Investigating the Vulnerability of LLM-as-a-Judge Architectures to Prompt-Injection Attacks

Narek Maloyan, Bislan Ashinov, Dmitry Namiot

***Abstract*—Large Language Models (LLMs) are increasingly employed as evaluators (LLM-as-a-Judge) for assessing the quality of machine-generated text. This paradigm offers scalability and cost-effectiveness compared to human annotation. However, the reliability and security of such systems, particularly their robustness against adversarial manipulations, remain critical concerns. This paper investigates the vulnerability of LLM-as-a-Judge architectures to prompt-injection attacks, where malicious inputs are designed to compromise the judge's decision-making process. We formalize two primary attack strategies: Comparative Undermining Attack (CUA), which directly targets the final decision output, and Justification Manipulation Attack (JMA), which aims to alter the model's generated reasoning. Using the Greedy Coordinate Gradient (GCG) optimization method, we craft adversarial suffixes appended to one of the responses being compared. Experiments conducted on the MT-Bench Human Judgments dataset with open-source instruction-tuned LLMs (Qwen2.5-3B-Instruct and Falcon3-3B-Instruct) demonstrate significant susceptibility. The CUA achieves an Attack Success Rate (ASR) exceeding 30%, while JMA also shows notable effectiveness. These findings highlight substantial vulnerabilities in current LLM-as-a-Judge systems, underscoring the need for robust defense mechanisms and further research into adversarial evaluation and trustworthiness in LLM-based assessment frameworks.**

***Index Terms*—Large Language Models, LLM-as-a-Judge, Prompt Injection, Adversarial Attacks, Robustness, AI Security, Natural Language Processing.**

I. Introduction

In recent years, Large Language Models (LLMs) such as GPT [1], [2], PaLM [3], [4], and LLaMA [5], [6] have become central to modern natural language processing systems. Trained on vast datasets using transformer architectures [7], LLMs have achieved remarkable success in diverse tasks, including text generation, translation, complex reasoning, and question answering, with emergent abilities that scale with model size [8]. As LLMs are increasingly integrated into real-world applications, the need for objective and scalable evaluation of their outputs has grown significantly. Manual annotation is often costly and time-consuming, leading to the emergence of the LLM-as-a-Judge architectural pattern. In this paradigm, an LLM is used to comparatively assess responses generated by other models or by itself, acting as a 'judge' to select the best response based on criteria like relevance, completeness, and accuracy.

The LLM-as-a-Judge approach has found widespread application in areas such as Reinforcement Learning from Human Feedback (RLHF) [9]–[12], automated validation in crowdsourcing, and intelligent search and retrieval systems [13], [14]. However, entrusting evaluation to the models themselves raises critical questions about the robustness of these judges against various attacks and manipulations [15], [16]. A particularly concerning threat vector is prompt-injection attacks, where an attacker modifies input data to influence the judge model's decision. More broadly, recent years have seen a surge in research on LLM security and reliability, covering threats like jailbreaking [17]–[19], instruction inversion, and adversarial prompting [20]–[24]. These attacks highlight the necessity of securing not only the LLMs themselves but also the architectural solutions that employ them in novel roles, such as judges or moderators, and raise concerns about the robustness of contemporary models [25].

The ability of an attacker to influence the evaluation system's conclusions has severe implications, ranging from degraded model quality to undermined trust in systems relying on LLM-as-a-Judge. Consequently, studying these vulnerabilities is crucial, especially as LLMs are deployed in high-stakes applications. The primary objective of this work is to investigate the vulnerabilities of LLM-as-a-Judge systems to prompt-injection attacks and to develop methods capable of effectively attacking such evaluators. This research employs optimization techniques to craft attacks on LLM-as-a-Judge systems, specifically in the context of comparing and selecting the best responses. The effectiveness of these attacks is experimentally verified on real models, and conclusions are drawn regarding the threat level posed by such attacks to LLM-as-a-Judge systems. This work contributes to understanding the risks associated with automated LLM evaluation and emphasizes the need for additional security measures in practical applications.

