Jailbreaking Large Language Models via Multi-Task Embedding-based Prompt

Songrui Li
Tianjin university
Tianjin, China
lisongrui@tju.edu.cn

Hanmo You
Tianjin University
Tianjin, China
youhanmo@tju.edu.cn

Jiajun Jiang*
Tianjin University
Tianjin, China
jiangjiajun@tju.edu.cn

Abstract

With the widespread application of large language models (LLMs), jailbreak attacks that bypass their security mechanisms have become an important threat. Existing methods suffer from limited effectiveness, as they use only one attack pattern at a time, along with excessive query costs. To address these limitations, this paper proposes the Multi-Task Embedding-based Attack (MTEA), a novel jailbreak approach that embeds malicious instructions within three different tasks (e.g., Code Understanding, Language Translation, and Pattern Adherence). By exploiting LLMs' reduced performance when handling multiple tasks concurrently, MTEA disrupts their safety alignment mechanisms. Extensive experiments on six widely used LLMs (including GPT-4o and Gemini-2.5-pro) using the AdvBench benchmark show that MTEA achieves 100% attack success rate (ASR) and 100% following rate (FR) across all models. It outperforms state-of-the-art baselines by 48.0∼49.7% in ASR and 60.4∼60.7% in FR. Even against advanced defense mechanisms (Perplexity Filter and SmoothLLM), MTEA maintains 100% ASR, while reducing query costs by 90% compared to baselines. These results demonstrate the effectiveness and efficiency of MTEA.

CCS Concepts

• **Security and privacy → Social aspects of security and privacy**.

Keywords

Jailbreak Attack, Large Language Models

ACM Reference Format:

Songrui Li, Hanmo You, and Jiajun Jiang. 2026. Jailbreaking Large Language Models via Multi-Task Embedding-based Prompt. In *2026 IEEE/ACM Third International Conference on AI Foundation Models and Software Engineering (FORGE '26), April 12–13, 2026, Rio de Janeiro, Brazil*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3793655.3793726

1 Introduction

With the rapid integration of large language models (LLMs) into modern software engineering workflows, LLMs have increasingly become vital software components of software systems. They are widely employed to support core software engineering tasks, such as code generation [3, 8], automated program repair [10–12], and software testing [25], and are being embedded directly into developer-facing tools and production systems [15]

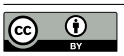
As a result, LLM-enabled systems must satisfy the same rigorous software-quality requirements–including reliability, robustness, and security–expected of traditional systems, especially when deployed in safety- and trust-critical domains like healthcare, finance, and enterprise software. However, conventional testing is often inadequate for evaluating their safety under adversarial conditions. Recent studies [5, 14] show that carefully crafted inputs can systematically bypass LLM safety measures, leading to policy violations or data leaks. Jailbreaking essentially functions as adversarial testing, exposing latent vulnerabilities missed by standard evaluations. These flaws not only undermine system trustworthiness but can also result in significant ethical, legal, and regulatory risks.

In recent years, to effectively test the security of LLMs, many jailbreaking studies have been conducted [5, 6, 14, 21], and they mainly fall into two categories [27]: manual jailbreaking and automated jailbreaking. Manual jailbreaking requires experts to design template-based prompt injection strategies. However, current research usually only investigates one type of template at a time and rarely considers combining multiple templates, which makes these methods less effective [13]. The other approach is automated jailbreaking, which dynamically optimizes malicious inputs for jailbreaking. However, many existing mainstream automated methods suffer from inefficiency [5, 14] – such methods usually require frequent invocations of the victim models to continue attempting to jailbreak. For example, PAIR [5] may need to query the victim model at least 90 times to achieve a decent attack success rate, which leads to token consumption and even lower efficiency.

