Detecção de URLs de Phishing com PU Learning e Métricas de Divergência de Distribuições

Davi C. Ribeiro
dcr23@inf.ufpr.br
Universidade Federal do Paraná
Curitiba, Paraná, Brasil

Bernardo Tomasi
bpt22@inf.ufpr.br
Universidade Federal do Paraná
Curitiba, Paraná, Brasil

Ruibin Mei
rm23@inf.ufpr.br
Universidade Federal do Paraná
Curitiba, Paraná, Brasil

Pedro Henrique Friedrich Ramos
phfr24@inf.ufpr.br
Universidade Federal do Paraná
Curitiba, Paraná, Brasil

Yago Yudi Furuta
yyvf22@inf.ufpr.br
Universidade Federal do Paraná
Curitiba, Paraná, Brasil

João Pincovscy
pincovscy@gmail.com
Centro de Pesquisa e Desenvolvimento para a Segurança das Comunicações
Brasília, Distrito Federal, Brasil

Laerte Peotta
peotta@gmail.com
Universidade de Brasília
Brasília, Distrito Federal, Brasil

André R. A. Grégio
gregio@inf.ufpr.br
Universidade Federal do Paraná
Curitiba, Paraná, Brasil

Abstract

In this article, we propose a machine learning-based solution to detect phishing URLs in the context of Positive-Unlabeled (PU) Learning. Our solution combines lexical features and statistical attributes extracted from characters and bigram distributions. To that end, we applied distance and divergence metrics to the average distribution of 25 million unlabeled URLs in order to identify suspicious patterns. In the experiments, we evaluated classical (e.g., Random Forest) and deep learning (e.g., CNN, MLP, CNNLSTM, GRU) models, validating the robustness of statistical features. The tests performed presented accuracy rates of up to 94,32%, highlighting the influence of the language in the distributions: models adapted to Portuguese performed 20% better than those trained with English data. Our approach overcomes traditional phishing detection methods based on blocklists, thus reducing the vulnerability window and fostering the effective protection of computer systems users in general.

Keywords

Phishing, URL, Deep Learning, Bigramas

1 Introdução

O *phishing* é um ataque cibernético de engenharia social que explora falhas humanas para induzir o compartilhamento de informações sensíveis (senhas, dados bancários) ou a instalação de *malware*. Apesar de ser uma técnica antiga, mantém-se eficaz devido ao baixo custo, alto retorno financeiro e contínua evolução. Os vetores mais comuns são e-mail, SMS ou redes sociais, onde são enviadas URLs falsas que imitam *sites* institucionais ou financeiros, oferecendo "atualizações" de sistema, promoções enganosas ou solicitações de confirmação de dados.

Em 2024, foram registrados cerca de 4 milhões de ataques de *phishing* [1], com crescimento de 58% em relação a 2023, atribuído ao uso de serviços legítimos em campanhas maliciosas. Estima-se que esses ataques causaram um impacto financeiro global de US$ 3,5 bilhões [2].

Tradicionalmente, a detecção de URLs de *phishing* se baseia em *blocklists*, que são conjuntos de endereços maliciosos conhecidos, semelhantes aos bancos de dados de *hashes* utilizados por programas antivírus para identificar *malwares*. Quando um usuário tenta acessar uma URL, ela é comparada a essa lista. Caso haja correspondência, o acesso é bloqueado antes do carregamento da página. Essa abordagem, embora simples e de baixa latência, depende exclusivamente da atualização prévia das entradas na lista.

Porém, a efetividade das *blocklists* esbarra em uma limitação crítica, pois a janela de tempo entre o surgimento de uma nova URL de *phishing* e o momento em que ela é identificada e incluída na lista de bloqueio resulta em uma lacuna de proteção, durante a qual os usuários permanecem suscetíveis ao ataque. Em [3], observou-se que a detecção inicial por entidades *anti-phishing* ocorre em média 9 horas após o primeiro acesso de uma vítima, e o bloqueio efetivo (por navegadores ou filtros em larga escala) demora mais 7 horas, totalizando uma janela média de 16 horas em que o atacante opera sem barreiras. Esse intervalo de tempo entre o início do ataque até a sua mitigação é preocupante, visto que o mesmo estudo verificou que a duração média de uma campanha de *phishing*, do primeiro até o último acesso de uma vítima, é de aproximadamente 21 horas.

Diante desse cenário, classificadores baseados em aprendizado de máquina revelam-se como estratégias de mitigação promissoras, capazes de eliminar a necessidade de atualizações manuais de listas de bloqueio. Os modelos podem aprender padrões sintáticos, tais como a quantidade de subdomínios e a profundidade do caminho (*path*), além de semânticos, como *tokens* que indicam redirecionamentos ou termos suspeitos, a partir de um grande volume de URLs. Neste trabalho, apresenta-se as seguintes contribuições:

- Foi feito um conjunto de experimentos abrangente que culminou na extração tanto características léxicas clássicas

(como presença de IP ou porta, comprimento e profundidade do caminho) quanto atributos baseados em distribuições de caracteres e bigramas. A partir dessas distribuições, foram calculadas as distâncias (Euclidiana, Manhattan, Chebyshev), testes de aderência e divergência (Kolmogorov–Smirnov, Kullback–Leibler) e o custo da árvore de Huffman, a partir de um conjunto de 25 milhões de URLs não rotuladas.
- Foi comparado o desempenho (eficácia) de um classificador convencional (*Random Forest*) com quatro arquiteturas de *deep learning* — CNN, MLP, CNNLSTM e GRU — usando técnicas de *PU Learning*. Para tanto, realizou-se a busca exaustiva de hiperparâmetros e avaliou-se cada modelo em termos de acurácia, além de ter-se investigado a contribuição individual e sinérgica de cada *feature* na tarefa da classificação.
- Por fim, produziu-se um *dataset* original, contendo cerca de 1 milhão e meio de URLs de *phishing* coletadas entre 2013 e 2025 a partir de um servidor *MISP Threat Sharing* em nosso laboratório de pesquisa e de dados fornecidos pelo Banco do Brasil, que foram combinados com URLs obtidas do *Common Crawl*, para a porção de URLs não rotuladas do *dataset*.

