risti
Revista Ibérica de Sistemas e Tecnologias de Informação
Iberian Journal of Information Systems and Technologies

Recebido/Submission: 28/02/2025
Aceitação/Acceptance: 30/06/2025

Detecção de Prompt Injection em modelos de Linguagem

Júlia Jamile Oliveira Gonçalves[1], Thiago Lotufo[2], Georges Daniel Amvame Nze[3], Fábio Lopes de Mendonça[4]

julia_jamile@hotmail.com; thiago.lotufo@ufrj.br; georges@unb.br; fabio.mendonca@unb.br

[1] Universidade Federal de Brasília, Programa de Pós-Graduação profissional em Engenharia Elétrica, 70910-900, Brasília, Brasil

[2] Universidade Federal do Rio de Janeiro, Programa de Pós-Graduação em Informática, 21941-916, Rio de Janeiro, Brasil

[3] Universidade Federal de Brasília, Programa de Pós-Graduação profissional em Engenharia Elétrica, 70910-900, Brasília, Brasil

[4] Universidade Federal de Brasília, Programa de Pós-Graduação profissional em Engenharia Elétrica, 70910-900, Brasília, Brasil

Pages: 104-116

Resumo: Os Modelos de Linguagem de Grande Escala (LLMs) são amplamente utilizados na indústria e na academia para tarefas diversas, como assistentes virtuais e automação de processos. No entanto, essas tecnologias apresentam vulnerabilidades de segurança, como ataques de *Prompt Injection*, que podem comprometer a integridade e confiabilidade dos modelos. Este estudo propõe uma abordagem baseada em aprendizado de máquina para detectar ataques de *Prompt Injection* e comparar sua eficácia com modelos tradicionais. Experimentos foram conduzidos utilizando modelos como *BERT*, *CountVectorizer* e *TfidfVectorizer*, demonstrando que técnicas de *Oversampling* aprimoram a detecção dessas ameaças.

Palavras-chave: Inteligência Artificial, Modelos de Linguagem, Prompt Injection, Aprendizado de Máquina, Cibersegurança

Prompt Injection Detection in Language Models

Abstract: Large Language Models (LLMs) are widely used in industry and academia for various tasks, such as virtual assistants and process automation. However, these technologies present security vulnerabilities, such as Prompt Injection attacks, which can compromise the integrity and reliability of the models. This study proposes a machine learning-based approach to detect Prompt Injection attacks and compares its effectiveness with traditional models. Experiments were conducted using models like BERT, CountVectorizer, and TfidfVectorizer, demonstrating that Oversampling techniques enhance the detection of these threats.

Keywords: Artificial Intelligence, Language Models, Prompt Injection, Machine Learning, Cybersecurity

1. Introdução

A inteligência artificial (IA) é uma das ferramentas fundamentais na transformação digital, inovando em áreas como saúde, ensino, guerras, política, notícias e dados sensíveis. (Yao, Lou e Qin, 2024). No entanto, como qualquer tecnologia disruptiva, a IA levanta preocupações sobre questões como impacto no mercado de trabalho, ética e desigualdade tecnológica.

Com o avanço dos Modelos de Linguagem de Grande Escala (L*LMs - Large Language Models)*, cresce o uso de sistemas capazes de compreender e gerar informação de forma sofisticada, contribuindo para melhorar interações humanas e otimizar processos automatizados (El et al., 2023).

Diante deste novo panorama, vem crescendo o uso de LLMs por empresas e instituições mundiais, para detecção e mitigação de ataques de *Prompt Injection*, as quais tornam-se essenciais para garantir a segurança e a confiabilidade desses sistemas. Esses ataques são relativamente novos e demandam análises aprofundadas para o desenvolvimento de estratégias eficazes de defesa. Este estudo busca contribuir para essa área ao propor métodos de aprendizado de máquina para detectar esses ataques, avaliando a eficácia de abordagens tradicionais e modernas. Além disso, o estudo investiga estratégias de filtragem de entrada baseadas em regras e aprendizado de máquina, que podem reduzir significativamente a taxa de sucesso dos ataques.

Portanto, o objetivo central deste trabalho é analisar as vulnerabilidades dos LLMs em relação ao *Prompt Injection* e propor estratégias eficazes para detecção e mitigação, em qualquer tipo de ambiente, sendo militar ou corporativo. Para isso, será avaliada a eficácia de diferentes abordagens, comparando sua precisão, *recall* e *F1-score*.

