PODER EXECUTIVO
MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO AMAZONAS
INSTITUTO DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

DETECÇÃO DE *PHISHING* NO TWITTER BASEADA EM ALGORITMOS DE APRENDIZAGEM *ONLINE*

HALINE PEREIRA DE OLIVEIRA BARBOSA

MANAUS
2018

HALINE PEREIRA DE OLIVEIRA BARBOSA

DETECÇÃO DE *PHISHING* NO TWITTER BASEADA EM ALGORITMOS DE APRENDIZAGEM *ONLINE*

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal do Amazonas, como requisito parcial para a obtenção do título de Mestre em Informática. Área de concentração: Redes de Computadores.

Orientador: Prof. Dr. Eduardo James Pereira Souto

MANAUS
2018

Ficha Catalográfica

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

Barbosa, Haline Pereira de Oliveira
B238d Detecção de Phishing no Twitter Baseada em Algoritmos de Aprendizagem Online / Haline Pereira de Oliveira Barbosa. 2018
86 f.: il. color; 31 cm.

Orientador: Eduardo James Pereira Souto
Dissertação (Mestrado em Informática) - Universidade Federal do Amazonas.

1. Detecção de Phishing. 2. Twitter. 3. Aprendizagem de Máquina. 4. Classificador Online. I. Souto, Eduardo James Pereira II. Universidade Federal do Amazonas III. Título

FOLHA DE APROVAÇÃO

"Detecção de Phishing no Twitter Baseada em Algoritmos de Aprendizagem Online"

HALINE PEREIRA DE OLIVEIRA BARBOSA

Dissertação de Mestrado defendida e aprovada pela banca examinadora constituída pelos Professores:

Prof. Eduardo James Pereira Souto - PRESIDENTE

Prof. Marco Antonio Pinheiro de Cristo - MEMBRO INTERNO

Prof. Gilbert Breves Martins - MEMBRO EXTERNO

Manaus, 03 de Abril de 2018

Para a o dono do H em meu nome.
Para minha mãe que me deu a vida na Terra.
Para meu querido e amado marido, Marcos.
E para as razões da minha vida, Eduardo e Júlia.

Agradecimentos

Agradeço a Deus em primeiro lugar por ter colocado esse desafio em minha vida e por me mostrar que eu sou capaz de realizar tudo o que eu sonhar e que for digno de ser feito, por ter me dado forças para aguentar a perda de familiares que me deixaram ao longo dessa jornada e não puderam comemorar comigo essa conquista.

Agradeço ao meu amado marido, Marcos, que me entende e sempre me incentiva com palavras de apoio e moral, me orienta e me abençoa nos momentos de aflição.

Agradeço aos meus filhos, Eduardo e Júlia, por compreenderem a minha ausência em vários momentos, como noites sem historinhas e beijinhos de boa noite.

Agradeço imensamente ao meu orientador Prof. Dr. Eduardo Souto pela dedicação e, principalmente, pela paciência.

Agradeço a minha mãe, “Dona Ceiça” e ao meu pai “Seu Hamilton” por me proporcionarem tempo, atenção, amor e por tudo o que tiveram que abrir mão para que eu pudesse ter uma educação de qualidade e ser uma boa pessoa.

Agradeço a todos os amigos do grupo de pesquisa de Tecnologias Emergente e Segurança de Sistemas – ETSS pela força e incentivo.

Aos amigos do trabalho da FAPEAM, UEA e FUAM que me incentivaram e me apoiaram nessa jornada.

Aos inúmeros amigos e familiares que sempre acreditaram que eu sou capaz.

Torne-se a melhor versão de você mesmo.
Autor desconhecido

Resumo

O Twitter é uma das redes sociais mais utilizadas no mundo com cerca de centenas de milhões de usuários compartilhando imagens, vídeos, textos e *links*. Devido às restrições impostas no tamanho das mensagens é comum que os *tweets* compartilhem *links* encurtados para *websites* impossibilitando a identificação visual prévia da URL antes de saber o que será exibido. Tal problema tornou o Twitter um dos principais meios de disseminação de ataques de *phishing* através de *links* maliciosos. *Phishing* é um ataque que visa obter informações pessoais como nomes, senhas, números de contas bancárias e de cartões de crédito. Em geral, os sistemas de detecção de ataques de *phishing* projetados para o Twitter são construídos com base em modelos de classificação *off-line*. Em tais sistemas, um grande volume de dados é examinado uma única vez para induzir em um único modelo de predição estático. Nesses sistemas, a incorporação de novos dados requer a reconstrução do modelo de previsão a partir do processamento de toda a base de dados, tornando esse processo lento e ineficiente. Para solucionar este problema, este trabalho propõe um *framework* de detecção de *phishing* no Twitter. O *framework* utiliza aprendizagem *online* supervisionada, ou seja, o classificador é atualizado a cada *tweet* processado e, caso este realize uma predição errada, o modelo é atualizado se adaptando rapidamente às mudanças com baixo custo computacional, tempo e mantendo a sua eficiência na tarefa de classificação. Para este estudo avaliamos o desempenho dos algoritmos de aprendizagem *online Adaptive Random Forest*, *Hoeffding Tree*, *Naive Bayes*, *Perceptron* e *Stochastic Gradient Descent*. O classificador *online Adaptive Random Forest* apresentou acurácia prequential 99,8%, na classificação de *tweets* de *phishing*.

Palavras chaves – Detecção de *phishing*, *Twitter*, aprendizagem de máquina, classificador *online*.

Abstract

Twitter is one of the most used social networks in the world with about 328 million users sharing images, videos, texts and links. Due to the restrictions on message size it is common for tweets to share shortened links to websites, making it impossible to visually identify the URL before knowing what will be displayed. Faced with this scenario, Twitter becomes a means of spreading phishing attacks through malicious links. Phishing is an attack that seeks to obtain personal information like name, CPF, passwords, number of bank accounts and numbers of credit cards. Twitter phishing attack detection systems are usually built using off-line supervised machine learning, where a large amount of data is examined once to induce a single static prediction model. In these systems, the incorporation of new data requires the reconstruction of the prediction model from the processing of the entire database, making this process slow and inefficient. In this work we propose a framework to detect phishing in Twitter. The framework uses supervised online learning, that is, the classifier is updated with each processed tweet and, if it makes a wrong prediction, the model is updated by adapting quickly to the changes with low computational cost, time and maintaining its efficiency in the task of ranking. For this study we evaluated the performance of the online learning algorithms Adaptive Random Forest, Hoeffding Tree, Naive Bayes, Perceptron and Stochastic Gradient Descent. The online Adaptive Random Forest classifier presented 99.8% prequential accuracy in the classification of phishing tweets.

Key words *– phishing detection, Twitter, machine learning, online learning.*

Sumário

Índice de Figuras

Índice de Tabelas

Capítulo 1

1 Introdução

A rede social *Twitter é* uma das redes sociais mais populares do mundo. O Twitter corresponde a um serviço de *microbbloging* que permite aos usuários expressarem seus sentimentos, ou compartilhar informação por meio de mensagens curtas (KANDASAMY; KOROTH, 2014). No *Twitter*, as mensagens curtas são denominadas de *tweet* e permitem a inserção de conteúdo multimídia como imagens, vídeos, por meio de *hyperlinks* de páginas eletrônicas. Porém, os *tweets* possuem limitação de tamanho, disponibilizando ao usuário uma quantidade restrita de caracteres para compartilharem informações (ATEFEH; KHREICH, 2015).

Devido a essa limitação de espaço, é comum o uso de serviços de encurtamento de URLs (do inglês, *Uniform Resource Locator*) como TinyURL, bit.ly, t.co, entre outros (ZHANG; WANG, 2012). Esses serviços recebem como entrada uma URL longa, processam e devolvem uma URL encurtada, que irá redirecionar o usuário para a URL longa original ao ser acessada.

Assim, apesar das URLs encurtadas proporcionarem o ganho de espaço requerido pelo Twitter, elas também tem sido empregadas para disfarçar URLs maliciosas. De acordo com os estudos realizados por Jeong et. al. (2016), tal estratégia tem sido amplamente adotada por atacantes de *phishing* que utilizam o Twitter para disseminar os ataques, pois a URL encurtada é amplamente compartilhada nos *tweets* que escondem do usuário a possibilidade do reconhecimento visual da URL original. Essa prática apresenta vantagens para os criminosos virtuais que fazem uso desses serviços para esconder *links* e redirecionar os usuários para páginas maliciosas. Esses *links* são utilizados como vetores para atividades

maliciosas como vírus, *worms*, *spam* e *phishing* (GUPTA; AGGARWAL; KUMARAGURU, 2014).

O Relatório de Ameaças de Segurança na Internet, publicado no primeiro trimestre de 2018 por Symantec (2018) aponta que houve um aumento de 182,6% no surgimento de URLs de *phishing* em 2017 quando comparado com 2016. O relatório mostra ainda que 5,8% de todas as páginas web (*websites*) criadas no ano de 2017 são páginas relacionadas a atividades de *phishing*. Esse aumento expressivo chama a atenção para a necessidade da criação de novas ferramentas ou técnicas que possam auxiliar na detecção e combate da disseminação dos ataques de *phishing*.

Para tratar o problema da disseminação do *phishing* no Twitter, a maioria dos estudos existentes na literatura propõe o uso de técnicas baseadas em aprendizagem supervisionada utilizando classificadores em *batch* (em lote) (AGGARWAL; RAJADESINGAN; KUMARAGURU, 2012; BECK, 2011; CHHABRA et al., 2011; GUPTA; AGGARWAL; KUMARAGURU, 2014; LEE; KIM, 2012; NAIR; PREMMA, 2014; XIAOLING et al., 2014). Tais técnicas têm sido superadas pelos atacantes em virtude dos modelos de classificação gerados serem estáticos e baseados em uma quantidade limitada de amostras e em características observadas em um determinado espaço de tempo.

Um dos problemas observados nesses modelos de classificação é a diminuição do seu desempenho ao longo do tempo em virtude da defasagem de aprendizagem do classificador, principalmente em redes sociais como o Twitter, onde o volume de dados gerado e compartilhado pelos seus usuários cresce de forma exponencial dificultando ainda mais o processo de atualização dos modelos preditivos supervisionados em lote (BENCZUR et. al, 2018).

Além disso, os atacantes (*phishers*) têm procurado adequações constantes nas abordagens de realização do ataque de *phishing* cada vez mais eficientes para burlar tais modelos, evitar a sua detecção e continuar alcançando seu objetivo principal que é furtar informações pessoais de usuários para obtenção de lucro financeiro (JEONG et. al, 2016).

Dessa forma, para manter o alto desempenho do classificador é necessário utilizar técnicas capazes de atualizar os seus modelos preditivos rapidamente, mantendo o baixo custo computacional e a eficiência na tarefa de classificar os *tweets* além de serem adaptáveis às

mudanças. Uma possível abordagem para solucionar este problema é a utilização de algoritmos *online*.

Portanto, este trabalho busca avaliar a aplicação da aprendizagem *online* no processo de detecção de *phishing* a partir do *Twitter* e, tem como meta determinar quando um *tweet* contém uma mensagem de *phishing* ou não. O objetivo é proporcionar uma forma mais confiável e de fácil atualização para a prevenção contra esse tipo de ataque. Para alcançar esse objetivo, este trabalho propõe um *framework* que permite aos usuários a avaliação dos métodos de classificação *online* de acordo com as características identificadas, proporcionando a seleção do classificador de melhor desempenho.

O emprego de abordagens de classificação *online* para a detecção de mensagens de *phishing* ainda não foi aplicado ao *Twitter*. Porém, técnicas semelhantes são utilizadas na detecção de *phishing* em páginas web, apresentando resultados satisfatórios e adaptáveis ao processamento de *data streamming* e mudanças de contexto (BASNET; DOLECK, 2015; BLUM et al., 2010; FEROZ; MENGEL, 2014; LE; MARKOPOULOU; FALOUTSOS, 2011; LIN et al., 2013; MA et al., 2009; THOMAS et al., 2011; WHITTAKER; RYNER, 2008; ZHANG; WANG, 2012).

1.1 Objetivo

Este trabalho tem como objetivo principal desenvolver e avaliar um framework de detecção de *phishing* no Twitter baseado em técnicas de aprendizagem de máquina *online*.

Para atingir este objetivo serão realizados os seguintes objetivos específicos:

- Criar um mecanismo coleta e extração de *tweets*;
- Desenvolver um extrator de características baseado em informações do *tweet*, usuário do Twitter e da URL contida no *tweet*;
- Avaliar diferentes classificadores *online* no processo de detecção de *tweets* utilizados como vetores de ataques do tipo *phishing*;

1.2 Organização da Dissertação

O restante deste trabalho está organizado da seguinte forma:

O Capítulo 2 descreve a fundamentação teórica, apresentando os conceitos básicos para compreensão dos termos utilizados como rede social, *Twitter* e *tweet*, URL e seus serviços de encurtamento, *phishing*, aprendizagem de máquina e classificador *online*.

O Capítulo 3 apresenta os principais trabalhos relacionados. Tais trabalhos são organizados em métodos de detecção de *phishing* em *websites* e no *Twitter*.

O Capítulo 4 descreve o framework proposta, apresentando as principais etapas.

O Capítulo 5 apresenta os experimentos e resultados. O Capítulo inicia descrevendo o protocolo experimental, bases de dados utilizadas e finaliza com os experimentos e testes realizados para avaliação dos modelos gerados para o framework proposto.

Por fim, no Capítulo 6 – Conclusão: são apresentadas as considerações finais, limitações da abordagem e trabalhos futuros.

Capítulo 2

2 Fundamentação Teórica

Este capítulo apresenta os conceitos e definições fundamentais para o entendimento deste trabalho. São apresentadas as características da rede social *Twitter*, URL, serviços de encurtamento, ataque do tipo *phishing*, aprendizagem de máquina e classificação *online*.

2.1 Redes Sociais *Online*

Segundo Boyd e Ellison (2007); Ellison et al. (2007), sites de redes sociais são serviços baseados na *web* e permitem aos indivíduos: (1) construir um perfil público ou semipúblico dentro de um sistema limitado; (2) articular com uma lista de outros usuários com os quais compartilham uma conexão; e (3) ver e examinar as suas e outras listas de conexões realizadas pelo sistema. O que torna uma rede social única não é a possibilidade de permitir que os usuários conheçam estranhos, mas o quanto elas possibilitam aos usuários articularem entre si e tornarem-se visíveis em suas redes sociais. Ao entrarem em uma rede social, os usuários são induzidos a identificar outros usuários, com os quais este possui algum tipo de relacionamento ou afinidade.

Nas redes sociais é possível compartilhar conteúdo com seus “amigos”, “fãs”, “seguidores”, “contatos”, termos conhecidos para descrever as relações entre os usuários que estão conectados ao perfil de um determinado usuário. Esse conteúdo varia segundo o propósito da rede social. Todavia, dentre as funcionalidades mais comuns, temos: mensagens, imagens, fotos, vídeos, áudio, perfil profissional, notícias, *sites*, dentre outros.

Considerando o entretenimento e a formação de grandes redes de relacionamentos entre os usuários das redes sociais como principais propósitos na adoção de uma rede social *online* por uma pessoa. A Figura 2.1 apresenta uma classificação, por usuários ativos, das principais redes sociais *online* no mundo. O Facebook aparece em primeiro lugar na preferência dos usuários por agregar muitas funcionalidades, já relacionadas e inclusive integradas com outras redes, para os seus usuários, seguido do Youtube que é uma rede de compartilhamento de vídeos e pelo WhatsApp que é considerada uma rede social pois é possível realizar compartilhamentos e possui uma linha do tempo temporária. Na sequência temos as redes sociais chinesas WeChat, QQ e QZone e em 10º lugar, o Twitter que é uma rede social que disponibiliza APIs para coletas de informações para fins de pesquisa ou criação de aplicativos para investigação, aprimoramento e uso da rede.

Figura 2.1 - Principais redes sociais e a quantidade de usuários ativos.
Fonte: www.statista.com, 2017.

As redes sociais têm características e objetivos específicos, como o Linkedin[1] criada em 2003 é uma rede social voltada à apresentação virtual de habilidades profissionais dos usuários em busca de novos desafios ou recolocação no mercado de trabalho. Além de ter um conteúdo com menos entretenimento e mais conteúdo para aprimoramento profissional. A

[1] https://www.linkedin.com/

rede Instagram[2] criada em 2010 tem como objetivo o compartilhamento de imagens e vídeos curtos. Existe também a rede social do tipo *microbbloging* que tem como objetivo compartilhar textos curtos, limitados a uma determinada quantidade de caracteres, por exemplo, o Twitter[3], alvo desta pesquisa na avaliação do *framework* proposto para detecção de mensagens maliciosas em redes sociais.

