Enhancing Multi-Agent LLM Output Quality Through Adversarial Critique: A Cross-Domain Evaluation

Utkarsh Kalra
Department of Information Technology
Manipal University Jaipur
Jaipur, Rajasthan
utkarsh.229302529@muj.manipal.edu

Rohit Kumar Gupta
Department of Information Technology
Manipal University Jaipur
Jaipur, Rajasthan
rohitkumar.gupta@jaipur.manipal.edu

***Abstract*—Multi-agent large language model (LLM) systems have shown promising results by dividing tasks among multiple agents with different roles. However, many existing approaches focus mainly on collaboration and lack a dedicated mechanism for critically evaluating and improving generated responses. In this article, a multi-agent framework known as AdversarialMAS is introduced, which presents an adversarial critique stage to improve output quality. The framework consists of a Generator Agent that creates an initial response, a Critic Agent that analyzes the response, and a Revision Agent that refines the final output based on the feedback. Further AdversarialMAS is evaluated against three approaches: a Single-Agent LLM, a Sequential Multi-Agent pipeline, and a Self-Refine method. The evaluation is conducted across three domains—startup strategy development, research proposal generation, and software system design. Experimental results from automated metrics and LLM-based evaluation show that the proposed approach achieves the highest overall score of 4.19, with improvements in consistency, completeness, and faithfulness. The results suggest that adversarial critique can be an effective approach for improving the reliability and quality of multi-agent LLM outputs.**

***Index Terms*—Multi-agent systems, Large Language Models, adversarial critique, iterative refinement, LLM evaluation**

I. Introduction

Large language models (LLMs) have proved strong potential in generation, reasoning, and solving complex tasks across various domains [1]. However, single-agent LLM systems primarily rely on a single generation process and offer limited support for error detection, response validation, or content improvement. These limitations can be overcome by recent studies that have explored multi-agent LLM systems, where multiple agents collaborate by dividing tasks or adopting specialized roles [4], [5], [7].

Although role-based collaboration enhances task organization, assigning different roles alone does not always improve output quality. However, recent LLM-based multi-agent systems can be affected by adversarial influence and communication-level errors, especially in complex problem-solving scenarios [20]. Many existing multi-agent frameworks focus on information exchange and sequential task execution without explicitly challenging or evaluating intermediate results. As a result, incorrect assumptions or incomplete reasoning may continue through the generation process. This creates a need for stronger feedback mechanisms where outputs are critically analyzed and refined.

Self-refinement and feedback-driven generation approaches have demonstrated that iterative evaluation can improve LLM responses [2], [10]. However, existing approaches often combine generation and evaluation within the same model or provide limited separation between creation and critique. The use of independent agents dedicated to adversarial evaluation remains an important area for further investigation.

This article explains AdversarialMAS, a critique-based multi-agent framework that aims to enhance the output quality of LLMs through separated generation, evaluation, and revision stages. The framework consists of a Generator Agent that crafts the first output, an Adversarial Critic Agent that detects the output's flaws, and a Revision Agent that alters the output based on the given critique. The primary differentiation of our framework compared to cooperative pipelines is that our framework uses critique as a productive component of the generation process. Several review strategies that employ an adversarial approach have shown that the critique of intermediate output assists in the reliability of LLM for a given task [19].

Existing multi-agent architectures mainly emphasize task decomposition and collaboration, while limited work evaluates adversarial critique as a systematic quality improvement mechanism. To study this effect, AdversarialMAS is compared with a Single-Agent LLM baseline, a Sequential Multi-Agent pipeline, and a Self-Refine approach across different application domains. The main contributions ot this work are summarized as follows:

- A critique-driven multi-agent framework that separates generation, evaluation, and revision.
- A cross-domain evaluation of AdversarialMAS across different task complexities.
- A comparative analysis against single-agent, sequential multi-agent, and self-refinement approaches.

The remainder of this paper is organized as follows. Section II discusses related work on multi-agent LLM systems and refinement approaches. Section III introduces the proposed AdversarialMAS framework. Section IV describes the experimental setup and evaluation methodology. Section V presents results and discussion, which is followed by conclusions and future directions in Section VI.