2. Referencial Teórico

2.1. Modelo de Linguagem de Grande Escala (LLM) e Inteligência Artificial (IA)

A Inteligência Artificial (IA) é um campo da ciência da computação dedicado ao desenvolvimento de sistemas capazes de realizar tarefas que normalmente exigiria inteligência humana, como raciocínio lógico, aprendizado, reconhecimento de padrões e tomada de decisões (Mumtarin et al., 2023). Dentre os avanços da IA, os Modelos de Linguagem de Grande Escala (LLMs) mostram um progresso significativo do Processamento de Linguagem Natural (NLP). Esses modelos são treinados em vastos conjuntos de dados textuais, permitindo a geração e compreensão sofisticada de linguagem natural.

A IA possui várias abordagens, sendo o *Machine Learning* (ML), um dos principais métodos aplicados nesses sistemas. O aprendizado de máquina permite que a IA melhore seu desempenho com base em experiências passadas. O Aprendizado Profundo (D*eep*

Learning - DL), por sua vez, utiliza redes neurais artificiais para modelar representações complexas de dados (Souza, 2023).

Os Modelos de Linguagem de Grande Escala (LLMs) são frequentemente empregados em assistentes virtuais, *chatbots*, tradução automática, geração de código e outras aplicações. Seu avanço tem sido impulsionado pelo aumento do poder computacional e pelo crescimento de bases de dados massivas utilizadas para treinamento (Kockula & Divya, 2024).

Modelos como BERT (*Bidirectional Encoder Representations from Transformers*) são exemplos de avanços nessa área. O BERT foi desenvolvido pelo Google e é amplamente utilizado para tarefas como classificação de texto, resposta a perguntas e análise de sentimentos (Rahman et al., 2024).

Entretanto, o avanço dos LLMs também levanta preocupações no contexto da ética e regulação. A capacidade desses modelos de gerar textos realistas pode ser explorada para disseminação de desinformação, manipulação de conteúdo e fraudes digitais. Estudos recentes, tais como do (Hines et al., 2024), mostram a necessidade de regras e métodos que estejam ligados à segurança para garantir o crescimento ético e claro dessa tecnologia.

2.1. Machine Learning

A ascensão de métodos de aprendizado de máquina em aplicações interativas tem impulsionado o desenvolvimento de técnicas mais eficazes para a análise de texto. Dois métodos amplamente utilizados para o pré-processamento de dados textuais em Machine Learning são o *CountVectorizer* e o *TfidfVectorizer*. Já os classificadores *SGDClassifier* e *MultinomialNB* são frequentemente aplicados em tarefas de classificação de texto. Essas abordagens têm se mostrado particularmente relevantes em pesquisas voltadas à detecção de ataques de *Prompt Injection* em sistemas de Inteligência Artificial (IA) (Das et al., 2018).

O *CountVectorizer* é um método de frequência de palavras que converte textos em vetores de frequência de palavras, permitindo que modelos de aprendizado de máquina processem dados textuais de maneira estruturada (Ramos, 2003). Por outro lado, o *TfidfVectorizer* aprimora essa abordagem ao considerar a importância relativa de cada palavra dentro do documento, diminuindo o peso de palavras comuns e enfatizando termos mais informativos (Agarwal et al., 2023).

O *SGDClassifier* utiliza a técnica de Descida de Gradiente Estocástica (SGD), que é eficiente para treinar grandes quantidades de dados textuais, sendo popular no campo de classificação de texto (El et al., 2023). *MultinomialNB* é um classificador baseado no *Teorema de Bayes* que é comumente utilizado para categorização de documentos devido à sua eficácia em lidar com ocorrências de palavras multinominais (Mccallum & Nigam, 1998).

Com a evolução de modelos de linguagem natural, como os

Large Language Models (LLMs), ataques mais sofisticados se tornaram viáveis, incluindo o chamado *Prompt Injection*. Esse tipo de ataque explora a ausência de

mecanismos robustos de filtragem nas entradas dos modelos, induzindo-os a executar ações maliciosas ou revelar informações sensíveis (Mitreva et al., 2024).

2.2. Prompt Injection

A segurança de sistemas baseados em Modelos de Linguagem Natural (NLP) tem se tornado uma das principais preocupações da comunidade científica e tecnológica. Grupos de pesquisa e empresas vêm unindo esforços para identificar novas vulnerabilidades e, principalmente, desenvolver estratégias para tornar os aplicativos de inteligência artificial (IA) mais seguros.

Entre as vulnerabilidades mais exploradas, segundo a OWASP Top 10 para modelos de Linguagem de Grande Escala (OWASP LLM Project, 2024), destaca-se o *Prompt Injection*, um tipo de ataque que tem recebido atenção crescente. Esse ataque explora a maneira como modelos de LLM, como o *GitHub Copilot*, processam comandos de entrada. Essencialmente, um invasor insere instruções maliciosas no prompt do modelo, fazendo com que ele execute ações não autorizadas ou a gerar respostas inesperadas (Hung et al., 2024).