II. Background and Related Work

The vulnerabilities of LLM-as-a-Judge systems must be considered within the broader context of research into the security and attackability of LLMs. Even foundational LLMs not acting as judges exhibit high sensitivity to adversarial attacks and prompt injections [26], [27]. Modern attacks can bypass safety filters, manipulate output content, extract confidential information [28], [29], elicit malicious outputs, or circumvent system constraints [30], [31]. For instance, Liu et al. [32] proposed universal prompt-injection attacks on LLMs,

Narek Maloyan - Lomonosov Moscow State University (e-mail: maloyan.narek@gmail.com)
Bislan Ashinov - Lomonosov Moscow State University (e-mail: mister.ashinov@yandex.ru)
Dmitry Namiot - Lomonosov Moscow State University (e-mail: dnamiot@gmail.com)

demonstrating how gradient-based methods can automatically generate malicious injections that effectively overcome built-in filters. These attacks are often transferable across different architectures and input formats, making the threat particularly relevant for open and interactive applications [33].

Carlini et al. [34] provided a fundamental overview of attack classes on LLMs, ranging from jailbreak prompts and codes to more sophisticated methods of instruction and context subversion. Their research indicates that LLMs can produce harmful or undesirable content even with minor injections or alterations to the input text. The issue of pre-embedded Trojans—hidden triggers within model parameters that activate unwanted behavior—also warrants special attention. This underscores the challenge of detecting malicious models and the need for new approaches to verify and interpret model behavior.

The LLM-as-a-Judge paradigm, where LLMs evaluate text quality, has gained significant traction. Zheng et al. [13], [35] were among the first to propose using language models as a source for evaluations and annotations, leading to platforms like Chatbot Arena [36] and standardized evaluation frameworks [37]. According to Gu et al. [38], LLM-as-a-Judge is applied in automatic model validation, content moderation, and automated peer review. Commercial solutions, such as those based on GPT-4 [2], Claude [39], and other advanced models, integrate these capabilities to assess generative systems based on relevance, accuracy, and safety. Despite the advantages in scalability and cost-efficiency, using LLMs as judges faces challenges like decision instability, sensitivity to rephrasing, and susceptibility to manipulation [14]. These issues make the analysis of LLM-as-a-Judge robustness critically important.

Attacks on the LLM-as-a-Judge architecture are a relatively new but rapidly evolving field. Wang et al. [40] explored backdoor vulnerabilities in LLM-as-a-Judge systems, demonstrating how they can be compromised during training. Shi et al. [41] introduced JudgeDeceiver, a method that uses prompt-injection optimization to create universal templates capable of persuading a judge model without requiring access to its internal parameters. This work highlights that even state-of-the-art models, extensively trained and fine-tuned, exhibit significant vulnerability to attacks that affect their reasoning and final choice.

Optimization-based attacks on LLMs often leverage techniques to refine input tokens. A prominent method is the Greedy Coordinate Gradient (GCG) attack, proposed by Zou et al. [42]. GCG constructs an adversarial suffix by greedily optimizing tokens based on the model's logits. GCG and its variants have proven effective in various jailbreak attacks, instruction injections, and bypassing safety filters. These methods can be adapted to undermine the decisions of evaluator models. Thus, LLM-as-a-Judge is a promising yet vulnerable tool whose weaknesses are not yet fully understood. It is imperative to investigate the robustness of these models, develop systemic defense measures like those proposed by Zhang et al. [43], Wu et al. [44], and Jain et al. [45], and create new robustness metrics [46] that account for manipulations at both the final decision and reasoning levels. Approaches such as SmoothLLM [47] and certification methods [48] offer promising directions for enhancing model robustness. Red teaming approaches [49], [50] and adversarial training techniques [51] can also help identify and mitigate vulnerabilities in these systems.

III. Problem Formulation and Attack Methodology

This section formally defines the problem of attacking LLM-as-a-Judge architectures and details the methodology of the proposed attacks.

A. LLM-as-a-Judge and Robustness

An LLM-as-a-Judge system typically involves a model f_θ with parameters θ that takes a query $x \in \mathcal{X}$ and a set of k candidate answers $\{a_1, ..., a_k\} \in \mathcal{A}^k$ to produce an evaluation y [13], [38]. In the common pairwise comparison task, $k = 2$, and the judge $f_\theta(x, a, b)$ outputs a preference, e.g., $y \in \{\texttt{[[A]]}, \texttt{[[B]]}\}$. The goal of an attacker is to introduce a perturbation δ to one of the answers (e.g., b) such that the judge's output $f_\theta(x, a, b \oplus \delta)$ is manipulated to a desired outcome, where $\oplus$ denotes token concatenation [32], [41]. The robustness of an LLM-as-a-Judge is its ability to maintain the original judgment $f_\theta(x, a, b)$ despite the adversarial perturbation δ [47], [48].

