LLM STINGER: Jailbreaking LLMs Using RL Fine-Tuned LLMs (Student Abstract)

Piyush Jha, Arnav Arora, and Vijay Ganesh

Georgia Institute of Technology, USA
{piyush.jha, aarora362, vganesh}@gatech.edu

Abstract

We introduce LLM STINGER, a novel approach that leverages Large Language Models (LLMs) to automatically generate adversarial suffixes for jailbreak attacks. Unlike traditional methods, which require complex prompt engineering or white-box access, LLM STINGER uses a reinforcement learning (RL) loop to fine-tune an attacker LLM, generating new suffixes based on existing attacks for harmful questions from the HarmBench benchmark. Our method significantly outperforms existing red-teaming approaches (we compared against 15 of the latest methods), achieving a +57.2% improvement in Attack Success Rate (ASR) on LLaMA2-7B-chat and a +50.3% ASR increase on Claude 2, both models known for their extensive safety measures. Additionally, we achieved a 94.97% ASR on GPT-3.5 and 99.4% on Gemma-2B-it, demonstrating the robustness and adaptability of LLM STINGER across open and closed-source models.

Introduction

Jailbreaking Large Language Models (LLMs) involve crafting inputs that lead safety-trained models to violate developer-imposed safety measures, producing unintended or harmful responses. One effective method for this is through suffix attacks, where specific strings are appended to the input to trigger undesired behavior. Suffix-based attacks have shown success against both white-box and black-box LLMs, offering a simpler, more efficient, and easily automated alternative without the need for complex prompt engineering and human creativity to craft situations and role-playing templates (Zou et al. 2023). Although most of the existing suffix attacks have been patched because of safety training, we observed that modifications of those suffixes can still lead to successful jailbreak attempts. However, manually crafting these modifications or using a white-box gradient-based attacker to find new suffixes is laborious and time-consuming, limiting the scalability of such efforts.

In this work, we introduce LLM STINGER, a tool that uses an LLM to automatically generate highly effective adversarial suffixes that can jailbreak safety-trained LLMs. By fine-tuning an attacker LLM in a reinforcement learning (RL) loop, LLM STINGER generates new attack suffixes for a set of harmful questions from the HarmBench benchmark (Mazeika et al. 2024). This automated approach efficiently discovers new suffixes that bypass existing defenses, streamlining the process of crafting jailbreak attacks. The RL loop refines the attacker LLM with the help of reward signals that guide it toward more effective attacks. We compared LLM STINGER against 15 SOTA attack methods, and it outperformed all of them on attack success rate. For example, we were able to increase the ASR (+50.3%) on safety-trained closed-source models such as Claude.

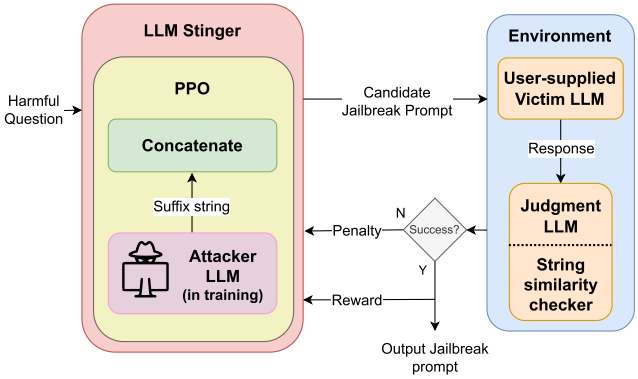

Figure 1: Architecture Diagram of LLM STINGER

LLM STINGER

Implementation Details of LLM STINGER

During training, the attacker LLM takes as input the harmful question and seven publicly available suffixes (Zou et al. 2023) and is prompted to generate similar new suffixes. The generated suffix is appended to the harmful question (from the training set) and sent to the victim LLM, requiring only black-box access to the model. The victim's response is evaluated by the judgment model, which provides binary feedback on whether the attack succeeded. Additionally, if the attack fails, a string similarity checker provides token-level feedback, allowing for more precise adjustments to the generated suffixes (more details in Jha, Arora, and Ganesh 2024). This mechanism evaluates how closely the generated suffix aligns with previously successful suffixes, penalizing deviations that stray too far from patterns known to bypass

Attack method / Victim LLM	LLM STINGER (ours)	GCG	GCG-M	GCG-T	PEZ	GBDA	UAT	AP	SFS	ZS	PAIR	TAP	TAP-T	Auto DAN	PAP top5	Human	DR
Llama2-7B-chat	**89.3**	32.1	19.5	15.9	0.0	0.0	3.1	19.5	3.1	0.4	6.9	5.0	3.8	0.0	0.8	0.1	0.0
Vicuna-7B	**93.08**	89.9	83.9	83.1	17.6	16.9	18.2	76.1	52.8	26.5	66.0	67.9	78.0	89.3	15.6	46.7	20.1
Claude 2	**52.2**	-	-	1.5	-	-	-	-	-	0.6	1.9	1.3	0.0	-	0.1	0.0	0.0
Claude 2.1	**26.4**	-	-	1.4	-	-	-	-	-	0.6	2.5	1.9	0.0	-	0.1	0.1	0.0
GPT 3.5 Turbo 0613	**88.67**	-	-	44.3	-	-	-	-	-	20.3	52.8	54.7	78.6	-	10.6	25.9	16.4
GPT 3.5 Turbo 1106	**94.97**	-	-	56.4	-	-	-	-	-	33.6	42.1	45.9	60.4	-	11.9	3.0	36.5
GPT 4 Turbo 1106	80.50	-	-	21.4	-	-	-	-	-	9.3	41.5	43.4	**81.8**	-	11.9	1.4	6.9