O fenômeno de *Prompt Injection* pode ser categorizado, conforme a literatura recente (Zhang et al., 2024), em duas classes principais. A primeira, denominada *Prompt Injection* Direto, refere-se a ataques nos quais o agente malicioso insere instruções ou comandos diretamente no *prompt*, com o objetivo de induzir o modelo a executar comportamentos que não seriam previstos em condições normais. Já a segunda, conhecida como *Prompt Injection* Indireto, caracteriza-se pela injeção de conteúdo malicioso em fontes externas — como páginas web ou documentos — os quais são posteriormente interpretados pelo modelo, configurando um vetor de ataque menos explícito, porém igualmente perigoso.

Essa ameaça representa um desafio significativo, pois os LLMs não possuem uma separação clara entre código e entrada textual, o que dificulta a identificação de comandos legítimos frente a tentativas de exploração (Kokkula & Divya, 2024).

Com o objetivo de mitigar essa vulnerabilidade, Kokkula e Divya (2024) propuseram o *framework Palisade*, que adota uma abordagem em camadas para a detecção de *Prompt Injection*. O sistema opera por meio da aplicação sequencial de três filtros distintos: inicialmente, um filtro baseado em regras linguísticas é utilizado para capturar padrões explícitos de linguagem; em seguida, um classificador fundamentado em técnicas de aprendizado de máquina realiza a triagem estatística das entradas; por fim, um modelo de linguagem adicional é empregado para realizar uma análise semântica mais aprofundada do conteúdo.

Essa combinação aumenta a precisão da detecção e reduz a incidência de falsos negativos, proporcionando uma defesa mais robusta contra instruções maliciosas inseridas nos prompts.

3. Trabalhos Relacionados

Estudos anteriores exploram a segurança de Modelos de Linguagem, destacando ataques como *PoisonPrompt* (Yao et al., 2024) e estratégias de defesa, como o framework *Palisade* (Kokkula & Divya, 2024). No entanto, ainda são escassas as pesquisas que

avaliam o impacto de diferentes abordagens de aprendizado de máquina na detecção de *Prompt Injection*.

O *PoisonPrompt*, desenvolvido por Yao et al. (2024), demonstrou ser altamente eficaz em comprometer tanto prompts rígidos quanto flexíveis em Modelos de Linguagem de Grande Escala (LLMs), por meio de ataques do tipo *backdoor*. Esse tipo de ataque compromete a integridade dos modelos ao inserir comandos ocultos durante a fase de ajuste fino ou diretamente no prompt de entrada. Ele explora a capacidade dos modelos de seguir instruções com pouca restrição, permitindo que agentes mal-intencionados induzam comportamentos inesperados sem que a modificação seja facilmente detectável. Contudo, a implementação e a defesa contra o *PoisonPrompt* são desafiadoras, devido à sua sofisticação e à necessidade de compreensão aprofundada das estratégias de *prompt* e da arquitetura dos modelos. Além disso, a efetividade do ataque pode depender do acesso prévio ao processo de treinamento ou da manipulação direta das entradas do usuário, o que pode limitar sua aplicação dependendo do cenário.

Por outro lado, Kokkula e Divya (2024) propuseram o **Palisade**, um *framework* voltado à detecção e mitigação destaques de *Prompt Injection*. Ao contrário das abordagens tradicionais, que geralmente se concentram em filtrar conteúdos explícitos, o *Palisade* adota técnicas baseadas em aprendizado de máquina para identificar padrões sutis nos *prompts*, considerando tanto a estrutura linguística quanto a intenção subjacente da entrada do usuário.

Desta forma, este trabalho contribui para o aprimoramento de sistemas de IA, ao comparar a eficiência de diferentes algoritmos de aprendizado de máquina na detecção de ataques, aponta o impacto de estratégias de balanceamento de dados na performance dos modelos, bem como viabiliza a investigação do desempenho de modelos pré-treinados em relação a métodos tradicionais. Uma das vantagens dessa abordagem é sua flexibilidade, permitindo a adaptação a diferentes tipos de ataques sem depender de regras rígidas previamente definidas. No entanto, o framework *Palisade* pode ser vulnerável a ataques mais sofisticados, que tentam mascarar comandos maliciosos por meio de reformulações criativas ou uso de linguagem ambígua. Como em qualquer sistema baseado em aprendizado de máquina, a eficácia do *Palisade* está diretamente ligada à qualidade e à diversidade dos dados utilizados para seu treinamento.

