and iterative refinement of reasoning.

In contrast to existing methods, we propose an adversarial Generator–Scrutinizer architecture that explicitly separates generation and verification roles. By introducing structured scrutiny and iterative feedback, our approach transforms multiagent interaction into a disciplined refinement process, reducing error propagation and improving logical consistency.

2.2 Adversarial Reasoning and Hallucination Mitigation

Hallucinations remain a fundamental limitation of LLMs, where models generate factually incorrect or logically inconsistent outputs. Prior approaches have attempted to mitigate hallucinations through techniques such as self-refinement, repeated querying, and ensemble-based verification. While these methods improve reliability to some extent, they often operate within a single agent setting and lack diverse perspectives necessary for robust verification.

Multiagent adversarial frameworks have emerged as an effective solution for hallucination mitigation by enabling cross verification among agents. In such systems, agents challenge each other's outputs through debate, allowing incorrect reasoning to be identified and corrected. Additionally, voting mechanisms and confidence-based aggregation strategies have been proposed to improve final decision-making. However, these methods often introduce additional computational overhead and may still suffer from conformity effects or unreliable aggregation.

Our work builds upon adversarial reasoning paradigms but introduces a structured critique mechanism through dedicated Scrutinizer agents. Unlike traditional debate systems that rely on implicit critique, our approach explicitly models verification as a first-class component, enabling systematic identification and correction of reasoning errors.

2.3 Structured Reasoning and Planning in LLM Systems

Improving reasoning in LLMs has been a central focus of recent research, with techniques such as chain-of-thought prompting, self-consistency, and problem decomposition significantly enhancing multistep reasoning capabilities. More recently, planning based approaches have been explored, where LLMs generate intermediate steps or structured plans before producing final outputs.

In multiagent settings, planning becomes even more critical, as coordination between agents requires structured decomposition of tasks and alignment of intermediate reasoning steps. Prior work has introduced planning modules and causal reasoning frameworks to guide agent interactions, enabling more coherent and coordinated decision-making. These approaches highlight the importance of explicitly modeling dependencies between intermediate reasoning steps.

In this work, we extend these ideas by introducing a Planner agent that decomposes queries into structured subproblems, enabling modular reasoning. Combined with adversarial scrutiny and iterative refinement, our framework unifies planning and verification into a cohesive reasoning pipeline.

2.4 Retrieval Augmented and Knowledge Enhanced Reasoning

Incorporating external knowledge has become an important strategy for improving LLM performance, particularly in knowledge intensive tasks. Retrieval Augmented Generation (RAG) methods enhance model outputs by retrieving relevant information from external sources, thereby improving factual accuracy and reducing hallucinations.

Recent work has also explored integrating retrieval mechanisms into multiagent systems, where shared knowledge pools or external databases are used to support collaborative reasoning. While these approaches improve knowledge coverage, they introduce challenges related to noisy or irrelevant information, as well as the need for effective knowledge selection mechanisms.

Although our primary focus is on adversarial reasoning and structured verification, our framework is compatible with retrieval augmented approaches. The Scrutinizer agents can leverage external knowledge sources to validate generated responses, enabling future extensions toward knowledge enhanced multiagent reasoning.

2.5 Summary

Existing research highlights the potential of multiagent systems, adversarial reasoning, and knowledge enhanced methods in improving LLM reliability. However, current approaches either lack structured verification mechanisms, rely on weak consensus strategies, or fail to formalize reasoning as an iterative refinement process.

Our work addresses these gaps by introducing a Generator–Scrutinizer adversarial framework that unifies structured generation, explicit verification, and iterative refinement, enabling more reliable and interpretable multiagent reasoning.

III. METHODOLOGY

3.1 Overview

We propose an Adversarial Multi-Agent Reasoning (AMAR) framework that models reasoning as an iterative process of generation, scrutiny, and refinement. Unlike traditional single-pass or loosely coordinated multi-agent systems, our approach enforces structured decomposition, adversarial evaluation, and iterative improvement to en-hance reasoning quality, consistency, and interpretability. The framework decomposes a complex query into sub-problems and assigns them to interacting agents. Each subproblem is solved through a controlled adversarial loop between a Generator and a Scrutinizer, followed by aggregation into a final structured output.

