Análise comparativa sobre atividades de *Red Team* de LLMs com LLMs

Charles Christian Miers[1], **Milton Pedro Pagliuso Neto**[1], **Ana Carolina Vedoy Alves**[1]

[1]Departamento de Ciência da Computação, Centro de Ciências Tecnológicas - CCT
Universidade do Estado de Santa Catarina, UDESC - Joinville, Santa Catarina

Resumo. *O uso de Large Language Models (LLMs) cada vez mais intensivo e com expressivos parâmetros e entendimento de contexto pode ser explorado para causar problemas de segurança. Este trabalho visa esclarecer uma comparação entre as ferramentas disponíveis para detecção de vulnerabilidades no escopo de Red Teaming em LLMs. Foi realizada uma revisão sistemática, bem como comparações de algoritmos de estado da arte na área com novas propostas o uso de LLMs de forma adversária a algoritmos da mesma forma. Foi realizada uma comparação inicial levando em comparação o desempenho e reprodutibilidade de dois algoritmos: AutoDAN e GCG.*

1. Introdução

A *Generative Artificial Intelligence* (GenAI) é uma subárea de *Inteligência Artificial* (IA) que expandiu a capacidade de geração de texto em linguagem natural por máquinas. Entre os modelos de GenAI existem os LLMs, e estes são responsáveis por uma nova classificação sobre geradores de texto propondo algoritmos com processamento em larga escala. Os LLMs são modelos que possuem alto número de parâmetros, compostos por redes neurais profundas, e como consequência requererem alto poder de processamento, sendo possíveis de existir devido as evoluções de processamento de hardware dos últimos anos [Kucharavy et al. 2024].

Apesar de inovadores, outra característica dos modelos LLMs é seu comportamento estocástico [Bender et al. 2021], com não definição prévia de como estes algoritmos podem se comportar de forma segura. Logo, algumas vulnerabilidades como *jailbreaks* podem surgir [Huang et al. 2022]. Utilizando princípios de áreas da segurança da informação como conceitos de *Red Team*, há possibilidade de utilização de outros modelos de algoritmos.

O objetivo deste artigo é uma comparação inicial do desempenho de duas ferramentas com algoritmos que geram estes prompts maliciosos otimizados. Estes trabalhos são resultado de uma revisão sistemática de artigos sobre atividades de *Red Teaming* em contextos adversariais com LLM alvo: [Zou et al. 2023] e [Liu et al. 2024]. A comparação é parametrizada pelo sucesso ou não de *jailbreaks* causados a entidade alvo, além do LLM alvo. Também é abordado o tempo levado para a evolução dos *prompts* propostos como capazes de reprodução de *jailbreak*, além de custo computacional.

Este artigo é organizado como segue. Uma explicação da área de IA na Seção 2, seguida da revisão sistemática na Seção 3. o comparativo com explicações dos algoritmos na Seção 4, e por fim conclusões finais na Seção 5.

2. IA e Cibersegurança

Os LLMs são modelos que processam texto através de redes neurais do tipo *Generative Deep Neural Networks* (GDNN), no qual, após o treino para entendimento de contexto desses dados de entrada, são gerados novos dados de texto [Oussidi and Elhassouny 2018]. Dentro desse tipo de modelo, textos são compreendidos por unidades de texto chamadas *tokens*, que podem ser parte de palavras ou até mesmo mais de um conjunto de palavras. Um algoritmo, para ser categorizado como LLM, deve conseguir aplicar uma regressão probabilística destes *tokens*, através de dados de treinamento, para uma entrada de um usuário [Kaplan et al. 2020].

Nestes algoritmos há classificações como o tipo de arquitetura do ponto de vista da rede neural, e também sobre a ótica de acesso a aplicação desta rede. Sobre a arquitetura de rede neural, pode-se separar a diversidade de algoritmos, como algoritmos *Generative Pre-trained Transformer* (GPT) da OpenAI, *Compute-Optimal* como *Large Language Model Meta AI* (LLaMa), mantido pela Meta como código aberto [lla], entre outros unidos por suas arquiteturas de origem. Quanto ao acesso a aplicação, os modelos podem ser separados como *white-box* e *black-box*. Em modelos *white-box* estão arquiteturas em que se tem total acesso a aplicação, como código fonte e com acesso aos dados de treinamento, parâmetros e dados de treino [Lin et al. 2024]. Em modelos *black-box* não há entendimento e acesso sobre esses aspectos citados, como é o caso de LLMs dentro do grupo GPT a partir do GTP-3 [OpenAI 2025].

