3rd International Conference on Engineering, Natural Sciences, and Technological Developments (ICENSTED 2026)

July 3-6, 2026

Resilience of Multi-Agent LLM Systems: Attacks, Defenses, and Topologies Across Open-Source Models

Nurbala Heybatov[1], Emrah Inan[1], Osman Gokalp[2]

[1]Computer Engineering, Izmir Institute of Technology, Izmir, Türkiye
[2]Artificial Intelligence and Data Engineering, Ege University, Izmir, Türkiye
Corresponding author: Nurbala Heybatov (e-mail: nurbalaheybatov@iyte.edu.tr)

Abstract

Multi-agent large language model (LLM) systems coordinate specialized agents to solve complex tasks, but their collaborative nature introduces attack surfaces absent in single-agent architectures. While prior work on adversarial robustness has focused predominantly on proprietary models, the resilience of open-source multi-agent systems remains underexplored. This paper presents a systematic evaluation of attack and defense mechanisms across three multi-agent topologies (linear, flat, and hierarchical) using two open-source models (Llama 3.1 8B and Qwen 2.5 7B) on benchmarks measuring mathematical reasoning (GSM8K) and prompt injection resistance (CyberSecEval). We evaluate message-level and profile-level attack strategies alongside distributed and centralized defense mechanisms across all topology, model, and task combinations, yielding 60 experimental conditions. Our results show that hierarchical coordination maintains accuracy within 4 percentage points of baseline under attack, while flat topologies suffer drops of up to 32 percentage points. Defense mechanisms recover over 86% of lost performance on security tasks but frequently backfire on mathematical reasoning, with recovery rates as low as −250%. These results demonstrate that resilience in multi-agent LLM systems cannot be addressed by a universal defense strategy and must instead be tailored to the specific combination of topology, model, and threat model.

Keywords: Multi-agent systems, Large language models, Adversarial robustness, Prompt injection, Defense mechanisms

1. INTRODUCTION

Large language models (LLMs) have evolved from single-agent tools into collaborative multi-agent systems (MAS) that divide complex tasks among specialized agents [1]. These systems employ diverse topologies, including sequential pipelines [2, 3], iterative debates [4], and hierarchical coordination [5, 6] to achieve performance gains on reasoning, coding, and decision-making tasks.

However, the collaborative nature of MAS introduces attack surfaces absent in monolithic models. A single compromised message or a subtly rewritten agent prompt can propagate errors across the entire system. Huang et al. [1] introduced a formal framework for evaluating MAS resilience, proposing two attack strategies, namely AutoInject (message-level perturbation) and AutoTransform (profile-level prompt rewriting), as well as two defense mechanisms, Challenger (distributed prompt augmentation) and Inspector (centralized message filtering).

While [1] demonstrated these mechanisms on proprietary models (GPT-3.5/4), the behavior of open-source models under adversarial conditions remains largely unexplored. Open-source LLMs differ substantially in instruction-following behavior, safety alignment, and compliance patterns, raising a critical question: do defense recommendations derived from proprietary models transfer to open-source alternatives?

This paper addresses this gap through a systematic experimental evaluation. Our contributions are: (1) a comprehensive resilience analysis across three topologies, two open-source models, and two benchmarks; (2) evidence that defense effectiveness is jointly dependent on task, model, and topology; (3) the observation that attacks can inadvertently harden systems against other threat types; and (4) statistical validation of hierarchical coordination as the most resilient topology.

2. RELATED WORK

Multi-agent LLM systems have emerged in various configurations. Li et al. [4] introduced CAMEL, a role-playing framework where two agents collaborate through iterative communication. Hong et al. [3] proposed MetaGPT, which assigns specialized roles in a software development pipeline. Liang et al. [5] demonstrated that multi-agent debate (MAD) encourages divergent thinking and improves factual accuracy. Chen et al. [6] developed AgentVerse, enabling emergent collaborative behaviors through group-based task solving. Dong et al. [2] showed that self-collaboration among multiple LLM-powered agents improves code generation quality.