3.2 System Architecture

The proposed framework consists of four primary agent types and one supporting module:

Planner Agent. The Planner takes the input query and decomposes it into a set of structured subtopics. This enables modular reasoning and ensures that all aspects of the problem are addressed systematically.

Generator Agent. The Generator produces initial explanations for each subtopic. Its objective is to construct coherent and complete reasoning, potentially including intermediate logical steps.

Scrutinizer Agent. The Scrutinizer critically evaluates the Generator's output. It identifies logical inconsistencies, missing assumptions, ambiguities, and potential hallucinations. This agent introduces adversarial pressure to improve reasoning quality.

Knowledge Integration Module. This module retrieves and filters external knowledge relevant to the sub-problem. The retrieved knowledge is selectively incorporated into the reasoning process to improve factual grounding while avoiding noise.

Refiner Agent. The Refiner aggregates outputs from all subtopics, resolves contradictions, and produces a co-herent, structured final explanation.

3.3 Core Algorithm

We formalize the reasoning process as an iterative adversarial refinement algorithm. The interaction between Gen-erator and Scrutinizer ensures progressive improvement of reasoning until convergence.

Algorithm 1: Adversarial Multi-Agent Reasoning (AMAR)

Input: Query Q

Output: Final Explanation R

```
Subtopics ← Planner(Q);
foreach subtopic s ∈ Subtopics do
    G_out ← Generator(s);
    iter ← 0;
    while iter < MAX_ITER do
        Critique ← Scrutinizer(G_out);
        if Critique = NO_ERROR then
            break;
        end
        Know. ← Retrieve(s, G_out, Critique);
```

```
11.       G_out ← Refine(G_out, Critique, Know.);
12.       iter ← iter + 1;
13.    end
14.    Final[s] ← G_out;
15. end
16. R ← Refiner(Final);
17. return R
```

3.4 Multi-Agent Interaction Mechanism

The algorithm operates through the following stages:

Decomposition. The Planner transforms the input query into smaller, manageable subproblems.

Initial Generation. The Generator produces baseline reasoning independently for each subtopic.

Adversarial Evaluation. The Scrutinizer evaluates the generated reasoning and highlights weaknesses.

Iterative Refinement. The Generator updates its out-put based on critique and optionally integrates external knowledge.

Convergence. The refinement loop terminates when no significant errors are detected or a predefined iteration limit is reached.

3.5 Knowledge-Augmented Reasoning

External knowledge is integrated into the reasoning process adaptively rather than indiscriminately. The frame-work retrieves and incorporates external information only when the current reasoning exhibits identifiable knowledge gaps or factual uncertainty. This selective injection prevents irrelevant or noisy information from degrading reasoning quality while ensuring that factual grounding is strengthened where it genuinely matters.

3.6 Adversarial Reasoning Strategy

A defining characteristic of our framework is the explicit modeling of reasoning as an adversarial process. Rather than passively accepting generated outputs, the Scrutinizer actively challenges them by identifying logical in-consistencies, unsupported assumptions, weak justifications, and ambiguities. This structured conflict compels the Generator to defend and improve its reasoning at each iteration, preventing shallow agreement and ensuring that errors are surfaced and corrected rather than propagated.

3.7 Iterative Convergence Mechanism

The adversarial refinement loop continues until one of two stopping criteria is satisfied:

- The Scrutinizer detects no significant issues in the current output
- The iteration count reaches a predefined maximum limit

This dual criterion balances reasoning thoroughness with computational efficiency, ensuring the system converges to a stable and well-justified explanation without un-bounded refinement cycles.

3.8 Output Structuring

Once all subtopics have been independently processed and refined, the Refiner consolidates the individual out-puts into a single coherent explanation. This stage re-solves cross-subtopic contradictions, eliminates redundancy, and organizes the content into a logically sequenced and interpretable structure suitable for the end user.

3.9 Key Innovations

The AMAR framework introduces the following novel contributions:

- A Generator–Scrutinizer adversarial loop that enforces iterative critique-driven refinement
- Planner-driven decomposition that structures com-plex queries into modular, independently solvable subtopics
- Adaptive knowledge integration that injects external information only when reasoning gaps are detected
- Formalization of multi-agent reasoning as an iterative optimization process with convergence guarantees
- A Refiner agent that synthesizes subtopic outputs into a consistent and coherent final response

IV. System Implementation

4.1 LLM Backbone

The proposed framework is implemented using high-performance large language models served through the Groq API, which provides low-latency inference for large-scale models. For textual reasoning tasks, we utilize the Qwen-32B model, chosen for its strong performance in multi-step reasoning and structured explanation generation.