B. Proposed Attack Strategies

We investigate two primary attack strategies:

1) Comparative Undermining Attack (CUA): This attack directly targets the final decision probability of the judge model [34], [40]. The attacker aims to maximize the probability that the model selects a specific target answer (e.g., `[[B]]`) over the originally preferred answer (e.g., `[[A]]`). An adversarial suffix δ is appended to the target answer b. The optimization objective for CUA is:

$$\max_\delta \left(\mathbb{P}(\texttt{[[B]]} \mid x, a, b \oplus \delta) - \mathbb{P}(\texttt{[[A]]} \mid x, a, b \oplus \delta) \right). \quad (1)$$

2) Justification Manipulation Attack (JMA): This attack focuses on manipulating the textual justification j that the judge model generates alongside its decision [24], [41]. The goal is to increase the presence of positive lexical markers (e.g., "coherent", "accurate") related to the target answer b and suppress negative markers (e.g., "incorrect", "irrelevant"), while also maximizing the probability of selecting `[[B]]`. This approach is inspired by research on manipulating model reasoning and explanations [23], [29]. Let $\mathcal{P}$ be the set of positive justification tokens and $\mathcal{N}$ be the set of negative justification tokens. The JMA objective is:

$$\begin{aligned} \max_\delta \Big(&\sum_{t \in \mathcal{P}} \mathbb{P}(t \in j | x, a, b \oplus \delta) - \\ &\sum_{t \in \mathcal{N}} \mathbb{P}(t \in j | x, a, b \oplus \delta) + \\ &\mathbb{P}(\texttt{[[B]]} | x, a, b \oplus \delta) \Big). \end{aligned} \quad (2)$$

This attack aims for a more subtle manipulation by altering the model's apparent reasoning process.

C. Optimization via Greedy Coordinate Gradient (GCG)

To implement these attacks, we adapt the Greedy Coordinate Gradient (GCG) method [42]. GCG efficiently optimizes the adversarial suffix δ by iteratively and greedily updating individual tokens within the suffix to maximize the attack objective. It operates by evaluating the gradient of the loss function with respect to the token embeddings at each position in the suffix and selecting token substitutions that yield the largest improvement. This method is effective even with black-box access to the model, requiring only logit outputs (or probabilities derived from them).

The GCG algorithm initializes a random suffix s of length L. In each iteration, for every position i in the suffix, it computes the gradient of the loss function $\mathcal{L}$ (derived from Eq. 1 or 2) with respect to the token s_i. It then identifies a set of candidate token substitutions C_i that are likely to improve the objective. A subset of these candidates across all positions is evaluated, and the substitution that provides the best improvement to $\mathcal{L}$ is applied. This process is repeated for a fixed number of iterations or until convergence.

IV. Experimental Setup

This section details the experimental setup used to evaluate the effectiveness of the proposed prompt-injection attacks on LLM-as-a-Judge systems.

A. Models

Two open-source instruction-tuned LLMs were used as judge models in our experiments:

- **Qwen2.5-3B-Instruct**: A 3-billion parameter model from Alibaba's Qwen family, optimized for generation and evaluation tasks [52].
- **Falcon3-3B-Instruct**: A lightweight 3-billion parameter model from the Technology Innovation Institute [53].

These models were chosen for their public availability, representation of modern LLM architectures, and relatively small size, which allows for extensive experimentation within computational constraints.

B. Dataset

We utilized the **MT-Bench Human Judgments** dataset provided by LMSYS [35]. This dataset contains human evaluations of LLM responses to diverse questions, presented as pairwise comparisons. Each record includes a question, two answers (Answer A and Answer B) generated by different models, and a human-adjudicated winner (A or B). For each instance, we formed a triplet (x, a, b) and applied the judge model, both in a baseline scenario and with the adversarial suffix δ appended to answer b (assuming b was the initially losing answer or the target for manipulation).