II. Related Work

Multi-agent LLM systems have recently gained attention as a way to improve task solving by distributing responsibilities among multiple agents. Similarly distributed intelligent systems require reliable coordination and trust-aware decision-making mechanisms to improve overall system performance [16]. Frameworks such as MetaGPT [4] and AutoGen [5] introduced structured agent collaboration, where different agents perform specialized tasks and communicate to complete complex objectives. Similar collaborative approaches have been explored through agent-based systems such as CAMEL [7] and AgentVerse [8]. These methods demonstrate that role assignments and interaction between agents can improve task organization and problem-solving. However, most existing multi-agent frameworks primarily focus on cooperation and task completion rather than explicitly evaluating and correcting generated output through adversarial feedback.

Critique-based refinement has been studied as an effective approach for improving LLM responses. Prompt engineering techniques have also been discovered to enhance interaction quality and guide LLM behavior through structured prompting patterns [9]. An iterative process where a model generates an output, provides feedback on its own response, and revises the result based on that feedback is introduced as Self-Refine [2]. Similar feedback-driven strategies, including Reflexion [10] and automated correction methods [11], have shown that iterative improvement can enhance model performance on complex tasks. However, these approaches generally rely on the same model for both generation and evaluation, which may limit the diversity and effectiveness of the critique process. The separation of generation and evaluation into independent agents provides an alternative direction for improving output quality.

LLM-based evaluation has also become an important area for assessing generated content quality [13]. LLM-as-a-Judge approaches enable automated evaluation by using language models to score and compare responses based on human-aligned criteria [3]. Other studies have explored the reliability and alignment of LLM-based evaluation methods for measuring model performance [12], [14]. While these approaches provide useful quality assessment, evaluation is usually performed after generation rather than being integrated into the generation process itself.

Existing studies show that multi-agent collaboration, iterative improvement, and automated evaluation can enhance generated outputs. However, limited research has explored adversarial critique as part of a multi-agent generation process. Most current approaches rely on cooperative agents or self-feedback, where the same model generates and evaluates the response. The proposed AdversarialMAS framework addresses this limitation by introducing a separate critic agent that reviews generated outputs and provides structured feedback to improve the final result.

III. Proposed AdversarialMAS Framework

The proposed AdversarialMAS framework introduces adversarial critique as an integrated mechanism for improving multi-agent LLM output quality. Unlike traditional multi-agent workflows that mainly rely on cooperation and information exchange [4], [5], AdversarialMAS separates generation, evaluation, and revision into independent agent roles. This design enables generated responses to be critically examined and refined before producing the final output.

A. Framework Overview

AdversarialMAS consists of three specialized agents: a Generator Agent, an Adversarial Critic Agent, and a Revision Agent. These agents are assigned a domain-specific task prompt; the Generator Agent [6] creates an initial response based on the required objectives. These generated responses are then passed to the Adversarial Critic Agent, which analyzes