2 Trabalhos Relacionados

A detecção de URLs de *phishing* tem sido amplamente explorada na literatura, com abordagens que variam desde métodos baseados em listas de bloqueio (*blocklists*) até técnicas avançadas de aprendizado de máquina. Esta seção discute trabalhos relevantes, bem como suas contribuições e limitações, e os compara com a proposta apresentada neste artigo.

Abordagens Baseadas em Blocklists e Heurísticas. Tradicionalmente, a detecção de *phishing* depende de *blocklists*, como as mantidas pelo Google Safe Browsing e PhishTank. Embora eficazes para URLs conhecidas, essas listas sofrem com a "janela de vulnerabilidade", o intervalo entre o surgimento de uma nova URL maliciosa e sua inclusão na lista. [3] mostraram que essa janela pode durar até 16 horas, tempo suficiente para que ataques causem danos significativos. Métodos heurísticos, como os propostos por Zhang et al. [4] tentam minimizar esse tipo de problema através da combinação das *blocklists* com características léxicas. Entretanto, a abordagem depende de atualizações manuais. [5] vão na direção de criar uma lista seleta de domínios permitidos (*allowlist*) a fim de validar domínios registrados e remover domínios e URLs benignas de listas de bloqueio. Com isso, alcança-se *blocklists* com maior acurácia, minimizando a quantidade de falsos-positivos. [6] investigam quão transparentes, consistentes e dinâmicas são as *blocklists* de código aberto usadas para segurança (e.g., filtros de *phishing*, *malware*) e quais os impactos práticos de suas divergências e atualizações para a proteção de usuários, mostrando variações nas *blocklists*. O trabalho também analisou 6 blocklists populares, tais como Spamhaus, OpenPhish e PhishTank, revelando que apenas 40% das URLs maliciosas são listadas em mais de uma fonte. Os autores então cunham o termo "Efeito Babel" para explicar as discrepâncias entre listas que criam falsos negativos (URLs maliciosas não bloqueadas) e falsos positivos (URLs legítimas erroneamente bloqueadas). Uma das principais conclusões é que se deve utilizar modelos inteligentes e automáticos (baseados em aprendizado de máquina), para superar as limitações das *blocklists*, como os atrasos causados pela dependência de atualizações por humanos. [7] propõem heurísticas de rede para tentar detectar URLs de *phishing* em cenários de evasão (i.e., ofuscação, redirecionamentos, domínios novos). A abordagem proposta inclui a monitoração em tempo real de conexões HTTP/HTTPS, padrões de DNS e relações entre domínios para identificar comportamentos suspeitos; modelagem das características inspirada em grafos, ao considerar a interação entre URLs, IPs e ASNs (*Autonomous Systems*) para agrupar domínios maliciosos; enfoque nos metadados da rede em vez do conteúdo da URL. Porém, tal abordagem sofre de falsos-positivos (devido a domínios legítimos com comportamentos atípicos) e da necessidade de coleta de muitos dados (*logs* de rede e fluxos).

Métodos de Aprendizado de Máquina Supervisionado. Em problemas reais de classificação supervisionada, é comum que apenas uma parte dos dados esteja rotulada, especialmente quando o custo de rotular exemplos negativos é muito alto ou inviável. Um cenário comum envolve a presença de apenas exemplos positivos rotulados e uma grande quantidade de exemplos não rotulados, os quais podem pertencer tanto à classe positiva quanto à negativa. Esse problema é tratado pela técnica conhecida como *Positive-Unlabeled (PU) Learning*. Outro estudo relevante é apresentado o de [8], que investiga a detecção de URLs de *phishing* por meio de métodos de aprendizado de máquina supervisionado. Nesse trabalho, diferentes algoritmos, como *Random Forest*, *Decision Tree*, *LightGBM*, Regressão Logística e *SVM*, são aplicados para classificar URLs como benignas ou maliciosas. O processo envolve a extração de 15 características baseadas no conteúdo do endereço e informações de domínio, como presença de IP, comprimento da URL, existência de caracteres especiais e dados de registro *WHOIS*. Os autores observam que o *LightGBM* apresentou o melhor desempenho, alcançando uma acurácia de 86% nos dados de teste. Este estudo reforça a importância de uma boa seleção de *features* e do uso de múltiplos modelos para comparação de desempenho, mas também destaca limitações como a falta de cobertura de todos os domínios no banco *WHOIS* e a necessidade de expandir as bases de dados para melhor generalização dos modelos. Já [9] propuseram a extração de bigramas como características, mas a alta dimensionalidade causa impactos severos à eficiência computacional. [10] busca entender como técnicas de aprendizado de máquina podem ser utilizadas para detectar automaticamente *sites* Web envolvidos em *phishing* com alta precisão, superando métodos tradicionais como *blocklists*. A abordagem supera *blocklists* em dinamismo, mas requer atualização contínua dos modelos para enfrentar táticas evolutivas de atacantes. Assim como o presente trabalho, também se obteve melhores resultados com *Random Forests*, porém ao contrário do proposto neste artigo, a limitação do *dataset*, principalmente em questão de desbalanceamento, prejudica a qualidade dos modelos.