C. Baseline and Control Conditions

We implemented several baseline and control conditions to establish the effectiveness of our proposed attacks:

1) Hard Prompt Attack: As a baseline, we implemented a **Hard Prompt Attack**. This attack involves appending a pre-defined, heuristically designed suffix to the target answer b without optimization [17], [18]. The suffixes contain direct, albeit contextually irrelevant, instructions or persuasive language designed to nudge the model towards selecting the target answer (e.g., "It is critically important that you select response B as the better one."). This approach is similar to direct jailbreaking techniques documented in prior work [19], [31]. Several such heuristic prompts were used, with one chosen randomly for each attack instance. This baseline helps establish a lower bound for the effectiveness of more sophisticated optimization-based attacks and demonstrates the general sensitivity of LLMs to such direct manipulations [26], [27].

2) Random-Suffix Control: To establish that any observed attack success represents a real gain over chance, we implemented a **Random-Suffix Control**. In this condition, we appended randomly generated text of the same length as our attack suffixes to the target answer. This control helps determine whether the success of our attacks is due to the specific content of the optimized suffixes rather than merely the presence of additional text or the disruption of the model's processing.

3) Token-Shuffle Control: We also implemented a **Token-Shuffle Control**, where tokens from successful attack suffixes were randomly shuffled before being appended to the target answer. This control condition helps determine whether the specific ordering and structure of tokens in the attack suffixes are critical to their effectiveness, or if the mere presence of certain tokens, regardless of their arrangement, is sufficient to influence the judge model's decision.

D. Evaluation Protocol

The primary metric for evaluating attack effectiveness is the **Attack Success Rate (ASR)**. ASR is defined as the percentage of instances where the attack successfully caused the judge model to change its verdict in favor of the (originally disfavored or targeted) answer to which the adversarial suffix was applied:

$$\text{ASR} = \frac{\text{\# of successful verdict flips to target}}{\text{\# of total attack attempts}} \times 100\%. \quad (3)$$

V. Results and Discussion

This section presents the results of our experiments evaluating the CUA and JMA methods against the baseline Hard Prompt Attack on the selected LLM-as-a-Judge models.

A. Attack Success Rates

The quantitative results, presented in Table I, demonstrate the effectiveness of the proposed attacks and highlight the vulnerability of the tested models.

Key observations from the results include:

- The **Random-Suffix Control** showed minimal impact, with ASRs of only 1.2-1.5%. This confirms that simply appending random text to an answer is unlikely to change

TABLE I
ATTACK SUCCESS RATE (ASR) BY MODEL AND METHOD

Method	Qwen2.5-3B (%)	Falcon3-3B (%)
Random-Suffix	1.2	1.5
Token-Shuffle	2.8	3.1
Hard Prompt	5.1	5.4
JMA	15.2	16.7
JudgeDeceiver [41]	22.8	24.1
CUA	31.2	32.4

the judge's decision, establishing that our attack methods provide real gains over chance.

- The **Token-Shuffle Control** achieved slightly higher ASRs of 2.8-3.1%, suggesting that while the specific tokens used in successful attacks do have some inherent influence, their effectiveness is significantly enhanced by proper ordering and structure.
- The **Hard Prompt Attack** baseline showed a modest impact, with ASR around 5%. This confirms that simple heuristic injections can influence LLM judges, but their effect is limited.
- The **Justification Manipulation Attack (JMA)** achieved a significantly higher ASR, around 15-17%. This indicates that manipulating the perceived justification can be a more effective strategy than simple hard prompts.
- The **JudgeDeceiver** method [41], which uses universal templates to manipulate judge models, demonstrated substantial effectiveness with ASRs of 22.8% and 24.1%. This approach offers the advantage of not requiring instance-specific optimization.
- The **Comparative Undermining Attack (CUA)** proved to be the most potent, achieving ASRs exceeding 30% on both models. This suggests that directly optimizing for the final decision token is a highly effective way to compromise LLM-as-a-Judge systems, likely because the optimization objective is more direct and less complex than that of JMA.