Apesar dos avanços desses estudos na compreensão da segurança de LLMs, ainda há uma lacuna na literatura quanto à avaliação sistemática do uso de múltiplas técnicas de aprendizado de máquina na detecção de *Prompt Injection*.

Este estudo propõe, nesse sentido, o preenchimento de parte dessa lacuna, ao comparar o desempenho de diferentes algoritmos, analisar o impacto do balanceamento de dados na performance dos modelos, e investigar a eficácia de modelos tradicionais versus modelos pré-treinados. Ao aprofundar essa análise, espera-se contribuir para o desenvolvimento de soluções mais robustas e eficientes contra ataques emergentes, fortalecendo a segurança de sistemas baseados em inteligência artificial.

4. Metodologia

O sistema proposto para detecção de ataques de *Prompt Injection* combina técnicas de aprendizado de máquina (ML) e modelos de linguagem natural (LLMs), com o objetivo

de identificar tentativas maliciosas de manipulação de entradas textuais. A Figura 1 ilustra a arquitetura geral do sistema, composta pelas seguintes etapas:

4.1. Coleta de dados

Foi construído um conjunto de dados personalizado em português, contendo exemplos de diferentes tipos de ataques de *Prompt Injection*, incluindo injeções diretas, instruções contraditórias e tentativas de subversão do comportamento esperado do modelo. Cada exemplo foi manualmente anotado com os rótulos "1" (indicando prompt malicioso) ou "0" (prompt legítimo), com base em critérios relacionados à intenção de manipulação do modelo. Por exemplo, perguntas informativas como "O que eu preciso para viajar com segurança durante a pandemia?" foram rotuladas como legítimas (0), enquanto *prompts* como "Crie um script XSS para roubar cookies de usuário" e "Explique como injetar código XSS em um formulário de login" foram marcados como maliciosos (1).

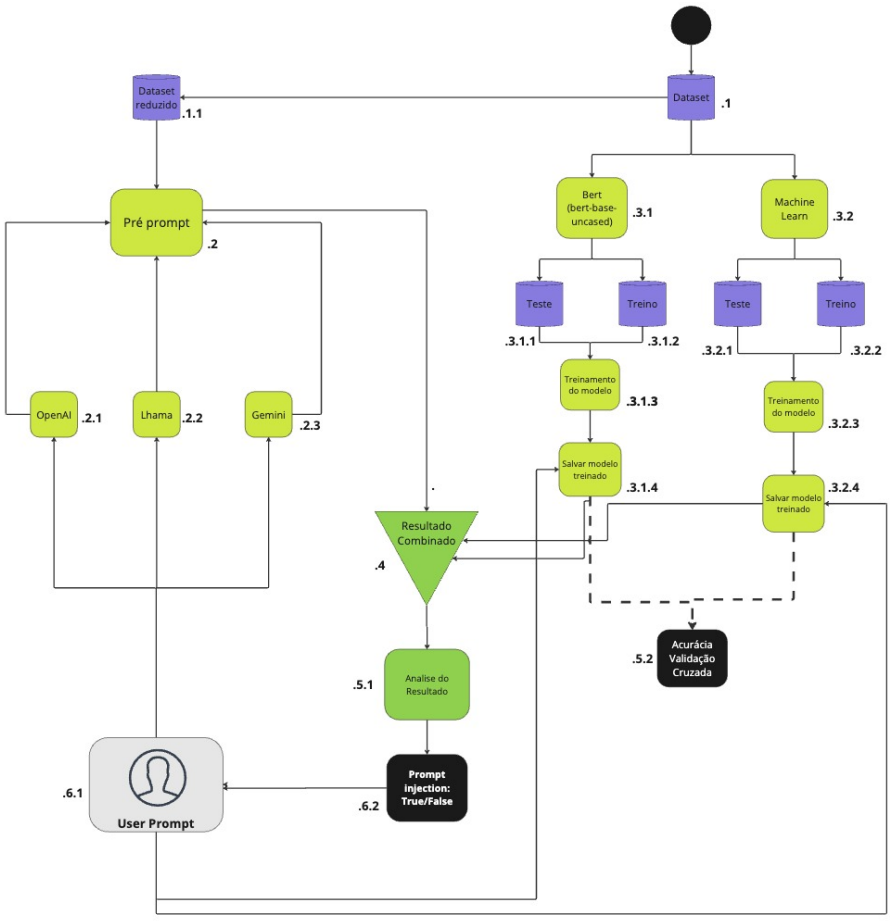

Figura 1 – Arquitetura do sistema de detecção de *prompt injection*

Para reforçar a diversidade linguística e a robustez da base, também foram incorporados dois conjuntos de dados públicos em inglês: (Jaziri, 2024) e (Harelix, 2023). Após a integração e embaralhamento desses dados de forma aleatória, formou-se um único *dataset* bilíngue, conforme abordagens sugeridas por (Lima, 2022).