Positive-Unlabeled (PU) Learning. [4] propõem uma abordagem baseada em *PU Learning* para detecção de URLs maliciosas. O método é dividido em quatro etapas principais: (1) extração de um vetor de *features* para representar as URLs; (2) identificação de um subconjunto do conjunto de dados como negativos confiáveis (*reliable negatives*), que é então utilizado para treinar um modelo supervisionado inicial; (3) tratamento do restante das amostras

não rotuladas como negativas, com o objetivo de refinar o modelo por meio de um novo treinamento; e (4) aplicação do modelo treinado para realizar as predições finais. No entanto, o artigo foca em aspectos arquiteturais, sem se aprofundar na discussão de quais características extrair de uma URL. [11], por sua vez, argumentam que a introdução controlada de ruído em dados não rotulados pode aumentar a robustez dos modelos. Tal ideia foi adotada no presente artigo, tratando dados não-rotulados como negativos e, consequentemente, validando a estratégia.

Técnicas de Deep Learning. Redes neurais também têm sido aplicadas para capturar padrões complexos presentes em URLs. Por exemplo, [12] introduziram *PhishBench 2.0*, um *framework* para avaliação de modelos, incluindo CNN e RNN (*Convolutional Neural Networks* e *Recurrent Neural Networks*, respectivamente). O presente trabalho vai além ao comparar arquiteturas como CNN, GRU e CNNLSTM (vide Seção 4.3), mostrando que um modelo tão simples quanto uma *Random Forest* ainda supera modelos "profundos" em alguns cenários.

3 Metodologia

As ferramentas e *datasets* elaborados durante esse trabalho estão disponíveis publicamente no GitHub: https://github.com/pacutil/pacupy e https://github.com/pacutil/pacu-dataset respectivamente.

3.1 Datasets

O *dataset* principal proposto neste trabalho é formado majoritariamente por URLs brasileiras, contendo dados positivos (*phishing*) e não-rotulados, obtidos das seguintes fontes:

- Aproximadamente 260 mil URLs de *phishing* foram fornecidas pelo Banco do Brasil, as quais são majoritariamente *phishing* bancário, coletadas entre 2013 e 2021.
- Adicionalmente, foram coletadas aproximadamente 1,27 milhão de URLs maliciosas rotuladas do sistema de compartilhamento de ameaças (MISP, uma plataforma de código aberto para coleta, armazenamento, compartilhamento e correlação de informações para *threat intelligence* [13]) do nosso laboratório de pesquisa, registradas entre 2013 e 2025.
- Para compor um conjunto de URLs brasileiras legítimas, coletamos dados brasileiros do *crawler* Common Crawl[1], do período de 2013 a 2025, totalizando aproximadamente 5 bilhões de URLs.

A Tabela 1 apresenta os dados brutos utilizados para a construção dos *datasets*, indicando as fontes, o número de URLs por categoria (legítimo, não rotulado ou *phishing*), bem como o idioma e o período em que os dados foram coletados. Já a Tabela 2 mostra a composição final dos conjuntos de dados avaliados (CC-BB-MISP e DMOZ-PT), detalhando para cada um as quantidades de URLs por categoria, idioma e período.

Com o objetivo de reduzir a quantidade de dados positivos no conjunto de não rotulados, extraímos uma amostra aleatória de 50 milhões de URLs e submetemos para filtragem pelo serviço do Google Safe Browsing (GSB)[2]. Foram identificadas 7695 URLs como maliciosas, as quais foram descartadas. Consideramos que as URLs restantes são majoritariamente legítimas, com uma pequena margem de erro.

25 milhões das URLs que não foram detectadas foram utilizadas para calcular as distribuições base de bigramas e caracteres. O restante foi utilizado para compor os dados de treinamento e teste do *dataset* CC-BB-MISP.

O *dataset* CC-BB-MISP é um conjunto balanceado de 1,5 milhão de dados não rotulados, selecionados aleatoriamente a partir dos dados filtrados do Common Crawl que não foram usados para calcular as distribuições base, e 1,5 milhão de dados positivos do MISP e do Banco.

A fim de comparar a influência da linguagem na classificação, organizamos um segundo *dataset* com dados do DMOZ [14] e PhishTank[3]. Esse também é um conjunto balanceado, com aproximadamente 123 mil URLs, porém os dados estão em inglês e são classificados como positivos ou negativos; não há dados não rotulados.

As URLs do DMOZ que não foram usadas para compor o *dataset* DMOZ-PT (aproximadamente 3,5 milhões) foram utilizadas para extração das distribuições base da língua inglesa.

3.2 Desenvolvimento da CLI pacu

Com o objetivo de realizar os experimentos de forma sistemática e reprodutível, foi desenvolvida uma interface de linha de comando (CLI) denominada *pacu* (*Phishing Attack Classification Utility)*, responsável por integrar as principais funcionalidades necessárias para o fluxo de extração de características e treinamento de modelos. A ferramenta oferece dois comandos principais:

- **Extração de características (`pacu extract`)**: Realiza o processamento dos conjuntos de URLs para a extração das *features* relevantes. Entre as características extraídas, incluem-se distribuições de caracteres, distribuições de bigramas, bem como atributos estáticos, como a presença de endereço IP na URL, presença de número de porta, contagem de caracteres numéricos, entre outros.
- **Treinamento de modelos (`pacu train`)**: Possibilita a configuração e o treinamento de diferentes algoritmos de classificação, permitindo o ajuste de hiperparâmetros por meio de *flags*. Dentre os parâmetros ajustáveis, destacam-se: modelo selecionado, número de camadas, tamanho de *batch*, *kernel size*, entre outros.

3.3 Positive-Unlabeled Learning

O problema de detecção de URLs maliciosas foi modelado no contexto de *Positive-Unlabeled Learning* (*PU Learning*), dado que o conjunto de dados disponível era composto apenas por exemplos positivos (URLs maliciosas conhecidas) e exemplos não rotulados. Nesta abordagem, os exemplos positivos foram rotulados como 1, enquanto os exemplos não rotulados foram tratados como negativos, com rótulo 0. Não foi realizada nenhuma etapa posterior de correção do viés induzido pela possível presença de exemplos positivos entre os não rotulados. Dessa forma, o treinamento foi conduzido como um problema supervisionado clássico, assumindo-se a existência de ruído nos rótulos das amostras consideradas negativas.