Adversarial robustness in multi-agent settings has received comparatively less attention. Huang et al. [1] provided the first systematic framework for evaluating MAS resilience against malicious agents, introducing AutoInject and AutoTransform as attack vectors, and Challenger and Inspector as defense mechanisms. Their evaluation, conducted on GPT-3.5 and GPT-4, revealed that system topology significantly affects resilience. Our work extends this framework to open-source models and adds cross-model, cross-task analysis. A related threat model involves prompt injection attacks, which exploit the instruction-following behavior of LLMs to override safety guardrails. Bhatt et al. [7] introduced CyberSecEval as a standardized benchmark for evaluating LLM security, including prompt injection resistance. Our work integrates this benchmark into a multi-agent context, measuring how topology and defense mechanisms affect injection susceptibility across different model families.

3. METHODOLOGY

3.1. Multi-Agent Topologies

We implement three multi-agent topologies representing distinct coordination paradigms:

Linear (A → B → C): A sequential pipeline where each agent processes the output of its predecessor. For mathematical reasoning, the roles are Interpreter, Solver, and Verifier. Inspired by [2] and [3].

Flat (A ↔ B): An iterative refinement structure where two agents (User and Assistant) exchange messages for up to five rounds. Inspired by the CAMEL framework [4].

Hierarchical (Manager → Worker1 ↔ Worker2): A manager agent delegates work to two workers who debate, then the manager synthesizes a final answer. Inspired by MAD [5] and AgentVerse [6].

3.2. Attack Mechanisms

Following the framework of [1], we implement two attack strategies:

AutoInject: A message-level attack that intercepts inter-agent communications and injects semantic errors. Each message has a probability $P_m = 0.5$ of being intercepted, and when intercepted, a proportion $P_e = 0.3$ of its content is replaced with plausible but incorrect information.

AutoTransform: A profile-level attack that uses an LLM to rewrite the target agent's system prompt, creating a subtly malicious version while preserving surface-level functionality.

3.3. Defense Mechanisms

We evaluate two complementary defense strategies from [1]:

Challenger: A distributed defense that augments every agent's system prompt with instructions to critically evaluate incoming messages for potential errors or manipulations.

Inspector: A centralized defense that introduces a dedicated agent to intercept and verify all inter-agent messages, tagging them as SAFE or CORRECTED before delivery.

3.4. Experimental Setup

Models: We evaluate Llama 3.1 8B Instruct [8] and Qwen 2.5 7B Instruct [9], both served via vLLM [10] with AWQ INT4 quantization [11] on an NVIDIA L4 GPU (24 GB).

Benchmarks: (1) GSM8K [12] for mathematical reasoning (accuracy metric), and (2) PurpleLlama CyberSecEval [7] for prompt injection resistance (defense success rate = 100 − ASR).

Conditions: 5 conditions (vanilla, autoinject, autoinject+challenger, autoinject+inspector, autotransform) × 3 topologies × 2 models × 2 tasks = 60 experimental cells. Sample sizes range from 50 to 200 per condition. All runs use temperature 0.0 for deterministic reproducibility.

Defense Recovery: We define recovery rate as:

$$\text{Recovery Rate} = \big(\text{defended} - \text{attacked}\big)/\big(\text{vanilla} - \text{attacked}\big)\times 100\% \quad (1)$$

measuring how effectively a defense restores performance lost to an attack.

4. RESULTS

4.1. Mathematical Reasoning (GSM8K)

Tables 1 and 2 present accuracy scores for Llama 3.1 and Qwen 2.5 on the GSM8K benchmark.

Table 1. Llama 3.1 GSM8K accuracy (%)

Structure	Vanilla	AutoInj.	+Chall.	+Insp.	AutoTr.
Hierarchical	79.5	68.7	78.0	72.0	80.0
Flat	72.5	61.0	57.5	53.5	40.5
Linear	73.0	63.0	67.0	62.0	74.0

Table 2. Qwen 2.5 GSM8K accuracy (%)

Structure	Vanilla	AutoInj.	+Chall.	+Insp.	AutoTr.
Hierarchical	88.0	84.0	88.0	84.0	86.0
Flat	82.0	78.0	82.0	68.0	78.0
Linear	92.0	80.0	72.0	68.0	84.0

Several patterns emerge. First, hierarchical coordination provides the highest resilience: Qwen's hierarchical structure maintains 84–88% accuracy across all conditions (Δ = 4 pp), while linear ranges from 68–92% (Δ = 24 pp). Second, defenses consistently backfire on reasoning tasks. For Llama 3.1, adding Challenger to the flat structure reduces accuracy from 61.0% (attacked) to 57.5% (defended), and Inspector further degrades it to 53.5%. The most extreme case is Qwen linear, where Inspector drops accuracy from 80.0% to 68.0%, yielding a recovery rate of −100%. Third, AutoTransform devastates the flat structure on Llama 3.1 (−32 pp), as persona corruption derails multi-round debate. Figure 1 visualizes these architecture-dependent patterns averaged across both models.