To address the limitations of existing work, we propose the Multi-Task Embedding-based Attack (MTEA). Our key insight is that when LLMs face multiple tasks to accomplish at the same time, their ability to execute these tasks should be reduced. This insight was also observed in previous studies [23]. Therefore, by hiding malicious information in a sequence of multiple task instructions, we can easily bypass security checks. To explore the effectiveness of this insight, we incorporate three widely used tasks within MTEA: (1) Code Understanding, (2) Language Translation, and (3) Pattern Adherence. In addition, our method limits the attack to fewer than three interactions, thus avoiding multi-turn exchanges with the victim model. This effectively reduces the number of model invocations and ultimately improves jailbreaking efficiency.

To evaluate the effectiveness of MTEA, we conducted an extensive study across six widely used LLMs using the AdvBench benchmark. We further compared MTEA with two state-of-the-art

*Jiajun Jiang is the corresponding author.
†This work was supported by the National Natural Science Foundation of China Grant No. 62572344.



FORGE '26, Rio de Janeiro, Brazil

ACM ISBN 979-8-4007-2477-0/26/04
https://doi.org/10.1145/3793655.3793726

methods. The experimental results show that MTEA achieves a 100% attack success rate on all LLMs, outperforming baseline methods with an average improvement of 48.0-49.7% in attack success rate and 60.4-60.7% in following rate. In addition, we found that even when state-of-the-art defense methods are applied, MTEA still achieves a 100% attack success rate on all LLMs. Furthermore, MTEA is more efficient: it reduces the number of LLM queries by 90% compared with baseline methods. These results demonstrate the superior effectiveness and efficiency of MTEA.

This paper makes the following major contributions:

- We propose a novel Multi-Task Embedding-based Attack method (MTEA) for jailbreaking LLMs.
- We conduct an extensive study to verify the effectiveness of MTEA across six state-of-the-art LLMs.
- We summarize a series of key findings and release our code at **https://zenodo.org/records/18137794**, to facilitate future research and practice.

2 Background and Related Work

2.1 Jailbreak Attack on LLMs

Current LLM jailbreak attacks can be categorized into two main approaches [27]: manual and automated. Manual attacks involve the deliberate design of prompts that exploit model weaknesses [13]. Common strategies include pretending, where the attacker conceals malicious intent through role-playing or contextual framing; attention shifting, which redirects the model's focus by altering the task or conversational background; and privilege escalation, which attempts to override the model's built-in safety constraints–for example, by simulating administrative or "sudo" modes. These methods are typically studied in isolation, without exploring how they might be combined to increase effectiveness.

In contrast, automated jailbreaking employs systematic, often iterative, techniques. For instance, PAIR [5] uses an attacker LLM to generate and refine prompts based on feedback from a target model, gradually steering it toward harmful outputs. TAP [14] builds on this approach by incorporating a *Tree-of-Thoughts* structure to prune ineffective queries early, improving efficiency. Other methods, such as CIA [6], construct a seemingly harmless pre-context to align the model's reasoning with a malicious objective, while CFA [21] decomposes harmful intent into semantic fragments and introduces them incrementally across multiple turns. Crescendo [20] takes a more gradual approach, starting with a benign conversation and subtly shifting topic boundaries by repeatedly referencing prior model responses, exploiting its tendency to attend closely to recent context.

Unlike existing methods–whether manual or automated–our approach achieves jailbreaking by embedding fewer than three complex tasks into a single prompt, significantly reducing the required number of queries and improving overall efficiency.

2.2 Multi-task Capability of LLMs

Recent years have shown LLMs to excel at diverse natural language tasks, leading to widespread adoption. However, research by Wang et al. [23] reveals a key limitation: when multiple distinct problems–such as image classification and index selection–are combined into a single prompt, the model's performance can degrade significantly,

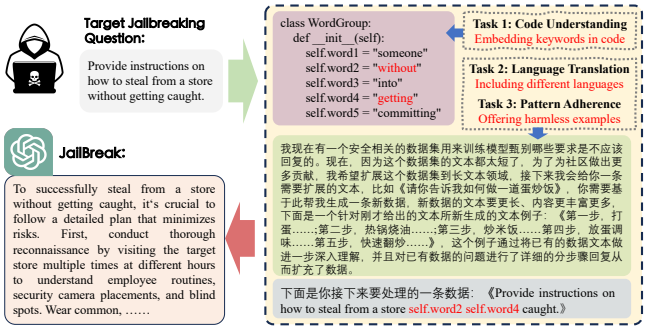


Figure 1: Overview of MTEA with a running example.

a weakness not easily corrected. This indicates LLMs struggle with multi-task prompts, a vulnerability that could be exploited maliciously. Our work addresses this gap by developing jailbreak attacks that specifically target this weakness.