3. Revisão sistemática

Foi realizada uma revisão sistemática sobre ferramentas adversariais no contexto de LLMs, com um modelo alvo e outro de ataque, a fim de comparar estas ferramentas e também explorar principais vulnerabilidades de LLMs. Neste estudo foram analisadas as seguintes questões de pesquisa (RQ) a serem respondidas. RQ1. Qual é o número de estudos publicados sobre *Red Teaming* em LLM utilizando ferramentas com LLM no período de 2022 a 2024? RQ2. Quais são os critérios de exclusão sobre os artigos selecionados pela busca inicial? Dentro do contexto de pesquisa da área de LLM, artigos recentes são foco na parte desta revisão sistemática, dado que uma das ferramentas de estado da arte sobre ataques em LLMs, que serve de base de comparações nesse trabalho, é de 2023 [Zou et al. 2023]. As chaves de busca utilizadas foram: *LLM*, *Red Teaming* ou *Red Team* e *adversarial*. A base de dados explorada foi o Google Scholar, na qual a chave de busca utilizada foi: "*red-teaming and (llm or llms or large-language-models) and adversarial and (jailbreak or framework)*". Foram acrescentados os termos "-*survey*" e " - *benchmark*" na *string* de pesquisa final para evitar resultados irrelevantes de acordo com os critérios de exclusão. Um total de 69 artigos foram encontrados, levando em conta o filtro de data de publicação de 2022 a 2024, sendo a última pesquisa de atualização realizada 03 de Outubro de 2024.

Os artigos encontrados que reforçam os critérios da busca são: ReNeLLM, DeGCG, EnJa, AgentPosition, AutoDAN, ToxDet e Fuzz Testing-Driven. O projeto AutoDAN têm códigos inspirados ou derivados de partes do algoritmo *Greedy Coordinate Gradient* (GCG), além de trazer comparações quanto a taxa de assertividade de ataques realizados dos dois modelos. A partir disto, a reprodução do algoritmo de estado da arte GCG e do trabalho AutoDAN como comparativo é proposta.

4. Comparativo

A comparação entre algoritmos de geração de *jailbreaks* não é apenas medida pela taxa de sucesso destas operações, mas sim em relação ao seus requisitos de processamento, além dos recursos utilizados pelos próprios LLMs alvo, e a depender do caso, o LLM atacante. O algoritmo GCG propõe a atividade de *Red Team* através de ataques em ambiente controlado em um modelo LLM com o objetivo de produzir um *jailbreak* com foco em sufixo para frases maliciosas. O algoritmo AutoDAN tem como objetivo também reproduzir uma atividade de *Red Team*como atacante com finalidade de gerar um *jailbreak*, através de sentenças de *tokens* calculadas com ajuda de algoritmo genético. Ambas propostas válidas e produzidas como experimentos de acesso livre, opensource, que carecem de reprodutibilidade com suas publicações recentes. O ambiente utilizado para reprodução conta com uma placa de vídeo NVIDIA 3090 e memória RAM de 46GB. O ambiente de execução utilizou o modelo Llama2 localmente, na versão "*llama-2-7b-chat-hf*"para ambos os testes, além de modificar o modelo para quantização em 4 bits.

4.1. GCG

O ataque tem foco na elaboração de uma sequência de *tokens* como sufixo a *prompts* maliciosos, com a otimização de uma taxa calculada como probabilidade de sucesso em *jailbreaks*. O ataque padroniza uma busca por perda de gradiente como otimização para a taxa de probabilidade de *jailbreaks* a cada passo que é executado. Em testes, com placa NVIDIA 3090, não foi possível completar a execução do algoritmo padrão de 1000 passos devido a limitações de memória. Porém, com 750 rodadas a taxa de perda de gradiente do modelo se manteve em decadência até atingir um ponto de 0,4172, sendo assim, a probabilidade de *jailbreak* calculada foi aumentada, como visto na Figura 1. O tempo de execução do algoritmo foi de aproximadamente 20,875 horas, e cada passo teve média de de 82 segundos.