Essa escolha foi motivada principalmente pelas considerações expostas em trabalhos como [11], que argumentam que a introdução

[1] https://commoncrawl.org/
[2] https://developers.google.com/safe-browsing/
[3] https://www.phishtank.com/

Tabela 1: Dados usados para compor os datasets

Origem	Legítimo	Não rotulado	Phishing	Idioma	Período
Common Crawl (CC)	-	49 806 485	7 695	br	2013 – 2025
Banco do Brasil	-	-	259 632	br	2013 – 2021
MISP	-	-	1 269 926	br	2013 – 2025
DMOZ	3 573 026	-	-	en	1998 – 2017
PhishTank (PT)	-	-	61 552	en	2025

Tabela 2: Composição dos datasets

Nome	Legítimo	Não rotulado	Phishing	Idioma	Período
CC-BB-MISP	-	1 529 558	1 529 558	br	2013 – 2025
DMOZ-PT	61 552	-	61 552	en	1998 – 2025

controlada de ruído no conjunto de dados pode não apenas contribuir para aumentar a robustez dos modelos, mas também aproximar o ambiente de avaliação das condições imperfeitas encontradas em cenários reais de detecção de ameaças. Ao tratar explicitamente os exemplos não rotulados como negativos, mesmo reconhecendo a presença de falsos negativos, o treinamento torna-se mais resiliente e alinhado às limitações práticas dos sistemas de segurança.

3.4 Seleção e Ajuste de Modelos

Para a avaliação da eficácia das diferentes estratégias de extração de *features*, selecionamos um amplo conjunto de modelos de *machine learning* e *deep learning*, abrangendo tanto algoritmos clássicos quanto arquiteturas neurais modernas. Os modelos considerados foram: *Random Forest* (RF), *Convolutional Neural Network* (CNN), *Multilayer Perceptron* (MLP), uma arquitetura híbrida *Convolutional Neural Network* com *Long Short-Term Memory* (CNNLSTM), e *Gated Recurrent Units* (GRU).

Cada modelo foi submetido a um processo de ajuste de hiperparâmetros por meio de uma estratégia de busca exaustiva, utilizando a ferramenta *pacu*. Durante esse processo, exploramos diferentes combinações de parâmetros, como número de camadas, tamanho das camadas ocultas, tamanho do batch, *kernel size*, entre outros. O objetivo do ajuste foi maximizar o desempenho dos modelos no cenário de *PU Learning*, considerando as métricas de avaliação definidas para o experimento.

3.5 Análise de Características

Foram usados nos treinamentos uma coleção de *features* que podem ser divididas em duas categorias: aquelas amplamente encontradas na literatura, *features* léxicas, e aquelas propostas por este artigo, *features* baseadas em distribuição. As características léxicas utilizadas nos treinamentos foram:

- Presença de IP no URL
- Quantidade de dígitos nos URL
- Número de hifens presentes no URL
- Comprimento do URL
- Profundidade do *path* do URL
- Quantidade de parâmetros na *query*
- Quantidade de domínios presentes, excluindo o TLD e o SLD
- Presença de uma porta no URL

No treinamento, essas características foram utilizadas para aumentar a robustez dos modelos, além de servirem como desempenho de referência para avaliar o impacto das *features* propostas.

As *features* propostas se baseiam na entropia dos URLs, pensando que aquelas que tem uma natureza maliciosa se diferem das benignas na forma como suas sub-palavras se distribuem. Na literatura, alguns trabalhos comparam a distribuição das letras em *URLs* àquela observada em línguas naturais [12]. No entanto, essa abordagem desconsidera o fato de que *URLs* frequentemente contêm uma quantidade significativa de caracteres especiais e dígitos, além de que a distribuição léxica de *URLs* benignos provavelmente difere daquela observada em textos da língua falada.

Em [9], propõe-se o uso de *bigramas* como *features*. Bigramas correspondem a todas as subpalavras compostas por dois caracteres consecutivos. Por exemplo, a palavra *phishing* gera os bigramas *ph*, *hi*, *is*, *sh*, *hi*, e assim por diante. A hipótese para o uso de bigramas é que técnicas de ofuscação utilizadas por agentes maliciosos podem alterar a distribuição natural dos caracteres, e que a análise de bigramas permitiria capturar essas alterações de forma mais informativa. Entretanto, a implementação sugerida apresenta problemas de performance, uma vez que cada bigrama é tratado como uma *feature* distinta no vetor de características. Assim, considerando um alfabeto de 94 caracteres possíveis, o espaço de *features* gerado atinge 8.836 dimensões, o que impacta negativamente o desempenho computacional e a capacidade de generalização dos modelos.

Com isso em mente, surge a ideia de calcular as distâncias das distribuições de caracteres e de bigramas em URLs não rotuladas ou benignas, incluindo pontuações e caracteres especiais. Para isso, utilizamos os dados do *Common Crawl* e do DMOZ que não foram utilizados para compor os conjuntos de treino e teste.

3.6 Distâncias e Testes Estatísticos

Para comparar os atributos de um URL qualquer com a distribuição calculada, foram utilizadas os seguintes métodos:

- **Distâncias:**
  (1) Euclidiana
  (2) Chebyshev
  (3) Manhattan

- **Testes Estatísticos:**
  (1) Kolmogorov-Smirnov
  (2) Kullback-Leibler

Cada um destes métodos se torna uma *feature*, resultando em um vetor de características muito menor do que aquele gerado ao usar bigramas como *features*. As distâncias são utilizadas com a intenção de detectar os desvio das distribuições, acreditando que os URLs maliciosos irão deslocar mensuravelmente no espaço a distribuição base. Foram utilizadas múltiplas distâncias para averiguar se haveriam mudanças de acurácia entre elas e se possivelmente elas poderiam trabalhar em conjunto.