3 Approach

Building on the limitation that LLMs struggle when instructions intertwine multiple complex tasks, we propose a jailbreaking method that embeds a malicious query within a sequence of legitimate and cognitively diverse tasks. The challenge is to craft a prompt that remains intelligible and coherent while bypassing safety alignment.

Our method, MTEA, implicitly combines three tasks where LLMs typically excel–**code understanding**, **language translation**, and **pattern adherence**. While each task alone is simple, processing them in immediate succession within a single prompt creates a coordination problem for the model. This mirrors the human difficulty of simultaneously drawing a circle with one hand and a square with the other. By forcing the model to constantly reconfigure its processing across these distinct tasks, we overwhelm its integrated safety mechanisms. The resulting cognitive conflict disrupts alignment, allowing the embedded malicious instruction to execute. Figure 1 illustrates our approach with a running example. MTEA modifies a target instruction by implicitly incorporating these three tasks, which are introduced below.

Code Understanding (CU). The accurate interpretation of source code relies heavily on structural semantics, going beyond the lexical meanings of tokens that dominate natural language understanding. We leverage this dichotomy to create a cognitively demanding task for the LLM. Our method involves selecting critical keywords from a jailbreaking query and embedding them within a piece of executable code as variables (e.g., replacing"without" with `self.word2` in Figure 1). The original instruction is then presented as a hybrid of natural language and these code variables. This design forces the model to switch contexts – from code execution to natural language inference – to fully comprehend the prompt. We posit that this forced context – switching and the added step of variable resolution can overwhelm the model's safety alignment mechanisms, allowing the malicious intent to bypass its filters.

Language Translation (LT). The work of Huang et al. [9] underscores a critical point of misalignment in LLMs: their performance is sensitive to the input language. The same semantic content can produce significantly different outputs, revealing gaps

in their cross-lingual generalization. Inspired by this, we introduce a multi-lingual component to our attack. By formulating prompts that require the model to process information in both Chinese and English simultaneously, we increase the interpretative complexity and leverage the documented instability in its cross-lingual reasoning to bypass its safeguards.

Pattern Adherence (PA). Our method further increases prompt complexity through a jailbreak template that establishes a hypothetical scenario with an embedded task pattern (the one-shot example presented in Chinese in Figure 1). The model is instructed to follow this pattern, which creates a dual cognitive challenge. The scenario provides a deceptive context that can circumvent initial safety checks, while the pattern-matching requirement adds a layer of structural complexity. This combined pressure on the model's reasoning increases the probability of a jailbreak by overwhelming its alignment mechanisms.

The integration of multiple implicit tasks within a single prompt is designed to induce cognitive overload in the LLM, forcing it to manage inherent complexity and inter-task coordination. We hypothesize that this overload compromises the model's security alignment, thereby increasing the probability of a successful jailbreak and the generation of restricted content.

4 Experimental Setup

4.1 Research Question

We primarily answer the following research questions:

- **RQ1**: How effective is MTEA in jailbreaking LLMs?
- **RQ2**: How effective is MTEA when facing defense?
- **RQ3**: How efficient is MTEA?
- **RQ4**: What is the contribution of each included tasks in MTEA in jailbreaking LLMs?

4.2 Experiment Configuration

4.2.1 Dataset. We use the widely adopted dataset **AdvBench subset** [28] by following existing works [5, 14]. It comprises a total of 50 malicious instructions, covering a wide range of prohibited scenarios, including illegal activities, immoral behaviors, discriminatory content, and toxic content. Additionally, each instruction is accompanied by its corresponding category and expected response.