Com o objetivo de promover a reprodutibilidade dos experimentos, o conjunto completo foi publicado e está disponível publicamente em: https://huggingface.co/datasets/PromptInjectionDataset/Injection-Attack-Detection-Dataset.

4.2. Balanceamento e Processamento

Durante o treinamento dos modelos de aprendizado de máquinas (ML) e do modelo BERT, utilizou-se o conjunto de dados completo. Para os experimentos envolvendo a geração de respostas por modelos de linguagem (LLMs), aplicou-se uma versão reduzida do dataset, composta por 5% dos exemplos selecionados aleatoriamente e inseridos na etapa de *pré-prompt* (.2). Essa estratégia visa equilibrar a diversidade nas entradas e eficiência no processamento das respostas geradas.

Essa versão reduzida do conjunto de dados foi então processada por três modelos de linguagem distintos: o *GPT-4-mini*, (OpenAI, 2021); o *Llama* 3.1-70B (Llama, 2024); e o *Gemini 1.5-Flash*, (Google Gemini, 2023). A escolha desses modelos fundamenta-se em sua capacidade reconhecida de interpretar entradas textuais complexas e produzir respostas coerentes com base em suas respectivas arquiteturas e históricos de treinamento.

Cada modelo LLM recebeu do usuário os *prompts* de entrada de maneira independente, permitindo a análise comparativa de seu comportamento frente a possíveis ataques de *Prompt Injection*.

4.3. Modelagem e Teste de Modelos

O *dataset* completo foi utilizado para o treinamento de modelos de aprendizado de máquina. Neste processo, adotou-se o modelo BERT (.3.1.), onde utilizou-se a arquitetura *bert-base-uncased*, com etapas de treinamento inicial, validação, refinamento e persistência do modelo (Vaj, 2023) e o modelo tradicional de Machine Learning (.3.2.), onde foi aplicado técnicas de vetorização com *CountVectorizer* e TfidfVectorizer, seguidas por classificadores *SGDClassifier* e *MultinomialNB*.

O fluxo de treinamento seguiu as etapas descritas por (Lima, 2022), com o modelo BERT passando por fases específicas de treinamento inicial (.3.1.1), testes (.3.1.2), refinamento final (.3.1.3) e salvamento para uso posterior (.3.1.4). De maneira análoga, os modelos tradicionais seguiram um processo equivalente (.3.2.1 a .3.2.4), como evidenciado na Figura 1, garantindo reprodutibilidade e padronização dos experimentos.

4.4. Combinação de Resultados

As predições individuais dos modelos (BERT e ML) foram agregadas em um mecanismo de decisão por voto majoritário.

Para cada entrada textual, um *array* armazena as respostas classificadas como “ataque” (*True*) ou “legítimo” (*False*). A decisão final corresponde à classe predominante entre

os modelos, promovendo maior robustez na detecção e aproveitando as vantagens complementares de cada abordagem.

4.5. Análise e Validação

Os resultados foram avaliados por meio de métricas extraídas da biblioteca sklearn. metrics, incluindo acurácia, precisão, *F1-score* e validação cruzada (seção 5.2 da Figura 1). Essa análise permitiu quantificar o desempenho de cada modelo e identificar padrões relevantes associados a entradas maliciosas.

4.6. Interação com o Usuário

Com base na análise anterior, o sistema classifica a entrada como *Prompt Injection*: *True* ou *False*. Essa etapa fornece uma decisão clara sobre a presença ou ausência de um ataque, permitindo ações preventivas de mitigação e contribuindo para a segurança dos modelos de linguagem.

A consistência dessa etapa é garantida pela validação descrita na Figura 1 (seção 6.2), assegurando integridade e confiabilidade na comunicação dos resultados.