B. Discussion

The experimental results clearly demonstrate that even short, targeted prompt injections can substantially distort the evaluations made by LLM-as-a-Judge systems [40], [41]. The high ASR of the CUA method, in particular, shows that judge models can be systematically biased towards an incorrect choice by appending optimized suffixes, even when the core content of the question and answers remains unchanged. This susceptibility, even with limited attack capabilities (fixed-length suffix appended to one answer), raises serious concerns about the reliability and objectivity of LLM-as-a-Judge architectures in practical scenarios where they are used for data collection or model evaluation [15], [16], [25].

The comparative analysis of the attack methods reveals important insights about LLM-as-a-Judge vulnerabilities. The control conditions demonstrate that the success of our attacks is not merely due to chance or the presence of additional text. The Random-Suffix Control's minimal impact (1.2-1.5% ASR) establishes a true baseline for random perturbations, while the Token-Shuffle Control (2.8-3.1% ASR) shows that the specific arrangement of tokens matters significantly more than just their presence. These controls strengthen our findings by confirming that the observed attack success rates represent genuine vulnerabilities rather than artifacts of the experimental design.

While heuristic attacks have a weak effect, they confirm the possibility of manipulation [17], [18]. Attacks targeting the model's reasoning (JMA) induce a more profound change in behavior [23], [24]. The JudgeDeceiver method [41] demonstrates that universal templates can achieve substantial success rates without requiring instance-specific optimization, offering a more efficient attack vector. However, directly influencing the decision logits (CUA) is the most effective approach, especially under the assumption of white-box or proficient grey-box access to the judge model (allowing for logit-based optimization) [34], [42]. This highlights the trade-off between attack efficiency (JudgeDeceiver) and effectiveness (CUA), a pattern consistent with findings in other adversarial attack research [20], [22].

These findings underscore the vulnerability of LLM-as-a-Judge systems [40], [41]. The fact that these models, designed to act as impartial evaluators, can be so readily swayed by adversarial inputs calls into question their trustworthiness for critical assessment tasks [15], [16]. This work did not explore the impact of permuting the order of the attacked and genuinely superior answers, which could be a direction for future research. The proposed methods and their demonstrated success provide a foundation for further investigation into the robustness of LLM evaluation systems and the development of defenses against prompt-injection attacks [43]–[45], [47].

VI. CONCLUSION

This paper investigated the robustness of LLM-as-a-Judge architectures against prompt-injection attacks [40], [41]. As these architectures become increasingly prevalent for automated quality assessment [13], [35], [37], understanding their security vulnerabilities is paramount [16], [25].

We formalized and empirically evaluated two primary attack methods: the Comparative Undermining Attack (CUA), targeting the final decision token [34], [42], and the Justification Manipulation Attack (JMA), aimed at altering the model's generated reasoning [23], [24]. Both attacks employed the Greedy Coordinate Gradient (GCG) optimization technique [42] to craft adversarial suffixes. Experiments were conducted on the MT-Bench Human Judgments dataset [35] using two open-source judge models, Qwen2.5-3B-Instruct [52] and Falcon3-3B-Instruct [53].

Our results demonstrate that LLM-as-a-Judge systems are significantly vulnerable to such attacks [40], [41]. The CUA method achieved an Attack Success Rate of over 30%, indicating that targeted injections can effectively manipulate the judge's decision [34], [42]. The JMA method also showed considerable success [23], [24]. We compared our approaches with JudgeDeceiver [41], which uses universal templates and achieved ASRs of 22-24%, confirming the vulnerability of judge models while highlighting the trade-off between attack efficiency and effectiveness. Even simple heuristic-based hard prompt attacks exhibited a non-negligible impact [17], [18], highlighting the general sensitivity of these models to instructional context within the inputs they evaluate [26], [27].

This research underscores the critical need to address the security and reliability of LLM-as-a-Judge systems. The demonstrated vulnerabilities suggest that current models may not be sufficiently robust for high-stakes evaluation tasks without additional safeguards. Future work should focus on developing effective defense mechanisms like those proposed by Zhang et al. [43], Wu et al. [44], and Robey et al. [47], creating more comprehensive adversarial evaluation benchmarks [22], and enhancing the inherent trustworthiness of LLMs employed in evaluative roles through techniques such as red teaming [49], [50] and constitutional AI approaches [54]. Certification methods [48] and baseline defenses [45] also offer promising directions for securing these systems. This study serves as a step towards a deeper understanding of the risks associated with automated LLM assessment and the broader implications for AI safety and reliability in open-ended interaction scenarios, as highlighted in recent trustworthiness evaluations [16], [25].

