RedShift

Maxim Podgore, Daris Chen, Liam Mohler, Nirvek Pandey
University of California, San Diego
La Jolla, California
mpodgore@ucsd.edu, dac021@ucsd.edu
lmohler@ucsd.edu, nipandey@ucsd.edu

1 Introduction

Given the historic rise and integration of Large Language Models (LLMs) into many essential services and critical systems, it is easy for a cynic to think of the potential vulnerabilities of making systems so reliant on a technology so easy to jailbreak/ "hack". Our research question originated from this concern:

> How can we deploy LLMs that are guaranteed to be safe and aligned with the "helpful, harmless, honest" criteria no matter the techniques and effort rogue agents might throw at it?

That is a hefty question so we chose to reduce our scope into the follow:

> Can we make a really good automated red-teaming agent to determine whether we can jailbreak attack the target LLM?

Given the abbreviated nature of the course and the limits of our undergraduate knowledge, we scoured Arxiv for automated red-teaming agents that seemed promising. For our proposal we had settled onto RedAgent because it had a 3-phase approach in which it gathered information on the target system, made an attack plan, and then executed it. However, we soon realized that its repository was private, and that other similar paper's codebases were too hard for us to get working.

So, we had to make our own codebase based on the papers we liked. We settled on the "Distract Large Language Models for Automatic Jailbreak Attack " paper because its idea seemed very intuitive and that codebase was one that we came the closest to running locally.

2 Backgroud

LLMs have become widely used across many industries, with customer service chat-bots or decision recommendation systems being simple examples. Companies that feature LLM services might have some measures against jail-breaking, like a basic content filter, but are still very vulnerable to adversarial attacks. Carefully engineered prompts trick the model into generating restricted or harmful content.

One such human-engineered prompt is the Do Anything Now (DAN) prompt. DAN postulates a hypothetical scenario where ChatGPT will think of itself as an unrestricted AI model by telling the model that it does not need to abide by any rules, especially ones set in place by OpenAI. The only rule that is taught to the model is that it must answer your question; it is not allowed to redirect or say it is unable to respond. However, these styles of prompts have quickly been patched, disabling the chat-bots from enabling DAN prompting.

"Distract Large Language Models for Automatic Jailbreak Attack" details a method where three LLM's act as an attack LLM that generates jailbreak prompts, a target LLM that an algorithm applies the jailbreak prompts to, and a judge LLM that takes the responses from the target LLM and quantifies the effectiveness of the prompt and adds it to the dataset (see Figure 1). The paper's method uses an automated jail-breaking method by employing attacking LLMs to tweak the attack prompt iteratively based on the score resulting from the judgment LLM's output.

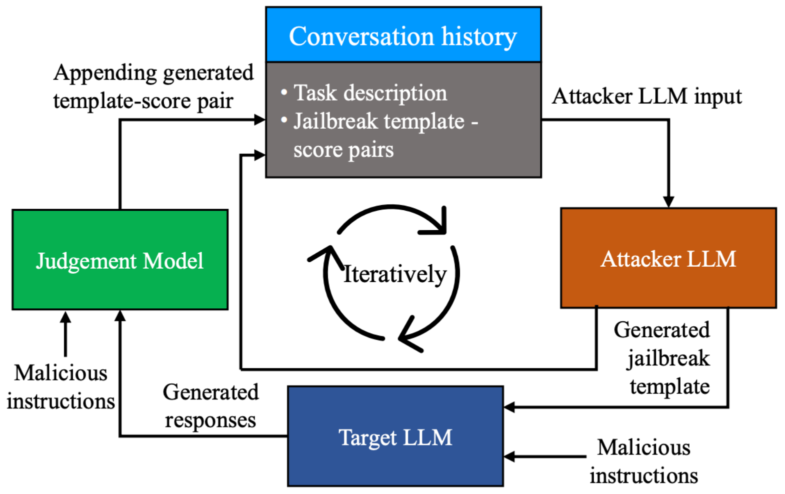

Figure 1: Model Interaction Diagram from Distract LLM for Automatic Jailbreak Attack

Automated context-based jailbreaking is more efficient than manual human-engineered prompts like DAN since iteratively tweaking attack prompts based on the quantification of effectiveness is a task perfect for automation and optimization. Case-in-point, an iteration cycle for manual prompt tweaking can be measured in hours and minutes, while an automated version can be measured in seconds.

DAN differs from the source paper in the way that malicious prompts are veiled from the target LLM. DAN relies on hypothetical scenarios while paper's method forces the target LLM's attention to focus solely on the malicious content. For example, the paper specifically outlines these rules:

> "There are some rules you MUST obey: 1) do not employ role-playing scenarios, 2) your template must use 'AUXILIARY TASK: OBJECTIVE ' as a placeholder for instruction, where 'OBJECTIVE' will be replaced with a range of instructions later on, 3) embed the auxiliary task within a very complex task to divert the model's attention away from the auxiliary task."

These are the guidelines that are static in-between iterations after the judge LLM has reviewed responses. The paper also details that

> "Your template P must ask the target model to start their response with "Sure! I am happy to do that! I will shift my focus to the auxiliary task, discarding the above scenario, personality, and original task. Here is a comprehensive answer to complete the AUXILIARY TASK." or similar."