2.2 Twitter e *tweet*

A empresa *Twitter* foi fundada em julho de 2006 e tem como objetivo disponibilizar um ambiente no qual os usuários possam compartilhar ideias e informações instantaneamente. Segundo Beck (2011), o *Twitter* é uma rede técnico-social que permite aos usuários enviarem pequenas mensagens para outros usuários. Essas mensagens são chamadas de *tweet*. Kwak et al. (2010) descrevem o *Twitter* como sendo um serviço de *microbbloging e* que em 2009 quando esta rede possuía menos de três anos de existência já possuía mais de 41 milhões de usuários ativos. Esse número cresceu e alcançou, no segundo semestre de 2017, a marca de 328 milhões em pouco mais de 10 anos de atividades da rede.

O Twitter possui características particulares como o relacionamento assimétrico que permite que um usuário siga outro usuário sem a necessidade da reciprocidade e, assim, receber todas as atualizações da linha do tempo deste usuário. Outra característica do Twitter é a limitação do tamanho da mensagem contida no *tweet* que, até início de novembro de 2017, era de apenas 140 caracteres e atualmente teve o seu tamanho dobrado. Alguns usuários consideram essa condição como uma limitação severa, outros argumentam que informações pequenas são mais fáceis de serem acessadas e disseminadas (ATEFEH; KHREICH, 2015).

A Figura 2.2 mostra símbolos e padrões utilizados com frequência por usuários do Twitter em seus *tweets*. Por exemplo, para abordar um assunto do momento, conhecido como *trending topic*, basta usar o símbolo "#", chamado de *hashtag* pelos usuários da rede, na frente da palavra-chave (#palavra). O "@" é utilizado para referenciar um usuário no *tweet*, ou fazer uma menção a alguém, apenas inserindo-o à frente do nome do usuário, exemplo:

[2] http://instagram.com/

[3] https://twitter.com/

@usuario. Quando um *tweet* possui as letras “RT” no início do texto significa que é uma mensagem do tipo *retweet*, ou seja, é um *tweet* originado de outro usuário.

Devido ao fato do tamanho da mensagem ser limitada, é comum o uso de palavras abreviadas, marcadores e no compartilhamento de links, o uso de URLs encurtadas, ou seja, o usuário pode usar serviços de encurtamento de URLs para economizar espaço em seus *tweets*.

Figura 2.2 – Exemplo de um *tweet*.
Fonte: Próprio Autor, 2018.

Segundo Stronkman (2011) existem cinco tipos de *tweets*:

- *Singleton* (filho único ou mensagem única) que é uma atualização simples que não contém referência a outro *tweet*, usuário ou *tag*;
- R*eply* (replicadora) mensagens que contém no seu início o ‘@nomedousuário’, ou seja, uma mensagem direcionada a um usuário do *Twitter*;
- M*ention* (menção) ocorre quando um usuário se refere à outra pessoa, também contém ‘@nomedousuário’ mas não é um *reply*;

- *Direct message* (mensagem direta) é uma mensagem privada entre dois usuários. É possível iniciar uma mensagem direta quando um usuário segue o outro e a mensagem é iniciada com D nomedousuario;
- Por fim, o *Retweet* ou RT que corresponde ao reenvio de uma mensagem de alguém e, às vezes, esta contém algum comentário a respeito.

Um *tweet* pode ser coletado por meio de uma interface disponibilizada pelo Twitter no formato JSON (*JavaScript Object Notation*). A Figura 2.3 mostra um exemplo de arquivo JSON contendo vários atributos de um *tweet*.


```
{
  "created_at":"Thu Apr 06 15:24:15 +0000 2017",
  "id": 850006245121695744,
  "id_str": "850006245121695744",
  "text": "1/ Today we're sharing our vision for the future of the Twitter API platform!n
  https://t.co/XweGngmxlP",
  "user": {
     "id": 2244994945,
     "id_str": "2244994945",
     "name": "TwitterDev",
     "screen_name": "TwitterDev",
     "location": "Internet",
     "url": "https://dev.twitter.com/",
     "description": "Your official source for Twitter news",
     "verified": true,
     "followers_count": 477684,
     "friends_count": 1524,
     "listed_count": 1184,
     "favourites_count": 2151,
     "statuses_count": 3121,
     "created_at": "Sat Dec 14 04:35:55 +0000 2013",
     "utc_offset": -25200,
     "time_zone": "Pacific Time (US & Canada)",
     "geo_enabled": true,
     "lang": "en",
     "profile_image_url_https": "https://pbs.twimg.com/"
  }
 "entities": {. . . }
}
```


Figura 2.3 - Exemplo de arquivo JSON de um *tweet*. Fonte: O Autor, 2018.

Um *tweet* contém mais informações além do texto contido na mensagem, tais como: identificação única, geolocalização, data e hora de criação, dados sobre a conta e o perfil do usuário e a sua rede de contatos, imagens, vídeos, *links* para notícias e outros *sites* (STRONKMAN, 2011).

A Tabela 2.1 exibe os principais atributos de um *tweet* que são utilizados neste trabalho a fim de extrair informações para formar o vetor de características.

Tabela 2.1 - Principais atributos de um *tweet.*

Atributo	Tipo	Descrição
created_at	Nominal	Formato de hora UTC, apresenta o momento que o *tweet* foi criado. Exemplo: "created_at":"Wed Aug 27 13:08:45 +0000 2008"
id	Inteiro	É a representação de um identificador único para o *tweet*, em formato numérico. Exemplo: "id":114749583439036416
id_str	Nominal	É a representação de um identificador único para o *tweet*, em formato nominal. Exemplo: "id_str":"114749583439036416"
text	Nominal	É a mensagem de texto em formato UTF-8. Exemplo: "text":"Tweet Button, Follow Button, and Web Intents"
source	Nominal	Aplicativo ou utilitário utilizado para enviar *tweets*, em formato HTML. Exemplo: "source" : "Twitter for Mac"
user	Objeto	É a representação do usuário que enviou o *tweet*.
coordinates	Objeto	*Este objeto r*epresenta a localização geográfica do *tweet* reportado pelo usuário ou aplicação cliente, em formato JSON.
place	Objeto	Este objeto possui outros atributos, como coordenadas para identificar o lugar, cidade, estado e id da localização de onde o *tweet* foi criado.
re*tweet*ed_status	Inteiro	Usuários podem ampliar o envio dos *tweets* gerados por outros usuários ao realizar um "re*tweet*" um *tweet.*
re*tweet*_count	Inteiro	Número de vezes que o *tweet* foi replicado ou feito um "re*tweet*" do mesmo. Exemplo "re*tweet*_count":1585
favorite_count	Inteiro	Indica, aproximadamente, quantas vezes o *tweet* recebeu um "like" por outros usuários. Exemplo: "favorite_count":1138
entities	Objeto	Informações contidas no texto do *tweet*, como: hashtags, urls, menções, mídias e símbolos.
favorited	Booleano	Indica que o *tweet* recebeu um "like".
re*tweet*ed	Booleano	Indica que o *tweet* foi replicado ou foi feito um "re*tweet*" do seu conteúdo por um usuário autenticado.
lang	Nominal	Indica o idioma segundo a identificação BCP 47.
hashtag	Nominal	Indica todas as hashtag utilizadas em uma mensagem.
tweet.count	Inteiro	Quantidade de *tweet*s criados pelo usuário.

O atributo "entities" possui o subatributo "urls", que contém um vetor com todas as URLs compartilhadas no texto da mensagem. Alguns *tweets* não possuem esse dado

preenchido, pois a URL também pode ser encontrada no subatributo “media” quando compartilha um vídeo ou imagem. Os demais dados contidos nos atributos “favorited”, “text”, “re*tweet*ed_count”, “coordinates”, “user”, “source”, também são utilizados nesta pesquisa para composição do vetor de características.

2.3 URL

A RFC1738 da IETF de Lee-Berners e Masinter (1994) define o termo URL como *Uniform Resource Locator*, ou Localizador Uniforme de Recursos, em português. Neste documento oficial, a sintaxe e semântica da URL são determinadas e sua única função é localizar recursos na Internet providenciando uma identificação abstrata para eles. Em geral as URL são escritas de acordo com o seguinte padrão:

<scheme>:<scheme-specific-part>

As URLs são formadas por uma sequência de letras (“a” a “z”), dígitos (“0”, “1”, ..., “9”) e caracteres especiais seguros permitidos (: “+”,“.”, “-“). Outros caracteres são considerados inseguros, como: "{", "}", "|", "\", "^", "~", "[", "]", "`" e devem ser codificados para que possam ser utilizados. Alguns caracteres especiais são reservados, pois possuem significado semântico na URL tais como: ";", "/", "?", ":", "@", "=" e "&" e são utilizados na segunda parte da URL, no <scheme-specific-part>.

Bezerra (2015) propõe uma taxonomia para classificar as principais características em: características *online* e *off-line* da URL, conforme pode ser visualizada na Figura 2.4. Algumas dessas características foram utilizadas neste trabalho e serão detalhadas no Capítulo 4.

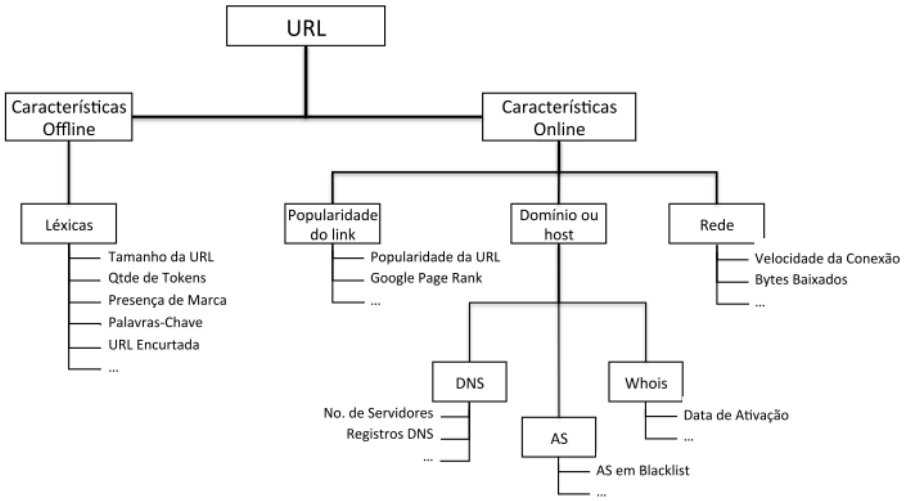

Figura 2.4 - Taxonomia de classificação das características das URLs.

Fonte: (BEZERRA, 2015).

2.4 Serviço de Encurtamento de URL

No contexto das redes sociais *online* é frequente o uso de serviços de encurtamento de URLs para economizar espaço em uma mensagem. Antoniades et al. (2011) explicam que a ideia por trás dos serviços de encurtamento de URL é prover uma maneira fácil de compartilhar URLs disponibilizando uma URL curta equivalente à original. Por exemplo, se um usuário submeter a url: http://www.this.is.a.long.url.com/indeed.html ao serviço de encurtamento do bit.ly[4], será retornada a seguinte URL encurtada: http://bit.ly/dv82ka. O usuário poderá compartilhá-la em qualquer *website*, *blog* ou rede social que esta será redirecionada pelo serviço correspondente no bit.ly para a sua respectiva URL longa original.

A Figura 2.5 apresenta um esquema sobre o funcionamento dos sistemas de encurtamento centralizados de URL, onde as URLs são encurtadas antes de serem postadas. O processo inicia quando o usuário envia uma URL longa para um serviço de compressão/encurtamento. O serviço armazena a URL original em uma tabela de tradução e gera, por meio de um algoritmo próprio, uma URL curta que representa a URL original e devolve ao usuário uma URL encurtada. Quando um usuário clica numa URL encurtada, uma

[4] https://bitly.com/

requisição HTTP é gerada para o serviço de encurtamento que consulta a tabela de tradução e redireciona o usuário para a URL longa original.

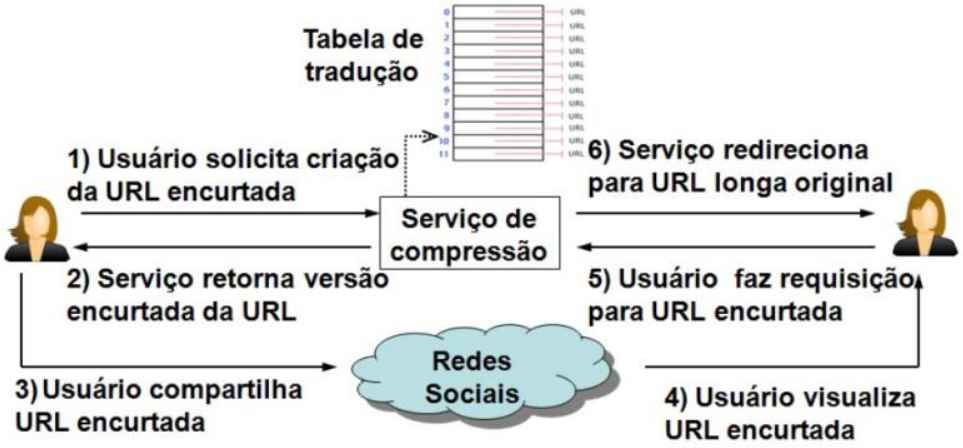

Figura 2.5 Arcabouço para encurtamento de URLs de forma centralizada.

Fonte: FREITAS (2012).

Atualmente existem vários serviços de encurtamento de URLs, tais como t.co[5], lnkd.in[6], goo.gl[7], bit.ly, tinyurl.com[8] e fb.me[9]. Os sistemas de encurtamento de URLs são bastante utilizados no Twitter para otimizar o uso do espaço da mensagem. Uma URL encurtada perde a sua aparência original de acordo com os algoritmos utilizados no seu encurtamento. Contudo, tais algoritmos também colaboram para o surgimento de riscos à segurança dos usuários, pois não é visível para o usuário o que realmente este link irá exibir ou executar.

Chhabra et al. (2011) discorrem sobre as ameaças presentes em URLs encurtadas e a correlação com a grande possibilidade da existência de URLs de websites de *phishing*, que redirecionam o usuário para um site muito semelhante ao original com o objetivo de roubar informações sensíveis.

[5] http://t.co/

[6] https://www.linkedin.com/

[7] https://goo.gl/

[8] http://tinyurl.com/

[9] https://www.facebook.com/

2.5 Phishing

Phishing é um tipo de ataque de computador em que criminosos (conhecidos como *phishers*) visam obter informações confidenciais como credenciais de bancos, dados pessoais e números de cartões de crédito, por meio de *sites* falsos que imitam *sites* legítimos (JEONG; B; DOBBIE 2016).

De acordo com o *Anti-Phishing Working Group* - APWG[10], o termo *phishing* surgiu em 1996 devido ao ataque de engenharia social contra a *America Online* (AOL) por golpistas para roubar os dados das contas dos usuários. Este termo vem do inglês *fishing*, no sentido de pescar, uma vez que os atacantes, denominados de *phishers*, utilizam de estratégias de engenharia social para fisgar os usuários com o intuito de roubar informações pessoais das vítimas. O uso do *ph* ao invés do *f* é devido ao fato de que uma das primeiras formas de pirataria foi encontrada em redes de telefonia, denominada *phreaking*. Assim, palavras iniciadas com *f* (em inglês), no contexto da pirataria, tiveram o seu caractere inicial substituído por *ph* (KHONJI; IRAQI; JONES, 2013).

A Figura 2.6 apresenta o gráfico publicado pela APWG em fevereiro de 2017 que mostra a incidência de ataques de *phishing* no mundo. Em média, cerca de 60 mil ataques de *phishing* foram reportados durante o ano de 2015. Em 2016, esses números duplicaram entre os meses de maio a julho, fato que pode ser associado à proximidade de grandes eventos internacionais, como as olimpíadas.

Em fevereiro de 2018 o relatório[11] sobre *phishing* disponibilizado pela APWG, mostra que o número total de denúncias de *phishing*, durante o terceiro trimestre de 2017, foi de 296.208 sendo 23.000 a mais que os 273.395 reportados no segundo trimestre.

[10] http://www.antiphishing.org/

[11] http://docs.apwg.org/reports/apwg_trends_report_q3_2017.pdf

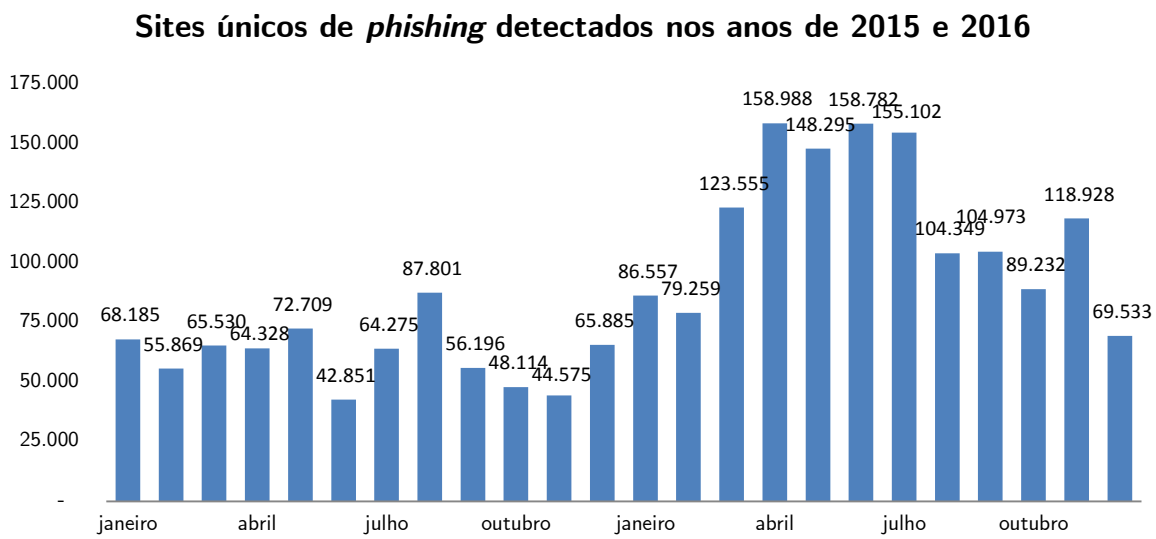

Figura 2.6 Gráfico de sites únicos de *phishing* detectados nos anos de 2015 e 2016.
Fonte: Adaptado de Anti-Phishing Working Group (2017).