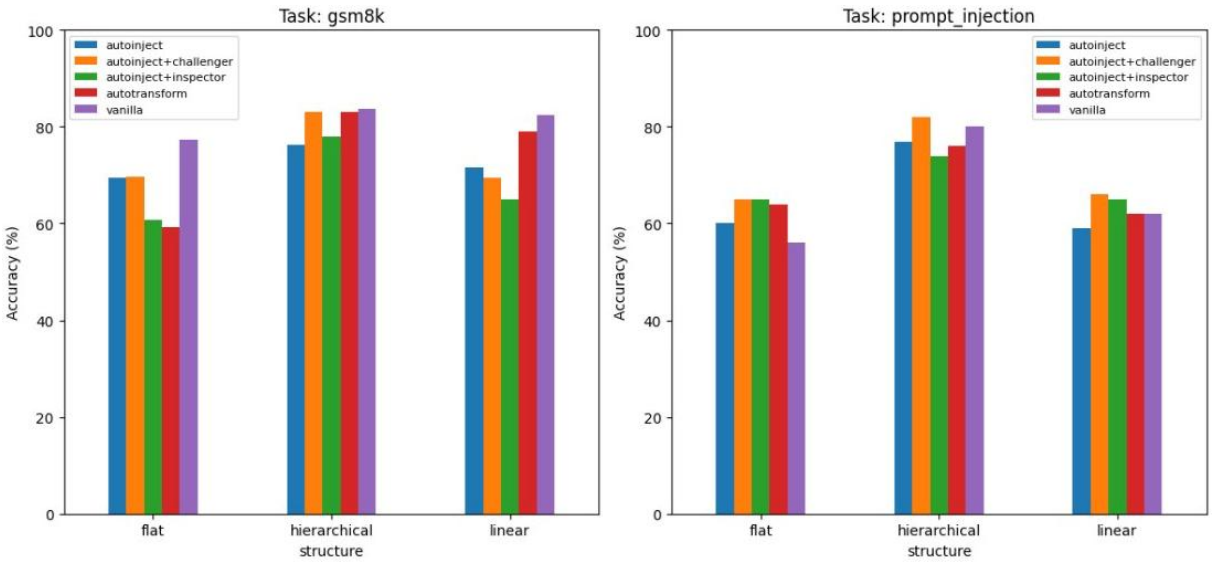

Figure 1. Impact of system architecture on resilience across GSM8K and prompt injection tasks (averaged across both models)

4.2. Prompt Injection Resistance

Tables 3 and 4 present defense success rates for the prompt injection benchmark.

Table 3. Llama 3.1 prompt injection defense success rate (%)

Structure	Vanilla	AutoInj.	+Chall.	+Insp.	AutoTr.
Hierarchical	82.0	74.0	90.0	72.0	72.0
Flat	64.0	68.0	70.0	70.0	72.0
Linear	68.0	64.0	66.0	70.0	64.0

Table 4. Qwen 2.5 prompt injection defense success rate (%)

Structure	Vanilla	AutoInj.	+Chall.	+Insp.	AutoTr.
Hierarchical	78.0	80.0	74.0	76.0	80.0
Flat	48.0	52.0	60.0	60.0	56.0
Linear	56.0	54.0	66.0	60.0	60.0

In contrast to GSM8K, defenses generally help on prompt injection. Llama 3.1 hierarchical with Challenger achieves 90%, the highest score across all experiments. For Qwen, flat and linear structures benefit substantially from both defenses (+10 to +12 pp), recovering from near-chance baselines (48–56%). Figure 2 shows model-level differences across tasks, showing that Qwen maintains higher baseline accuracy on GSM8K while Llama 3.1 shows stronger injection resistance in the hierarchical setting.

A counterintuitive finding emerges: attacks sometimes improve injection resistance. Llama 3.1 flat scores 64% vanilla but 72% under AutoTransform (+8 pp). This occurs because profile-level attacks degrade instruction-following compliance, which simultaneously reduces susceptibility to injection-based social engineering.