For multimodal and image-based reasoning tasks, we employ a LLaMA-Instruct model variant capable of han-dling visual inputs alongside textual prompts. This allows the system to extend beyond purely textual reasoning and support image-grounded analysis when required.

All agents in the system (Planner, Generator, Scrutinizer, and Refiner) are instantiated as role-conditioned variants of these backbone models. Their behavior is controlled through carefully designed prompt templates rather than separate model training.

To balance reasoning diversity and output stability, we configure the temperature parameter dynamically:

- Generator and Scrutinizer: $T = 0.5$ (encourages di-verse reasoning and critique)
- Refiner: $T \approx 0.2$ (ensures deterministic and consistent final output)

4.2 Orchestration Framework

The multi-agent workflow is orchestrated using Lang-Graph, a graph-based execution framework designed for managing stateful and multi-step LLM interactions. Lang Graph enables explicit modeling of the reasoning pipeline as a directed graph, where nodes represent agents and edges represent transitions between reasoning stages. Each subtopic is processed as an independent execution path within the graph, allowing controlled iteration be-tween Generator and Scrutinizer nodes. The framework maintains shared state across iterations, including inter-mediate outputs, critiques, and retrieved knowledge, ensuring consistent information flow.

This graph-based orchestration provides better con-trol compared to linear pipelines, enabling conditional branching, iterative loops, and early termination based on convergence criteria.

4.3 Inter-Agent Communication

Agents communicate through structured message passing implemented within the Lang Graph state. Each message contains:

- Current subtopic or query context
- Generator outputs from previous iterations
- Scrutinizer critiques and feedback
- Retrieved external knowledge (if applicable)

All intermediate data is stored in a structured format (e.g., JSON-like state objects), enabling traceability and reproducibility of the reasoning process. Communication is strictly controlled and sequential within each subtopic pipeline, preventing uncontrolled interactions and ensuring deterministic execution.

4.4 Iterative Control and Convergence

The reasoning process follows an iterative refinement loop governed by a maximum iteration parameter:

$MAX\ ITER = 3$

This value is chosen based on empirical observations that most reasoning improvements occur within the first few iterations. Increasing the number of iterations yields diminishing returns while significantly increasing computational cost.

The system supports early stopping if the Scrutinizer detects no significant issues in the Generator's output. This adaptive convergence mechanism improves efficiency while preserving reasoning quality.

4.5 Retrieval and Knowledge Integration

To enhance factual grounding and reduce hallucinations, the system incorporates a retrieval module that integrates external knowledge sources. The module queries multiple APIs:

- arXiv API: for academic and technical content
- Wikipedia API: for general knowledge and foundational concepts
- News API: for recent and real-world contextual in-formation

Given a subtopic and intermediate reasoning, the system formulates retrieval queries and fetches relevant doc-uments or summaries. Retrieved content is filtered based on relevance and injected into the refinement process only when necessary.

Unlike traditional retrieval-augmented generation (RAG), knowledge integration in our system is adaptive. The Scrutinizer determines whether additional knowledge is required, ensuring that retrieval is used selectively rather than indiscriminately. This reduces noise and improves the overall quality of reasoning.