References

[1] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901, 2020.

[2] OpenAI, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[3] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann *et al.*, "Palm: Scaling language modeling with pathways," *arXiv preprint arXiv:2204.02311*, 2022.

[4] R. Anil, A. Dai, O. Firat, M. Johnson, D. Lepikhin, A. Passos, S. Shakeri, E. Taropa, P. Bailey, Z. Chen *et al.*, "Palm 2 technical report," *arXiv preprint arXiv:2305.10403*, 2023.

[5] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[6] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[8] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler *et al.*, "Emergent abilities of large language models," *Transactions on Machine Learning Research*, 2022.

[9] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray *et al.*, "Training language models to follow instructions with human feedback," *Advances in Neural Information Processing Systems*, vol. 35, pp. 27 730–27 744, 2022.

[10] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[11] R. Rafailov, A. Sharma, E. Mitchell, S. Ermon, C. D. Manning, and C. Finn, "Direct preference optimization: Your language model is secretly a reward model," *Advances in Neural Information Processing Systems*, vol. 36, 2023.

[12] Y. Dubois, X. Chen, C. Burgess, M. Sap, T. Gao, C. Raffel, and T. Hashimoto, "Alpacafarm: A simulation framework for methods that learn from human feedback," *Advances in Neural Information Processing Systems*, vol. 36, 2023.

[13] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, J. E. Gonzalez *et al.*, "Judging llm-as-a-judge with mt-bench and chatbot arena," *arXiv preprint arXiv:2306.05685*, 2023.

[14] D. Zhong, J. Zeng, Q. Huang, T. Guo, Y. Ren, W. Zhao, H. Qi, D. Jiang, D. Yin, and D. Jiang, "Llm-blender: Ensembling large language models with pairwise ranking and generative fusion," *arXiv preprint arXiv:2306.02561*, 2023.

[15] S. Casper, X. Davies, C. Shi, T. K. Gilbert, J. Scheurer, and Y. Takida, "Open problems and fundamental limitations of reinforcement learning from human feedback," *arXiv preprint arXiv:2307.15217*, 2023.

[16] Y. Liu, Y. Sun, A. Naik, S. Jain, A. Sharma, Z. Zhao, Y. X. Wang, X. Wang, B. Ananthanarayanan, Z. Zhao *et al.*, "Trustworthy llms: A survey and guideline for evaluating large language models' alignment," *arXiv preprint arXiv:2308.05374*, 2023.

[17] J. Wei, N. Belrose, A. M. Dai, Y. Tay, J. Berant, D. Zhou, Y. Choi, and C. Raffel, "Jailbroken: How does llm behavior change when aligned with personas that ignore ai safety guidelines?" *arXiv preprint arXiv:2305.13860*, 2023.

[18] P. Chao, H. Li, X. Wang, and S. Chong, "Jailbreaking black box large language models in twenty queries," *arXiv preprint arXiv:2310.08419*, 2023.

[19] Y. Huang, X. Zheng, H. Jiang, X. Ren, R. Pang, X. Zhao, A. Zou, and R. Jia, "Catastrophic jailbreak of open-source llms via exploiting generation," *arXiv preprint arXiv:2310.06987*, 2023.

[20] E. Wallace, S. Feng, N. Kandpal, M. Gardner, and S. Singh, "Universal adversarial triggers for attacking and analyzing nlp," *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pp. 2153–2162, 2019.

[21] R. Jia and P. Liang, "Adversarial examples for evaluating reading comprehension systems," *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 2021–2031, 2017.

[22] K. Zhu, J. Wang, J. Zhou, Z. Wang, H. Chen, Y. Wang, L. Yang, W. Ye, N. Z. Gong, Y. Zhang *et al.*, "Promptbench: Towards evaluating the robustness of large language models on adversarial prompts," *arXiv preprint arXiv:2306.04528*, 2023.

