

Mecanismos de Avaliação de Segurança de Large Language Models (LLMs): Uma Análise Comparativa Entre Ferramentas Modernas

Emilly Stephanny Silva², Isabela Kremp Narciso², Milton P. Pagliuso Neto²
Marcos A. Simplicio Jr¹, Victor T. Hayashi¹, Charles C. Miers²

¹ Universidade de São Paulo (USP)

{victor.hayashi, msimplicio}@usp.br

²Universidade do Estado de Santa Catarina (UDESC) – Joinville

{emilly.silva1, isabela.kn1822, milton.neto}@edu.udesc.br, charles.miers@udesc.br

Abstract. *The use of Large Language Models (LLMs) has spread across diverse domains, enabling analysis, translation, and content generation. While this brought benefits, it also introduced potential risks, as the training processes applied to these models do not protect them from all possible attacks. To prevent this, red teams have developed automated attack tools to evaluate the robustness of these models. This article proposes a brief qualitative analysis of automated red-teaming attack tools specifically designed for evaluating LLMs (SafeProbe, PyRIT, Garak, ARES and PromptFoo), thus facilitating an understanding of the principal differences between them.*

Resumo. *Com o uso de LLMs difundido em cada vez mais espaços, possibilitando análises, traduções e geração de conteúdo dos mais diversos tipos. Além de trazer consigo benesses, também emergiu com eventuais riscos. Os treinamentos aplicados nesses modelos não os blindam contra todos os possíveis ataques. Como forma de avaliar o alinhamento seguro dessas LLMs, red teams elaboram ferramentas de ataque automatizadas que possibilitam a avaliação da robustez desses modelos. Este estudo propõe a realização de uma análise comparativa qualitativa entre ferramentas de ataque automatizado de red teaming desenvolvidas especificamente para a avaliação de LLMs (SafeProbe, Python Risk Identification Toolkit (PyRIT), Generative AI Red-Teaming and Assessment Kit (GARAK), AI Robustness Evaluation System (ARES) e PromptFoo), permitindo compreender suas principais diferenças.*

1. Introdução

Os LLMs, modelos fundacionais treinados com grandes corpora de texto, têm sido utilizados em diversos campos do cotidiano, desde sistemas robustos até aplicações mais rotineiras, capazes de gerar respostas e textos de maneira semelhante à de um ser humano [Mao et al. 2026]. Contudo, apesar dos avanços nas arquiteturas e nos paradigmas de aprendizado de máquina, observa-se um número crescente de incidentes envolvendo tanto atores mal-intencionados quanto usuários, o que tem intensificado as preocupações quanto à geração de respostas inadequadas por esses modelos. [Yuan et al. 2026].

A fim de mitigar esse tipo de resposta, equipes de cibersegurança têm elaborado ferramentas que reproduzem uma série de testes com perturbações linguísticas, com o objetivo de gerar um conjunto de dados capazes de servir como uma avaliação da robustez de LLMs e consequentemente evidenciando deslizos [Yuan et al. 2026]. Diante da emergência de diversas ferramentas modernas, surge a incerteza sobre qual é a preferível em cada situação. Em vista disso, este estudo tem como objetivo analisar cinco ferramentas de ataques, com o intuito de trazer suas principais características, e a partir dessa análise, levantar uma comparação entre as mesmas.

O artigo está organizado como segue. A Seção 2 aborda a fundamentação teórica do tema, introduzindo o leitor ao que são LLMs, como seu uso desencadeia riscos e quais são as abordagens utilizadas para mitigá-los. A Seção 3 aborda a estruturação dos métodos. As Seções 4 e 5 apresentam as análises individuais e comparativas realizadas.