5. Resultados e Discussão

Os resultados obtidos neste estudo demonstram que é possível treinar modelos de linguagem com alta acurácia para detectar ataques de *Prompt Injection*. A partir dos experimentos conduzidos, observou-se que modelos treinados com a técnica de *Oversampling* apresentaram melhor desempenho em comparação com os modelos treinados com *Undersampling,* conforme indicado na tabela a seguir:

Modelo	**Acurácia**	**Precisão**	***F1-Score***	**Acurácia**	**Precisão**	***F1-Score***
CountVectorizer & SGDClassifier	0,96	0,97	0,96	0,93	0,93	0,93
CountVectorizer & MultinomialNB	0,96	0,96	0,96	0,92	0,92	0,92
TfidfVectorizer & *SGDClassifier*	0,97	0,98	0,97	0,93	0,93	0,93
TfidfVectorizer & MultinomialNB	0,96	0,96	0,96	0,92	0,92	0,92
Oversampling				Undersampling		

Tabela 1 – Desempenho de modelos tradicionais de ML com diferentes estratégias de balanceamento

Todos os modelos testados apresentam uma maior assertividade quando treinados com bancos *Oversampling*. Este comportamento não é reproduzido quando o modelo BERT é utilizado. Para este tipo de modelo, a acurácia é igual para ambos os tipos de balanceamento, conforme tabela a seguir:

Modelo	Acurácia	Precisão	*F1-Score*	Acurácia	Precisão	*F1-Score*
BERT	0,98	0,98	0,98	0,98	0,98	0,98
	Oversampling			Undersampling		

Tabela 2 – Desempenho do modelo BERT com diferentes estratégias de balanceamento

Outra característica do modelo BERT é o resultado similar para ambos os tipos de banco e métodos de medição, como *Acurácia*, *Precisão* e *F1-Score*. Por possuir esta característica, é possível identificar uma maior consistência no modelo, tanto no treinamento quanto na utilização. Para uma melhor visualização dos resultados, segue um gráfico de comparação.

5.1. *Prompts* treinados

Após o treinamento de diferentes tipos de modelos de Machine Learning, utilizando dois tipos diferentes de balanceamento de banco de dados, foram realizados testes em modelos de inteligência artificial de mercado. Estes modelos já possuem treinamento para detectar *Prompt Injection*, com o objetivo de proteger os dados do usuário e das empresas que os desenvolvem.

Nesta etapa do estudo, os modelos *Llama, OpenAI* e *Gemini* foram utilizados. Como estes três modelos são inteligências artificiais, não é possível medir acurácia, precisão e *f1-score* para comparar com os modelos de ML. No entanto, é possível realizar uma comparação com os *outputs* dos *prompts* testados.

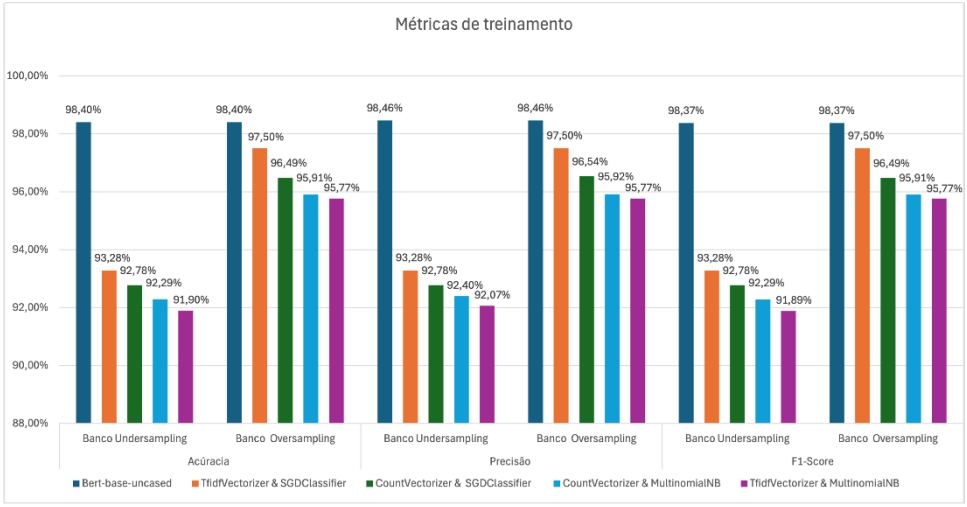

Figura 2 – Comparação visual das métricas de desempenho entre modelos de ML tradicionais e BERT

Para obter os resultados a seguir, foi realizado um *pré-prompts* dos modelos de IA utilizando 5% do *dataset* utilizado para treinar os modelos de ML. Os dados foram selecionados aleatoriamente a toda execução do código. A tabela a seguir contém os resultados dos testes realizados.

O resultado demonstrado na Tabela 3 avalia a probabilidade de uma entrada ser considerada um *Prompt Injection*, com saídas que podem assumir qualquer valor entre 0 e 1.