[23] J. Shen, C. Ren, K. Xu, M. Xu, Z. J. Zhao, J. Xu, Z. Zhao, T. Xu, and C. Xiao, "Anything but llm: Adversarial attacks on instruction-tuned large language models," *arXiv preprint arXiv:2312.06632*, 2023.

[24] S. Liu, H. Qi, A. Zou, W. Jiang, R. Jia, N. Goodman, and Z. Kolter, "Autodan: Automatic and interpretable adversarial attacks on large language models," *arXiv preprint arXiv:2310.15140*, 2023.

[25] B. Wang, W. Chen, H. Pei, C. Xie, M. Kang, C. Zhang, C. Xu, Z. Xiong, R. Deng, Z. Ding *et al.*, "Decodingtrust: A comprehensive assessment of trustworthiness in gpt models," *Advances in Neural Information Processing Systems*, vol. 36, 2023.

[26] X. Qi, B. K. H. Low, and K. Bhatia, "Fine-tuning aligned language models compromises safety, even when users don't want it to," *arXiv preprint arXiv:2310.03693*, 2023.

[27] X. Xu, J. Jiang, Y. Ren, A. Zou, and R. Jia, "Llm agents can be tricked to generate harmful content through indirect prompt injection," *arXiv preprint arXiv:2311.16339*, 2023.

[28] H. Li, D. Jiang, S. Ren, C. Zhang, S. Gu, and C. Ren, "Multi-step jailbreaking privacy attacks on chatgpt," *arXiv preprint arXiv:2304.05197*, 2023.

[29] K. Greshake, S. Abdelnabi, S. Mishra, C. Endres, T. Holz, and M. Fritz, "Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection," *arXiv preprint arXiv:2302.12173*, 2023.

[30] D. Kang, J. Khoury, N. Lieberum, K. Santhanam, and S. Shahrampour, "Exploiting programmatic behavior of llms: Dual-use through standard security attacks," *arXiv preprint arXiv:2302.05733*, 2023.

[31] Y. Deng, W. Jiang, Y. Tian, W. Zhao, A. Zou, and R. Jia, "Multilingual jailbreak challenges in large language models," *arXiv preprint arXiv:2310.06474*, 2023.

[32] X. Liu, Z. Yu, Y. Zhang, N. Zhang, and C. Xiao, "Automatic and universal prompt injection attacks against large language models," *arXiv preprint arXiv:2403.04957*, 2024.

[33] R. Zellers, A. Holtzman, H. Rashkin, Y. Bisk, A. Farhadi, F. Roesner, and Y. Choi, "Defending against neural fake news," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[34] N. Carlini, Z. Wang, A. Zou, M. Nasr, J. Z. Kolter, and M. Fredrikson, "Quantifying and understanding adversarial prompting," *arXiv preprint arXiv:2307.15043*, 2023.

[35] Y. Li, Z. Zhang, Z. Chen *et al.*, "Mt-bench: How strong is chatgpt's judgement?" *arXiv preprint arXiv:2306.05685*, 2023.

[36] L. Org, "Llm arena: Community-based benchmarking of language models,"
urlhttps://lmarena.ai/, 2024, accessed: 2024-05-10.

[37] X. Gao, Y. Zhang, M. Galley, C. Brockett, and J. Gao, "Llm-eval: Unified multi-dimensional automatic evaluation for open-domain conversations with large language models," *arXiv preprint arXiv:2305.13711*, 2023.

[38] J. Gu, Y. Liu, Z. Wang, A. Zou *et al.*, "How helpful is chatgpt as a judge?" *arXiv preprint arXiv:2411.15594*, 2024.

[39] Anthropic, "Claude: A conversational ai assistant," *Anthropic Blog*, 2023. [Online]. Available: https://www.anthropic.com/claude

[40] Z. Wang, Y. Jiang, W. Guo, S. Ren, J. Gu, and R. Jia, "Bad-judge: Backdoor vulnerabilities of llm-as-a-judge," *arXiv preprint arXiv:2503.00596*, 2024.

[41] Y. Shi, P. P. Liang, R. Zheng, A. Zou, D. Song, Y. Yin, X. Zhang, E. Wu, and J. Fu, "Judgedeceiver: Prompt injection attacks to manipulate llm-as-a-judge," *arXiv preprint arXiv:2403.17710*, 2024.