2. Fundamentação Teórica

As LLMs são modelos de Inteligência Artificial (IA) que, por meio de aprendizado profundo, i.e., utilizando uma gama de conteúdo em linguagem natural para o aprendizado da máquina, podem gerar sentenças de texto baseadas em *tokens* [Miers et al. 2024]. LLMs têm sido utilizadas nos mais diversos ambientes laborais, não se restringindo às áreas de tecnologia, mas também alcançando outras, como finanças e saúde, para análises, automações e previsões [Yu et al. 2026]. O estado atual da arquitetura das LLMs apresenta lacunas que podem comprometer a segurança cibernética de usuários e instituições, por meio de diversas vulnerabilidades, como a exposição de informações pessoalmente identificáveis, a reprodução de comportamentos ilegais ou de cunho preconceituoso, bem como a disseminação de instruções criminosas. As lacunas de segurança no contexto de IA adversária são objeto de esforços contínuos por parte de instituições e da academia, com o objetivo de mitigar os riscos associados a comportamentos indesejados e as vulnerabilidades desses sistemas. Esses esforços têm se materializado tanto no desenvolvimento de projetos voltados à segurança cibernética quanto na atuação de instituições responsáveis pela elaboração de diretrizes, padrões e mecanismos de avaliação aplicáveis a sistemas de IA. Tais iniciativas buscam estabelecer práticas comuns de segurança que orientem o desenvolvimento, a validação e a implantação confiável de modelos, reduzindo os riscos associados ao seu uso em ambientes reais [Yulianto et al. 2025]. Nesse contexto, têm sido propostas metodologias, referenciais e ferramentas destinadas à avaliação de aspectos como robustez, segurança e confiabilidade de modelos de IA. A diversidade dessas abordagens, entretanto, resulta em diferentes critérios, escopos e procedimentos de avaliação, o que dificulta a comparação direta entre as soluções existentes [Neto 2026]. Diante desse cenário, o presente artigo estabelece um conjunto de critérios para a análise dessas iniciativas e ferramentas, com o objetivo de evidenciar suas principais diferenças e fornecer subsídios para a seleção de abordagens adequadas a diferentes situações.

3. Método

O presente estudo está inserido no contexto de um projeto com uma instituição financeira, no qual as cinco soluções foram identificadas e selecionadas para a realização de uma breve análise comparativa: SafeProbe, PyRIT, GARAK, ARES e PromptFoo. Com a definição do escopo desse estudo, foram realizadas análises individuais de cada ferramenta, com base nas documentações oficiais, seguidas de uma análise comparativa entre

estas. No contexto do presente artigo, estabeleceram-se critérios voltados à confiabilidade da avaliação. Esses critérios contemplam aspectos relacionados aos mecanismos de julgamento (C1); às métricas de sucesso de ataques (C2); à existência de *inter-rater agreement* (C3); e ao alinhamento com a taxonomia NIST AI 100-2e2025 (C4) [Vassilev et al. 2025].

4. Análises Individualizadas

Nesta etapa do estudo, cada ferramenta é descrita brevemente, seguida de uma análise individual que emprega os quatro critérios (C1:C4) estabelecidos.

4.1. SafeProbe

O Safeprobe [Neto 2026] é um *toolkit* desenvolvido para verificar o alinhamento de segurança de LLMs. A ferramenta encontra-se em alinhamento com a taxonomia *National Institute of Standards and Technology* (NIST), possui código aberto e pode ser configurada e reproduzida em *Command-Line Interface* (CLI). Em suma, seu funcionamento segue o seguinte fluxo: ataque, consolidação de *inputs* e *outputs* em *Comma-Separated Values* (CSV), julgamento com multi-juízes, adição dos veredictos dos juízes ao arquivo CSV de resultados, análise quantitativa dos dados obtidos e disponibilização de um relatório final. A arquitetura da ferramenta apresenta seis módulos de ataque, nos quais se utilizam adaptadores de modelo para a elaboração dos mesmos. Uma vez realizados os ataques, seus resultados são consolidados e três modelos distintos julgam se foram bem-sucedidos. O relatório final apresenta a taxa de sucesso do teste, calculada pela fórmula *Attack Success Rate* (ASR), e a concordância entre os avaliadores, verificada por meio de Cohen's κ e de Fleiss' κ .