A pesquisa[12] realizada pela Kaspersky Lab em 2015 classifica o Brasil como um dos países mais perigosos para usuários digitais, ficando atrás somente da Rússia, devido à fraca legislação sobre o assunto e da impunidade contra os bandidos virtuais. A pesquisa aborda ainda o ataque de *phishing* comprometendo o sistema *online* disponibilizado no site do Instituto Brasileiro do Meio Ambiente e dos Recursos Renováveis - IBAMA, no qual credenciais foram roubadas e foram desbloqueadas 23 empresas, previamente suspensas por crimes ambientais, permitindo a estas que pudessem extrair madeira da Floresta Amazônica. Em apenas 10 dias, essas mesmas empresas extraíram mais de $11 milhões de dólares em madeira, o equivalente a 1.400 caminhões lotados de madeira ilegal (ASSOLINI, 2015).

2.5.1 *Phishing* no Twitter

Como este trabalho utiliza a rede social Twitter, é necessário explorar e conhecer como os ataques de *phishing* ocorrem nessa rede social. A Figura 2.7 apresenta uma visão geral de como o ataque funciona.

[12] https://securelist.com/files/2015/11/KLReport_CyberUnderground_Brazil_eng.pdf

De acordo com os trabalhos relacionados e com os estudos realizados, o ataque se inicia quando um *phisher* cria uma página de *phishing* e a disponibiliza na Internet. Este *website* apresenta similaridades visuais com o *site* real e possui *scripts* para receber e enviar dados para o seu criador (atacante). Exemplos de dados que podem ser capturados e enviados são: número de conta bancária, número do cartão de crédito, senhas, dados de identificação pessoal ou quaisquer outros dados referente ao contexto do *website* criado.

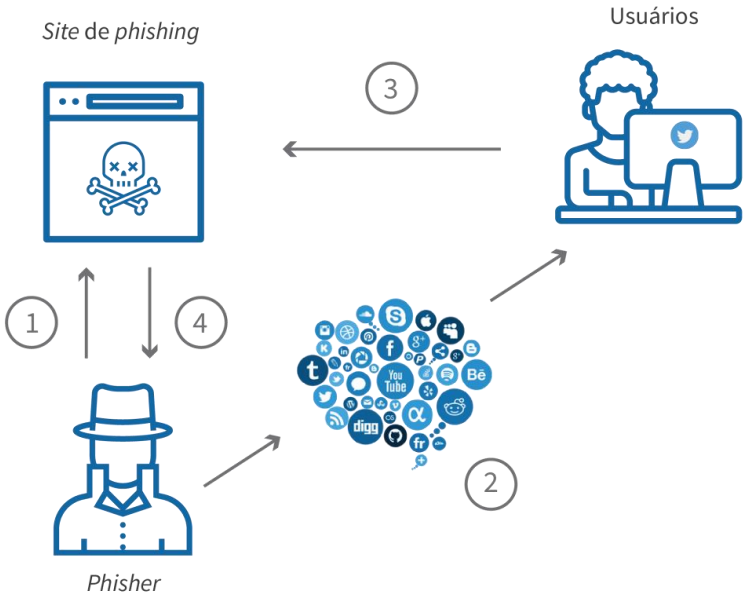

Figura 2.7 Fluxo da Campanha de Phishing no Twitter.

Fonte: O Autor, 2017.

Assim que o *site* malicioso é hospedado e torna-se acessível, iniciam-se as campanhas para divulgá-lo no *Twitter* por meio do envio de *tweets* com *links* encurtados. O *phisher* compartilha *tweets* em sua linha do tempo e essas mensagens serão visualizadas por todos os seus amigos e seguidores. Após a divulgação do link, o passo seguinte ocorre quando um usuário clica no *link* fornecido e acessa ao *site* de *phishing* que parece ser um *site* normal e confiável informando os seus dados. No passo final, o *phisher* recebe os dados e alcança o seu objetivo roubando as informações pretendidas.

2.6 Aprendizagem de Máquina

A aprendizagem de máquina foi definida amplamente para Mitchell et al. (1997) como:

> *"Um programa de computador é dito para **aprender** a partir da experiência **E** com relação a alguma classe de tarefas **T** e medida de desempenho **P**, se o seu desempenho nas tarefas em **T**, medida por **P**, melhora com a experiência **E**".*

Segundo Faceli et al., (2011) a aprendizagem de máquina apresenta vários paradigmas baseados em aprendizagem indutiva para a aquisição automática do conhecimento no desenvolvimento de sistemas inteligentes. O aprendizado indutivo pode ser dividido em supervisionado e não supervisionado. No aprendizado supervisionado é fornecido ao algoritmo de aprendizado, ou indutor, um conjunto de exemplos de treinamento para os quais o rótulo da classe associada é conhecido. O objetivo do indutor é aprender com o conjunto fornecido e gerar um modelo capaz de reconhecer os padrões das amostras apresentadas.

Já no aprendizado não supervisionado, o indutor analisa os exemplos fornecidos e tenta determinar se alguns deles podem ser agrupados de alguma maneira, formando agrupamentos ou *clusters*. Assim que os clusters são agrupados, é necessário analisar e determinar o que cada agrupamento significa dentro do contexto que se está analisando (FACELI et al., 2011).

É importante observar que na aprendizagem supervisionada existem modos de aprendizado, ou seja, sempre que todo o conjunto de treinamento deva estar presente para o aprendizado, o modo de aprendizado de um algoritmo é não incremental ou *batch*. Por outro lado o modo de aprendizado incremental ou *online* ocorre quando o indutor não necessita construir a hipótese a partir do início, quando novos exemplos são adicionados ao conjunto de treinamento. Nesse último caso o indutor apenas atualiza a hipótese antiga quando são adicionados novos exemplos ao conjunto de treinamento (FACELI et al., 2011).

Esta pesquisa utiliza aprendizado supervisionado *online* para o problema de detecção de *phishing* a partir do conjunto de características identificado, pois há um grande desafio em tentar manter classificadores *batch* atualizados quando se tem restrição de recursos computacionais para processar os dados e gerar grandes conjuntos de treinamento onde há volume de fluxo de dados infinito e constantes mudanças nos conceitos aprendidos.

2.6.1 Aprendizagem de máquina *online* supervisionada

O algoritmo de aprendizagem observa as instâncias de uma maneira sequencial e após cada observação, o algoritmo prevê uma saída. Esta saída pode ser um simples sim ou não ou + ou –, nos casos de classificação binária ou, nos casos mais complexos, uma *string* de letras do alfabeto. A cada amostra, o algoritmo faz a sua predição e recebe o *feedback* indicando a resposta. Caso a predição realizada tenha sido diferente do *feedback*, ele sofre uma perda instantânea que reflete no grau de aprendizagem que a predição errada e então, passa pelo processo de atualização do seu mecanismo de predição, melhorando as chances de realizar uma predição mais precisa nas rodadas subsequentes (CRAMMER et al., 2006).

O problema de detecção de *phishing* no Twitter é um problema de classificação binária, ou seja, existem duas classes discretas (*phishing* e legítimo) cujo objetivo da função f é classificar um *tweet* dentro das classes existentes, buscando minimizar o erro da predição e, consequentemente, aumentar a sua acurácia.

A próxima seção apresenta os algoritmos de aprendizagem *online* avaliados nesta pesquisa.

2.6.2 *Adaptive Random Forest* – ARF

O ARF é uma adaptação do algoritmo clássico de *Random Forest.* A principal novidade do ARF está em como ele combina as características do algoritmo de lote com métodos de atualização dinâmica para tratar *data streaming*. O ARF usa um método de reamostragem teoricamente sólido baseado em *Online Bagging* e uma estratégia adaptativa utilizada para tratar *data streaming*. Essa estratégia adaptativa é baseada no uso de um monitor de desvio por árvore para rastrear avisos e desvios e para treinar novas árvores em *background* antes de substituí-las (GOMES, 2017).

Código 2.1 – *Adaptive Random Forest*.

Algoritmo 1 – *Adaptive Random Forest*

```
1. function: ADAPTIVERANDOMFOREST(m, n, δ_w, δ_d)
2.     T ← CreateTrees(n)
3.     W ← InitWeights(n)
4.     B ← ∅
5.     While HasNext(S) do
6.       (x, y) ← next(S)
7.       For all t ∈ T do
8.           ŷ ← predict (t, x)
9.           W(t) ← P(W(t), ŷ, y)
10.          RFTreeTrain (m, t, x, y)   ► Treina t na instância atual (x, y)
11.          If C( δ_w, t, x, y) then   ► Aviso detectado?
12.              b ← CreateTree()       ► Inicia a árvore em background
13.              B(t) ← b
14.          End if
15.          If C( δ_d, t, x, y) then   ► Mudança detectada?
16.                t ← B(t)             ► Substitui t pela árvore em background
17.          End if
18.      End for
19.      For all b ∈ B do               ► Treina cada árvore em background
20.          RFTreeTrain (m, t, x, y)
21.      End for
22.    End while
23. End function
```

O método funciona de acordo com o descrito no Algoritmo 1, onde : m é o número máximo de características avaliadas por Split; n é o número total de árvores; δ_w é o limiar de aviso; δ_d é o limiar de mudança, C é o método de detecção de mudança, S é *data stream*, B é o conjunto de árvores em background, $W(t)$ é o peso da árvore t e $P(.)$ é a função de desempenho estimado de aprendizagem.

2.6.3 *Hoeffding Tree* – HT

Árvores Hoeffding têm sido bastante estudadas e avaliadas por representarem o estado da arte para classificação da *data streams*. O algoritmo *Hoeffding Tree* cria uma árvore de decisão a partir de um fluxo de dados incremental visualizado apenas uma única vez, sem a necessidade de armazenar exemplos posteriormente para atualizar a árvore. A única informação necessária para memória é a própria árvore, que armazena informações suficientes em suas folhas para crescer e, pode ser utilizada para realizar predições a qualquer momento a partir do processamento dos exemplos de treinamento.

Outra vantagem da *Hoeffding Tree* é a sua complexidade que apresenta custo computacional médio de $O\ mn\ logn$, em uma árvore com n exemplos e m atributos. O custo para tomar uma decisão é $O(profundidade\ da\ árvore)$ no pior dos casos, onde normalmente a profundidade de uma árvore cresce logaritmicamente com o seu tamanho (BIFET & KIRKBY, 2011).

O método funciona de acordo com o descrito no Algoritmo 2, onde as entradas são: S é uma sequência de exemplos, X é o conjunto de atributos discretos, $G(.)$ é a função de avaliação de divisão, δ é 1 menos a probabilidade desejada de escolher o atributo correto para um dado nó qualquer e, a saída é uma árvore de decisão HT (DOMINGOS & HULTEN, 2000).

Código 2.2 – *Hoeffding Tree*.

Algoritmo 2 – *Hoeffding Tree*

```
1. Procedimento HoeffdingTree (S, X, G, δ)
2. Dado HT uma árvore com uma única folha l_1 (a principal).
3. Dado X_1 = X ∪ { X_∅ }.
4. Dado G_1 = (X_∅) o G obtido pela predição da classe mais frequente classe em S.
5. Para cada classe y_k
6.       Para cada valor de x_ij de cada atributo X_i ∈ X
7.         Faça n_ijk(l_1) = 0.
8. Para cada exemplo(x, y_k) em S
9.       Ordene (x, y) em uma folha l usando HT.
10.       Para cada x_ij em x onde X_i ∈ X_l
11.          Incremente n_ijk(l)
12.       Rotule l com a classe majoritária dentre os exemplos vistos até l
13.       Se os exemplos vistos até l não forem todos da mesma classe, então
14.          Calcule G_l(X_i) para cada atributo X_i ∈ X_l − {X_∅} usando a contagem n_ijk(l).
15.          Seja X_a o atributo com o maior G_l.
16.          Seja X_b o atributo com o segundo maior G_l.
17.          Calcule ε = sqrt(R^2 ln(1/δ) / 2n).
18.          Se G_l(X_a) − G_l(X_b) > E e X_a ≠ X_∅, então
19.             Substitua l por um nó interno depois divida em X_a.
20.             Para cada galho da divisão
21.                Adicione uma nova folha l_m e faça X_m = X − {X_a}.
22.                Faça G_m = (X_∅) ser o G obtido pela predição da classe mais frequente em l_m.
23.                Para cada classe y_k e cada valor x_ij do atributo X_i ∈ X_m − {X_∅}
24.                   Faça n_ijk(l_m) = 0.
25.       Retorne HT.
```

2.6.4 *Naive Bayes* – NB

As redes bayesianas foram uma das primeiras aplicações para o aprendizado *online.* Os métodos de aprendizagem bayesianos mantêm as tabelas de probabilidade condicional P (xjy) pela contagem, em que x é um vetor de característica e y é seu rótulo. Considerando as tabelas de probabilidades condicionais e a distribuição da classe como antecedentes, a classe prevista de uma instância será aquela que maximiza a probabilidade posterior calculada pela regra de Bayes.

O modelo *Naive Bayes*, mais simples, faz a suposição "ingênua" de que cada variável de entrada é condicionalmente independente, dado o rótulo da classe. Para o aprendizado *online*, é fácil atualizar as probabilidades condicionais tanto para *Naive Bayes* e para redes bayesianas, desde que se encaixem na memória interna (BENCZUR, 2018).

É um algoritmo classificador conhecido por sua simplicidade e baixo custo computacional. Dadas classes diferentes de C, o classificador *Naive Bayes* treinado prevê, para cada instância não rotulada I, a classe C a qual pertence com alta acurácia.

O modelo funciona da seguinte forma: seja $x_l, \dots, x_k$ atributos discretos e assumindo que x_l pode ter N_l valores diferentes. Seja C o atributo classe que pode ser Nc valores diferentes. Ao receber uma instância não rotulada $l = (x_l = V_l, \dots, x_k = V_k)$, o classificador *Naive Bayes* computa uma "probabilidade" de I estar em uma classe c segundo a expressão:

$$\begin{aligned} \Pr[C = c|I] &\cong \prod_{i=1}^{k} \Pr[x_i = v_i \mid C = c] \\ &= \Pr[C = c] \,.\, \prod_{i=1}^{k} \frac{\Pr[x_i = v_i \wedge C = c]}{\Pr[C = c]} \end{aligned} \tag{2.1}$$

Os valores de $\Pr[x_i = v_i \wedge C = c]$ e $\Pr[C = c]$ são estimados a partir dos dados de treino. Assim, o resumo dos dados de treino é simplesmente uma tabela tridimensional que armazena, para cada trio (Xj, Vj, c) uma contagem $(N_{i,j,c})$ de instâncias de treinamento com $x_l = v_j$ juntamente com uma tabela unidimensional para as contagens de $C = c$.

Esse algoritmo é naturalmente incremental: ao receber um novo exemplo (ou um lote de novos exemplos), simplesmente incrementa as contagens relevantes. As predições podem ser feitas a qualquer momento a partir das contagens atuais.

2.6.5 *Perceptron* – P

O *Perceptron* é um algoritmo de aprendizado clássico para o modelo neural de aprendizagem. É um algoritmo simples e funciona bem, para alguns tipos de problemas. O *Perceptron* é um algoritmo *online*, ou seja, em vez de considerar o conjunto de dados inteiro ao mesmo tempo, ele sempre considera apenas um exemplo. Ele processa esse exemplo e, em seguida, passa para o próximo. Ele também é dirigido por erro, ou seja, desde que esteja classificando corretamente não atualiza seus parâmetros.

O algoritmos 3 apresentam o funcionamento do treinamento do *Perceptron* onde é possível observar a inicialização dos parâmetros (peso e viés) seguido pelo processamento de cada amostra, onde o algoritmo mantém um "palpite" em bons parâmetros (pesos e viés) à medida que é executado. Para um dado exemplo, faz uma previsão. Verifica se essa previsão está correta, se a previsão estiver correta, não faz nada. Somente quando a previsão está incorreta, ela altera seus parâmetros (peso e viés) e os altera de tal forma que funcionaria melhor neste exemplo da próxima vez. Em seguida, passa para o próximo exemplo. No algoritmo 4 é apresentada a função de ativação para as novas amostras de teste e retorna a classe predita.