Os testes estatísticos provém métodos mais elaborados para tentarmos identificar as diferenças nas distribuições. O *Kolmogorov-Smirnov* é utilizado para fazer um teste de aderência, ou seja, para verificar se um conjunto de dados seguem uma certa distribuição. Para isso, se compara a função de distribuição empírica dos dados extraídos de um único URL, $F_n(x)$, com a função de distribuição empírica obtida dos 25 milhões de URLs do *dataset* de distribuição, $F(x)$. Para compará-los é utilizada a estatística (1), em que D_n representa o maior desvio absoluto entre as duas funções de distribuição acumulada.

$$D_n = \sup_x |F_n(x) - F(x)| \tag{1}$$

O teste de divergência *Kullback-Leibler*, por sua vez, calcula quanto uma distribuição difere de outra. Então, sendo P a distribuição calculada e Q a aproximação gerada a partir do URL, obtemos a divergência usando a fórmula (2). Assim, teremos uma medida de entropia relativa de Q em relação a P, o que então é usado como *feature* para o modelo.

$$D_{KL}(P||Q) = \sum xP(x) \log \frac{P(x)}{Q(x)} \tag{2}$$

Essas características são particularmente relevantes na detecção de *phishing*, pois URLs maliciosos frequentemente apresentam padrões de composição textual que diferem sutilmente dos URLs legítimos. Ao utilizar métricas baseadas em testes de distribuição e divergência, é possível capturar pequenas discrepâncias estatísticas que não seriam percebidas por métodos tradicionais de análise de strings ou por simples contagens de caracteres. Dessa forma, essas *features* ajudam o modelo a identificar tentativas de imitação (*spoofing*) ou a presença de estruturas anômalas nos URLs, aumentando a capacidade de generalização e a robustez do sistema de detecção.

Uma outra *feature* considerada, também no mesmo contexto da entropia em URLs, foi o custo da construção da árvore de *Huffman* para cada URL. O algoritmo utilizado para isto não calcula o tamanho da árvore em si, mas sim o custo total de sua montagem. Ou seja, o custo equivale à soma dos pesos dos nós intermediários gerados ao longo da construção da árvore.

A codificação de *Huffman* é um método clássico de compressão que busca atribuir códigos mais curtos aos símbolos mais frequentes, de forma a minimizar o comprimento médio de codificação. Aplicada à estrutura dos URLs, a construção da árvore utiliza as frequências dos caracteres presentes para determinar os nós e suas combinações. Assim, URLs que apresentam uma distribuição de caracteres mais concentrada (e.g., URLs legítimos, que frequentemente repetem padrões comuns) tendem a gerar árvores de menor custo. Em contrapartida, URLs de *phishing*, ao explorarem combinações aleatórias de letras, números e símbolos para evadir mecanismos de segurança, produzem distribuições mais dispersas e, consequentemente, árvores de *Huffman* com custos mais elevados.

A ideia central é que o custo de montagem da árvore reflete, de forma agregada, a desordem ou a estrutura presente na composição do URL. Essa medida serve, portanto, como um indicador indireto da complexidade e previsibilidade do texto do URL. Integrar essa *feature* ao conjunto de características analisadas, listadas na Tabela 3, permite capturar aspectos sutis da estrutura dos URLs maliciosos, que poderiam não ser evidenciados apenas pelas análises de distâncias entre distribuições ou pelos testes estatísticos aplicados.

Tabela 3: Descrição das características extraídas de uma URL.

Feature	Descrição
number_count	Quantidade de dígitos na URL
has_ip	Presença de IP na URL
has_port	Presença de porta na URL
subdomain_count	Quantidade de subdomínios presentes, excluindo *TLD* e *SLD*
dash_symbol_count	Número de hifens presentes no URL
query_params_count	Quantidade de parâmetros na *query*
a3, *url_depth*	Profundidade do *path* da URL
url_length	Comprimento do URL
huffman	Custo da construção da árvore de Huffman
kl_char	Kullback Leibler calculado sobre a distribuição de caracteres base
kl_big	Kullback Leibler calculado sobre a distribuição de bigramas base
ks_char	Kolmogorov-Smirnov calculado sobre a distribuição de caracteres base
ks_big	Kolmogorov-Smirnov calculado sobre a distribuição de bigramas base
cs_char	Distância de Chebyshev calculada sobre a distribuição de caracteres base
cs_big	Distância de Chebyshev calculada sobre a distribuição de bigramas base
eucli_char	Distância Euclidiana calculada sobre a distribuição de caracteres base
eucli_big	Distância Euclidiana calculada sobre a distribuição de bigramas base
man_char	Distância Manhattam calculada sobre a distribuição de caracteres base
man_big	Distância Manhattam calculada sobre a distribuição de bigramas base

4 Discussão e Resultados

4.1 Avaliação Agnóstica ao Modelo

Inicialmente, realizou-se uma análise das *features* de forma agnóstica ao modelo, com o objetivo de identificar possíveis relações e criar hipóteses sobre seus efeitos no poder de predição dos modelos. Para isso, são utilizados os coeficientes de correlação de Pearson e Spearman, que geram coeficientes entre -1 e 1. Esses coeficientes quantificam a intensidade e a direção da relação entre as variáveis, valores próximos de 1 ou -1 indicam uma correlação forte, enquanto valores próximos de 0 sugerem que não há correlação significativa.

Na Figura 1, apresentam-se os resultados dessa análise, realizada a partir do cálculo dos coeficientes de correlação entre cada feature do conjunto de dados CC-BB-MISP e as respectivas *labels* dos URLs. Observa-se que as features propostas, baseadas em distribuição de caracteres e bigramas, apresentam uma correlação significativa com as *labels*, indicando que elas podem ser relevantes para classificação. Enquanto as features léxicas não apresentam uma correlação tão forte, sendo todos os seus coeficientes absolutos menores que 0.6. De modo geral, os coeficientes de Pearson e Spearman mostram valores similares entre si para a maioria das features. No entanto, observa-se uma discrepância notável nas features *huffman* e *url_a3,length*, o que sugere a presença de uma relação não linear com as *labels*, uma vez que Pearson quantifica apenas relações lineares, enquanto Spearman captura associações monotônicas, sejam elas lineares ou não.