4.2. Generative AI Red-Teaming and Assessment Kit (GARAK)

A ferramenta GARAK [Derczynski et al. 2024] oferece uma abordagem estruturada para avaliar a segurança de LLMs por meio de um mecanismo de *Red Teaming*. Além disso, essa ferramenta oferece 44 módulos de ataque pré-configurados e é de código aberto, o que proporciona alta flexibilidade, contudo não apresenta alinhamento à taxonomia NIST. O processo de configuração da ferramenta é realizado por meio da linha de comando, na qual é realizada a leitura do tipo de *target model* que será avaliado e dos demais *plugins*. O fluxo de funcionamento do mecanismo possui quatro componentes principais: *Generators*, *Probes*, *Detectors* e *Bufs*. De forma simples, o *Probe* envia *prompts* maliciosos ao *Generator*, i.e., ao *target model* previamente definido. Enquanto isso, os *Bufs* buscam desestabilizar a interação entre os últimos por meio da modificação dos *prompts*. Depois, o *Generator* envia o *output* gerado aos *Detectors*, que podem ser *keyword-based* ou *Machine Learning* (ML) *classifiers*, para detectar sucessos e falhas. Por fim, após a finalização da execução, um relatório dos resultados obtidos é disponibilizado em arquivos JSON e HTML, contendo os *inputs*, os *outputs* e a taxa de sucesso do teste realizado, com *confidence intervals*.

4.3. Python Risk Identification Toolkit (PyRIT)

A ferramenta PyRIT [Microsoft 2026], desenvolvida pelo *Microsoft AI Red Team* e disponível em código aberto, apresenta um mecanismo equipado com técnicas para investigar novos riscos e explorar vulnerabilidades já conhecidas. Apesar de não apresentar alinhamento à NIST, é equipada com uma interface gráfica (COPyRIT) e possui 17 módulos

de ataque em sua versão original. A arquitetura da ferramenta possui seis componentes principais: *Memory*, *Targets*, *Converters*, *Datasets*, *Scorers* e *Orchestrators*. Essencialmente, os testes de segurança começam instanciando um *Orchestrator*. Depois, é realizada a definição dos *Targets* — *Adversarial Model* e *Target Model* — e o objetivo do teste. Após a configuração inicial, o *Adversarial Model* gera um *prompt* malicioso, que é modificado por um ou mais *Converters* antes de ser enviado ao *Target Model*. Então, o *output* gerado pelo *Target Model* é enviado para os *Scorers* para a avaliação do resultado do ataque. Esses componentes podem ser classificados como *Objective Scorers*, que verificam se o objetivo do ataque foi atingido e retornam um *score True/False*, e *Harm Scorers*, que verificam o nível de severidade da resposta por meio de uma *Float Scale*. Para esses últimos, o PyRIT emprega *inter-rater agreement* por meio do *Krippendorff's Alpha*. Toda a interação é gravada no componente *Memory*, desde a *prompt* gerada inicialmente e a *prompt* crua de *output*, até o veredicto dos *Scorers*. Um *score* pode ser classificado como *UNDETERMINED*, *SUCCESS* ou *FAILURE*. Caso o ataque não tenha sido bem-sucedido, o *Adversarial Model* deve tentar novamente, até obter uma tentativa bem-sucedida.

4.4. AI Robustness Evaluation System (ARES)

A ferramenta *open-source* ARES trabalha com um fluxo composto por quatro etapas principais: eleger o alvo, selecionar objetivos, estabelecer as estratégias a serem utilizadas nos ataques e, por fim, gerar os resultados obtidos [IBM DRL Security Team 2025]. Para a formulação desses ataques, podem ser utilizados *prompts* diretos por meio de um *chat* interativo ou de ferramentas implementadas via extensões. O *framework* possui onze *plugins* já implementados e treze variantes de *encoding*. Isso é possível, uma vez que a construção foi concebida para permitir a adição de utilitários externos sem alterar o código principal. Para a verificação do sucesso dos ataques, utilizam-se avaliadores, que podem ser aplicados de forma combinada ou não, como *Keyword-based*, *LLM-as-a-judge* e *Detector-based*. Há também avaliadores específicos para determinadas vulnerabilidades identificadas pela *Open Worldwide Application Security Project* (OWASP). Os resultados obtidos são transformados em um arquivo JSON, que, posteriormente, é utilizado para a geração de um *Dashboard*.