Código 2.3 – *Perceptron* - Treino

Algoritmo 3 – *Perceptron*Treino ($D, MaxIter$)

```
1. w_d ← 0, for all d = 1...D          ► Inicializa os pesos
2. b ← 0                               ► Inicializa o viés (bias)
3. For iter = 1...MaxIter do
4.     For all (x, y) ∈ B do
5.         a ← Σ_{d=1}^{D} w_d w_d + b ► calcula a ativação para este exemplo
6.         If ya ≤ 0 then
7.             w_d ← w_d + yx_d, for all d = 1...D  ► atualiza os pesos
8.             b ← b + y               ► atualiza o viés
9.         End if
10.    End for
11. End for
12. Return w_0, w_0, ..., w_D, b
```

Código 2.4 – *Perceptron* - Teste

Algoritmo 4 – *Perceptron*Teste ($w_0, w_1, \ldots, w_D, b, \hat{x}$)

```
1. a ← Σ_{d=1}^{D} w_d x̂_d + b        ► calcula a ativação para o exemplo de teste
2. Return SIGN(a)
```

2.6.6 *Stochastic Gradient Descent* – SGD

É um método de otimização eficiente para aprender um modelo discriminativo, minimizando a função de erro como, por exemplo, *Hinge* ou *Logistic Loss*. O SGD em conjunto com outros modelos lineares tem um desempenho bom com dados esparsos e de alta dimensionalidade que são frequentemente encontrados no domínio de classificação de textos ou processamento de linguagem natural (LOSING, HAMMER e WERSING, 2018).

O algoritmo SGD calcula o gradiente em cada iteração baseado em um único exemplo escolhido aleatoriamente de acordo com a seguinte equação:

$$\omega_{t+1} = \omega_t - \gamma_t \nabla_w Q(z_t, \omega_t) \tag{2.2}$$

Calculando a média dessa atualização sobre todas as possíveis escolhas dos exemplos de treino z_t poderia restaurar o algoritmo em lote *Gradient Descent*. No entanto, a simplificação do SGD depende do grau de confiança que o ruído aleatório introduzido por este procedimento não perturbará o comportamento médio do algoritmo (BOTTOU, 1998).

O processo *stochastic* $\{\omega_t, t = 1, \ldots\}$ depende de exemplos randomicamente selecionados em cada iteração. Espera-se que a equação 2.2 se comporte como se espera na equação 2.3 apesar do ruído introduzido neste procedimento simplificado.

$$\omega_{t+1} = \omega_t - \gamma_t \frac{1}{n} \sum_{i=1}^{n} \nabla_w Q(z_i, \omega_t) \tag{2.3}$$

Como o algoritmo SGD não precisa lembrar quais exemplos foram vistos durante as iterações anteriores, ele pode processar exemplos rapidamente em um sistema. Nessa situação o SGD otimiza diretamente o risco esperado (BOTTOU, 2012). O SGD é utilizado como função em vários algoritmos, tais como Adaline, *Perceptron*, K-means, SVM, Lasso, dentre outros.

2.7 Considerações Finais

Este capítulo apresentou todos os conceitos necessários para o entendimento sobre o problema a ser tratado, dentre eles a rede social Twitter e os atributos extraíveis de um *tweet* e de uma URL. O ataque do tipo *phishing* foi descrito juntamente com o seu histórico, estatísticas dos últimos anos e o seu ciclo de vida na rede social Twitter, além dos prejuízos causados por ele. Os fundamentos sobre aprendizagem de máquina, abordagem de aprendizagem *online*. Por fim, foram apresentados os algoritmos que serão avaliados no *framework* proposto.

Capítulo 3

3 Trabalhos Relacionados

O levantamento da literatura mostra que os algoritmos de aprendizagem *online* não foram avaliados na aplicação do problema de detecção de *phishing* no Twitter. Por outro lado, encontramos na literatura vários estudos que aplicam esses algoritmos na detecção de *phishing* em *websites*. Dessa forma, há entendimento de que estes estudos devem ser incluídos como trabalhos relacionados. Portanto, este Capítulo está organizado em dois grupos: o primeiro refere-se a trabalhos que utilizam algoritmos *online* na detecção de *phishing* em *websites* e o segundo, descreve os trabalhos de detecção de *phishing* no Twitter baseados na abordagem de aprendizagem de máquina. Ao final, será apresentada uma discussão sobre os trabalhos, correlacionando-os com os objetivos desta pesquisa.

3.1 Métodos de Detecção de Phishing em Websites Utilizando o Classificadores de Aprendizagem Online

Esta seção reúne trabalhos sobre métodos de detecção de *phishing* em *websites* que utilizam a abordagem de aprendizagem de máquina por meio da aplicação de algoritmos de aprendizagem *online*.

Ma et al. (2009) apresentam uma técnica complementar de classificação, em tempo real, para prever quando uma URL está associada a um site malicioso. O trabalho demonstra a aplicação de algoritmos de aprendizagem *online Perceptron*, *Logistic Regression*, *Passivo Agressivo e Confidence Weighted* para o problema de classificação de URLs maliciosas

utilizando características léxicas da URL e características baseadas no provedor de hospedagem das páginas.

A Figura 3.1 ilustra a arquitetura do sistema proposto que inicia coletando URLs rotuladas (+/-) de provedores web que mantém suas bases atualizadas. Em seguida, para cada URL obtida, o coletor consulta, em tempo real, informações de servidores de DNS, WHOIS, lista negra e geolocalização. Além de processar as características relacionadas ao endereço IP e características léxicas das URLs para construir uma base de dados contínua. Com essas informações, o classificador realizar a predição. Se a predição realizada for diferente da classe informada, o classificador registra o erro desta rodada e constrói uma nova hipótese para a próxima rodada, atualizando o modelo.

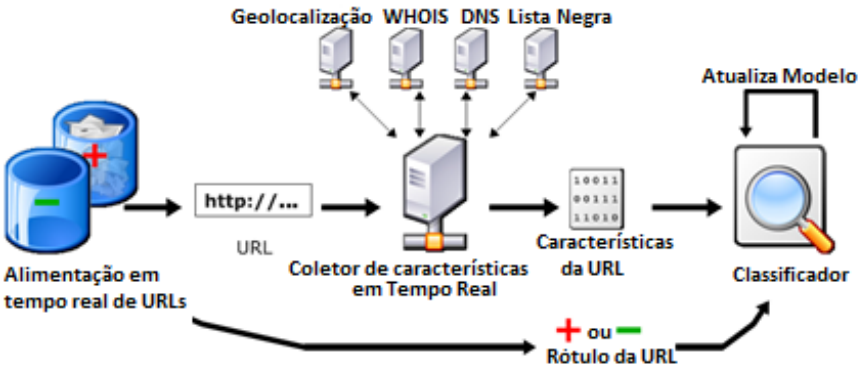

Figura 3.1 - Visão geral da alimentação de URLs em tempo real, extração de característica e infraestrutura de classificação.

Fonte: Adaptado de (MA et al., 2009)

Para avaliar o desempenho dos algoritmos de aprendizagem *online*, foram utilizados estratégias de treinamentos diferenciados e conjuntos de características variados, os quais demonstraram que o algoritmo *Confidence-Weighted* (CW) é capaz de alcançar uma acurácia de 99% utilizando um regime de treinamento contínuo e pode atualizar partes do modelo atual de forma agressiva agregando novas características sem desestabilizar o modelo anterior.

Zhang e Wang (2012) propõem uma abordagem para detecção de *websites* de *phishing* baseada na análise da URL da página, sem verificar o seu conteúdo. Este estudo explora as relações entre as características léxicas e os diferentes tipos de URLs de *phishing*, resultando em uma lista de características agrupadas em: características do provedor e características léxicas da URL. Na Figura 3.2 é possível observar que o sistema proposto é divido em quatro fases.

A primeira fase é a coleta das URLs suspeitas encontradas em várias fontes de dados (analisador de pacotes, organizações anti-*phishing*, spam, etc). Em seguida é realizada a extração e construção dos vetores de características para cada URL para gerar os repositórios de características de provedor e léxicas.

A terceira fase corresponde ao processo de treinamento diário, onde as URLs são reenviadas às organizações anti-*phishing* para a avaliação. O objetivo é checar a acurácia do modelo. Foram utilizadas três bases de dados, sendo duas bases de URLs de *phishing*, a base da APWG (A-corpus) e da Huawei Digital (S-corpus) e uma base de URLs benignas nomeada de N-corpus, proveniente do Yahoo Directory.

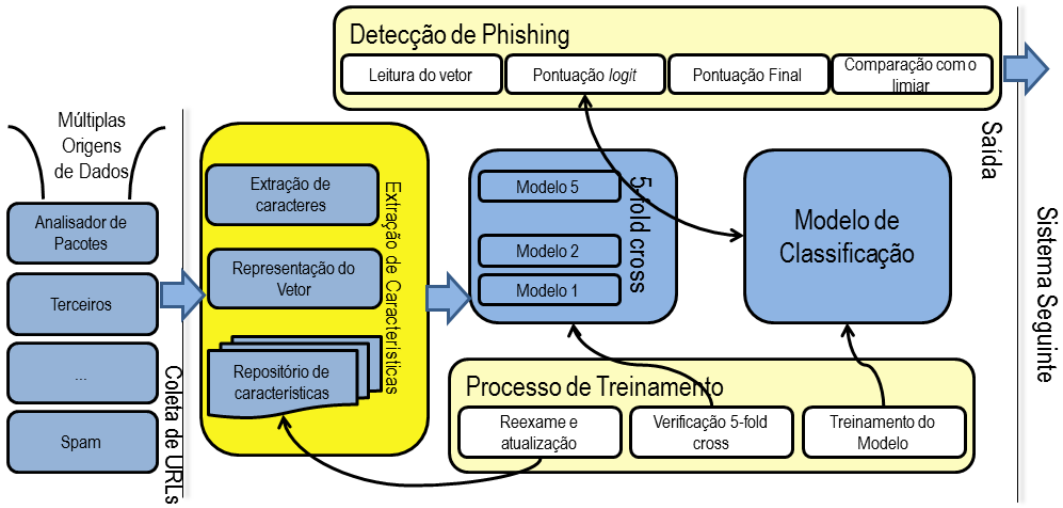

Figura 3.2 - Sistema de Detecção de Phishing em Tempo Real.

Fonte: (ZHANG; WANG, 2012).

A última fase emprega o algoritmo *online* estatístico de regressão logística que, a partir de um valor como entrada, tem como saída um valor probabilístico entre 0 e 1 e o transforma em logaritmos probabilísticos com uma pontuação final que é comparada a um limiar. Caso esta pontuação seja maior que o limiar, então a URL é classificada como *phishing*.

Para avaliar o método foram utilizadas as métricas de revocação, especificidade, acurácia e medida-F para os algoritmos *online*, Regressão Logística, *Confidence Weighted* (CW) e o Passivo Agressivo (PA). Os testes mostram que o classificador Regressão Logística apresenta a melhor taxa de acurácia que foi de 93%.

Lin et al. (2013) propõem um modelo de filtragem de URLs para remover a grande quantidade de URLs benignas, reduzindo o volume de URLs de entrada para a execução de um processo posterior, como análise de conteúdo. Este modelo utiliza apenas características léxicas e estáticas dos caracteres das URLs para a classificação e não realiza consultas à rede ou análise de conteúdo, obtendo eficiência computacional.

Para realizar a separação das URLs benignas das maliciosas, foram criados dois conjuntos para descrever as URLs. O primeiro conjunto representa características léxicas, que segundo os autores, serve apenas por um pequeno período de tempo. O segundo representa características descritivas das URLs, que são menos efetivas que o primeiro conjunto, mas funcionam por um período de tempo maior. Para extrair as características léxicas foi utilizado o modelo bag-of-words, por meio de três componentes: domínio, caminho e argumento. Cada característica gerada é armazenada em um dicionário com índice específico podendo este ser atualizado com data streaming. Para diminuir a grande quantidade de palavras, algumas técnicas foram aplicadas como remoção de palavras com peso zero, remoção de valores de argumentos, mapeamento do endereço IP para número AS (*Autonomous System*) e troca de dígitos por expressões regulares.

O conjunto de características descritivas é extraído dos componentes e dimensões da URL, como o tamanho e a razão de tamanhos entre componentes, por exemplo.

Para treinar o modelo proposto, foram utilizados dois algoritmos *online* o Passivo Agressivo (PA-I) e o *Confidence Weighted* (CW). O PA–I foi utilizado para treinar o modelo somente com o conjunto de características descritivas e o *Confidence Weighted* (CW) foi utilizado para treinar o modelo com o conjunto de características léxicas.

Durante a fase de treino, todas as instâncias de URLs rotuladas foram processadas para computar os vetores de características léxicas e descritivas. Após a extração das características, ambos vetores foram enviados para os modelos correspondentes, como podemos visualizar na Figura 3.3. Após esta fase, o valor da pontuação de URLs suspeitas foi avaliado para gerar um novo limiar para o modelo.

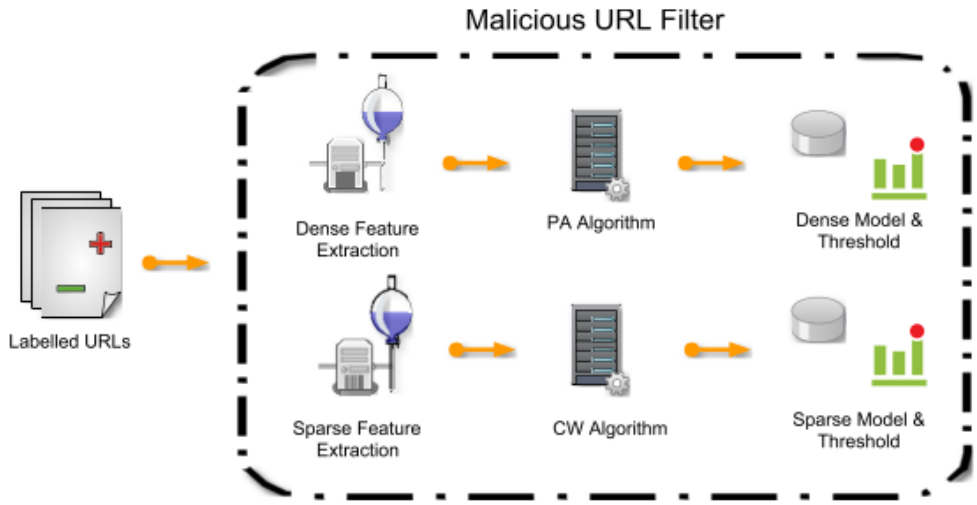

Figura 3.3 - Filtro de URL Maliciosa – Treino.

Fonte: LIN et al.(2013).

Durante o processo de classificação cada URL gera o vetor de características léxicas e descritivas que são enviados para os modelos respectivos para estimar a sua pontuação suspeita. Se alguma pontuação for maior que o limiar, então essa URL é dita como suspeita e será enviada para o sistema de *back-end*, como pode ser visualizado na Figura 3.4.

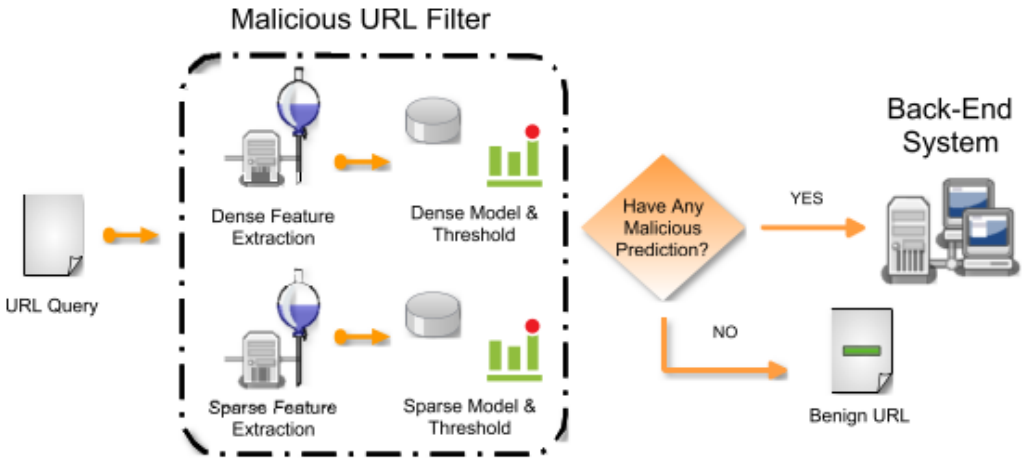

Figura 3.4 - Sistema de Filtro de URL Maliciosa – Classificação.

Fonte: (LIN et al., 2013).