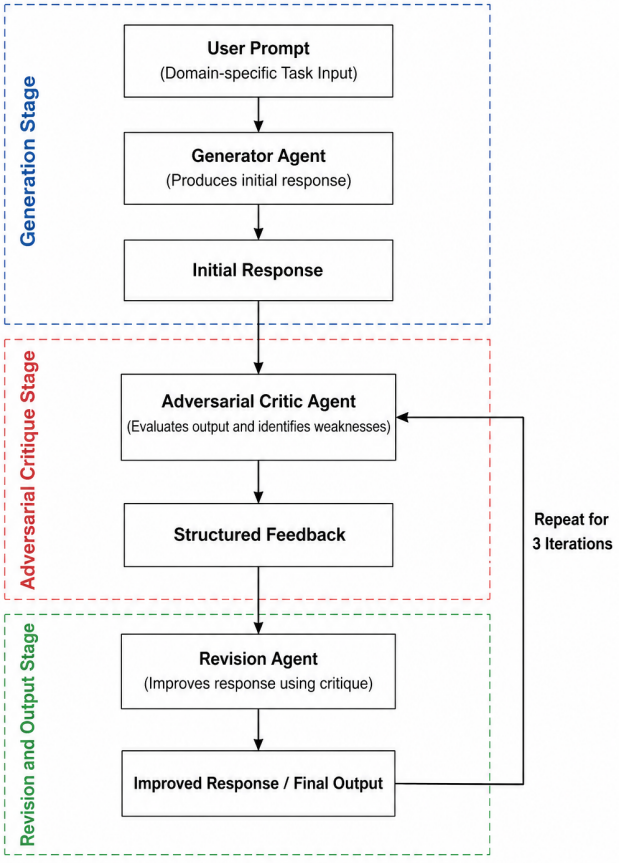

Fig. 1. Proposed AdversarialMAS framework consisting of generation, adversarial critique, and iterative revision stages.

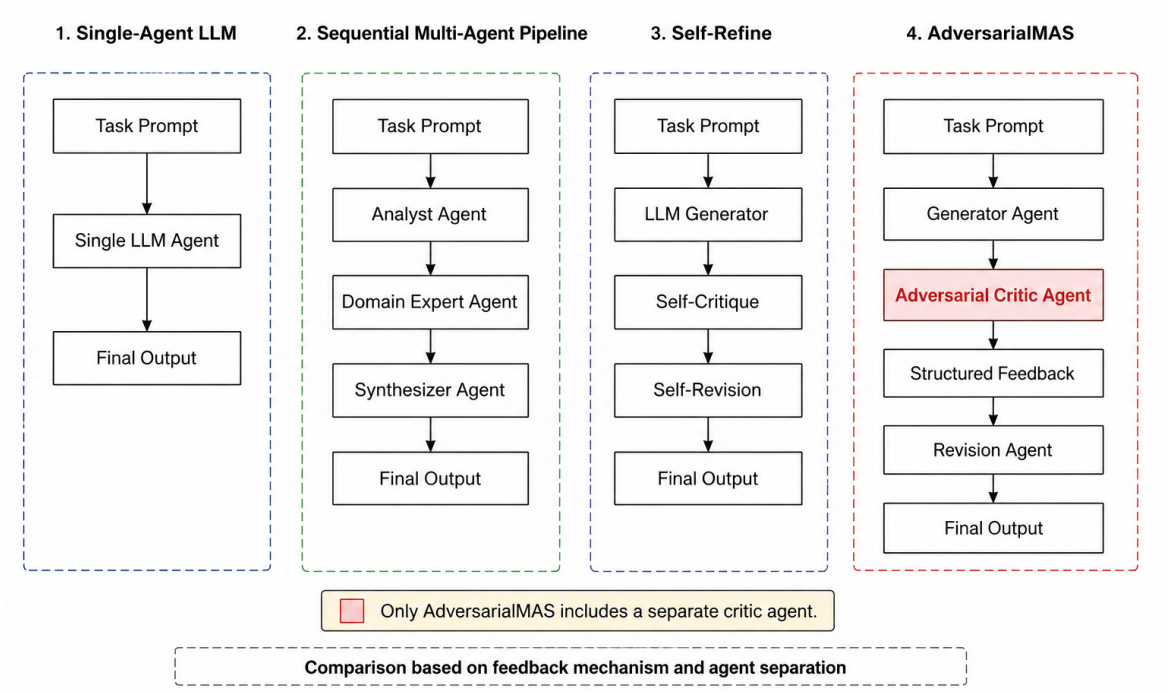

Fig. 2. Architectural comparison of the evaluated approaches.

the output to identify missing information, inconsistencies, and feasible improvements. Both the original response and structured critic feedback are utilized by the Revision Agent, and produce an enhanced version by addressing the identified limitations. This creates a feedback-driven generation process where critique directly influences the final response rather than being used only as an external evaluation step.

The proposed framework workflow is illustrated in Fig. 1. The architecture follows three main stages: response generation, adversarial evaluation, and iterative revision. The process is repeated for multiple cycles, which allows the technique to gradually improve output quality.

B. Agent Roles and Interactions

The Generator Agent creates the first output for a given task in a structured format. Its role is only to generate the output content, as the self-evaluation of the output will be performed by the Adversarial Critic Agent.