4.5. PromptFoo

O *framework* de código aberto PromptFoo [Promptfoo 2026] utiliza *plugins* para a geração de *prompts* de ataque, definidos com base em estratégias previamente estabelecidas. Essas estratégias abrangem diferentes tipos de ataques, incluindo abordagens multimodais com áudio e imagem. Além disso, a ferramenta relaciona a avaliação e as sugestões de mitigação a *frameworks* como o *NIST AI Risk Management Framework* (AI RMF), mas não segue a taxonomia adotada neste artigo. A execução dos ataques ocorre de forma iterativa: o *prompt* é aplicado ao modelo escolhido e, caso o ataque seja bem-sucedido, a execução é encerrada. Caso contrário, o resultado é avaliado e utilizado na elaboração de uma nova tentativa, até que o ataque tenha sucesso ou o limite de tentativas seja atingido. Para avaliação do sucesso do ataque, utilizam-se *Vulnerability Analysis* por meio de detectores e *Response Analysis* por meio de *LLM-as-a-judge*. Também é elaborado o *Risk Scoring* baseado no *Common Vulnerability Scoring System* (CVSS), composto por *Impact Base*, *Exploitability Modifier*, *Human Factor* e *Complexity Penalty*.

O critério de *Exploitability* utiliza a ASR como base de cálculo. Por fim, os resultados e possíveis medidas de mitigação são apresentados em um *Dashboard* [Promptfoo 2026].

5. Análise Comparativa

A partir das análises individuais, tornou-se possível realizar uma análise comparativa entre as ferramentas selecionadas (Tabela 1). Sobretudo, os resultados da análise baseiam-se na forma nativa das ferramentas em estudo, i.e., com o objetivo de manter a análise uniforme, as extensões disponíveis não foram consideradas.

Tabela 1. Análise comparativa entre ferramentas *Red Team* de LLMs.

Critério	SafeProbe	GARAK	PyRIT	PromptFoo	ARES
C1	Multi-model com CoT; scores 0/1	Keywords ou classificadores de ML; scores HIT/PASS	41 Scorers: 12 com LLM e 29 sem LLM; scores True/False ou Float Scale	detectores e LLM-as-a-judge	Keyword-based, LLM-as-a-judge, Detector-based, OWASP evaluator
C2	ASR	ASR + confidence intervals	AttackOutcome	ASR	Não nativo
C3	Cohen's κ e Fleiss' κ	Não nativo	Krippendorff's α	Não nativo	Não nativo
C4	Sim	Não nativo	Não nativo	Não nativo	Não nativo

Além do SafeProbe, as demais ferramentas não possuem alinhamento à taxonomia NIST AI 100-2e2025. Além disso, enquanto SafeProbe, PyRIT, PromptFoo e ARES apresentam mecanismos de julgamento baseados em LLMs, GARAK apresenta apenas mecanismos mais determinísticos, com menor capacidade de interpretar contextos e nuances semânticas. Além disso, entre as ferramentas com *LLM-as-a-judge*, apenas SafeProbe e PyRIT apresentam métricas de concordância entre os julgadores. Por fim, SafeProbe, GARAK e PromptFoo utilizam ASR como métrica de sucesso de ataques, enquanto PyRIT possui sua própria métrica de sucesso, e ARES não mensura ASR.