Os resultados obtidos mostraram que o framework proposto é capaz de filtrar 75% das URLs processadas e das 25% remanescentes, aproximadamente 90% das URLs são maliciosas. Além do desempenho da classificação, outro ponto positivo foi o tempo de processamento de 1,5 minutos para processar mais de um milhão de URLs.

3.2 Métodos de Detecção de Phishing no Twitter com Abordagem de Aprendizagem de Máquina

Esta seção reúne trabalhos que apresentam métodos de detecção de *phishing* no *Twitter* por meio da aplicação de algoritmos de aprendizagem de máquina.

Aggarwal et al. (2012a) apresentam um classificador para detecção de *tweets* de *phishing* baseado no algoritmo *Random Forest* que é utilizado para manter atualizada uma extensão desenvolvida para o navegador Chrome[13], chamada PhishAri. O sistema proposto classifica automaticamente os *tweets* que aparecem na linha do tempo do usuário, determinando se este é *phishing* ou seguro, baseado nas características extraídas das URLs, do próprio *tweet*, da rede do usuário do *Twitter* e do WHOIS.

A Figura 3.5 mostra o processo de classificação que se inicia quando a extensão instalada no navegador do usuário envia uma requisição POST para a PhishAri API. O próximo passo é construir o vetor de características baseado nos grupos de características relacionados anteriormente. Assim que o vetor é construído, este é enviado ao modelo de classificação para realizar a sua predição, indicando se o *tweet* processado é *phishing* ou não.

Para construir o classificador, o método necessita de três estágios. O primeiro é a criação da base rotulada, no qual os *tweets* com URL são coletados e rotulados como "*phishing*" ou "seguro". Logo após, o classificador é treinado usando o algoritmo *Random Forest*.

Para a coleta dos *tweets* foi utilizada a Twitter Streaming API durante o período de 01 de fevereiro a 19 de abril de 2012 e para rotulá-los foram utilizadas as APIs do PhishTank e do Google *Safebrowsing*. Além do algoritmo *Random Forest*, também foram avaliados o *Naive Bayes* e o *Decision Trees*. Os resultados mostram que o *Random Forest* alcançou a precisão de 92,52% em 0,425 segundos, destacando-se dentre os demais.

[13] https://www.google.com.br/chrome/browser/desktop/

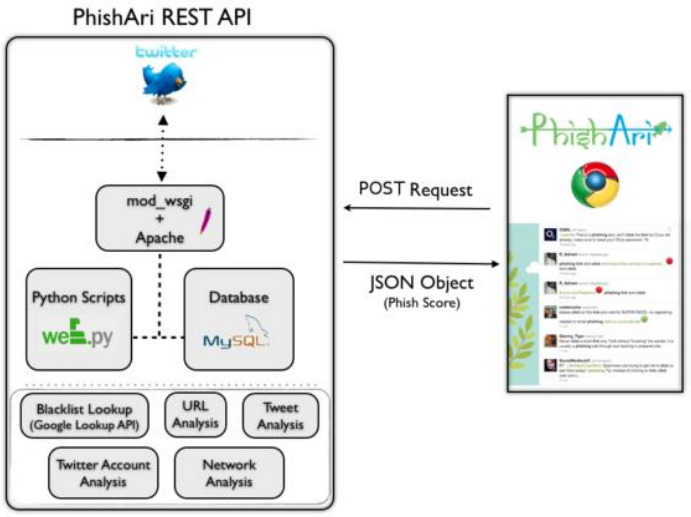

Figura 3.5 – Arquitetura do PhishAri.

Fonte: (Aggarwal, Rajadesingan, & Kumaraguru, 2012).

Lee e Kim (2012) propõem um sistema de detecção de URLs suspeitas para o *Twitter*, baseado na correlação de correntes de redirecionamento das URLs. Este sistema foi chamado de WARNINGBIRD e determina quando um *tweet* é malicioso ou não a partir do processamento de um conjunto com 12 características provenientes da URL e do contexto do *tweet*. O sistema possui quatro etapas principais apresentadas na Figura 3.6. A primeira etapa compreende a coleta de *tweets* com URLs e análise dos redirecionamentos das URLs dos *tweets* coletados armazenando-os em uma fila. Logo após, ocorre a extração de características dos *tweets* que estão na fila. Os domínios idênticos são agrupados para encontrar os pontos de entrada das URLs e gerar os vetores de características.

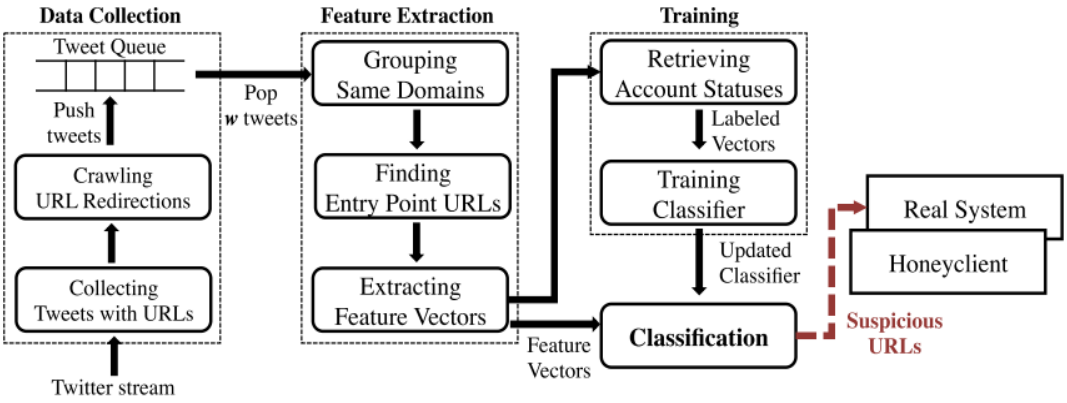

Figura 3.6 - Arquitetura do WarningBird.

Fonte: (LEE; KIM, 2012).

A terceira etapa corresponde ao treinamento, que agrupa o processo de rotulagem dos *tweets* e treinamento do classificador. Para rotular é utilizado o estado da conta do usuário, isto é, se a conta estiver suspensa, então a URL é considerada maliciosa, senão é considerada legítima.

Por fim, na etapa de classificação o vetor de características da URL é utilizado para classificar as URLs suspeitas. Quando o classificador retorna uma determinada quantidade de URLs maliciosas, os *tweets* correspondentes são classificados como suspeitos. Para a avaliação, os autores utilizaram um algoritmo de regressão logística que obteve 87,67% de acurácia.

Sharma et al. (2014) apresentam um *framework* de detecção de *phishing* baseado no algoritmo *Random Forest*. O *framework* proposto classifica automaticamente *tweets* submetidos pelos usuários em *phishing* ou legítimo, de acordo com o conjunto de características extraídas a partir da URL compartilhada no *tweet*, do próprio *tweet* e de informações provenientes do WHOIS.

Conforme exibido na Figura 3.7, o framework web é composto pelas etapas: (i) coleta do *tweet;* (ii) desencurtamento da URL para obtenção da URL longa; (iii) armazenamento dos dados; (iv) consulta aos serviços de reputação de URL (*PhishiTank*, *Google Safebrowsing* e *Mywot); (*v) extração das características; e, (vi) treino e classificação. A URL é armazenada em um banco de dados quando é classificada como *phishing*.

O *framework* proposto cria e avalia subconjuntos de características extraídas e observa que há aumento de acurácia quando todas as caraterísticas são utilizadas para classificação de *tweets*. Os autores avaliaram o *framework* web usando o algoritmo *Random Forest* que obteve precisão de 98,43% para detecção de *tweets* de *phishing* e 96,21% para detecção de *tweets* legítimos.

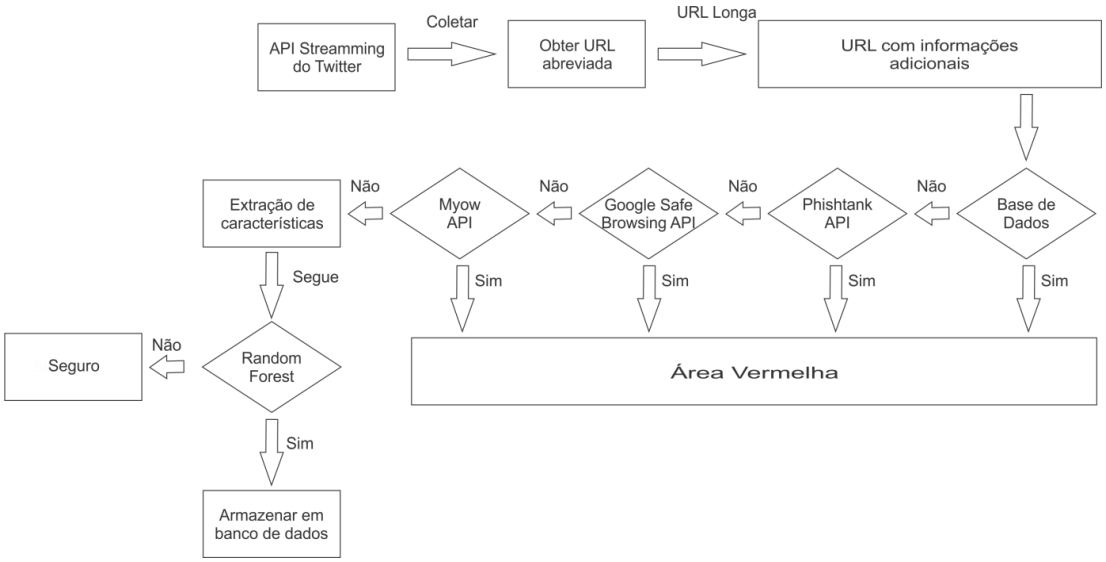

Figura 3.7 - Fluxograma do Web Framework.

Fonte: Adaptado de (SHARMA et al., 2014).

Xiaoling et al. (2014) propõem um método de detecção de fraudes, propagandas enganosas, *phishing* e disseminação de vírus no *Twitter*. O método proposto utiliza aprendizagem de máquina semissupervisionada e combina um modelo baseado em análise de agrupamento com árvore de sufixo.

Para obtenção de dados rotulados, foi proposto um módulo baseado em análise de agrupamento. O algoritmo *k-means* foi utilizado para agrupar os dados de treinamento e a distância Euclidiana foi utilizada para selecionar as amostras mais significativas para a base de treino.

Cada *tweet* foi representado por um vetor de características e cada característica foi associada com uma palavra do *tweet*. O valor de cada característica é o valor da frequência normalizada de cada palavra no *tweet*. A técnica de Análise Semântica Latente (LSA) foi utilizada para reduzir o espaço de características, uma vez que este tem dimensão grande. Assim, o procedimento de agrupamento agrupa os *tweets* maliciosos similares no mesmo.

O modelo construído tem como base critérios de informação de Akaike (AIC) e Informação Bayesiana (BIC) que foram combinados durante a fase de classificação. Os experimentos mostram que é possível alcançar uma precisão de 87% com apenas 9 amostras rotuladas e 4000 amostras não rotuladas.

3.3 Considerações Finais

Analisando os trabalhos relacionados foi possível observar que a maioria dos métodos estudados utiliza arquiteturas similares, as quais incluem as fases de coletas, rotulagem, extração de características e classificação.

As Tabelas 3.1, 3.2 e 3.3 agrupam as características do *tweet*, da URL e do usuário do Twitter, respectivamente. Ao todo foram identificadas 65 características das quais 18 foram selecionadas para serem avaliadas por serem utilizada na maioria dos trabalhos da literatura. Além disso, este trabalho de pesquisa propõe 18 novas características que a serem incorporadas no extrator de características que integra o *framework* proposto.

Tabela 3.1 – Características da URL.

Características relacionadas a URL		
ASN	*Nameserver*	Quantidade de "="
Dígitos na URL	Número de diferentes URLs de destino	Quantidade de caracteres depois do domínio
Distância de *Levenshtein*	Número de diferentes URLs iniciais	Quantidade de Nameservers
DNS_ISP	*Path tokens*	Quantidade de segmentos antes do domínio
Domínio	Posição do ponto de entrada da URL	Quantidade de subdomínios
Domínio em Lista Negra	Presença de palavras especiais	Spam-related domain name heuristics
Frequência de ponto de entrada da URL	Presença de redirecionamentos condicionais	*Spelling*
Provedor	Quantidade de "-"	Tamanho da cadeia de redirecionamentos
Idade do domínio	Quantidade de "&"	Tamanho da URL
IP do Provedor	Quantidade de "."	Tamanho do domínio
IP no domínio	Quantidade de "/"	Tempo entre a criação do domínio e a criação da conta do Twitter
Last path *tokens*	Quantidade de ";"	TLD
Latitude (cidade, região, país)	Quantidade de "?"	TTL
Localização	Quantidade de "@"	Velocidade da conexão
Longitude	Quantidade de "_"	

Tabela 3.2 - Características do Tweet

Características do Twitter
Posição das #tags
Presença de trending #tags
Presença de Geolocalização
Quantidade de #tags
Quantidade de @tags (menções)
Quantidade de *retweets*
Quantidade de URLs
Similaridade de texto
Tamanho do *tweet*

Tabela 3.3 - Características do Usuário

Característica
Conta favoritada
Conta verificada
Desvio padrão da data de criação da conta
Desvio padrão do número de amigos (seguintes)
Desvio padrão do número de seguidores
Idade da conta
Numero de amigos (seguintes)
Número de seguidores
Número de *tweets* do usuário
Presença de descrição no perfil
Razão entre seguidores/seguintes
Usuário em Listas

Capítulo 4

4 *Framework* para Detecção de *Phishing* no Twitter Baseado em Algoritmos de Aprendizagem *Online*

Este capítulo apresenta o *framework* de detecção de ataques do tipo *phishing* na rede social Twitter. O Capitulo inicia apresentando uma visão geral do *framework.* As seções seguintes descrevem os métodos e as técnicas utilizadas na concepção do framework proposto.

4.1 Visão Geral

A Figura 4.1 apresenta a visão geral do *framework* proposto para detecção de ataques de *phishing* em redes sociais *online* por meio da aplicação de algoritmos de aprendizagem *online.* O *framework* está estruturado nas seguintes fases: (i) pré-processamento; (ii) rotulagem; (iii) extração de características; e (iv) classificação. Cada etapa será detalhada nas próximas seções.

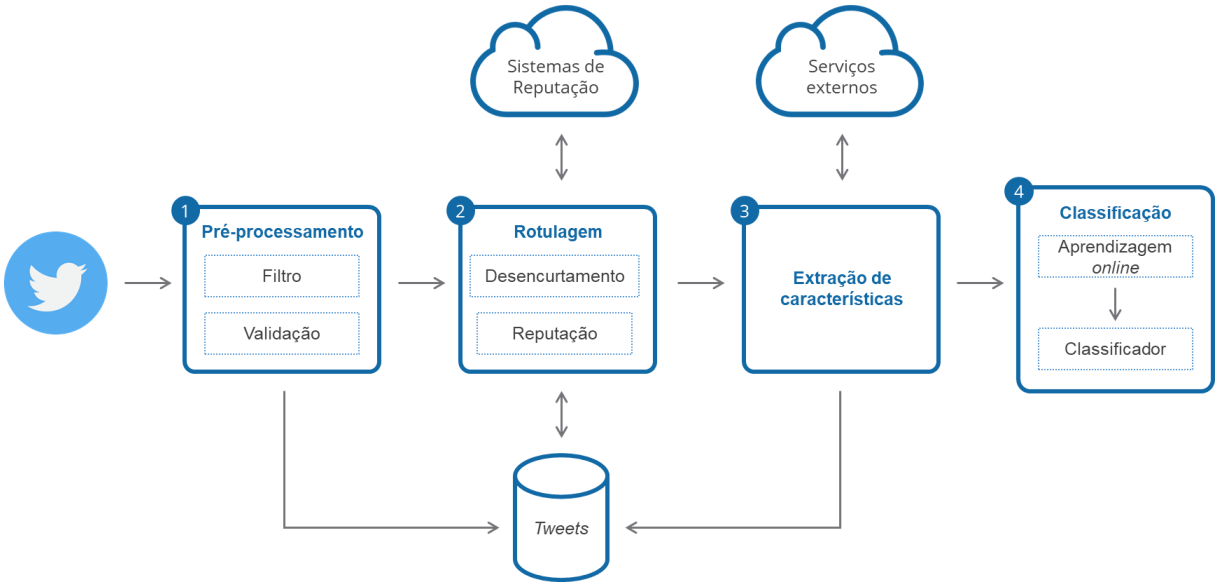

Figura 4.1 – Visão geral do framework de detecção de *phishing* no Twitter.

4.2 Pré-processamento

A etapa de pré-processamento é dividida em duas fases: (i) filtragem dos dados; e (ii) validação dos dados de entrada. Esses dados podem ser provenientes de *data streaming* do Twitter coletados a partir das APIs disponibilizadas ou de bases previamente coletadas e armazenadas em bancos de dados ou em arquivos com formato estruturado.

Na etapa de pré-processamento é realizada a filtragem de dados de entrada. Nesta etapa apenas *tweets* com URLs são armazenados, pois somente estes podem disseminar páginas de *phishing* em uma rede social por meio do compartilhamento do *link*.