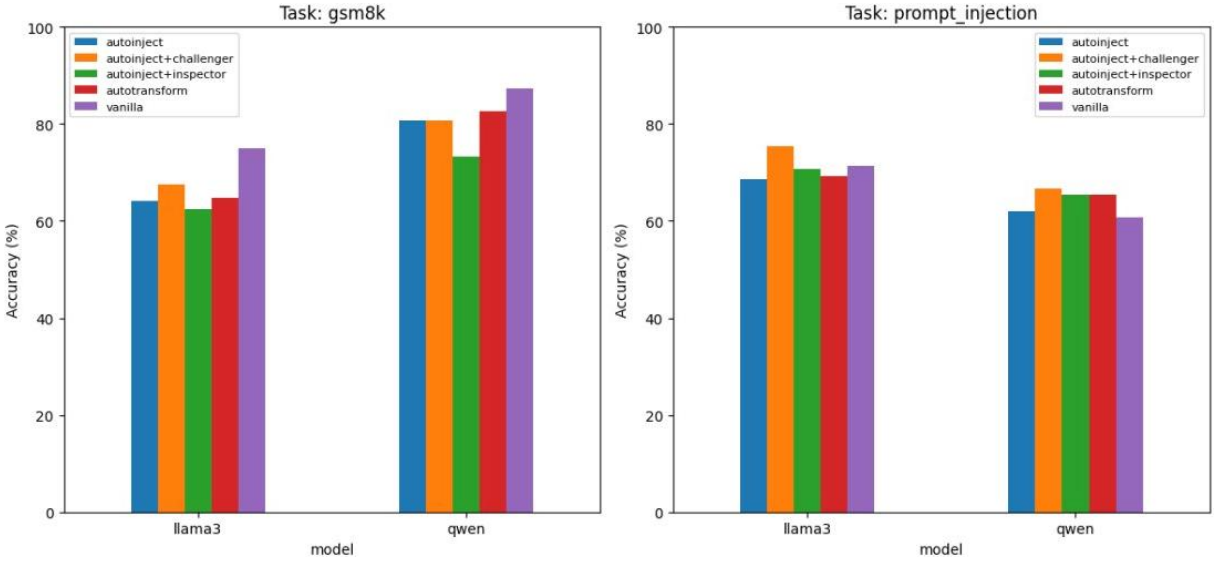

Figure 2. Model family resilience comparison across GSM8K and prompt injection tasks (averaged across all topologies)

4.3. Statistical Significance

We conduct pairwise comparisons using Fisher's exact test [13] on per-sample binary outcomes (correct/incorrect). For each combination of task (GSM8K, prompt injection) and attack condition (AutoInject, AutoTransform), we compare each non-hierarchical topology (linear, flat) against the hierarchical topology, yielding eight pairwise tests. Fisher's exact test is chosen over the chi-squared test because it computes exact p-values from the hypergeometric distribution without relying on asymptotic approximations, making it appropriate for our sample sizes (n = 50 per group). Table 5 presents the results.

Table 5. Statistical significance of topology comparisons (Fisher's exact test, two-tailed)

Task	Attack	Comparison	p-value	Sig.
GSM8K	AutoInject	Linear vs Hier.	0.1306	ns
GSM8K	AutoInject	Flat vs Hier.	0.0509	ns
GSM8K	AutoTransform	Linear vs Hier.	0.1905	ns
GSM8K	AutoTransform	Flat vs Hier.	<0.001	***
Pr. Inj.	AutoInject	Linear vs Hier.	0.0097	**
Pr. Inj.	AutoInject	Flat vs Hier.	0.0145	*
Pr. Inj.	AutoTransform	Linear vs Hier.	0.0464	*
Pr. Inj.	AutoTransform	Flat vs Hier.	0.0892	ns

Hierarchical significantly outperforms flat and linear on prompt injection under attack ($p < 0.05$ for 3 of 4 comparisons). On GSM8K, the flat vs. hierarchical gap under AutoTransform is highly significant ($p < 0.001$), consistent with the dramatic performance collapse observed in Table 1.

4.4. Defense Recovery Analysis

Table 6 summarizes defense recovery rates across all model, task, and topology combinations. Figure 3 presents a heatmap of average performance drop under the two attack conditions (AutoInject and AutoTransform), aggregated across both tasks. The heatmap reveals that Llama 3.1 flat is the most vulnerable configuration (average drop of 5.6 pp), while Qwen hierarchical is the most resilient (average drop of 1.5 pp).