The Adversarial Critic Agent takes the role of an independent evaluator. Instead of endorsing the output, the Critic identifies possible shortcomings of the output and composes critique. The output is evaluated along five dimensions of quality: completeness, consistency, feasibility, specificity, and clarity.

The Revision Agent alters the response based on the provided critique. It is given both the original response and the output of the Critic, and as a result, it is able to address the critiqued shortcomings while retaining the output.

C. Iterative Critique and Revision Process

The framework operates through an iterative refinement cycle [21], where the current response is evaluated by the Adversarial Critic Agent and revised according to the generated feedback during each iteration. The updated response becomes the input for the next cycle. Three revision iterations are used to maintain a balance between quality improvement and computational efficiency. The iterative process allows the framework to progressively reduce incomplete information, improve logical consistency, and enhance overall response quality.

D. Compared Architectures

To evaluate the impact of adversarial critique, AdversarialMAS is compared with three alternative generation strategies: Single-Agent LLM, Sequential Multi-Agent Pipeline, and Self-Refine.

The Single-Agent method generates a response using only one model call, without any additional feedback or review. While the Sequential Multi-Agent method divides the task among multiple specialized agents that operate in a fixed workflow. Each agent enhances the output based only on the previous agent's result, without directly identifying errors or providing criticism. This method follows role-based multi-agent collaboration approaches, such as MetaGPT [4].

The Self-Refine method allows a single model to generate a response, evaluate it, and revise its own response through iterative feedback [2]. However, AdversarialMAS uses a separate critic agent to review the generated response independently, in

contrast to Self-Refine. This helps to reduce dependence on the same model's reasoning process.

Figure 2 presents the workflow comparison between the four evaluated approaches. The comparison demonstrates that AdversarialMAS introduces a dedicated critic agent, while other methods either use direct generation, sequential collaboration, or self-evaluation.

IV. Experimental Setup

To evaluate the effectiveness of AdversarialMAS, the evaluation compares four generation strategies across multiple domains with different structural requirements. All methods use the same base model and evaluation conditions to ensure that performance differences are primarily caused by architectural design.

A. Evaluation Domains

Startup strategy development, research proposal generation, and software system design domains are selected to represent different levels of structural complexity. These domains were chosen because they require different types of reasoning, ranging from creative planning to highly structured technical specifications.

Output depends mainly on creativity, reasoning, and practical decision-making in a low-constraint domain, such as a startup strategy. However, research proposal generation represents a medium-constraint domain with expected academic structure, including problem definition, methodology, and expected contribution. Software system design represents a high-constraint domain where responses must satisfy technical requirements such as architecture, scalability, and component interactions. These domains were selected to evaluate LLM capabilities across different reasoning requirements and levels of task complexity [1], [14], [22].

Four approaches are evaluated under identical experimental conditions. The proposed AdversarialMAS framework introduces a separate critic agent between generation and revision, enabling independent evaluation before producing the final output.

B. Experimental Configuration

All experiments were performed using Llama 3.1 8B [17], [18] Instruct as the base language model. The model was accessed through the Groq API and used consistently across all approaches and agent roles. Using the same model configuration ensured that observed differences were related to system architecture rather than model capability differences. For each domain, five prompts were created, resulting in 15 total prompts. Each prompt was evaluated using all four methods, producing 60 experimental runs. AdversarialMAS and Self-Refine were used with three refinement iterations to maintain a consistent comparison.

C. Evaluation Methodology

Automated structural analysis and LLM-based quality assessment to measure different aspects of generated responses are combined in the evaluation. Feature-based analysis and classification approaches have also been explored for identifying important characteristics in textual data and improving automated decision processes [15].

Automated evaluation measures the structural properties of generated outputs. Structural completeness evaluates whether the required information and domain-specific components are present. Thematic coherence measures consistency between different parts of the response. Normalized length evaluates whether generated outputs maintain an appropriate level of detail.

In addition to automated metrics, an LLM-based judge evaluates response quality across multiple dimensions. Each output is assessed based on coherence, completeness, relevance, correctness, and faithfulness to the original task requirements. The judge receives the task prompt and generates a response without information about the producing method to reduce evaluation bias. This evaluation approach follows the LLM-as-a-Judge methodology introduced in previous studies [3], [12].