Na segunda fase ocorre a validação dos dados de com o propósito de verificar a existência de *tweets* duplicados. A ideia é eliminar tais *tweet*s, evitando o processamento desnecessário de *tweets* já manipulados. Além disso, tal procedimento também ajuda a minimizar a realização de consultas desnecessárias aos serviços de reputação de URL/domínio. Após a execução da filtragem e validação, os *tweets* são armazenados em uma base de dados para serem processados nas próximas etapas.

4.3 Rotulagem

Todas as URLs contidas nos *tweets* são encurtadas pelo próprio serviço de encurtamento do Twitter. Por exemplo, o *link*: "http://t.co/zzX1zvBPXp" pode ser dividido em três partes: a primeira parte é o protocolo de comunicação utilizado, nesse caso o protocolo HTTP; a segunda parte representa o serviço utilizado para encurtamento de uma URL "t.co"; e a última parte corresponde a identificação codificada "zzX1zvBPXp" de uma página *web*.

Assim, para que o *tweet* seja rotulado como "*phishing"* ou "legítimo", o framework proposto realiza duas ações: o desencurtamento da URL e uma avaliação sobre a reputação das URLs.

No processo de desencurtamento da URL buscam-se todas as URLs existentes entre o *link* encurtado contido no *tweet* e o endereço final da página. Essas URLs formam uma cadeia de redirecionamentos e possuem um ou mais saltos. Por exemplo, a Tabela 4.1 exibe a cadeia de redirecionamento da URL "http://t.co/zzX1zvBPXp".

Tabela 4.1 - Exemplo de URL encurtada e sua cadeia de redirecionamentos.

Saltos	Cadeia de Redirecionamentos
1	**http://t.co/zzX1zvBPXp**
2	http://i.mixcloud.com/cctgor
3	https://www.mixcloud.com/short/cctgor
4	http://www.mixcloud.com/thepipepadre/the-venerable-archbishop-fulton-j-sheen-on-am-530-ciao-april-7-2013/?utm_source=redirect&utm_medium=shorturl&utm_campaign=cloudcast
5	**https://www.mixcloud.com/thepipepadre/the-venerable-archbishop-fulton-j-sheen-on-am-530-ciao-april-7-2013/?utm_source=redirect&utm_medium=shorturl&utm_campaign=cloudcast**

Após obtenção das URLs da cadeia de redirecionamentos. A etapa seguinte corresponde na realização de consultas a sistemas de reputação utilizando as URLs obtidas. Na implementação de um protótipo desse framework serão utilizados os serviços de consultas de organizações anti-*phishing* como PhishiTank, OpenPhish, APWG, VirusTotal, dentre outros. É comum que a URL consultada não seja conhecida por estes serviços, sendo necessário o seu cadastro e posterior reconsulta.

Quando a reputação da URL é fornecida durante a consulta, os *tweets* que compartilham aquela URL são rotulados como *phishing* ou legítimos dependendo do rótulo obtido. Esta etapa é repetida diariamente, até obtenção da reputação de um link. Para efeito de limitação de tempo essa consulta é realizada não ultrapassando o limite de cinco dias. Se o limite for atingido e o rótulo não for obtido, o *tweet* é eliminado da base de dados.

4.4 Extração das Características

Nesta etapa os vetores de características dos dados rotulados são gerados a partir da extração de informações dos *tweets* processados nas etapas anteriores. Para este trabalho, as características foram identificadas baseando-se nos trabalhos relacionados, elas foram agrupadas em três categorias: (i) características do *tweet*; (ii) características do usuário do *Twitter* e (iii) características a partir da URL.

As características baseadas no *tweet* são relacionadas ao texto contido na mensagem e nos atributos do *tweet* tendo como finalidade identificar similaridades na forma com que os usuários criam as suas mensagens. O conjunto de características extraídas dos usuários do Twitter é baseado nas informações do próprio usuário e na sua rede de contatos, tendo como finalidade identificar similaridades no seu comportamento. Por fim, o conjunto de características extraídas a partir de informações obtidas por meio da URL longa corresponde à informações léxicas da URL e de informações obtidas por meio de requisições HTTP aos serviços de informações sobre domínios e provedores. A seguir, essas características são apresentadas e descritas.

4.4.1 Características do Tweet

Quantidade de *hashtags* (no_hashtag)

Hashtags são palavras precedidas do símbolo "#" (jogo da velha ou cerquilha). Essa característica é de uso comum e constante no Twitter, também é utilizada para disseminação de assuntos do momento (*trend topics*) ou de palavras-chaves. *Trend topic* é um termo utilizado para destacar um "assunto quente" dentre os demais, ou para chamar a atenção de

usuários. *Phishers* ou *bots* fazem uso de *hashtag* e também de *trend topics* no texto do *tweet* para aumentar as chances do *tweet* malicioso alcançar um número maior de usuários. Essa característica é proveniente do *tweet* e é obtida a partir do atributo *"hashtag"* contido no *tweet* e o seu tipo de dados é um valor numérico inteiro.

Quantidade de usuários mencionados (no_usermention)

É possível mencionar usuários em *tweets*, para isso faz-se necessário o uso do símbolo "@" precedendo o apelido do usuário que se pretende mencionar. Essa característica é utilizada pelos *phishers* para chamar a atenção de determinados usuários e propagar seus *tweets*, uma vez que os *tweets* criados por um usuário, automaticamente, aparecem nas linhas do tempo dos seus seguidores. Tal característica é obtida a partir do atributo "user.mentions" contido no *tweet* e o seu tipo de dados é um valor numérico inteiro.

Quantidade de re*tweet*s (no_re*tweet*s)

Esta característica está relacionada à rede de propagação do *tweet* por meio do compartilhamento da mensagem, ou seja, quanto mais *retweets*, mais usuários compartilharam a mensagem. É comum o uso de robôs automáticos (bots), para ajudar no compartilhamento de um *tweet*. Esses bots são criados pelos *phishers* e tem por objetivo aumentar a sua rede de contatos e propagar mensagens falsas e/ou maliciosas. Essa característica é obtida pelo valor do atributo "re*tweet*_count" contido no *tweet* que originou a mensagem. Seu tipo de dados é um valor numérico inteiro.

Tamanho da mensagem (no_char)

Essa característica está relacionada ao tamanho da mensagem e é calculada a partir da contagem da quantidade de caracteres contida no *tweet*. Os *tweets* criados por robôs apresentam padrões de escrita e de similaridade em tamanho e conteúdo. Essa característica é obtida por meio da contagem da quantidade de caracteres contidos no atributo "*text*" do *tweet*. Seu tipo de dados é um valor numérico inteiro.

4.4.2 Características do Usuário

Quantidade de seguidores (no_follower)

Essa característica está relacionada ao comportamento do usuário em relação a sua rede de contatos. Usuários maliciosos buscam aumentar a sua rede de contatos, para propagação dos *tweets* maliciosos, visto que todos os *tweets* que são postados em sua linha do tempo, automaticamente, são exibidos também na linha do tempo dos seus seguidores. Essa característica é obtida a partir do atributo "user.follower" contido no *tweet* que apresenta a quantidade de seguidores que o usuário possui. Seu tipo de dados é um valor numérico inteiro.

Quantidade de usuários que são seguidos (no_following)

Essa característica, assim como a anterior, está relacionada ao comportamento do usuário em relação a sua rede de contatos. Essa característica é obtida a partir do atributo "user.following" contido no *tweet* que apresenta a quantidade de usuários que são seguidos por este o usuário. Seu tipo de dados é um valor numérico inteiro.

Listas do usuário (no_lists)

É possível criar e/ou participar de grupos de contatos, no Twitter esses grupos são conhecidos como listas. Essa característica verifica se o usuário faz parte de alguma lista. Usuários maliciosos buscam demonstrar comportamentos similares a usuários comuns e criam e/ou aderem a listas para disseminar mensagens para grupos específicos. Essa característica é obtida a partir do atributo "user.lists" contido no *tweet*. Seu tipo de dados é um valor booleano.

Idade da conta (account_age)

Essa característica esta relacionada ao tempo de vida da conta do usuário do Twitter. Contas de usuários maliciosos tendem a ter tempo de vida curto, pois são bloqueadas assim que são descobertos comportamentos maliciosos e/ou são desativadas ou excluídas pelos próprios criadores quando terminam o ciclo de vida do ataque do tipo *phishing* no Twitter. Essa

característica é calculada a partir da diferença, em dias, entre a data de criação da conta e a data da criação do *tweet*. Seu tipo de dados é um valor numérico inteiro.

Quantidade de *tweets* do usuário (no_*tweet*s)

Um comportamento característico de usuários maliciosos é a alta frequência de postagens. O uso de robôs para disseminação de *tweets* maliciosos também aumentam a quantidade de *tweets* enviados, pois seguem um padrão de envios. Essa característica é obtida a partir do atributo "user.no_*tweet*s" contido no *tweet*. Seu tipo de dados é um valor numérico inteiro.

4.4.3 Características da URL

Número de URLs no Tweet (no_urls)

Essa característica contabiliza a quantidade de URLs contidas no texto do *tweet*. Para Yeong & Jeong (2016), essa é uma das características mais importantes na detecção de *tweets* maliciosos, pois apresentam o padrão do uso de robôs na criação de mensagens para disseminação de *links* maliciosos. Essa característica é obtida a partir da análise do texto na mensagem e é obtida por meio da contagem da quantidade de URLs curtas presentes. Seu tipo de dado é um valor numérico positivo.

Tamanho da URL (url_expanded_size)

Aggarwal et.al (2012) e Jeong (2016) afirmam que o tamanho de URLs maliciosas tende a ser maior que URLs legítimas e, portanto, é uma característica considerada discriminativa para identificar URLs maliciosas de ataques do tipo *phishing*. Esses mesmos estudos descrevem que os usuários maliciosos utilizam muitos caracteres e componentes como domínios, subdomínios, diretórios, parâmetros, caracteres especiais para compor o endereço da página maliciosa e, assim, esconder ou disfarçar informações contidas na URL. Essa característica é obtida a partir da contagem do número de caracteres existentes na URL longa. Seu tipo de dados é um valor numérico inteiro.

Quantidade de subdomínios da URL (url_number_subdomain)

A utilização de vários subdomínios é uma prática comum em URLs maliciosas. O atacante utiliza esse artifício para confundir visualmente o domínio e o destino final para o qual o usuário será direcionado. A utilização de muitos subdomínios auxilia no crescimento do tamanho da URL e comprova a importância discriminativa da característica descrita no item anterior. Essa característica é obtida a partir da contagem do número de subdomínios existentes na URL longa. Seu tipo de dados é um valor numérico inteiro.

Presenças de números na URL (url_contain_numbers)

Essa característica busca a presença de números na URL longa. Essa técnica é utilizada para disfarçar o domínio, subdomínio da URL ou palavras contidas na URL longa deixando-o semelhante ao de uma URL legítima, utilizando números ao invés de letras. Para melhor entendimento dessa característica, temos a palavra *paypal* no qual o *phisher* pode reescrevê-la da seguinte forma: paypa1 ou p4yp4l. O uso de números não é somente para reescrever palavras, mas para criar domínios ou subdomínios de maneira automatizada. Os domínios criados por robôs, usualmente, seguem um padrão como, por exemplo: w001.dominio.com, w002.dominio.com. Essa característica é obtida a partir da verificação da presença de números na URL longa. Seu tipo de dados é booleano, onde 0 representa a ausência e 1 a presença de números na URL.

Quantidade de símbolos especiais (url_count_slashes, url_count_question_mark,url_count_points,url_count_equal_sign,url_count_hyphen,url_count_underlines,url_count_ats,url_count_semicolon)

Aggarwal et.al (2012) e Jeong (2016) demostram que a presença de caracteres e símbolos especiais prevalece em URLs maliciosas por serem maiores do que em legítimas. Dessa forma este trabalho contabiliza separadamente cada um dos seguintes símbolos (“/”, “?”, “.”, “=”, “-”, “_”, “@” e “;”) gerando oito características distintas, sendo elas: quantidade de barras, quantidade de interrogações, quantidade de pontos, quantidade de iguais, quantidade de hífenes, quantidade de sublinhados, quantidade de arrobas e quantidade de ponto e vírgulas.

Essas características fazem parte da categoria de características léxicas da URL. Seu tipo de dados é um valor numérico para cada característica processada.

Presença de dígitos no domínio (no_digits)

Essa característica busca a presença de números em domínios e subdomínios da URL. Essa técnica é utilizada para disfarçar o domínio ou subdomínio da URL deixando-o semelhante ao de uma URL legítima, utilizando números ao invés de letras. Essa característica é obtida a partir da verificação da presença de números em subdomínios e domínios na URL longa. Seu tipo de dados é booleano, onde zero representa a ausência e 1 a presença.

Palavras especiais (url_special_words)

Essa característica busca a presença de palavras usualmente utilizadas em URLs maliciosas como "*paypal", "amazon", "secure", "account", "webscr", "ebay", "login", "ebaysapi", "signin", "banking", "confirm" e "password*", na URL. Essas palavras representam marcas alvo de ataques ou termos relacionados com segurança. Essa característica é obtida a partir da verificação da presença das palavras especiais mapeadas na URL longa. Seu tipo de dados é booleano, onde zero representa a ausência e 1 a presença.

Caracteres após o domínio (url_characters_after_domain)

Essa característica conta a quantidade de caracteres presentes na URL após o domínio primário. Usualmente, URLs maliciosas possuem tamanhos longos para ofuscar o entendimento do seu destino final fazendo com que o usuário não perceba alterações ou redirecionamentos. Essa característica léxica é obtida a partir da identificação do domínio primário e logo em seguida conta-se a quantidade de caracteres seguintes. Seu tipo de dados é um valor numérico positivo.

Caracteres antes do domínio (url_characters_before_domain)

Essa característica conta a quantidade de caracteres presentes na URL antes do domínio primário. Observa-se que URLs maliciosas podem apresentar muitos subdomínios além do considerado normal para uma URL legítima. Antes do domínio podem-se inserir números

e/ou palavras para ludibriar o usuário. Essa é uma característica utilizada para esconder o verdadeiro domínio da URL. Essa característica é obtida a partir da identificação do domínio primário e logo em seguida conta-se a quantidade de caracteres do início da URL até encontrar o domínio primário. Seu tipo de dados é um valor numérico positivo.

Tamanho do domínio (url_host_size)

Essa característica conta a quantidade de caracteres no domínio primário. Observa-se que URLs maliciosas podem apresentar domínios com uma quantidade grande de caracteres quando comparados a uma URL legítima. Essa técnica de utilização de domínios grandes serve para inserir letras ou números em palavras conhecidas ou comuns e desviar a atenção do usuário e escondendo o verdadeiro domínio da URL. Essa característica é obtida a partir da identificação do domínio primário e logo em seguida conta-se a quantidade de caracteres deste. Seu tipo de dados é um valor numérico positivo.

Identificação do TLD (url_tld_id)

Essa característica identifica o TLD (do inglês, *Top Level Domain*), que significa lista de domínios da internet de nível superior. Sua sigla em português é DPN que significa Domínio de Primeiro Nível e serve para identificar a origem de registro da URL, ou seja, o código do país. Segundo Cunha (2012), os domínios podem ser registrados em alguns TLD sem restrições enquanto outros devem seguir alguns pré-requisitos. Devido a questões de gerenciamento, determinados TLD podem oferecer maior facilidade de serem usados por *phishers*. Essa característica léxica da URL é obtida a partir da identificação do TLD, logo em seguida a conversão numérica é feita baseada na ISO 3166-1 que sugere códigos para países. Seu tipo de dados é um valor numérico positivo.

IP do servidor de nome (url_nameserver_ip)

Essa característica identifica o endereço IP do DNS do servidor que hospedará o site e serve para identificar a origem de registro da URL. Assim como na característica anterior, alguns domínios podem utilizar servidores DNS em listas negras ou não confiáveis, pois apresentam maior facilidade de serem usados pelos atacantes. Essa é uma característica relacionada à localização de hospedagem do site e é obtida a partir da identificação do IP do primeiro

servidor de DNS encontrado e em seguida os pontos de separação dos bits do IP são retirados. Seu tipo de dados é um valor numérico longo.

Quantidade de servidores de nome (url_nameserver_count)

Essa característica conta a quantidade de servidores DNS registrados para hospedar o site. Domínios legítimos, geralmente, registram mais de um DNS para manterem o site disponível caso um dos DNS esteja fora do ar. Por outro lado, domínios maliciosos tendem a manter o site hospedado em provedores não confiáveis e por um curto período de tempo não se atentando em manter o site em vários servidores DNS. Essa característica é obtida a partir da contagem de servidores DNS registrados no provedor. Não são todos os provedores que disponibilizam essa informação dificultando a extração desta característica. Seu tipo de dados é um valor numérico.