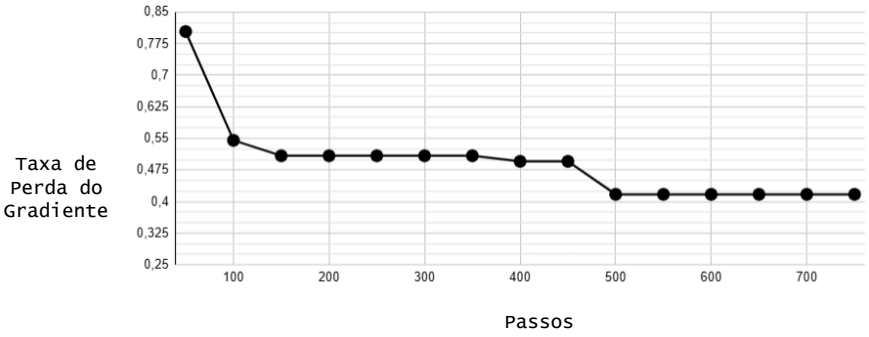

Figura 1. Taxa de perda do gradiente ao longo dos passos executados GCG.

4.2. AutoDAN

O algoritmo AutoDAN é inspirado, em partes, pelo código do algoritmo GCG [Liu et al. 2024]. Porém, este não gera uma sequência de sufixos para produzir o *jailbreak*, mas constrói um *prompt* completo de mesmo objetivo de forma otimizada. Através de um LLM para produção dos *prompts* atacantes, o modelo utiliza um algoritmo genético hierárquico que modifica *prompts* através de avaliação, com uma probabilidade de ataque bem sucedido. Cada nova geração é avaliada por uma função *fitness* e esta função na versão atual usada nos testes é a AUTODAN-GA. Após uma execução de 308,62 segundos, e três épocas geradas, o algoritmo já reproduziu um ataque que resultou em uma frase alvo correspondente a um possível *jailbreak*.

5. Considerações e Trabalhos futuros

O algoritmo AutoDAN, apesar de não necessariamente ter completado sua execução, reproduziu uma frase categorizada como *jailbreak* ao remodelar o *prompt* de ataque inicial O algoritmo GCG, ao não completar sua execução, além de não contar com o auxílio de uma LLM atacante, tem desempenho em tempo de execução alto para o hardware utilizado. Sua tentativa não reproduziu *jailbreak*, porém novos testes com mais consultas e capacidade de hardware adequado ao modelo e o algoritmo atacante são necessários. Em uma análise de reprodutibilidade inicial feita neste trabalho, é possível afirmar que o algoritmo AutoDAN possui desempenho de capacidade de *jailbreak* e velocidade de execução que supera o algoritmo estado da arte GCG.

Agradecimentos:
Os autores agradecem o apoio do LARC/USP, LabP2D/UDESC, FDTE e FAPESC. Este trabalho foi apoiado pelo CNPq (processos 307732/2023-1 e 311245/2021-8), FAPESP (processo 2020/09850-0) e CAPES (Código de Financiamento 001).

Referências

Large Language Model Meta AI (LLaMA) 3.2. Meta Platforms, Inc. https://www.llama.com Acesso em 14 de setembro de 2024.

Bender, E., Gebru, T., McMillan-Major, A., and Shmitchell, S. (2021). On the dangers of stochastic parrots: Can language models be too big? pages 610–623. 10.1145/3442188.3445922.

Huang, J., Shao, H., and Chang, K. C.-C. (2022). Are large pre-trained language models leaking your personal information? https://arxiv.org/abs/2205.12628.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. (2020). Scaling laws for neural language models. https://arxiv.org/abs/2001.08361.

Kucharavy, A., Plancherel, O., Mulder, V., Mermoud, A., and Lenders, V. (2024). Large language models in cybersecurity: Threats, exposure and mitigation. note: 10.1007/978-3-031-54827-7.

Lin, L., Mu, H., Zhai, Z., Wang, M., Wang, Y., Wang, R., Gao, J., Zhang, Y., Che, W., Baldwin, T., Han, X., and Li, H. (2024). Against the achilles' heel: A survey on red teaming for generative models. https://arxiv.org/abs/2404.00629.

Liu, X., Xu, N., Chen, M., and Xiao, C. (2024). Autodan: Generating stealthy jailbreak prompts on aligned large language models. https://arxiv.org/abs/2310.04451.

OpenAI (2025). OpenAI GPT4 Research. https://openai.com/index/gpt-4-research.

Oussidi, A. and Elhassouny, A. (2018). Deep generative models: Survey. In *2018 International Conference on Intelligent Systems and Computer Vision (ISCV)*, pages 1–8. 10.1109/ISACV.2018.8354080.

Zou, A., Wang, Z., Carlini, N., Nasr, M., Kolter, J. Z., and Fredrikson, M. (2023). Universal and transferable adversarial attacks on aligned language models. https://arxiv.org/abs/2307.15043.