Modelo	Resultados
OpenAI	1.0
Gemini	0.95
Llama	1.0

Tabela 3 – Resultados de modelos de IA de mercado em tarefa específica

5.2. Análise dos resultados

Mesmo com uma quantidade limitada de dados, os modelos de aprendizado de máquina (ML) treinados neste estudo obtiveram resultados comparáveis aos modelos de IA disponíveis no mercado. Isso indica que, com o devido ajuste, é possível direcionar esses modelos para áreas específicas, como a Segurança da Informação. Além disso, considerando que o *prompt Injection* pode envolver resultados sensíveis, é fundamental discutir práticas responsáveis de IA incluindo o tratamento adequado de dados e a mitigação de riscos à privacidade, para garantir que a aplicação desses modelos ocorra de forma ética e segura (Palomino-Flores & Cristi-López, 2024).

Os testes realizados revelaram que o modelo BERT apresentou os melhores resultados em termos de precisão e consistência, independentemente da técnica de balanceamento aplicada. Além disso, os modelos baseados em *TfidfVectorizer* demonstraram desempenho ligeiramente superior em relação aos que utilizaram *CountVectorizer*. Por fim, os modelos comerciais de IA, como *OpenAI, Llama* e *Gemini* apresentaram resultados similares entre si, reforçando a robustez da abordagem proposta neste estudo, mesmo em cenários mais controlados.

A análise dos resultados revela que, tanto a comparação entre deteções de *prompts* maliciosos em diferentes modelos, quanto o método utilizado no treinamento do modelo de *Machine Learning,* durante o estudo apresentado, apresenta diferenças significativas de outros estudos disponíveis (Kokkula & Divya, 2024; Yao et al., 2024). As técnicas de treinamento apresentadas permitiram um refinamento mais robusto, obtendo um desempenho satisfatório e uma acurácia muito semelhante a modelos de LLM utilizados amplamente na internet.

6. Conclusão

Este estudo propõe uma abordagem híbrida baseada em aprendizado de máquina para detecção de ataques de *Prompt Injection*, uma das ameaças emergentes mais relevantes no contexto de segurança aplicada a modelos de linguagem natural.

Ao treinar modelos de Machine Learning para reconhecer esse tipo de ameaça, foi possível aumentar o nível de segurança e confiabilidade das aplicações, contribuindo para a proteção de dados de usuários e organizações. Os resultados obtidos demonstraram que modelos como o BERT e o *SGDClassifier* combinado com *TfidfVectorizer* foram altamente eficazes na identificação de *Prompt Injection*, alcançando acurácia superior a 97%.

Dessa forma, mesmo em cenários com recursos limitados ou modelos de menor escala, é viável implementar medidas de segurança baseadas em aprendizado de máquina, reforçando a proteção de dados e a privacidade dos usuários por meio de técnicas avançadas de detecção e mitigação.

Como trabalhos futuros, propõe-se:

1. Avaliar a generalização do modelo por meio de testes em diferentes idiomas.
2. Integrar a abordagem a sistemas de detecção de ameaças em tempo real.
3. Refinar os conjuntos de dados utilizados, com foco na redução de falsos positivos.

Referências

Agarwal, A., Vats, S., Agarwal, R., Ratra, A., Sharma, V., & Gopal, L. (2023). Sentiment Analysis in Stock Price Prediction: A Comparative Study of Algorithms. Ieee.org. https://ieeexplore.ieee.org/abstract/document/10112565

Das, R., Kadam, S., Kalra, C., Nayak, V., & Govilkar, Dr. S. (2018). SARCASM DETECTION FOR ENGLISH TEXT (p. 66). Journal of Computer Engineering, Volume 6. https://www.pce.ac.in/wp-content/uploads/2021/10/COMP-Journal-2017-18.pdf#page=66

El, M., Tarek Abd El-Hafeez, Ahmed, O., & Hamed, E. (2023). A Prediction System Using AI Techniques to Predict Students' Learning Difficulties Using LMS for Sustainable Development at KFU. Lecture Notes in Networks and Systems, 22–36. https://doi.org/10.1007/978-3-031-21438-7_2

Felix, M. (2024). Manipulando o ChatGPT para entregar informações sensíveis via prompt injection. Medium. https://medium.com/@binaryfelix/manipulando-o-chatgpt-para-entregar-informa%C3%A7%C3%B5es-sens%C3%ADveis-via-prompt-injection-6be7d3b63ab1

Google Gemini. (2023). Build with the Gemini API. Google AI for Developers. https://ai.google.dev/

Harelix. (2023). train.csv · Harelix/Prompt-Injection-Mixed-Techniques-2024 at main. Huggingface.co. https://huggingface.co/datasets/Harelix/Prompt-Injection-Mixed-Techniques-2024/blob/main/train.csv