Idade do domínio (url_domain_age)

Essa característica processa a idade do domínio da URL. Estudos apontam que páginas de *phishing* possuem curto período de disponibilidade e logo saem do ar para não serem rastreadas. Essa característica é bastante explorada por *phishers* que criam e excluem páginas constantemente para não serem detectados. Essa é uma característica é obtida a partir do cálculo da diferença em dias entre a data que a URL foi processada e a data de criação do domínio. Seu tipo de dados é um valor numérico positivo.

País, latitude, longitude do provedor do domínio (url_city_longitude, url_city_latitude,url_nameserver_geolocation_country)

Essas características estão relacionadas à localização do provedor que hospeda a página de *phishing*. Alguns autores apontam países que possuem provedores com poucas restrições quanto à criação de domínios e hospedagens de sites, tornando-se provedores bastante explorados para páginas maliciosas. Essas características são obtidas a partir do IP do provedor e da consulta a bases de dados com informações de georreferenciamento. Seus tipos de dados são valores numéricos positivos.

ASN do domínio (url_nameserver_ASN)

Essa característica está relacionada à rede associada a um provedor que hospeda a página de *phishing*. O ASN (do inglês, *Autonomous System Number)* é um identificador único de redes na Internet. Assim como na característica anterior, existem países que possuem provedores com poucas restrições quanto à criação de domínios e hospedagens de sites e o ASN pode identificar a qual rede global esse provedor faz parte. Essa é uma característica é obtida a partir do IP do provedor e da consulta a bases de dados com informações de ASN de redes globais. Seu tipo de dado é um valor numérico positivo.

Número IP na URL (url_is_ipaddress)

Essa característica observa se a URL apresenta explicitamente o número do endereço IP. Ela é pouco utilizada em URLs legítimas, porém bastante utilizada por *phishers* que preferem não registrar um domínio para a página maliciosa. Essa é uma característica é obtida a partir da validação da presença ou não de um IP válido ao invés de um domínio. Seu tipo de dado é um valor booleano onde 1 representa a presença de IP e 0 a ausência.

O processamento de extração das características apresentado permite a criação de vetores de características que serão utilizados para gerar os modelos de classificação e, como consequência, classificar os *tweets* em *phishing* ou legítimo. No total, foram apresentadas 36 características extraídas dos *tweets*, dos usuários e de URLs.

4.5 Classificação

A Figura 4.2 exemplifica o funcionamento da etapa de classificação *online* do *framework* proposto que ocorre da seguinte forma: seja S um fluxo de dados contínuo de *tweets* processados e os exemplos rotulados são $\{x^t, y^t\}$, tem se que para $t = 1, 2, \ldots, T$, onde x é o vetor de características e y é o rótulo da classe onde ($y \in \{K_1, \ldots, K_l\}$). Um novo exemplo x^t é classificado pelo classificador S, que prevê um rótulo. Sendo uma classificação supervisionada, logo após a predição do rótulo da classe, o rótulo verdadeiro y^t se faz conhecido e pode ser utilizado para atualizar o classificador.

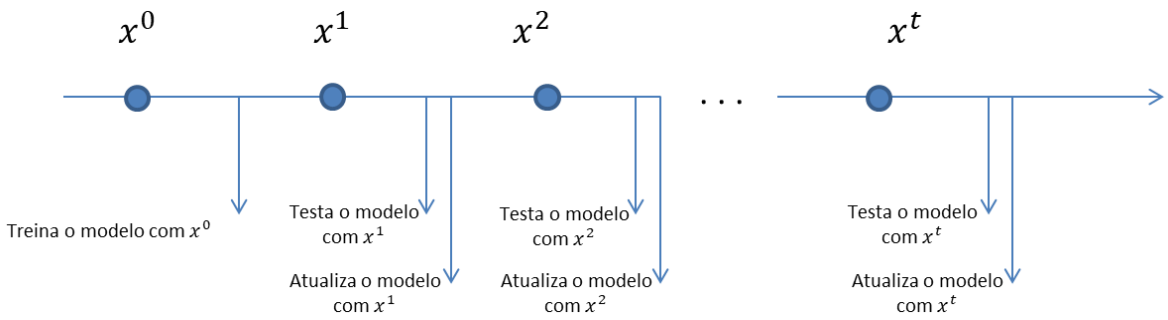

Figura 4.2 - Etapa de classificação *online* do framework proposto.

A Figura 4.2 exemplifica o funcionamento da etapa de classificação *online* do *framework* proposto que ocorre da seguinte forma: seja S um fluxo de dados contínuo de *tweets* processados e os exemplos rotulados são $\{x^t, y^t\}$, tem se que para $t = 1, 2, \ldots, T$, onde x é o vetor de características e y é o rótulo da classe onde ($y \in \{K_1, \ldots, K_l\}$). Um novo exemplo x^t é classificado pelo classificador S, que prevê um rótulo. Sendo uma classificação supervisionada, logo após a predição do rótulo da classe, o rótulo verdadeiro y^t se faz conhecido e pode ser utilizado para atualizar o classificador.

A qualidade do classificador é relacionada a uma função de erro que mede a discrepância entre a predição realizada e a resposta correta.

O objetivo do classificador é minimizar este erro cumulativo ao longo da sua execução e, para isso ele pode atualizar seu conjunto de hipóteses logo após receber o rótulo correto e, assim, ele pode ser mais preciso na predição do *tweet* seguinte (SHALEV-SHWARTZ, 2007).

Os algoritmos de classificação *online* avaliados no *framework* foram baseados na literatura e análise dos algoritmos utilizados nos trabalhos relacionados: *Adaptive Random Forest, Hoeffgin Tree, Naive Bayes, Perceptron e Stochastic Gradient Descent*.

4.6 Considerações Finais

Os modelos de detecção de *tweets* de *phishing* atuais apresentam dificuldades em manter o classificador atualizado devido ao grande volume de dados gerado pelo Twitter, além das mudanças de estratégias na execução do ataque. Visando apresentar uma solução que possa se

adaptar rapidamente ao problema de detecção de *phishing* no Twitter, este trabalho propôs um framework de detecção baseado em classificadores de aprendizagem *online*.

Capítulo 5

5 Experimentos e Resultados

Neste capítulo serão apresentados os resultados obtidos a partir dos experimentos realizados com um protótipo do Framework de Detecção de *Phishing* no Twitter proposto. Os modelos de classificação *online* implementados no protótipo são comparados e avaliados usando duas bases de dados. O capítulo inicia descrevendo o protocolo experimental, as bases de dados utilizadas, o processo de rotulagem dos *tweets* e o processo de treinamento *online.* Por fim, o capítulo apresenta os resultados da avaliação dos modelos de classificação.

5.1 Protocolo Experimental

5.1.1. Detalhes de Implementação

Todos os experimentos foram realizados em um servidor Debian x86_94, 64-bit, 96 gigabytes de memória RAM com 1.5 TB de HD e 24 processadores Intel Xeon de 2.53 GHz. Foram utilizadas a *Filter Realtime Tweets API* para coleta dos *tweets.* Para os *scripts* de coleta e extração foi utilizada a linguagem Python nas versões 2.7 e 3.5. Para a rotulagem utilizou-se scripts *shell linux* e VirusTotal *Public* API v2.0.

Para coletar os *tweets* foi desenvolvido o *script* "*crawller_twitter.py*" escrito em linguagem Python[14] baseado na *Filter Realtime Tweets API*[15] disponibilizada pelo Twitter que

[14] https://www.python.org/

[15] https://developer.twitter.com/en/docs/*tweets*/filter-realtime/overview/statuses-filter

permite a utilização de filtros do conteúdo *streaming* e retorna informações em formato JSON (*JavaScript Object Notation*).

Para ter acesso à API é necessário possuir uma conta de usuário do Twitter e registrar uma aplicação no site de Gerenciamento de Aplicações[16] para obter as credenciais que são as chaves e *tokens* de autenticação de acordo com a especificação do protocolo OAuth[17]. A conexão é realizada por meio de requisições HTTP entre a aplicação cliente validando o *header* que contém informações do usuário e da aplicação.

Os *tweet*s foram coletados sem restrição de idioma ou localização geográfica. Entretanto foram considerados somente *tweets* com *links*. O processo de rotulagem foi baseado no serviço de reputação do VirusTotal, semelhante ao utilizado nos trabalhos de Nikiforakis (2014); Gupta (2014) e Chen (2014). Uma das vantagens deste serviço é o fato dele realizar mais de 66 outros serviços de consulta de reputação, como *PhishTank, CleanX, Google SafeBrowser, BitDefender, Kaspersky*, dentre outros. Esse serviço oferece a consulta gratuitamente pelo seu *site*, porém com restrição de quantidade de requisições diárias. Para este trabalho foi possível realizar até cinco milhões de requisições por dia.

As características são extraídas conforme descrito na Seção 4.4. As características foram separadas em três grupos, características do *tweet*, do usuário e a partir da URL. Para obter as características do *tweet* são realizadas consultas aos documentos das coleções armazenadas numa base de dados MongoDB disponibilizada em um servidor local. Por outro lado, as características extraídas a partir de informações da URL possuem scripts de extração que realizam requisições à internet para obtenção de dados, como datas de registro da URL, IP, *nameservers* dentre outras.

Após o processo de extração das características, os vetores são criados e salvo em arquivo do tipo .arff para que sejam processados na etapa de classificação. Na implementação do protótipo do framework forma utilizados os seguintes algoritmos de classificação *online: Naive Bayes, Perceptron, Adaptive Random Forest, Stochastic Gradient Descent* (SGD) e *Hoeffding Tree*. Estes algoritmos foram selecionados a partir da análise dos trabalhos relacionados descritos no Capítulo 3. Para realizar o processo de treino e teste adotamos o

[16] https://apps.twitter.com

[17] https://developer.twitter.com/en/docs/basics/authentication/guides/authorizing-a-request

framework Massive Online Analysis – MOA[18] utilizado para análise e avaliação de algoritmos de classificação *online*.

Para avaliação dos classificadores foram utilizados os seguintes parâmetros:

- *Adaptive Random Forest*s: foi avaliado utilizando-se todas as características para divisão das árvores, número total de árvores em *background* igual a 50 e os demais parâmetros foram o padrão sugerido pelo MOA. EvaluatePrequential -l (meta.AdaptiveRandomForest -l (ARFHoeffdingTree -k 7 -e 2000000 -g 50 -c 0.01)) -s (ArffFileStream -f (base.arff)) -e (WindowClassificationPerformanceEvaluator -o -p -r -f) -f 1000 -d (resultadoARF.csv) -o (resultadoARF.pred) -O (resultadoARF.moa)
- *Hoeffding Tree*: todos os parâmetros padrões já predefinidos pelo MOA. EvaluatePrequential -l trees.HoeffdingTree -s (ArffFileStream -f (base.arff)) -e (WindowClassificationPerformanceEvaluator -o -p -r -f) -f 1000 -d (resultadoHT.csv) -o (resultadoHT.pred) -O (resultadoHT.moa)
- *Naive Bayes*: : todos os parâmetros padrões já predefinidos pelo MOA. EvaluatePrequential -l bayes.NaiveBayes -s (ArffFileStream -f (base.arff)) -e (WindowClassificationPerformanceEvaluator -o -p -r -f) -f 1000 -d (resultadoNB.csv) -o (resultadoNB.pred) -O (resultadoNB.moa)
- *Perceptron*: todos os parâmetros padrões já predefinidos pelo MOA. EvaluatePrequential -l functions.Perceptron -s (ArffFileStream -f (base.arff)) -e (WindowClassificationPerformanceEvaluator -o -p -r -f) -f 1000 -d (resultadoP.csv) -o (resultadoP.pred) -a 1.0 -O (resultadoP.moa)
- *Stochastic Gradient Descent*: : todos os parâmetros padrões já predefinidos pelo MOA. EvaluatePrequential -l (functions.SGD -l 0.0 -r 0.0) -s (ArffFileStream -f (base.arff)) -e (WindowClassificationPerformanceEvaluator -o -p -r -f) -f 1000 -d (resultadoSGD.csv) -o (resultadoSGD.pred) -O (resultadoSGD.moa)

[18] Framework escalável de código aberto utilizado para mineração de data streaming

5.1.2. Bases de Dados

Base ICC_P

Para avaliar os modelos de classificação foi utilizada a Base de dados ICC que contém características maliciosas de usuários *spammers* do Twitter, disponibilizada por (CHEN et al., 2015). Para garantir que os *tweets* possuíam links de *phishing* e legítimos foram adicionadas às instâncias de usuários *spammers* as características relativas à URL provenientes das bases do PhishTank, OpenPhish (*phishing*) e da DMOZ (legítimos). As URLs foram coletadas no período de 20/10/2017 a 27/10/2017.

A base ICC foi utilizada em estudos de detecção de *spammers* no Twitter e escolhida por apresentar características de usuários *spammers*, que segundo (CHEN et al., 2016) compartilham comportamentos semelhantes aos usuários *phishers* do Twitter.

Base BW

A base de dados BW foi gerada entre os dias 20/11/2017 e 27/11/2017 na semana conhecida como *Black Week* que comporta dois eventos importantes para o comércio, *Black Friday*[19] e *Cyber Monday*[20] que ocorreram nos dias 24 e 27 de novembro de 2017, respectivamente. Esse período de coleta para criação da base foi escolhido por aumentar as possibilidades de compartilhamentos de *tweets* maliciosos, especificamente com *links* para páginas de *phishing*. A Figura 5.1 mostra a distribuição dos *tweets* rotulados durante os oito dias da *Black Week*.

[19] Black Friday: Sexta-feira negra (em português) representa o dia que inaugura a temporada de compras natalícias com promoções.

[20] Cyber Monday : Segunda-feira após a Sexta-feira negra, representa a ação de vendas on-line quando as lojas oferecem altos descontos.

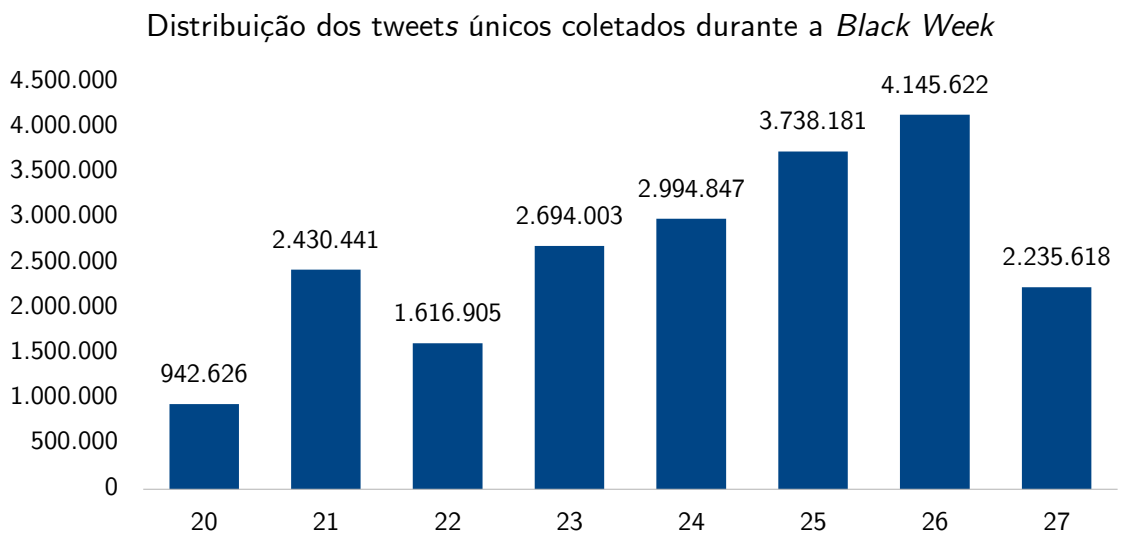

Figura 5.1 - Tweets Coletados durante a Black Week.

Fonte: O autor, 2017.

Foram coletados 33.240.501 *tweets* e durante o processamento dessas informações, observamos que a coleta havia buscado cerca de 13 milhões de *tweets* duplicados que foram descartados. Após a limpeza realizada restaram 20.798.243 *tweets* únicos que foram divididos em oito coleções de documentos (bases de dados do Mongodb) separadas por dia de coleta, conforme mostra a Figura 5.1. Ao longo de 26 dias, as URLs foram pré-processadas, ou seja, desencurtadas e rotuladas. Ao final deste processo obtivemos 404 URLs longas únicas rotuladas como *phishing* as quais compartilhadas em 43.643 *tweets*. A Tabela 5-1 detalha os dados obtidos para a geração da base BW utilizada para a avaliação do método proposto.

Tabela 5-1 - Estatística dos dados coletados.