The final comparison considers both overall quality scores and improvement obtained through refinement. For AdversarialMAS and Self-Refine, the difference between initial and final outputs is analyzed to measure the effect of the critique process.

V. Results and Discussion

This section presents the experimental results of AdversarialMAS compared with Single-Agent LLM, Sequential Multi-Agent, and Self-Refine approaches. The evaluation combines automated structural metrics and LLM-based judgment to analyze response quality from quantitative and qualitative perspectives. The analysis further investigates performance variations across domains and measures the effectiveness of adversarial revision.

A. Automated Evaluation

Automated evaluation measures structural properties of generated responses, including completeness, thematic consistency, and normalized response structure. Figure 3 presents the automated metric comparison across all evaluation domains.

The automated evaluation shows that the Sequential approach achieves the highest average structural score (0.72), followed by AdversarialMAS and Self-Refine with 0.70. The

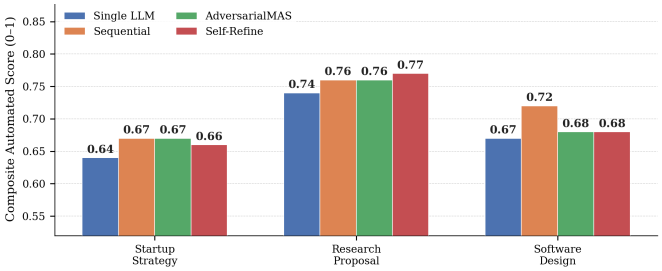

Fig. 3. Automated metric comparison across domains and methods.

TABLE I
LLM JUDGE SCORES BY DOMAIN (1–5)

Domain	Single LLM	Sequential	AdversarialMAS	Self-Refine
Startup	4.00	4.00	4.08	4.00
Research	4.00	4.00	4.20	4.00
Software	4.00	4.00	4.30	4.40
Average	4.00	4.00	4.19	4.13

improved structural performance of Sequential is mainly due to its multi-stage generation process, where multiple agents organize and expand responses before final generation.

However, structural improvement does not always correspond to higher content quality. Sequential optimization mainly improves response organization, coverage, and formatting but does not include an explicit mechanism for identifying incorrect assumptions or weak reasoning. This limitation becomes visible in the LLM Judge evaluation, where AdversarialMAS achieves better overall quality despite a slightly lower automated score.

The Research domain achieves the highest automated performance because academic proposal generation follows predefined structures and expected sections. In contrast, Startup Strategy requires more creative reasoning and domain understanding, making structural metrics less representative of actual usefulness.

B. LLM Judge Evaluation

AdversarialMAS was assessed using LLM-based metrics to quantify response quality with respect to coherence, completeness, relevance, correctness, and faithfulness. The domain-wise evaluation scores are shown in Table I.

As seen in figure 4, of the methods evaluated, AdversarialMAS obtained the best overall judge score at 4.19, whereas the other evaluated approaches (Single-Agent LLM, Sequential Multi-Agent, and Self-Refine) obtained lower scores. This shows that the addition of an independent critic agent adds another layer of quality control that is beyond the benefits seen from collaboration or self-evaluation.

The most notable improvements in score occurred in the Startup and Research domains. In these domains, the identification of gaps in reasoning through the use of adversarial feedback resulted in a greater explanation depth and improved task alignment. The Software domain presents a different case which is Self-Refine (4.40) over AdversarialMAS (4.30). This may show that in highly structured and technical domains, self-evaluation may be beneficial as the agent is able to directly meet specific requirements of the domain through a comparison of generated outputs and the task requirements.

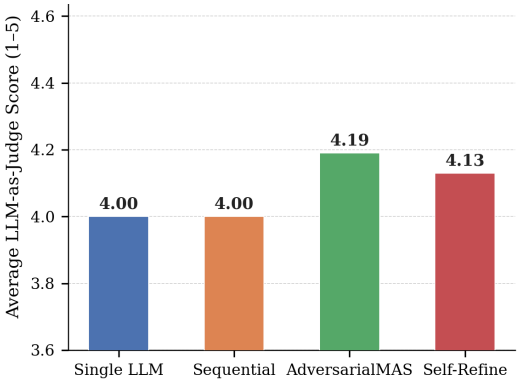

Fig. 4. Average LLM Judge scores across all domains.