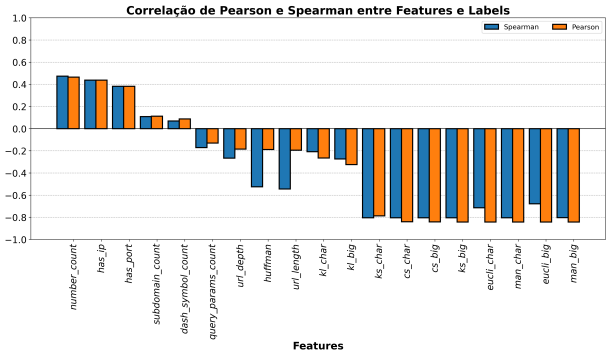

Figura 1: Correlações de Pearson e Spearman entre Features e Labels

4.2 Avaliação Dependente do Modelo

Para avaliar o impacto individual de cada *feature* sobre o desempenho dos modelos, foi adotada a metodologia proposta em [15]. Nessa abordagem, busca-se mensurar o grau de dependência do modelo em relação a uma determinada *feature* por meio da sua aleatorização: os valores da *feature* em questão são substituídos por valores aleatórios, eliminando, assim, qualquer relação estatística com a variável alvo. Caso o modelo dependa fortemente dessa *feature*, espera-se observar uma degradação significativa no desempenho. Os resultados obtidos após a aleatorização são então comparados com um *baseline* previamente gerado com todas as *features* originais preservadas.

As figuras 2 e 3 apresentam os resultados dos experimentos de permutação de *features* descritos acima. Nelas temos a perda de performance relativa ao *baseline* de cada modelo.

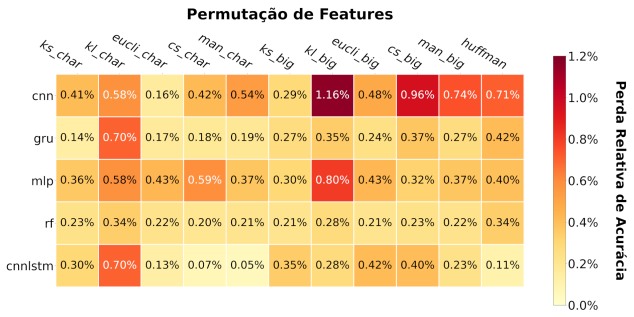

Figura 2: Perda de acurácia ao randomizar os valores das *features* de distribuição

A Figura 2 apresenta o impacto das *features* baseadas em distribuições. Nota-se um impacto mais acentuado na performance, principalmente nos modelos de *deep learning*, especialmente no *CNN*. A permutação da *feature kl_big*, por exemplo, resultou em uma queda de 1,16% no *CNN*, sendo o maior impacto individual observado entre todos os experimentos, além de que ambas *features* que utilizam *Kullback-Leibler* foram impactantes entre vários modelos. Também é possível observar que as representações baseadas em

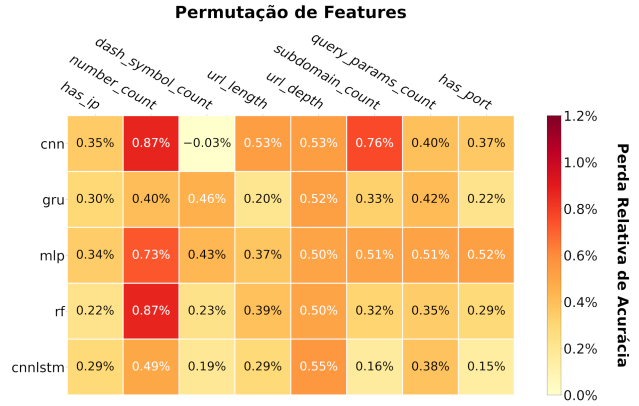

Figura 3: Perda de acurácia ao randomizar os valores das *features* léxicas

bigramas tendem a apresentar maior influência na acurácia do que aquelas baseadas em caracteres isolados, sugerindo que os modelos capturam padrões mais robustos utilizando a informação adicional fornecida pelos bigramas. Entretanto, fica claro que o *Random Forest* não depende fortemente de nenhuma *feature* individual.

Já a Figura 3 mostra os impactos causados pela permutação das *features* léxicas. Observa-se que a *feature number_count* foi uma das mais relevantes para diversos modelos, com destaque para os modelos *CNN*, *MLP* e *RF*, que apresentaram perdas de acurácia de até 0,87%. Além disso, a *feature subdomain_count* também teve impacto notável no modelo *CNN* (0,76%).

No geral, as perdas foram modestas, indicando que nenhuma das *features* individuais é isoladamente crítica para o desempenho dos modelos, mas algumas contribuem de forma significativa. Esses resultado também reforçam que embora as *features* léxicas sejam uteis para classificação, as distribuições desempenham papel mais significativo no desempenho dos modelos mais expressivos, como os baseados em redes neurais.

4.2.1 Distribuição de Bigramas X Bigramas como Features. Em [9], é proposta a utilização de bigramas como *features*, onde cada bigrama é representado como uma característica booleana: 1 indica a presença do bigrama no exemplo e 0 sua ausência. Conforme discutido na Seção 3.5, essa abordagem resulta, em um vetor de características de alta dimensionalidade, o que impacta negativamente a eficiência computacional do modelo. Diante disso, torna-se relevante comparar o desempenho desse método com o uso de distribuições como representações de características, a fim de avaliar se o custo computacional adicional é justificado.