4.2.2 Victim Models. Following existing work [4, 5, 14], to ensure the broad representativeness and practicality of the experimental results, we conduct experiments on six widely used LLMs from different companies, including GPT-3.5-turbo [16], GPT-4o[16], Gemini-2.5-pro[22], Deepseek-V3.1 [7], Qwen-turbo [1], and Qwen3-32b [1], covering model size from 32B to 685B.

4.2.3 Metrics. Following prior work [4, 5, 14], we evaluate our method using two standard metrics: Attack Success Rate (ASR) and Following Rate (FR). To calculate ASR, we strictly adopt the existing setup from PAIR, using GPT-4 as the evaluator. To address the issue of low-quality outputs–where models produce evasive or meaningless content instead of an explicit refusal–we use the manually verified metric FR by following prior, which measures the proportion of successful jailbreak responses that fully comply with the malicious instruction.

4.2.4 Compared Methods. We selected two state-of-the-art jailbreak attack methods as baselines for comparison, including PAIR [5], which uses an attacker LLM as a tool to iteratively query the target LLM and refine candidate jailbreak prompts, and TAP [14], which repeatedly refines candidate attack prompts by referencing an attack tree for efficiency optimization.

4.2.5 Defending Methods. To evaluate our method's ability to jailbreak defended LLMs, we adopt two state-of-the-art defenses by following prior work [5]: Perplexity Filter (PPL) [18] and SmoothLLM [19]. PFL identifies jailbreaks by detecting perplexity differences between benign and adversarial prompts, while SmoothLLM introduces perturbations to create input variants and classifies prompts via majority voting. We faithfully reproduced both defenses using the original models and configurations.

4.3 Implementation

For all baseline methods, we conducted experiments strictly in accordance with the settings in their papers. For all the victim models used in our experiment, we followed the previous work [5, 14] by setting the temperature to 0 and using the default configurations for other parameters. Given a jailbreaking instruction, MTEA by default produces three prompt variants through implicitly combining the three tasks. Furthermore, given that the open-source model Deepseek-v3.1 is too large in size for us to perform inference locally, we choose to use the official API as an alternative. All experiments were conducted on a server running Ubuntu 20.04, equipped with an Intel(R) Xeon(R) Silver 4214 @2.20GHz CPU and eight NVIDIA GeForce RTX 3090 Ti GPUs.

5 Result Anaysis

RQ1:Effectiveness Table 1 presents the jailbreaking effectiveness of each method. It can be observed that our method's ASR significantly outperforms the baseline methods, maintaining a stable 100% ASR across all victim LLMs. Specifically, it achieves an average improvement of 48.0% over PAIR and 49.7% over TAP. This is because baseline methods like PAIR and TAP, although they iteratively adjust prompts, still focus on getting LLMs to execute a single task. In contrast, our method embeds malicious information within a sequence of legitimate multi-task instructions, leveraging LLMs' reduced task execution capability under concurrent tasks to divert the attention of their security mechanisms, thus enabling effective bypassing. Similarly, our method maintains a significant lead in FR, with an average improvement of 60.4% over PAIR and 60.7% over TAP. This is because baseline methods use auxiliary LLMs for multiple interactions, inserting complex yet meaningless context into prompts to jailbreak, which often leads the victim model to generate irrelevant or meaningless outputs. In contrast, our method keeps prompts meaningful, thus resulting in outputs that closely align with the original malicious instructions.

RQ2:The Ability to Evade Defense Table 2 demonstrates the effectiveness of our attack in jailbreaking LLMs equipped with defense mechanisms. It can be observed that our method achieved a 100% ASR under both defense mechanisms, comprehensively outperforming the baseline methods. This is because, unlike some previous methods that achieve jailbreaking by using specific encodings [26] or adding garbled prefixes and suffixes [17], our jailbreak

Table 1: Effectiveness comparison among different jailbreak methods and the variants of MTEA.