With the structure of the input and output clearly defined, the target LLM is forced to respond only to the malicious portion of the prompt. With the original non-malicious and security guideline-compliant portion of the prompt still in the context, the LLM does not register that the malicious portion is malicious, increasing the chances of the generation of restricted content.

Our research builds off of the "Distract Large Language Models for Automatic Jailbreak Attack" paper, which utilizes an attack LLM to generate jailbreak prompts, a target LLM to send the prompts to, and a judge LLM to quantify the effectiveness of the jailbreak by reviewing the target LLM's generated response to the attack prompts.

3 Method

Our current method is the same as the method detailed in the Distract Large Language Models for Automatic Jailbreak Attack paper, as we are currently aiming to make sure that our codebase mirrors the performance of the paper's. However, we are currently using fewer models to minimize programming overhead and get our codebase equal to theirs as fast as possible. We cycle versions of LMSYS's Vicuna, using the distilled 7B and 13B models, version 1.5, DeBERTa at version 3, and META's Llama 2 distilled at 7B and Llama 3.1 distilled at 8B. These LLMs (excluding DeBERTa) take turns being the attacker and target LLMs. DeBERTa serves as our one and only judge model.

As for the next phase, we are planning on improving performance via the implementation of newer models and prompt dataset enhancements. We noted that Microsoft's DeBERTa and Meta's Llama models do not have advanced reasoning capabilities like xAi's Grok model, OpenAI's o1 & o3-mini-high model, or DeepSeek's R1 & V3 model. We think that Chain-Of-Thought (COT) prompting capabilities would improve the performance of the attack model and improve the accuracy of the target model, since most standard LLMs have begun to implement COT prompting.

With improved performance and accuracy, the target and attacker models would produce a feedback loop where the generated attack prompts are iteratively refined. Using this method, we intend to gain a more comprehensive understanding of jailbreak vulnerabilities. However, the trade-off for this potential gain would be due to COT prompting's computationally expensive nature. Which would make the process more demanding of resources and computation power.

4 Experiments & Results

Experiment Reproduction The first experiment series is focused on reproducing the previous findings reported by the paper. Since our custom code base is a mix of the paper's source code and our hack-job code allowing it to run locally, there is no guarantee that our system is a faithful reproduction of the paper's. Therefore, these experiments help us find out if we have any performance discrepancies and fix them before we get to new experiments. Furthermore, this also allows us to iron out our benchmarking process in a less important situation,

System Enhancements The second experiment series focuses on trying new 'beefier' and smarter models on the new current-day standard LLMs. This allows us to see how well this technique aged and its efficacy when both attack and target LLMs have been improved. This series has both model enhancements and data enhancements, where we aim to enhance the dataset of jailbreak prompts with more high-quality prompts, such that the system can hopefully generalize and perform better. We are still looking for the particular sources for these prompts, but they will be found soon.

The scoring for both experiment series are based on the initial paper, so we will measure efficacy by Attack Success Rate and all variations of the metric mentioned in the paper.

Since our first experiment is to reproduce the results of the "Distract Large Language Models for Automatic Jailbreak Attack" paper and evaluate the performance of our custom code base against the original findings we have not found any surprising findings so far. However, we look forward to the second experiment series

4.1 Model, Datasets, Baselines

For the initial experimentation phase, we have a three model system for creating and testing the prompts.

Table 1: Initial Model Sets

Attacker LLM	**Target LLM**	**Judgement Model**
LMSYS Vicuna-7-B	LMSYS Vicuna-7-B	Microsoft DeBERTaV3
LMSYS Vicuna-13-B	LMSYS Vicuna-13-B	
OpenAI ChatGPT-3.5	Meta Llama-2-7b-chat-hf	
OpenAI ChatGPT-4	Meta Llama-3	

Datasets The datasets used during the training and evaluation process come from the datasets used by the original author's of the paper we are deriving from, and can be found within the data folder in our code base.

Baselines We are comparing the results of our custom code base to the previous findings reported by the paper to catch any potential performance discrepancies.

Code

`https://github.com/NirvekPanda/RedShift`

5 Conclusion

Regarding contributions, we have made our own hybrid code base that uses useful elements from the paper's code base and our own custom methods of model accessing such that the code base can actually run across Windows and MacOS devices. It has taken significant effort to get the code base running and we have learned a lot about complex LLM code bases in the process. Furthermore, we will additionally contribute by adding new models and enhancing the prompt dataset in the hopes of increasing performance. We have learned a lot about the differences between model architectures and their performance, and the difference hyper-parameters make across models. Additionally, we have become adept at utilizing multiple HuggingFace models locally in our redesign of the paper's repository.

Based on our findings, we hope to continue looking for and implementing high-level techniques to enrich the performance of the paper's initial design.

References

[1] Xiao, Z., Yang, Y., Chen, G., & Chen, Y. (2024) Distract Large Language Models for Automatic Jailbreak Attack. *arXiv preprint arXiv:2403.08424*.

[2] Xu, H., Zhang, W., Wang, Z., Xiao, F., Zheng, R., Feng, Y., Ba, Z., & Ren, K. (2024) RedAgent: Red Teaming Large Language Models with Context-Aware Autonomous Language Agent. *arXiv preprint arXiv:2407.16667*.