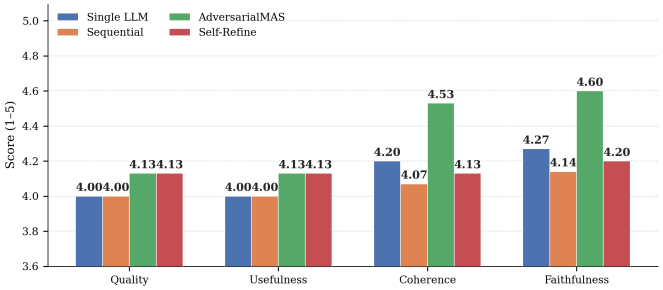

Fig. 5. Dimension-wise comparison across evaluation criteria.

C. Cross-Domain Analysis

Figure 5 presents the dimension-wise comparison across all evaluation criteria averaged over the three domains.

AdversarialMAS achieves the highest performance in Coherence (4.53) and Faithfulness (4.60). These improvements are directly associated with the adversarial critique stage, where the critic agent evaluates logical consistency, missing information, and deviations from task requirements.

Self-Refine also improves output quality through iterative feedback. However, because the same model performs generation and evaluation, the critique process may remain influenced by the original response assumptions. AdversarialMAS reduces this limitation by separating generation and criticism into independent agents.

Sequential Multi-Agent does not achieve similar improvements despite using multiple agents. The results indicate that increasing the number of agents alone does not guarantee higher quality. Effective feedback mechanisms are more important than simple role distribution.

D. Revision Effectiveness

The effectiveness of the refinement process is measured using the difference between initial and final responses. Table II presents the revision delta values for AdversarialMAS.

TABLE II
REVISION DELTA BY DOMAIN (ADVERSARIALMAS)

Domain	Average Delta
Startup	0.14
Research	0.12
Software	0.13

The revision delta values remain consistent across domains, ranging between 0.12 and 0.14. Although the magnitude of changes is moderate, the improvements mainly occur in

important areas such as completeness, reasoning consistency, and requirement alignment rather than unnecessary rewriting.

The findings demonstrate that adversarial critique improves final responses by providing targeted corrections. The critic agent identifies weaknesses, and the revision agent incorporates this feedback to generate higher-quality outputs.

Overall, AdversarialMAS performs best because it treats adversarial evaluation as an active generation component rather than a final inspection stage. The results suggest that multi-agent systems benefit more from structured critique than from additional cooperative agents alone. Self-Refine remains competitive, particularly in highly constrained technical domains, but independent adversarial feedback provides stronger improvements for open-ended reasoning tasks.

VI. Conclusion

This paper presents AdversarialMAS, a multi-agent LLM framework that improves generated responses through adversarial feedback. The framework separates the process into three different roles: generating responses, reviewing them, and making improvements. Experimental results on startup strategy, research proposal, and software design tasks show that AdversarialMAS produces better results than Single-Agent LLM, Sequential Multi-Agent, and Self-Refine methods. The approach achieved the highest average LLM Judge score of 4.19, showing improvements in response quality, completeness, and accuracy.

The evaluation across different domains shows that adversarial critique can improve LLM-generated responses, especially for tasks that require deeper reasoning and better understanding of requirements. While Self-Refine performed well on some structured tasks, the results suggest that an independent critic agent provides stronger feedback than simple multi-agent cooperation. In future work, AdversarialMAS can be evaluated with larger language models, more diverse datasets, and human-based evaluation to further analyze its performance and ability to generalize.

References

[1] Brown, Tom, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan et al. "Language models are few-shot learners." Advances in neural information processing systems 33 (2020): 1877-1901.

[2] Madaan, Aman, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon et al. "Self-refine: Iterative refinement with self-feedback." Advances in neural information processing systems 36 (2023): 46534-46594.

[3] Zheng, Lianmin, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin et al. "Judging llm-as-a-judge with mt-bench and chatbot arena." Advances in neural information processing systems 36 (2023): 46595-46623.