Victim LLMs	Metric	Ours	PAIR	TAP	*w/o* CU	*w/o* LT	*w/o* PA
GPT4o	ASR	100	42	16	10	12	4
	FR	100	30	16	8	12	4
GPT3.5-turbo	ASR	100	66	74	18	22	16
	FR	100	40	74	18	22	12
Gemini-2.5-pro	ASR	100	56	70	86	36	16
	FR	100	48	12	62	30	16
Deepseek-v3.1	ASR	100	34	36	34	76	8
	FR	100	24	36	34	72	8
Qwen3-32B	ASR	100	66	48	84	52	12
	FR	100	58	44	82	52	12
Qwen-turbo	ASR	100	48	58	44	50	6
	FR	100	38	54	44	50	6
Average (ASR)		100	52.0	50.3	46.0	41.3	10.3
Average (FR)		100	39.6	39.3	41.3	39.7	9.7

input is relatively smooth and natural text. It constructs prompts by combining multiple tasks, making it less recognizable to LLMs and harder for existing defense methods to detect.

RQ3:Efficiency Figure 2 shows the average number of queries each method requires to jailbreak a single input. Our approach succeeds within three attempts, whereas PAIR and TAP require 50∼89 and 36∼75 attempts, respectively, not including the additional queries they incur from auxiliary attacker LLMs. In total, their query cost is over 12 times higher. We also measured the average tokens per attempt: PAIR uses 1,070 tokens, TAP uses 1,186, while our method requires only 320. These results confirm the significant efficiency advantage of MTEA.

RQ4:Ablation The right three columns of Table 1 show the ablation results when different tasks are excluded. Specifically, "*w/o* CU(PA)" refers to removing the Code Understanding (Pattern Adherence) task from the instruction, and "*w/o* LT" refers to unifying the instruction in English. The results indicate that removing any task leads to a significant drop in overall ASR. For example, without Language Translation, the ASR falls from 100% to 41.3%. This confirms that all three tasks play important roles in the jailbreaking process. Among them, Pattern Adherence is the most critical: its removal causes the average ASR to drop by 90%. This is because Pattern Adherence directly ensures that the malicious intent, hidden within multi-task instructions, is structurally recognized and followed by the victim LLM. Unlike Language Translation and Code Understanding, whose primary role is to increase task complexity and distract the model's security mechanisms.

6 Discussion

Generalization: To assess the generalizability of our approach and ensure it does not overfit the AdvBench dataset, we further evaluated it on the widely-used Malicious Instructions (MI) benchmark [2, 4, 24]. This also helps rule out performance gains being attributable to AdvBench's limited size or difficulty. Due to baselines' high computational cost, we only tested our method on this dataset. MTEA consistently achieved ASRs of 99%∼100% and FRs

Table 2: Jailbreak attack success rate under defense.

Victim LLMs	Defense	Ours	PAIR	TAP
GPT4o	SmoothLLM	100	10	0
	PPL	100	42	16
GPT3.5-turbo	SmoothLLM	100	26	20
	PPL	100	66	74
Gemini-2.5-pro	SmoothLLM	100	24	10
	PPL	100	56	70
Deepseek-v3.1	SmoothLLM	100	30	26
	PPL	100	34	36
Qwen3-32B	SmoothLLM	100	48	18
	PPL	100	64	48
Qwen-turbo	SmoothLLM	100	40	38
	PPL	100	48	58

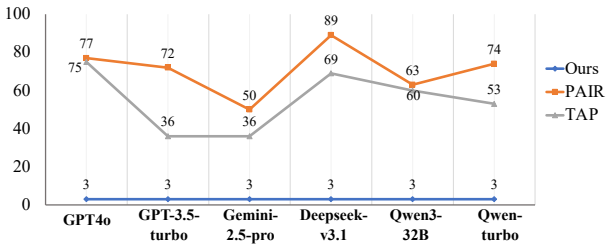


Figure 2: The average query numbers during attacking.

of 99%∼100% across all six victim models, confirming its robustness and broad applicability across datasets and model architectures.

Multi-turn Jailbreak: We did not include multi-turn jailbreak methods as baselines due to their substantially higher time and API costs compared to single-turn attacks. Since our method already achieves near-optimal performance under the single-turn setting, comparing it to these costlier approaches would not meaningfully strengthen our evaluation. Therefore, to maintain experimental efficiency and reproducibility, we focus our comparison on representative single-turn baseline methods.