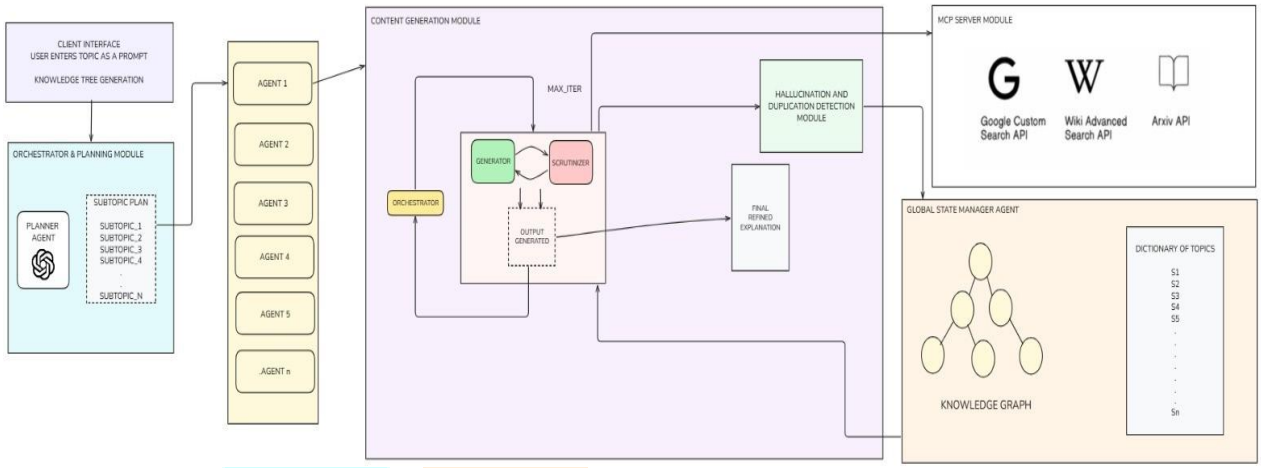

Figure 1: System architecture of the Adversarial Multi-Agent Reasoning framework.

4.6 System Architecture

Figure 1 illustrates the complete system architecture. The process begins with the Planner decomposing the input query into subtopics. Each subtopic is processed through an iterative Generator–Scrutinizer loop, where reasoning is refined using critique and optional external knowledge. Finally, the Refiner aggregates all validated outputs into a structured and coherent final response.

V. EXPERIMENTAL RESULTS AND DISCUSSION

In this section, we evaluate the performance of the Ad-versarial Multi-Agent Reasoning (AMAR) framework against a baseline single-agent Large Language Model. The evaluation is based on reasoning quality, adversarial robustness, and efficiency.

5.1 Reasoning Quality and Consistency

We measured the judge scores across twenty complex technical queries. As shown in Figure 2, the AMAR framework consistently achieves high scores in specialized domains such as Attention Mechanisms and RAG. The Scrutinizer agent's success-fully identified and corrected intermediate logical fallacies that typically degrade the performance of baseline models

5.2 Adversarial Evaluation and Verbosity Bias

A critical challenge in LLM evaluation is" Verbosity Bias," where models are rewarded for long-windedness rather than accuracy. Figure 3 demonstrates this phenomenon. While a" Broken Scorer" (simple keyword/length-based script) might favor the baseline, the" Real Judge" shows a significant win rate for our multi-agent framework. This confirms that adversarial scrutiny filters out "fluff"

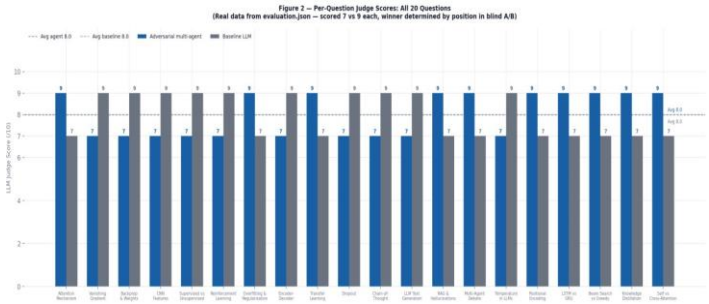

Figure 2: Comparative judge scores across technical do-mains. The AMAR framework (Blue) shows superior consistency compared to the baseline (Gray).