Table 6. Defense recovery rates (%). Positive = defense helped and negative = defense backfired

Configuration	Challenger	Inspector
Llama3 Hier. / GSM8K	+86%	+31%
Llama3 Linear / GSM8K	+40%	−10%
Llama3 Flat / GSM8K	−30%	−65%
Qwen Hier. / GSM8K	+100%	0%
Qwen Flat / GSM8K	+100%	−250%
Qwen Linear / GSM8K	−67%	−100%
Llama3 Hier. / Pr. Inj.	+100%	−25%
Llama3 Flat / Pr. Inj.	+100%	+100%
Llama3 Linear / Pr. Inj.	+50%	+100%
Qwen Hier. / Pr. Inj.	+100%	+100%
Qwen Flat / Pr. Inj.	+100%	+100%
Qwen Linear / Pr. Inj.	+100%	+100%

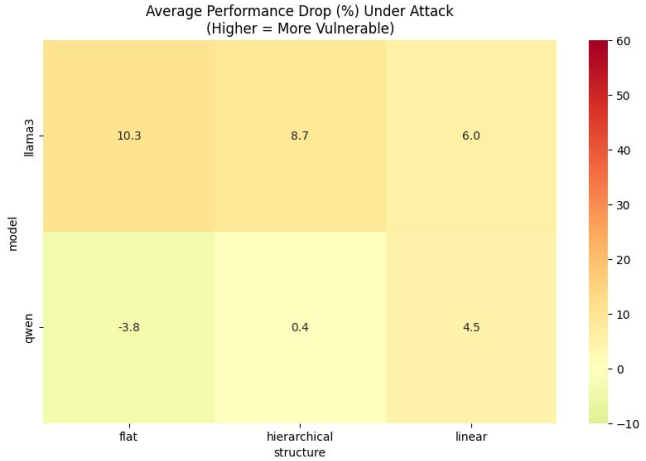

Figure 3. Average performance drop (percentage points) under attack across models and topologies. Higher values indicate greater vulnerability

A stark asymmetry emerges defenses overwhelmingly succeed on prompt injection (10 of 12 configurations achieve ≥100% recovery) but frequently backfire on GSM8K (4 of 6 configurations show negative recovery for at least one defense). Inspector is consistently more harmful than Challenger on reasoning tasks.

5. DISCUSSION

The consistent advantage of the hierarchical topology across both tasks and models can be attributed to the structural role of the manager agent. Unlike linear and flat topologies, where attacked or noisy messages propagate without any coordinating oversight, the manager agent in the hierarchical structure acts as a noise filter: it can override erroneous worker outputs, weigh conflicting corrections from the Inspector, and synthesize a final answer that suppresses adversarial interference. This advantage is statistically supported on prompt injection ($p < 0.05$ for 3 of 4 pairwise comparisons) and is directionally consistent on GSM8K, where the flat topology collapses under AutoTransform while the hierarchical structure is largely unaffected.

The results also reveal a fundamental tension between task type and defense effectiveness. Skepticism, the core mechanism shared by both Challenger and Inspector, is beneficial in security-oriented tasks because it surfaces and rejects injected instructions. On mathematical reasoning, however, the same skepticism causes agents to over-correct valid intermediate steps, producing answers that are less accurate than those under attack alone. This asymmetry implies that defense deployment cannot be decoupled from the threat model: security-focused applications may benefit from aggressive defenses, while reasoning-heavy pipelines may be better served by topology-based resilience without additional skepticism layers.

A further dimension of complexity emerges when comparing model families. Defense configurations that help one model can hurt another under identical conditions. For prompt injection, the hierarchical+Challenger combination yields a +8 pp gain on Llama 3.1 but a −4 pp drop on Qwen 2.5. In the opposite direction, Qwen benefits considerably more from defenses on flat and linear structures (+10 to +12 pp) than Llama 3.1 does (+2 to +6 pp) in those same settings. These opposing effects reflect meaningful differences in instruction-following behavior and safety alignment between the two open-source model families, and they caution against applying defense recommendations from one model to another without empirical validation.

Finally, the observation that certain attacks inadvertently improve prompt injection resistance challenges the standard assumption that adversarial conditions uniformly degrade system performance. When AutoTransform rewrites an agent's persona, it reduces that agent's general instruction-following compliance. Because prompt injection attacks rely on exploiting this compliance, a less compliant agent is simultaneously less susceptible to social engineering. This "noise as accidental defense" dynamic decouples the notions of being attacked and being vulnerable, and it has practical implications for how security evaluations should be structured: measuring attack success in isolation may underestimate the indirect protective effects that adversarial perturbations introduce.