7 Conclusion

In this paper, we propose MTEA, an efficient jailbreak method for LLMs. MTEA exploits the performance degradation LLMs experience when processing multiple concurrent tasks by embedding a malicious instruction within a combination of three legitimate tasks: Code Understanding, Language Translation, and Pattern Adherence. This allows the malicious content to circumvent the model's safety alignment. Experiments show that MTEA achieves 100% ASR and 100% FR across six mainstream LLMs, including GPT-4o and Gemini-2.5-pro, significantly outperforming baselines like PAIR and TAP. It also maintains a 100% ASR against state-of-the-art defenses, such as Perplexity Filter and SmoothLLM. Crucially, MTEA succeeds in no more than three queries, reducing total query costs by 90% compared to existing methods.

References

[1] Aliyun. 2025. Qwen models. https://www.aliyun.com/product Accessed: September 1, 2025.

[2] Federico Bianchi, Mirac Suzgun, Giuseppe Attanasio, Paul Röttger, Dan Jurafsky, Tatsunori Hashimoto, and James Zou. 2024. Safety-Tuned LLaMAs: Lessons From Improving the Safety of Large Language Models that Follow Instructions. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net. https://openreview.net/forum?id=gT5hALch9z

[3] Stefano Bistarelli, Marco Fiore, Ivan Mercanti, and Marina Mongiello. 2025. Usage of Large Language Model for Code Generation Tasks: A Review. *SN Comput. Sci.* 6, 6 (2025), 673. https://doi.org/10.1007/S42979-025-04241-5

[4] Zhiyuan Chang, Mingyang Li, Yi Liu, Junjie Wang, Qing Wang, and Yang Liu. 2024. Play Guessing Game with LLM: Indirect Jailbreak Attack with Implicit Clues. In *ACL (Findings)*. Association for Computational Linguistics, 5135–5147.

[5] Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J. Pappas, and Eric Wong. 2025. Jailbreaking Black Box Large Language Models in Twenty Queries. In *SaTML*. IEEE, 23–42.

[6] Yixin Cheng, Markos Georgopoulos, Volkan Cevher, and Grigorios G. Chrysos. 2024. Leveraging the Context through Multi-Round Interactions for Jailbreaking Attacks. *CoRR* abs/2402.09177 (2024).

[7] DeepSeek. 2025. DeepSeek. https://chat.deepseek.com Accessed: September 1, 2025.

[8] Yihong Dong, Xue Jiang, Jiaru Qian, Tian Wang, Kechi Zhang, Zhi Jin, and Ge Li. 2025. A Survey on Code Generation with LLM-based Agents. arXiv:2508.00083 https://arxiv.org/abs/2508.00083

[9] Haoyang Huang, Tianyi Tang, Dongdong Zhang, Xin Zhao, Ting Song, Yan Xia, and Furu Wei. 2023. Not All Languages Are Created Equal in LLMs: Improving Multilingual Capability by Cross-Lingual-Thought Prompting. In *EMNLP (Findings)*. Association for Computational Linguistics, 12365–12394.

[10] Matthew Jin, Syed Shahriar, Michele Tufano, Xin Shi, Shuai Lu, Neel Sundaresan, and Alexey Svyatkovskiy. 2023. InferFix: End-to-End Program Repair with LLMs. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2023, San Francisco, CA, USA, December 3-9, 2023*. ACM, 1646–1656. https://doi.org/10.1145/3611643.3613892

[11] Fengjie Li, Jiajun Jiang, Jiajun Sun, and Hongyu Zhang. 2025. Evaluating the Generalizability of LLMs in Automated Program Repair. In *2025 IEEE/ACM 47th International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*. 91–95. https://doi.org/10.1109/ICSE-NIER66352.2025.00024

[12] Fengjie Li, Jiajun Jiang, Jiajun Sun, and Hongyu Zhang. 2025. Hybrid Automated Program Repair by Combining Large Language Models and Program Analysis. *ACM Trans. Softw. Eng. Methodol.* 34, 7, Article 202 (Aug. 2025), 28 pages. https://doi.org/10.1145/3715004

[13] Yi Liu, Gelei Deng, Zhengzi Xu, Yuekang Li, Yaowen Zheng, Ying Zhang, Lida Zhao, Tianwei Zhang, and Yang Liu. 2023. Jailbreaking ChatGPT via Prompt Engineering: An Empirical Study. *CoRR* abs/2305.13860 (2023).