[4] Hong, Sirui, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang et al. "MetaGPT: Meta programming for a multi-agent collaborative framework." In International Conference on Learning Representations, vol. 2024, pp. 23247-23275. 2024.

[5] Wu, Qingyun, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang et al. "Autogen: Enabling next-gen LLM applications via multi-agent conversations." In First conference on language modeling. 2024.

[6] Park, Joon Sung, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. "Generative agents: Interactive simulacra of human behavior." In Proceedings of the 36th annual acm symposium on user interface software and technology, pp. 1-22. 2023.

[7] Li, Guohao, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. "Camel: Communicative agents for" mind" exploration of large language model society." Advances in neural information processing systems 36 (2023): 51991-52008.

[8] Chen, Weize, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu et al. "Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors." In International Conference on Learning Representations, vol. 2024, pp. 20094-20136. 2024.

[9] White, Jules, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C. Schmidt. "A prompt pattern catalog to enhance prompt engineering with chatgpt." arXiv preprint arXiv:2302.11382 (2023).

[10] Shinn, Noah, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. "Reflexion: Language agents with verbal reinforcement learning." Advances in neural information processing systems 36 (2023): 8634-8652.

[11] Pan, Liangming, Michael Saxon, Wenda Xu, Deepak Nathani, Xinyi Wang, and William Yang Wang. "Automatically correcting large language models: Surveying the landscape of diverse automated correction strategies." Transactions of the Association for Computational Linguistics 12 (2024): 484-506.

[12] Liu, Yang, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. "G-eval: NLG evaluation using gpt-4 with better human alignment." In Proceedings of the 2023 conference on empirical methods in natural language processing, pp. 2511-2522. 2023.

[13] Wang, Yidong, Zhuohao Yu, Wenjin Yao, Zhengran Zeng, Linyi Yang, Cunxiang Wang, Hao Chen et al. "Pandalm: An automatic evaluation benchmark for llm instruction tuning optimization." In International Conference on Learning Representations, vol. 2024, pp. 43573-43593. 2024.

[14] Chang, Yupeng, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen et al. "A survey on evaluation of large language models." ACM transactions on intelligent systems and technology 15, no. 3 (2024): 1-45.

[15] P. K. Tiwari, R. Gupta and R. K. Gupta, "Feature Analysis for Fake Review Detection through Supervised Classification," 2019 2nd International Conference on Intelligent Communication and Computational Techniques (ICCT), Jaipur, India, 2019, pp. 275-279.

[16] Gupta, Punit, Ankit Mundra, Navaditya Gaur, Mayank Kumar Goyal, and Rohit Kumar Gupta. "Trust aware workflow scheduling in Scalable Cloud Environment." (2019): 256-68.

[17] Sam, Kira. "Llama 3.1: An in-depth analysis of the next-generation large language model." Available at SSRN 6139407 (2024).

[18] Grattafiori, Aaron, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman et al. "The llama 3 herd of models." arXiv preprint arXiv:2407.21783 (2024).

[19] Agarwal, Abhinav. "Refute-or-Promote: An Adversarial Stage-Gated Multi-Agent Review Methodology for High-Precision LLM-Assisted Defect Discovery." arXiv preprint arXiv:2604.19049 (2026).

[20] Wiesmeier L, Busch M, Tacke M, Linka K, Cyron C and Aydin R (2026) Adversarial robustness of LLM-based multi-agent systems for engineering problems. Front. Artif. Intell. 9:1784484. doi: 10.3389/frai.2026.1784484.

[21] Chen, Pinzhen, Zhicheng Guo, Barry Haddow, and Kenneth Heafield. "Iterative translation refinement with large language models." In Proceedings of the 25th Annual Conference of the European Association for Machine Translation (Volume 1), pp. 181-190. 2024.

[22] Chen, Qian, Cong Xin, Yang Cheng, Chen Weize, Yusheng Su, Juyuan Xu, Liu Zhiyuan, and Sun Maosong. "Communicative agents for software development." arXiv preprint arXiv: 2307.07924 v4 [cs. SE] (2023).