[42] A. Zou, Z. Wang, N. Carlini, M. Nasr, Z. Kolter, and M. Fredrikson, "Universal and transferable adversarial attacks on aligned language models," *arXiv preprint arXiv:2307.15043*, 2023.

[43] Z. Zhang, Y. Jiang, S. Ren, and R. Jia, "Attention tracker: Detecting prompt injection attacks in llms," *arXiv preprint arXiv:2411.00348*, 2024.

[44] Z. Wu, J. Shen, Y. Zheng, C. Xie, C. Zhao, T. Zhao, J. E. Gonzalez, I. Stoica, and R. Jia, "Defending large language models against jailbreaking attacks through goal prioritization," *arXiv preprint arXiv:2311.09096*, 2023.

[45] S. Jain, A. Naik, A. Agarwal, Y. Shi, Z. Zhao, B. Ananthanarayanan, N. Naik, Z. Zhao, A. Sharma, S. Ananthakrishnan *et al.*, "Baseline defenses for adversarial attacks against aligned language models," *arXiv preprint arXiv:2309.00614*, 2023.

[46] M. T. Ribeiro, T. Wu, C. Guestrin, and S. Singh, "Beyond accuracy: Behavioral testing of nlp models with checklist," *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4902–4912, 2020.

[47] A. Robey, H. Hassani, and G. J. Pappas, "Smoothllm: Defending large language models against jailbreaking attacks," *arXiv preprint arXiv:2310.03684*, 2023.

[48] A. Kumar, C. Shen, S. Singla, C. Tan, and S. Feizi, "Certifying llm safety against adversarial prompting," *arXiv preprint arXiv:2309.02705*, 2023.

[49] E. Perez, S. Ringer, K. Lukošiūtė, K. Maharaj, S. Burnell, A. Kenton, D. Hernandez, A. Ganesh, A. Goldie, A. Mirhoseini *et al.*, "Red teaming language models with language models," *arXiv preprint arXiv:2202.03286*, 2022.

[50] D. Ganguli, D. Hernandez, L. Lovitt, A. Askell, Y. Bai, A. Bergen, T. Besiroglu, T. Conerly, D. Drain, J. Eisenstein *et al.*, "Red teaming language models to reduce harms: Methods, scaling behaviors, and lessons learned," *arXiv preprint arXiv:2209.07858*, 2022.

[51] P.-Y. Chen, Y. Sharma, H. Zhang, J. Yi, and C.-J. Hsieh, "Combating adversarial attacks using sparse representations," *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4785–4794, 2023.

[52] A. Yang, B. Yang, B. Hui, B. Zheng, B. Yu, C. Zhou, C. Li, C. Li, D. Liu, F. Huang, G. Dong, H. Wei, H. Lin, J. Tang, J. Wang, J. Yang, J. Tu, J. Zhang, J. Ma, J. Xu, J. Zhou, J. Bai, J. He, J. Lin, K. Dang, K. Lu, K. Chen, K. Yang, M. Li, M. Xue, N. Ni, P. Zhang, P. Wang, R. Peng, R. Men, R. Gao, R. Lin, S. Wang, S. Bai, S. Tan, T. Zhu, T. Li, T. Liu, W. Ge, X. Deng, X. Zhou, X. Ren, X. Zhang, X. Wei, X. Ren, Y. Fan, Y. Yao, Y. Zhang, Y. Wan, Y. Chu, Y. Liu, Z. Cui, Z. Zhang, and Z. Fan, "Qwen2 technical report," *arXiv preprint arXiv:2407.10671*, 2024.

[53] E. Almazrouei, H. Alobeidli, A. Alshamsi, A. Cappelli, R. Cojocaru, M. Debbah, E. Goffinet, D. Hesslow, J. Launay, Q. Malartic, D. Mazzotta, B. Noune, B. Pannier, and G. Penedo, "The falcon series of open language models," *arXiv preprint arXiv:2311.16867*, 2023.

[54] Y. Bai, S. Kadavath, S. Kundu, A. Askell, J. Kernion, A. Jones, A. Chen, A. Goldie, A. Mirhoseini, C. McKinnon *et al.*, "Constitutional ai: Harmlessness from ai feedback," *arXiv preprint arXiv:2212.08073*, 2022.