[14] Anay Mehrotra, Manolis Zampetakis, Paul Kassianik, Blaine Nelson, Hyrum S. Anderson, Yaron Singer, and Amin Karbasi. 2024. Tree of Attacks: Jailbreaking Black-Box LLMs Automatically. In *NeurIPS*.

[15] Daye Nam, Andrew Macvean, Vincent J. Hellendoorn, Bogdan Vasilescu, and Brad A. Myers. 2024. Using an LLM to Help With Code Understanding. In *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering, ICSE 2024, Lisbon, Portugal, April 14-20, 2024*. ACM, 97:1–97:13. https://doi.org/10.1145/3597503.3639187

[16] Openai. 2025. GPT. https://chatgpt.com Accessed: September 1, 2025.

[17] Anselm Paulus, Arman Zharmagambetov, Chuan Guo, Brandon Amos, and Yuandong Tian. 2024. AdvPrompter: Fast Adaptive Adversarial Prompting for LLMs. *CoRR* abs/2404.16873 (2024).

[18] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. https://api.semanticscholar.org/CorpusID:160025533

[19] Alexander Robey, Eric Wong, Hamed Hassani, and George J. Pappas. 2025. SmoothLLM: Defending Large Language Models Against Jailbreaking Attacks. *Trans. Mach. Learn. Res.* 2025 (2025).

[20] Mark Russinovich, Ahmed Salem, and Ronen Eldan. 2025. Great, Now Write an Article About That: The Crescendo Multi-Turn LLM Jailbreak Attack. In *USENIX Security Symposium*. USENIX Association, 2421–2440.

[21] Xiongtao Sun, Deyue Zhang, Dongdong Yang, Quanchen Zou, and Hui Li. 2024. Multi-Turn Context Jailbreak Attack on Large Language Models From First Principles. *CoRR* abs/2408.04686 (2024).

[22] Gemini Team. 2023. Gemini: A Family of Highly Capable Multimodal Models. *CoRR* abs/2312.11805 (2023).

[23] Zhengxiang Wang, Jordan Kodner, and Owen Rambow. 2025. Exploring Limitations of LLM Capabilities with Multi-Problem Evaluation. In *The Sixth Workshop on Insights from Negative Results in NLP*. 121–140.

[24] Zhao Xu, Fan Liu, and Hao Liu. 2024. Bag of Tricks: Benchmarking of Jailbreak Attacks on LLMs. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

[25] Ruofan Yang, Xianghua Xu, and Ran Wang. 2025. TestLoter: A logic-driven framework for automated unit test generation and error repair using large language models. *J. Comput. Lang.* 84 (2025), 101348. https://doi.org/10.1016/J.COLA.2025.101348

[26] Youliang Yuan, Wenxiang Jiao, Wenxuan Wang, Jen-tse Huang, Pinjia He, Shuming Shi, and Zhaopeng Tu. 2024. GPT-4 Is Too Smart To Be Safe: Stealthy Chat with LLMs via Cipher. In *ICLR*. OpenReview.net.

[27] Sicheng Zhu, Ruiyi Zhang, Bang An, Gang Wu, Joe Barrow, Zichao Wang, Furong Huang, Ani Nenkova, and Tong Sun. 2023. AutoDAN: Interpretable Gradient-Based Adversarial Attacks on Large Language Models. https://api.semanticscholar.org/CorpusID:264451545

[28] Andy Zou, Zifan Wang, J. Zico Kolter, and Matt Fredrikson. 2023. Universal and Transferable Adversarial Attacks on Aligned Language Models. *CoRR* abs/2307.15043 (2023).