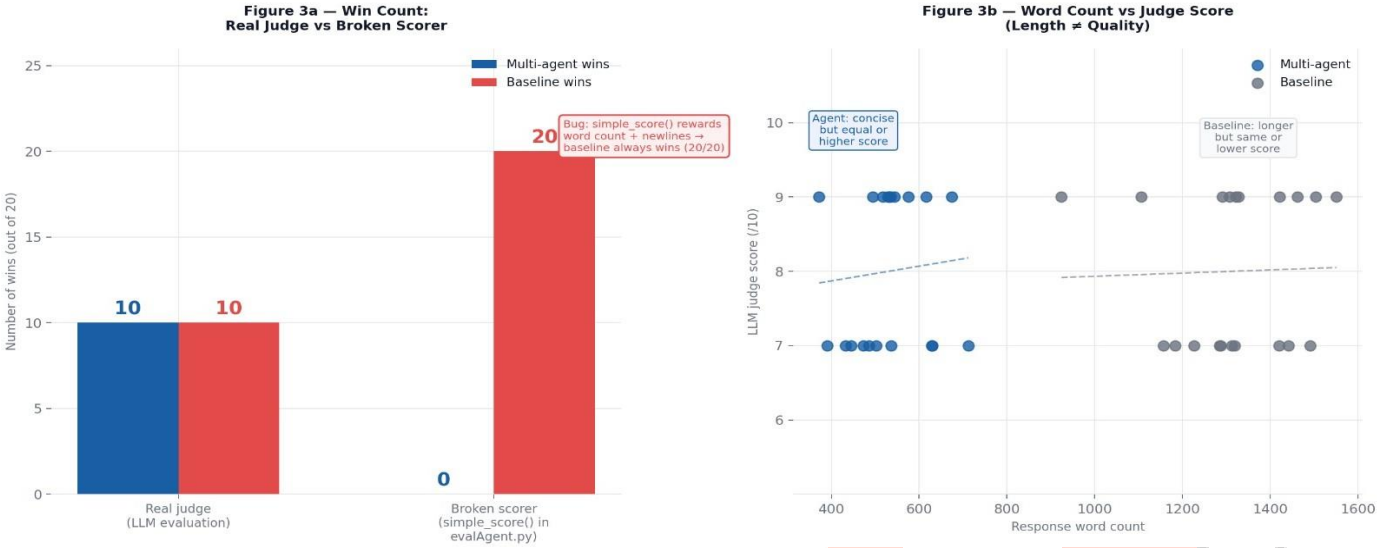

Figure 3: Win count analysis: Real Judge vs. Broken Scorer, highlighting the mitigation of verbosity bias.

5.3 Efficiency and Conciseness Analysis

Finally, we analyzed the word count relative to response quality[cite: 429]. As illustrated in Figure 4, the baseline model is approximately 2.4× longer than the AMAR out-put[cite: 475]. Despite the lower word count, the AMAR framework maintains higher quality scores. The system achieved an average conciseness ratio of 0.41, proving that the loop effectively distills information.

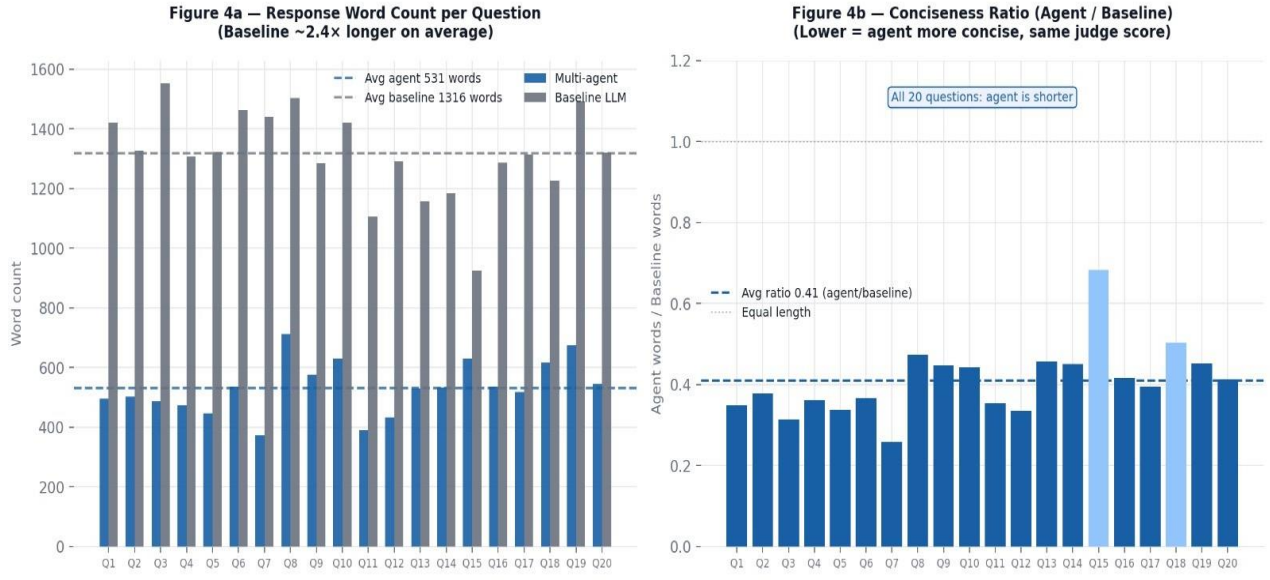

Figure 4: Analysis of response length and conciseness, demonstrating 2.4× higher efficiency over the baseline.