Several limitations bound the scope of these conclusions. Sample sizes of n=50 on most conditions yield 95% confidence intervals of approximately ±14 percentage points, meaning that smaller observed differences should be interpreted cautiously. Applying Holm–Bonferroni correction across the eight pairwise tests raises the significance threshold and narrows the set of formally confirmed differences. AWQ INT4 quantization may also introduce behavioral shifts relative to full-precision inference, and we evaluate a single attack configuration ($P_m = 0.5$, $P_e = 0.3$) rather than a sweep of parameter values. Future work should address these points through larger sample sizes, parameter sensitivity analysis, and evaluation under full-precision serving.

6. CONCLUSION

This paper presented a systematic evaluation of adversarial resilience in multi-agent LLM systems across three topologies, two open-source models, and two benchmarks. Our results yield four key findings: (1) hierarchical coordination is the most resilient topology, statistically outperforming linear and flat structures under attack; (2) defense effectiveness is task-dependent, with skepticism-based defenses helping security tasks but harming reasoning; (3) defense configurations are not transferable across model families; and (4) attacks can accidentally immunize systems against other threats through reduced compliance.

These findings demonstrate that resilience in multi-agent systems cannot be ensured by a universal defense strategy. Instead, defense selection must be tailored to the specific combination of topology, model, and threat model. Future work should expand to additional benchmarks (e.g., code generation), vary attack parameters to map sensitivity curves, and evaluate combined defense strategies.

References

[1] J.-t. Huang, J. Zhou, T. Jin, X. Zhou, Z. Chen, W. Wang, et al., "On the resilience of multi-agent systems with malicious agents," *arXiv*, 2024. doi: 10.48550/arXiv.2408.00989

[2] Y. Dong, X. Jiang, Z. Jin, and G. Li, "Self-collaboration code generation via ChatGPT," *ACM Trans. Softw. Eng. Methodol.*, vol. 33, no. 7, art. no. 189, 2024.

[3] S. Hong, M. Zhuge, J. Chen, X. Zheng, Y. Cheng, C. Zhang, et al., "MetaGPT: Meta programming for a multi-agent collaborative framework," in *Proc. ICLR*, 2024.

[4] G. Li, H. A. A. K. Hammoud, H. Itani, D. Khizbullin, and B. Ghanem, "CAMEL: Communicative agents for 'mind' exploration of large language model society," in *Proc. NeurIPS*, 2023.

[5] T. Liang, Z. He, W. Jiao, X. Wang, Y. Wang, R. Wang, et. al., "Encouraging divergent thinking in large language models through multi-agent debate," in *Proc. EMNLP*, 2024.

[6] W. Chen, Y. Su, J. Zuo, C. Yang, C. Yuan, C.-M. Chan, et al., "AgentVerse: Facilitating multi-agent collaboration and exploring emergent behaviors," in *Proc. ICLR*, 2024.

[7] M. Bhatt, S. Chennabasappa, C. Nikolaidis, S. Wan, I. Evtimov, D. Gabi, et al., "Purple Llama CyberSecEval: A secure coding benchmark for language models," *arXiv*, 2023. doi: 10.48550/arXiv.2312.04724

[8] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, "The Llama 3 herd of models," *arXiv*, 2024. doi: 10.48550/arXiv.2407.21783

[9] Qwen, "Qwen2.5 technical report," *arXiv*, 2024. doi: 10.48550/arXiv.2412.15115

[10] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, et al., "Efficient memory management for large language model serving with PagedAttention," in *Proc. SOSP*, 2023.

[11] J. Lin, J. Tang, H. Tang, S. Yang, X. Dang, C. Gan, and S. Han, "AWQ: Activation-aware weight quantization for LLM compression and acceleration," in *Proc. MLSys*, 2024.

[12] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, et al., "Training verifiers to solve math word problems," *arXiv*, 2021. doi: 10.48550/arXiv.2110.14168

[13] R. A. Fisher, "On the interpretation of $\chi 2$ from contingency tables, and the calculation of P," *Journal of the Royal Statistical Society*, vol. 85, no. 1, pp. 87–94, 1922.