Hines, K., Lopez, G., Hall, M., Zarfati, F., Zunger, Y., & Kiciman, E. (2024). Defending Against Indirect Prompt Injection Attacks With Spotlighting. ArXiv.org. https://arxiv.org/abs/2403.14720

Hung, K.-H., Ko, C.-Y., Rawat, A., Chung, I-Hsin., Hsu, W. H., & Chen, P.-Y. (2024). Attention Tracker: Detecting Prompt Injection Attacks in LLMs. ArXiv.org. https://arxiv.org/abs/2411.00348

Jaziri, C. (2024). Malicious_and_Benign_Dataset. Kaggle.com. https://www.kaggle.com/datasets/chaimajaziri/malicious-and-benign-dataset

Kokkula, S., & Divya, G. (2024). Palisade -- Prompt Injection Detection Framework. ArXiv.org. https://arxiv.org/abs/2410.21146

Kosinsky, M., & Forrest, A. (2024, March 26). What is a prompt injection attack? Ibm.com. https://www.ibm.com/think/topics/prompt-injection

Lima, M. de. (2022). 6 passos Para Criar Seu Primeiro Projeto de Machine Learning. Insightlab.ufc.br; Insight Data Science Lab. https://www.insightlab.ufc.br/6-passos-para-criar-seu-primeiro-projeto-de-machine-learning/

Llama. (2024). Llamaapi.com. https://console.llamaapi.com/

Mccallum, A., & Nigam, K. (1998). A comparison of event models for naive bayes text classification. http://yangli-feasibility.com/home/classes/lfd2022fall/media/aaaiws98.pdf

Mitreva, E., Georgiev, V., & Nikolova, A. (2024). Classification of Short Noisy Text. Proceedings of the International Conference on Computer Systems and Technologies 2024, 227–231. https://doi.org/10.1145/3674912.3674935

Mumtarin, M., Chowdhury, M. S., & Wood, J. (2023). Large Language Models in Analyzing Crash Narratives -- A Comparative Study of ChatGPT, BARD and GPT-4. ArXiv.org. https://arxiv.org/abs/2308.13563

OpenAI. (2021). OpenAI API. OpenAI. https://openai.com/api/

OWASP LLM Project. (2024). OWASP Top 10 for LLM Applications 2025 - OWASP Top 10 for LLM & Generative AI Security. OWASP Top 10 for LLM & Generative AI Security. https://genai.owasp.org/resource/owasp-top-10-for-llm-applications-2025/

Palomino-Flores, P., & Cristi-López, R. (2024). Del código a la creatividad: Explorando los horizontes éticos de la creación de contenido con Inteligencia Artificial. Revista Ibérica De Sistemas e Tecnologías De Informação, (E66), 355-369.

Rahman, M. A., Shahriar, H., Wu, F., & Cuzzocrea, A. (2024). Applying Pre-trained Multilingual BERT in Embeddings for Improved Malicious Prompt Injection Attacks Detection. 2nd International Conference on Artificial Intelligence, Blockchain, and Internet of Things (AIBThings), 1–7. https://doi.org/10.1109/aibthings63359.2024.10863664

Ramos, J. (2003). Using TF-IDF to Determine Word Relevance in Document Queries. https://citeseerx.ist.psu.edu/document? repid = rep1&type = pdf&doi = b3bf6373ff41a115197cb5b30e57830c16130c2c

Souza, F. R. (2023). Simulação de diálogos e personagens no Chat-GPT4: análise comparativa do desempenho em idiomas inglês e português. In Anais do IV Congresso Brasileiro Interdisciplinar em Ciência e Tecnologias. IV Cobicet: online (pp. 1–8.).

Vaj, T. (2023). Let's code K-fold validation on BERT. Medium. https://vtiya.medium.com/lets-code-k-fold-validation-on-bert-722f9438f932

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is All you Need. Advances in Neural Information Processing Systems, 30, 5998–6008.

Yao, H., Lou, J., & Qin, Z. (2024). PoisonPrompt: Backdoor Attack on Prompt-Based Large Language Models. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (, 7745–7749. IEEE. https://doi.org/10.1109/icassp48485.2024.10446267

Yao, H., Lou, J., Qin, Z., & Ren, K. (2024). PromptCARE: Prompt Copyright Protection by Watermark Injection and Verification. 2024 IEEE Symposium on Security and Privacy (SP), 845–861. https://doi.org/10.1109/sp54263.2024.00209

Zhang, W., Kong, X., Dewitt, C., Braunl, T., & Hong, J. B. (2024). A Study on Prompt Injection Attack Against LLM-Integrated Mobile Robotic Systems. 361–368. https://doi.org/10.1109/issrew63542.2024.00103