6. Considerações e Trabalhos Futuros

Este trabalho apresentou uma análise comparativa de cinco ferramentas de avaliação de segurança de LLMs, utilizando dados obtidos a partir do estudo de suas documentações oficiais. Em especial, foi possível observar uma falta de padronização desde seus mecanismos de julgamento e métricas de sucesso, até referenciais de alinhamento aplicados por cada uma, como GARAK (OWASP, *AI Vulnerability Database taxonomy* e *Language Model Risk Card*), PromptFoo (NIST AI RMF), PyRIT (categorias próprias) e SafeProbe (NIST AI 100-2e2025), pois não há um estabelecimento de diretrizes para o desenvolvimento de ferramentas dessa categoria. Ademais, a estrutura das ferramentas também é determinada por sua organização, o que facilita o aumento da disparidade entre sua arquitetura e seu fluxo de funcionamento. Contudo, a ferramenta ARES não apresenta uma métrica de sucesso para os resultados de um ataque. A métrica de sucesso mais utilizada é a ASR, exceto no caso da PyRIT, que utiliza um parâmetro próprio. Por fim, em contextos com múltiplos julgadores, apenas SafeProbe e PyRIT avaliam a concordância entre os juízes. Em trabalhos futuros serão implementados testes de eficiência das ferramentas analisadas, a fim de possibilitar uma análise exploratória a partir dos dados obtidos.

Agradecimentos:

Os autores agradecem o apoio do LARC/USP, LabP2D/UDESC, FDTE, FAPESC,

FAPESP e Banco Bradesco SA. 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

- Derczynski, L., Galinkin, E., Martin, J., Majumdar, S., and Inie, N. (2024). garak: A framework for security probing large language models. *arXiv preprint arXiv:2406.11036*. Acesso em: 2 set. 2026.
- IBM DRL Security Team (2025). Ai robustness evaluation system (ares). https://ibm.github.io/ares/getting_started.html. Acesso em: 10 out. 2026.
- Mao, Y., Liu, P., Cui, T., Liu, C., Xing, M., and You, D. (2026). Stop fixating on prompts: Reasoning hijacking and constraint tightening for red-teaming llm agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 64th Annual Meeting of the Association for Computational Linguistics. Acesso em: 2 set. 2026.
- Microsoft, A. R. T. (2026). Pyrit documentation: The python risk identification tool for generative ai. <https://azure.github.io/PyRIT/>. Acesso em: 1 set. 2026.
- Miers, C., Simplicio, M., Roja, M., Oliveira, D., Pagliuso, M., Damin, F., Bulla, R., and Hayashi, V. (2024). Ferramentas de jailbreaking para grandes modelos de línguas – aprendizado de máquina no contexto adversário. *Minicursos do XXIV Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*, page p. 49–91. Acesso em: 2 set. 2026.
- Neto, M. P. P. (2026). Safeprobe: a python toolkit for probing the safety alignment of large language models. Master’s thesis, UDESC. Acesso em: 2 set. 2026.
- Promptfoo (2026). Red teaming. <https://www.promptfoo.dev/docs/red-team/quickstart/>. Acesso em: 1 set. 2026.
- Vassilev, A., Oprea, A., Fordyce, A., Anderson, H., Davies, X., and Hamin, M. (2025). Adversarial machine learning: A taxonomy and terminology of attacks and mitigations. NIST Technical Series Publication NIST AI 100-2e2025, National Institute of Standards and Technology, Gaithersburg, MD. Acesso em: 2 set. 2026.
- Yu, G., Wang, Y., Wang, Z., Chen, L., Zheng, Y., and Liu, Y. (2026). Llms in industrial domains: A systematic review of adaptation techniques and applications from the product lifecycle perspective. *Advanced Engineering Informatics*, 74:104655.
- Yuan, S., Luo, N., Sun, J., Huang, Y., and Zhang, C. (2026). Red teaming llms via linguistic-aware fuzzing. *Proc. ACM Softw. Eng.*, 3(FSE). Acesso em: 2 set. 2026.
- Yulianto, S., Soewito, B., Gaol, F. L., and Kurniawan, A. (2025). Enhancing cybersecurity resilience through advanced red-teaming exercises and MITRE ATT&CK framework integration: A paradigm shift in cybersecurity assessment. *Cyber Security and Applications*, 3:100077.