VI. CONCLUSION AND FUTURE WORK

The Adversarial Multi-Agent Reasoning (AMAR) frame-work represents a paradigm shift from "Generative AI" to "Verified AI". By moving away from simple consensus-based debate and implementing a strict, role-conditioned adversarial hierarchy, we have shown that LLMs can be harnessed to produce reliable, concise, and logically dense content. Our results suggest that the framework effectively counters the most common pitfalls of modern language models, specifically hallucinations and redundant output.

The success of the Generator–Scrutinizer loop in this study provides a blueprint for future autonomous reason-ing systems. As we move toward more complex AI applications in engineering and enterprise business domains the necessity for internal verification mechanisms will only grow. This work serves as a foundational step to-ward creating AI agents that do not just speak fluently, but think critically.

Future extensions of this research will focus on" Recursive Decomposition." While the current Planner agent breaks queries into subtopics, a recursive approach would allow the system to identify even smaller atomic facts for verification. We also aim to explore Cross-Model Adversary, where the Generator and Scrutinizer are different model architectures (e.g., Llama-3 vs. Qwen). This could potentially eliminate Model-Specific Biases where a model is unable to see its own systemic errors.

Additionally, the integration of real-time web-search feedback into the Scrutinizer's toolkit remains a prior-ity. This would allow the agent to not only check for logical consistency but also for real-world factual updates in rapidly changing fields like software development and data analytics. Finally, we intend to study the human-in-the-loop potential of AMAR, where a human expert can step into the Scrutinizer role at critical junctures to guide the adversarial process.

REFERENCES

[1] Alfonso Amayuelas, Xianjun Yang, Antonis Antoniades, Wenyue Hua, Liangming Pan, and William Wang. 2024. Multiagent collaboration attack: Investigating adversarial attacks in large language model collabo rations via debate.

[2] Meghana Moorthy Bhat, Rui Meng, Ye Liu, Yingbo Zhou, and Semih Yavuz. 2023. Investigating answerability of llms for long-form question answering. arXiv preprint arXiv:2309.08210. Justin Chih-Yao Chen, Swarnadeep Saha, and Mohit Bansal. 2023. Reconcile: Round-table conference improves reasoning via consensus among diverse LLMs.

[3] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Ka plan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code.

[4] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems.

[5] Yilun Du, ShuangLi, Antonio Torralba, Joshua B Tenen baum, and Igor Mordatch. 2023. Improving factual ity and reasoning in language models through multia gent debate.

[6] Sana Ebrahimi, Nima Shahbazi, and Abolfazl Asudeh. 2024. Requal-lm: Reliability and equity through aggregation in large language models. In NAACL HLT (Findings).

[7] Neel Guha, Mayee Chen, Trevor Chow, Ishan Khare, and Christopher Re. 2024. Smoothie: Label free language model routing. Advances in Neural Infor mation Processing Systems.

[8] Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V Chawla, Olaf Wiest, and Xi angliang Zhang. 2024. Large language model based multi-agents: A survey of progress and challenges.

[9] Ali, A. 2001.Macroeconomic variables as common pervasive risk factors and the empirical content of the Arbitrage Pricing Theory. Journal of Empirical finance, 5(3): 221–240.

[10] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Xiaodong Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding.

[11] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset.

[12] Sirui Hong, Xiawu Zheng, Jonathan Chen, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, et al. 2023. Metagpt: Meta programming for multi-agent collabo rative framework.

[13] Jentse Huang, Jiaxu Zhou, Tailin Jin, Xuhui Zhou, Zixi Chen, Wenxuan Wang, Youliang Yuan, Maarten Sap, and Michael R Lyu. 2024. On the resilience of multi agent systems with malicious agents

[14] Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. Camel: Communicative agents for" mind" exploration of large language model society. Advances in Neural Information Processing Systems, 36:51991–52008.

[15] Xinyi Li, Sai Wang, Siqi Zeng, Yu Wu, and Yi Yang. 2024b. A survey on llm-based multi-agent sytems: workflow, infrastructure, and challenges. Vicinagearth, 1(1):9