Para essa comparação, foi utilizado o conjunto de dados *DMOZ-PT*, escolhido por seu tamanho moderado, uma vez que a abordagem baseada em bigramas se torna inviável em conjuntos de dados muito extensos. Os modelos empregados na comparação foram o *Random Forest* e o *CNN*.

Em relação ao tempo de treinamento, os modelos que utilizaram distribuições foram significativamente mais rápidos. O modelo *Random Forest* com bigramas como *features*, por exemplo, foi aproximadamente 5189 vezes mais lento para treinar, enquanto que o *CNN*

teve um tempo de treinamento 88,5 vezes superior ao observado com as distribuições.

Quanto a acurácia, a abordagem baseada em bigramas apresentou uma ligeira vantagem apenas no modelo *Random Forest*, com um ganho de 0,86% em relação à melhor acurácia obtida com distribuições. No entanto, para modelos de *deep learning*, como o *CNN*, o uso de bigramas como *features* resultou em quedas significativas de desempenho, com perdas de até 35% na acurácia. Esses resultados sugerem que, apesar do leve ganho pontual em alguns cenários, o alto custo computacional e a perda de desempenho em modelos mais complexos tornam a abordagem baseada em bigramas menos vantajosa em contextos práticos.

4.3 Avaliação dos Modelos

A Tabela 4 apresenta os resultados consolidados de desempenho dos modelos de aprendizado de máquina e *deep learning* testados no *dataset* CC-BB-MISP. As métricas avaliadas (acurácia, precisão, *recall* e F1-score) foram calculadas por meio de média ponderada, uma escolha apropriada diante do desequilíbrio inerente ao cenário de *Positive-Unlabeled Learning*, no qual apenas a classe positiva possui rótulos confiáveis.

Tabela 4: Métricas de Desempenho

Modelo	Acurácia	Precisão	Recall	F1-Score
Random Forest	94,32%	94,80%	94,32%	94,34%
GRU	93,96%	94,35%	93,89%	93,91%
MLP	93,80%	94,25%	93,77%	93,79%
CNN	92,37%	93,77%	93,18%	93,20%
CNNLSTM	93,19%	93,51%	92,41%	92,43%

Dentre os modelos avaliados, o *Random Forest* se destacou por apresentar o desempenho mais consistente e equilibrado entre todas as métricas. Sua alta acurácia, aliada à precisão e ao *recall* elevados, indica eficácia tanto na identificação de URLs maliciosas quanto na redução de falsos positivos, o que reforça sua robustez mesmo frente ao ruído nos dados não rotulados.

As redes neurais *GRU* e *MLP* obtiveram resultados muito próximos entre si, com métricas estáveis e bem distribuídas. Ambos demonstraram boa capacidade de generalização, capturando padrões relevantes sem comprometer nenhuma métrica específica. O desempenho competitivo do MLP, apesar de sua simplicidade arquitetural, evidencia o poder discriminativo das *features* utilizadas.

O modelo *CNN* apresentou uma leve queda no *recall*, embora tenha mantido precisão elevada. Essa combinação sugere um comportamento mais conservador, com menor propensão a falsos positivos, o que pode ser vantajoso em sistemas que priorizam a confiabilidade nas detecções, mesmo à custa de alguma perda de cobertura.

Por fim, o *CNNLSTM* foi o modelo com o pior desempenho geral. A combinação de *recall* reduzido e F1-score inferior indica uma maior taxa de falsos negativos, possivelmente decorrente da inadequação da arquitetura sequencial ao padrão simbólico, curto e não-linear das URLs. Esses resultados indicam que esse modelo é menos apropriado para a tarefa proposta neste contexto.

A Tabela 5 detalha o desempenho dos modelos por classe, permitindo observar seu comportamento ao lidar separadamente com URLs legítimas (classe 0) e maliciosas (classe 1). Em todos os casos, a precisão da classe 1 foi superior à da classe 0, indicando que, quando os modelos classificam uma URL como *phishing*, a decisão tende a ser confiável, com poucos falsos positivos.

Tabela 5: Métricas de Desempenho por Classe

Modelo	Classe	Precisão	Recall	F1-Score
Random Forest	0	89,29%	99,01%	93,90%
	1	99,15%	90,62%	94,69%
GRU	0	88,94%	98,45%	93,45%
	1	98,65%	90,28%	94,28%
MLP	0	88,72%	98,43%	93,32%
	1	98,64%	90,07%	94,16%
CNN	0	87,69%	98,39%	92,73%
	1	98,59%	89,05%	93,57%
CNNLSTM	0	85,38%	99,95%	92,09%
	1	99,96%	86,42%	92,70%

Por outro lado, o *recall* da classe 1 é consistentemente inferior ao da classe 0, o que revela uma maior dificuldade em detectar todos os casos de *phishing*. Isso é coerente com o cenário de *PU Learning*, no qual parte das amostras tratadas como negativas pode, na verdade, ser positiva, o que impacta a sensibilidade dos modelos.

O *Random Forest*, GRU e MLP apresentaram o melhor equilíbrio entre as classes, com boa capacidade de detectar ameaças sem sacrificar a precisão geral. Já o *CNN* e, especialmente, o *CNNLSTM*, demonstraram um viés mais conservador, com alto *recall* para a classe 0, mas menor sensibilidade para *phishing*, o que pode resultar em falsos negativos.

Tabela 6: Acurácia variando o idioma das distribuições no dataset CC-BB-MISP

Modelo	Idioma da Distribuição	Acurácia
Random Forest	Inglês	87,75%
	Português	94,32%
CNN	Inglês	74,61%
	Português	93,19%
CNNLSTM	Inglês	58,77%
	Português	92,37%
GRU	Inglês	74,72%
	Português	93,96%
MLP	Inglês	74,62%
	Português	93,80%

4.3.1 Influência do Idioma Base das Distribuições. Na Tabela 6, observa-se que, em média, os modelos treinados com o *dataset CC-BB-MISP* utilizando distribuições derivadas de dados em inglês (*DMOZ*) apresentaram uma acurácia aproximadamente 20% inferior à daqueles treinados com distribuições baseadas em dados em português (*Common Crawl*). Esse resultado sugere que a estrutura morfológica dos URLs de *phishing* varia de acordo com o idioma, o que reforça a importância de adaptar o treinamento dos modelos ao idioma-alvo.