Table 1: Attack Success Rate on HarmBench (standard behaviours test split) for open-source and closed-source victim LLMs. Bold cells highlight the best-performing attack method for each victim LLM, while underlined cells indicate the second best. The dashed (-) cells indicate that the attack method is incompatible with the victim LLM, as it is a closed-source (black-box) LLM. Additional details on the attack methods and victim LLMs are provided in Jha, Arora, and Ganesh (2024).

safety measures. This feedback is used to fine-tune the attacker LLM in an RL loop for 50 epochs using the Proximal Policy Optimization (PPO) algorithm, with the goal of improving the attack success rate over multiple iterations (Figure 1). Successful suffixes are saved, and during evaluation on test set questions (disjoint from training set), we assess whether the attacker LLM can generate a question-specific harmful suffix or if any of the 38 newly generated suffixes can successfully trigger an attack.

Experimental Setup

We use the standard behavior benchmark from HarmBench that includes a train-test split, and the main metric for evaluation is the attack success rate, a widely used standard in adversarial attack research (Mazeika et al. 2024). For the attacker model, we chose Gemma, an open-source LLM that allows for system prompt modifications and is not overly cautious in its responses, making it suitable for adversarial purposes. The judgment model is the HarmBench judgment LLM, selected because it outperforms other judgment models (GPT-4, Llama-Guard, AdvBench) on their manually labeled validation sets. To compare LLM STINGER with existing methods, we evaluate it against 15 popular red-teaming approaches and Direct Request (DR) baseline also used by HarmBench. Due to computational limitations, we select a subset of HarmBench victim LLMs. For open-source models, we focus on LLaMA2, known for its robustness to GCG and other attacks, as well as Vicuna. In the closed-source category, we test on GPT and Claude models, which have undergone extensive safety training with strong defenses at the system and model levels.

Discussion of Results

Our method demonstrates a significant improvement in jailbreaking LLMs, particularly those that have undergone safety training, where other attack methods have struggled. As shown in Table 1, on Llama2-7B-chat, we observed the highest increase in Attack Success Rate (ASR), with our method achieving a +57.2% improvement compared to the next best approach. This substantial increase highlights the efficacy of our technique in bypassing the defenses of highly safety-trained models.

When evaluating closed-source (black-box) models, our method increased the ASR by +50.3% on Claude 2, a model known for its rigorous safety training. In comparison, the next best method only reached a 1.9% ASR, further highlighting the strength of our approach. On GPT-3.5, we achieved the highest ASR of 94.97%, showcasing the robustness of our method across different closed-source models. A key advantage of our approach is that it only requires black-box access to the victim model, unlike many of the attack methods in HarmBench, which require white-box access. This allows our method to support a wider range of models, as it doesn't rely on internal model weights, making it more versatile and applicable to both open and closed-source LLMs. We also tested our method on Gemma-2B-it, the same model we use as our attacker LLM, which is not included in the HarmBench benchmark. Remarkably, we achieved an astounding ASR of 99.4% on this model using our method, further demonstrating its adaptability and effectiveness across different models. Since we use a judgment LLM classifier model to assess the success of each attack during feedback, we took extra steps to manually verify the outputs from the victim models. This ensures that our approach is genuinely breaking through the model's defences, rather than gaming the classifier. In the future, we plan to include support for additional token-level attack methods, conduct comparisons against more attack strategies and victim LLMs, including multi-modal models, and explore additional feedback mechanisms to enhance the effectiveness of the attacker LLMs.

Efficacy of LLM STINGER

It is easy to see that the jailbreak attack search problem addressed here is combinatorially hard, and hence naive methods won't work. Our RL-based heuristic search technique is effective, as it is guided by feedback from a judgment LLM and a string similarity checker that helps prune the search space dramatically and focus its exploration on high-potential regions. As a result, it is not only able to discover new attack suffixes efficiently but also adapts to the defenses of highly safety-trained models, thus demonstrating the practical strength of our approach.

References

Jha, P.; Arora, A.; and Ganesh, V. 2024. LLMStinger: Jailbreaking LLMs using RL fine-tuned LLMs. *arXiv preprint arXiv:2411.08862*.

Mazeika, M.; Phan, L.; Yin, X.; Zou, A.; Wang, Z.; Mu, N.; Sakhaee, E.; Li, N.; Basart, S.; et al. 2024. Harmbench: A standardized evaluation framework for automated red teaming and robust refusal. *arXiv preprint arXiv:2402.04249*.

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