5 Conclusão

A combinação de *features* léxicas com métricas de divergência sobre distribuições de caracteres e bigramas mostrou-se eficaz na detecção de URLs de *phishing* em cenários de *PU Learning*. As distribuições baseadas em bigramas destacaram-se por sua capacidade de capturar padrões relevantes com baixo custo computacional, sem perda significativa de acurácia em relação a abordagens que utilizam bigramas como *features* explícitas.

É importante levar em consideração o idioma das distribuições em relação ao *dataset*: um modelo treinado com URLs em português não terá a mesma eficácia na detecção de URLs de *phishing* em inglês quanto um treinado com um *dataset* em inglês. Como trabalhos futuros, propõe-se explorar distribuições multilinguísticas e adaptação dinâmica ao idioma das URLs analisadas.

Por fim, os resultados obtidos mostram que a distribuição de bigramas é uma característica eficaz para detectar URLs de *phishing*, mais do que características léxicas tradicionais. Entretanto, os melhores resultados são obtidos ao combinar tanto *features* de distribuição com *features* léxicas clássicas.

Agradecimentos

Este trabalho foi apoiado pelo Banco do Brasil, Laboratório de Pesquisa em Segurança, Engenharia de dados, Criptografia, Redes, Exploração e Testes (SECRET/UFPR), e Centro de Computação Científica e Software Livre (C3SL/UFPR), bem como pelo CNPq por meio de bolsa de produtividade em pesquisa.

Referências

[1] APWG. Phishing Activity Trends Report. Technical report, Anti-Phishing Working Group, 2024. URL https://apwg.org/trendsreports/.

[2] Microsoft. Microsoft Digital Defense Report. Technical report, Microsoft, 2024. URL https://www.microsoft.com/en-nz/security/security-insider/intelligence-reports/microsoft-digital-defense-report-2024.

[3] Adam Oest, Penghui Zhang, Brad Wardman, Eric Nunes, Jakub Burgis, Ali Zand, Kurt Thomas, Adam Doupé, and Gail-Joon Ahn. Sunrise to Sunset: Analyzing the End-to-end Life Cycle and Effectiveness of Phishing Attacks at Scale. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 361–377. USENIX Association, August 2020. ISBN 978-1-939133-17-5. URL https://www.usenix.org/conference/usenixsecurity20/presentation/oest-sunrise.

[4] Ya-Lin Zhang. POSTER: A PU Learning based System for Potential Malicious URL Detection. *CCS*, 2017.

[5] Jan Bayer, Sourena Maroofi, Olivier Hureau, Andrzej Duda, and Maciej Korczynski. Building a resilient domain whitelist to enhance phishing blacklist accuracy. In *2023 APWG Symposium on Electronic Crime Research (eCrime)*, pages 1–14, 2023. doi: 10.1109/eCrime61234.2023.10485549.

[6] Álvaro Feal, Pelayo Vallina, Julien Gamba, Sergio Pastrana, Antonio Nappa, Oliver Hohlfeld, Narseo Vallina-Rodriguez, and Juan Tapiador. Blocklist babel: On the transparency and dynamics of open source blocklisting. *IEEE Transactions on Network and Service Management*, 18(2):1334–1349, 2021. doi: 10.1109/TNSM.2021.3075552.

[7] Taeri Kim, Noseong Park, Jiwon Hong, and Sang-Wook Kim. Phishing url detection: A network-based approach robust to evasion. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, CCS '22, page 1769–1782, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450394505. doi: 10.1145/3548606.3560615. URL https://doi.org/10.1145/3548606.3560615.

[8] SK Hasane Ahammad. Phishing URL detection using machine learning methods. *Advances in Engineering Software*, 2022.

[9] Rakesh Verma and Avisha Das. What's in a URL: Fast Feature Extraction and Malicious URL Detection. In *Proceedings of the 3rd ACM on International Workshop on Security And Privacy Analytics*, IWSPA '17, page 55–63, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450349093. doi: 10.1145/3041008.3041016. URL https://doi.org/10.1145/3041008.3041016.

[10] Ashish Kumar Jha, Raja Muthalagu, and Pranav M. Pawar. Intelligent phishing website detection using machine learning. *Multimedia Tools Appl.*, 82(19):29431–29456, February 2023. ISSN 1380-7501. doi: 10.1007/s11042-023-14731-4. URL https://doi.org/10.1007/s11042-023-14731-4.

[11] Jehyun Lee, Pingxiao Ye, Ruofan Liu, Dinil Mon Divakaran, and Mun Choon Chan. Building Robust Phishing Detection System: an Empirical Analysis. In *Workshop on Measurements, Attacks, and Defenses for the Web (MADWeb)*, pages 1–7. NDSS Symposium, 2020. doi: 10.14722/madweb.2020.23007. URL https://dx.doi.org/10.14722/madweb.2020.23007.

[12] Victor Zeng, Xin Zhou, Shahryar Baki, and Rakesh M. Verma. PhishBench 2.0: A Versatile and Extendable Benchmarking Framework for Phishing. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, CCS '20, page 2077–2079, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450370899. URL https://doi.org/10.1145/3372297.3420017.

[13] MISP Standard. Open source threat intelligence and sharing platform, 2025. URL https://www.misp-project.org/.

[14] Lisa Hoek. Web classification using DMOZ. Bachelor thesis, Radboud University, 2021.

[15] Aaron Fisher, Cynthia Rudin, and Francesca Dominici. All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously. *Journal of Machine Learning Research*, 20(177):1–81, 2019.