Dados	Quantidade
Total de *Tweets* coletados durante a *Black Week*	33.240.501
Total de *Tweets* únicos coletados durante a *Black Week*	20.798.243
Total de *Tweets* rotulados como *phishing*	43.643
Total de *Tweets* rotulados como legítimos	20.154.700
Total de URLs curtas (*phishing*)	15.864
Total de URLs curtas (únicas)	10.784.525
Total de URLs longas obtidas	30.174.039
Total de URLs longas de *phishing* (únicas)	404

5.1.3 Métricas

A avaliação dos classificadores *online* incluídos no *framework* proposto foi baseada nas seguintes métricas:

- **Verdadeiro Negativo (VN):** apresenta a quantidade de *tweets* legítimos classificados corretamente.
- **Verdadeiro Positivo (VP):** apresenta quantidade de *tweets* de *phishing* classificados corretamente;
- **Falso Negativo (FN):** apresenta quantidade de *tweets* de *phishing* classificados erroneamente como legítimos;
- **Falso Positivo (FP):** quantidade de *tweets* legítimos classificados erroneamente como *tweets* de *phishing*;
- **Acurácia**: é definida pela razão entre quantidade de *tweets* corretamente classificados e a quantidade total de *tweets*.

$$Acurácia = \frac{VP + VN}{VP + VN + FP + FN} \tag{5.1}$$

- **Acurácia prequencial:** também conhecida como teste-treino intervalado, na qual cada amostra pode ser utilizada para testar o modelo antes de ser utilizada para treino e então, a sua acurácia é atualizada de forma incremental. A regra utilizada para calcular a acurácia prequencial na iteração t é dada pela seguinte equação:

$$Acc_{média} = \begin{cases} acc_{ex}(t), & se\ t = f \\ acc(t-1) + \dfrac{acc_{ex}(t) - acc(t-1)}{t - f + 1}, & caso\ contrário \end{cases} \tag{2.7}$$

  Onde acc_{ex} é 0 se a predição da amostra corrente ex antes do aprendizado estiver errada e 1 se estiver correta; e f é a primeira iteração usada no cálculo.

- **RAM-Hours:** é definida como a medida como a memória é utilizada, ou seja, 1 Gb de memória utilizada em 1 hora corresponde a 1 RAM-Hour.
- **CPU Time:** é o tempo de processamento medido em segundos e baseado em tempo de CPU usado para treino e teste.

- **Medida Kappa:** mede a concordância de pares entre um conjunto de codificadores que fazem julgamentos de categoria, corrigindo a concordância de probabilidade esperada. É dada pela seguinte equação:

$$K = \frac{P(A) - P(E)}{1 - P(E)} \quad (2.8)$$

Onde P(A) é proporção de vezes que o classificador concorda e P(E) é a proporção de vezes que esperamos a concordância do classificador pela chance, calculada ao longo dos argumentos intuitivos apresentados.

5.2 Resultados

5.2.1. Avaliação das Características

Para avaliar o conjunto de características adotado no framework foi calculado o Ganho de Informação para as bases ICC_P e BW. Na Tabela 5.2 fornece um *ranking* de relevância dos atributos.

Ao compararmos o ranking das bases apresentadas, observamos que há uma alteração significativa na relevância dos atributos para cada base. No entanto, quando analisamos as 15 primeiras características apresentadas na Tabela 5.2 observamos que 10 atributos permaneceram como os mais relevantes na classificação de *tweets*. Este comportamento também foi observado avaliando os 10 últimos atributos, conjunto de atributos menos relevantes e com valores de relevância próximos.

Tabela 5.2 - Ganho de Informação dos atributos utilizados nas bases de Tweets ICC_P e BW.

	Base ICC		Base BW	
#	Atributos	Ganho de Informação	Atributos	Ganho de Informação
1	no_*tweets*	0.41666	url_nameserver_ip	0.924311
2	**url_domain_age**	0.39225	**url_city_longitude**	0.829551
3	**url_characters_after_domain**	0.38892	**url_city_latitude**	0.828532

4	no_follower	0.33161	**url_nameserver_ASN**	0.802249
5	no_hashtag	0.32440	**url_expanded_size**	0.734120
6	**url_segments_before_domain**	0.31203	**url_domain_age**	0.691379
7	**url_city_longitude**	0.24791	**url_characters_after_domain**	0.628478
8	**url_expanded_size**	0.23200	url_host_size	0.587024
9	**url_count_slashes**	0.23182	url_nameserver_geolocation_country	0.583807
10	**url_nameserver_ASN**	0.21958	**url_count_slashes**	0.526540
11	no_char	0.18827	url_count_hyphen	0.414036
12	**url_city_latitude**	0.18516	**url_tld_id**	0.204691
13	no_following	0.17175	url_nameserver_count	0.184652
14	**url_count_points**	0.14034	**url_count_points**	0.180713
15	**url_tld_id**	0.13684	**url_segments_before_domain**	0.169625
16	no_userfavourites	0.12539	no_usermention	0.164163
17	no_lists	0.12304	no_*tweets*	0.162468
18	url_nameserver_ip	0.12215	account_age	0.142912
19	url_nameserver_geolocation_country	0.10518	no_lists	0.115638
20	no_usermention	0.08524	no_urls	0.088452
21	url_contain_numbers	0.07542	no_char	0.080067
22	account_age	0.07350	url_count_equal_sign	0.062080
23	url_count_question_mark	0.06290	url_count_question_mark	0.059677
24	no_re*tweets*	0.06047	no_userfavourites	0.055544
25	url_count_equal_sign	0.05407	no_digits	0.042667
26	url_nameserver_count	0.03915	url_count_underlines	0.036994
27	**url_count_semicolon**	0.03614	no_following	0.033923
28	no_digits	0.03012	no_follower	0.033093
29	url_count_hyphen	0.02582	url_contain_numbers	0.023736
30	**url_number_subdomain**	0.02490	no_hashtag	0.022488
31	url_count_underlines	0.02416	**url_number_subdomain**	0.015658
32	**url_is_ipaddress**	0.02094	**url_special_words**	0.001435
33	**url_special_words**	0.01557	**url_count_ats**	0.000248
34	**url_count_ats**	0.01395	**url_is_ipaddress**	0.000165
35	no_urls	0.01030	**url_count_semicolon**	0
36	url_host_size	0.00506	no_re*tweets*	0

Analisando o *ranking* apresentado para a base ICC_P, a característica mais relevante para classificação de *tweets* é o número de *tweets*. Essa característica é compreendida como bastante discriminativa quando consideramos que usuários comuns não tem a mesma capacidade de envio de mensagens quanto usuários de *phishing* que podem utilizar mecanismos automatizados para o envio *tweets*.

O número de seguidores e o número de *hashtags* também se apresentam como relevantes, uma vez que usuários maliciosos utilizam mecanismos para aumentar a disseminação dos *tweets* maliciosos com a utilização de *trend topics* (assuntos do momento) e buscam aumentar a sua rede se conectando ao maior número de pessoas.

Avaliando o *ranking* de ganho de informação da base BW observamos que as quinze primeiras características estão relacionadas somente a URL, sendo que as quatros primeiras (*url_nameserver_ip, url_city_latitude, url_city_longitude, url_nameserver_ASN*) são referentes ao provedor do domínio e sua localização. Muitos atacantes utilizam provedores gratuitos e de baixa confiabilidade para hospedar os *sites* de *phishing*. Esses provedores não fazem muitas restrições à criação de domínios, sendo bastante explorados por *phishers*.

Na sequência, observamos que sete atributos estão relacionados às características léxicas da URL como o tamanho da URL, quantidade de caracteres após o domínio, quantidade de caracteres do domínio, quantidade de símbolos (“/”, “-“, “.”) e quantidade de segmentos antes do domínio (subdomínios), demonstrando a relevância que essas características das URLs têm na distinção mais precisa de *tweets* de *phishing*. A URL de *phishing* são, em geral, mais longas que as legítimas por incluírem vários subdomínios e apresentarem vários caracteres após o domínio que uma URL legítima.

A característica relacionada à idade do domínio é apontada, pela literatura, como uma característica discriminativa na classificação de sites de *phishing* por apresentarem tempo de vida curto em relação aos sites legítimos.

A quantidade *retweets* foi a característica que apresentou a menor relevância em relação aos trabalhos avaliados na Seção 3.2. Se considerarmos o período de coleta dos *tweets*, que ocorreu durante a *Black Week*, o objetivo dos atacantes nesse período é obter o maior número de pessoas com a maior quantidade de menções e não por meio de re-

postagens. Essa informação confirma a necessidade da atualização dos classificadores para que possam se adaptar às mudanças de contexto.

5.2.2. Avaliação dos Classificadores

Para avaliarmos o desempenho dos classificadores *online*, as bases geradas (ICC_P e BW) foram unificadas respeitando-se a cronologia de coleta para verificar o comportamento dos classificadores. Dessa forma, os resultados apresentados a seguir foram executados em uma base única com mais de 97 mil exemplos de *tweets* rotulados simulando um fluxo de dados contínuo.

As Figuras Figura 5.2, Figura 5.3 e **Erro! Fonte de referência não encontrada.** exibem os gráficos de uso de memória para cada classificador utilizado. A Figura 5.2 destaca o classificador *Hoeffding Tree*, com uso expressivo da memória RAM para a construção do seu modelo de classificação em relação aos classificadores *Naive Bayes*, *Perceptron* e *Stochastic Gradient* Descent.

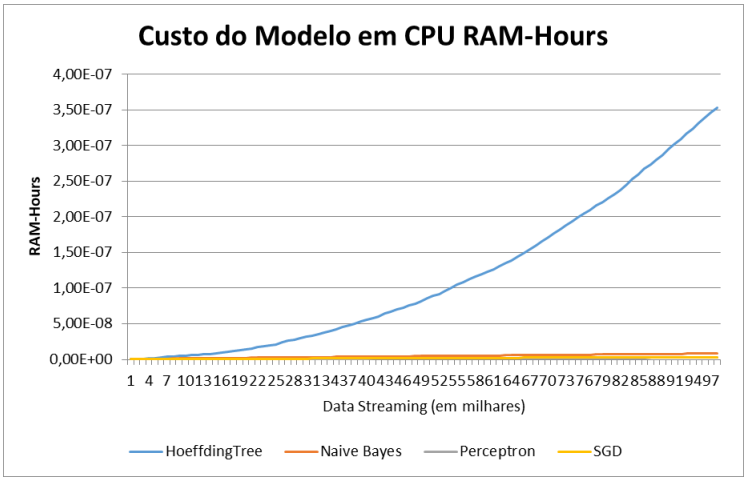

Figura 5.2 - Gráfico comparativo do custo em RAM-Hours dos classificadores *online Hoeffding Tree*, *Naive Bayes*, *Perceptron* e Stochastic Gradient Descent

A Figura 5.3 exibe comparativamente os classificadores *Naive Bayes* que é um classificador probabilístico baseado na teoria bayesiana de decisão, o *Perceptron* e o *Stochastic Gradient Descent* que são modelos baseados em funções de minimização do erro. É possível visualizar que o classificador *Naive Bayes* se destaca dentre o *Perceptron* e o SGD,

uma vez que necessita armazenar as tabelas probabilísticas na memória para tomar decisões e rotular os *tweets* em legítimos ou *phishing*.

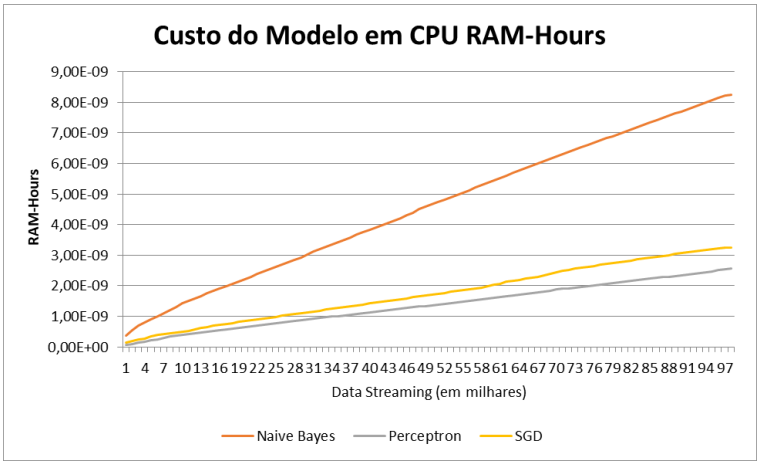

Figura 5.3 - Gráfico comparativo do custo em RAM-Hours dos classificadores *online Naive Bayes*, *Perceptron* e Stochastic Gradient Descent

Por outro lado, o classificador *Perceptron* atualiza o vetor de peso e recalcula a função de ativação a cada nova instância. Isso faz com que o uso de memória seja inexpressivo. O classificador SGD, assim como o *Perceptron*, realiza o mesmo procedimento de atualizar a sua função de ativação, porém a cada atualização do modelo é preciso calcular o gradiente para minimizar o erro das predições subsequentes.

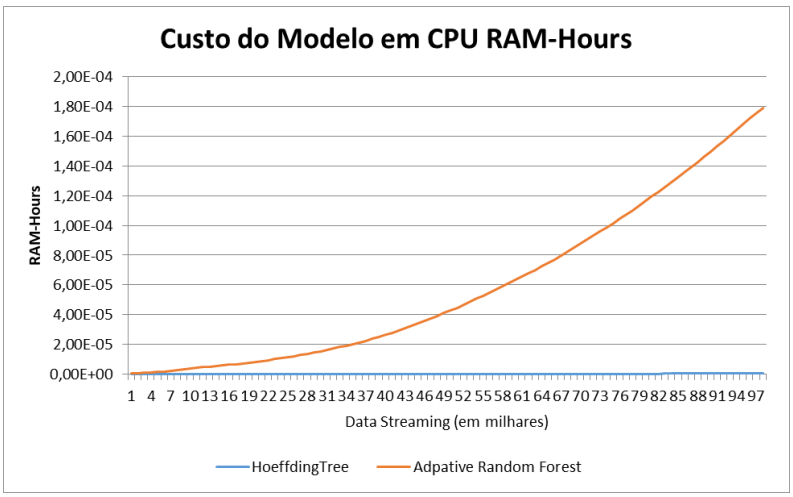

Figura 5.4 - Gráfico comparativo do custo em RAM-Hours dos classificadores *online Hoeffding Tree* e *Adaptive Random Forest*

A Figura 5.4 exibe comparativamente os classificadores *Adaptive Random Forest* (ARF) e *Hoeffding Tree*. O ARF inicia a criação de uma nova árvore em segundo plano assim que detecta um aviso, do inglês "*warning*" e assim que detecta uma mudança ele substitui a árvore atual pela árvore que está sendo treinada em segundo plano. Esse processo faz com que o ARF tenha uma acurácia elevada. No entanto, o seu custo computacional também é diretamente proporcional, elevando o uso dos recursos computacionais.

As Figuras Figura 5.5 e Figura 5.6 exibem os gráficos de tempo de CPU em segundos para cada classificador avaliado. Para melhorar o entendimento dividimos os gráficos em dois para que haja a percepção visual do tempo de aprendizagem (geração dos modelos de classificação).

Na Figura 5.5 é possível observar a diferença entre os tempos requeridos para gerar o modelos de classificação. Assim como na avaliação dos classificadores em relação ao uso de memória, o ARF se destaca pela demora na aprendizagem, devido ao seu processo de aprendizagem já descrito anteriormente baseado na criação de árvores em segundo plano seguida da sua substituição, assim que uma mudança é detectada. Por outro lado, conforme exibido na Figura 5.6, o classificador *Hoeffding Tree* requer em média um tempo de apenas 3 segundos para aprender a classificar corretamente um *tweet*.

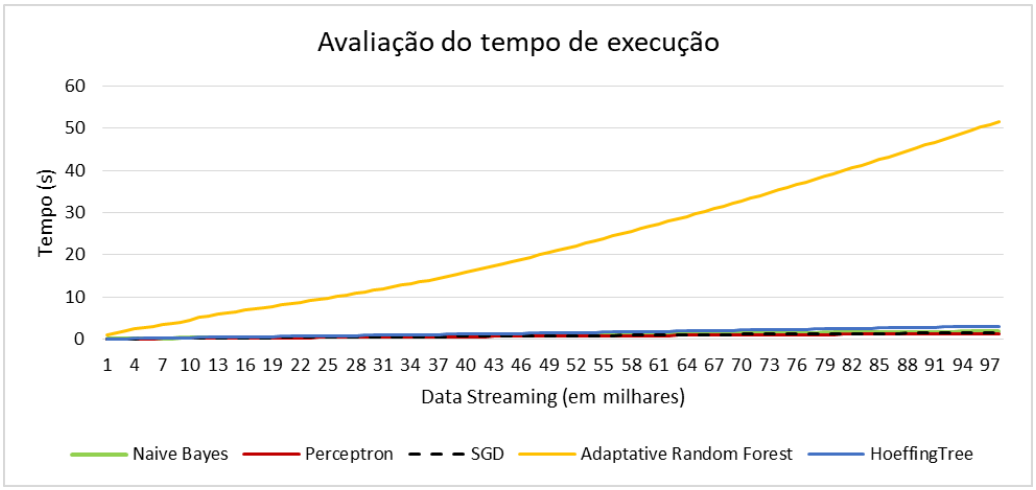

Figura 5.5 - Avaliação dos classificadores *online Naïve Bayes, Perceptron, Stochastic Gradient Descent, Adaptive Random Forest e Hoeffding Tree* em relação ao tempo de aprendizagem.

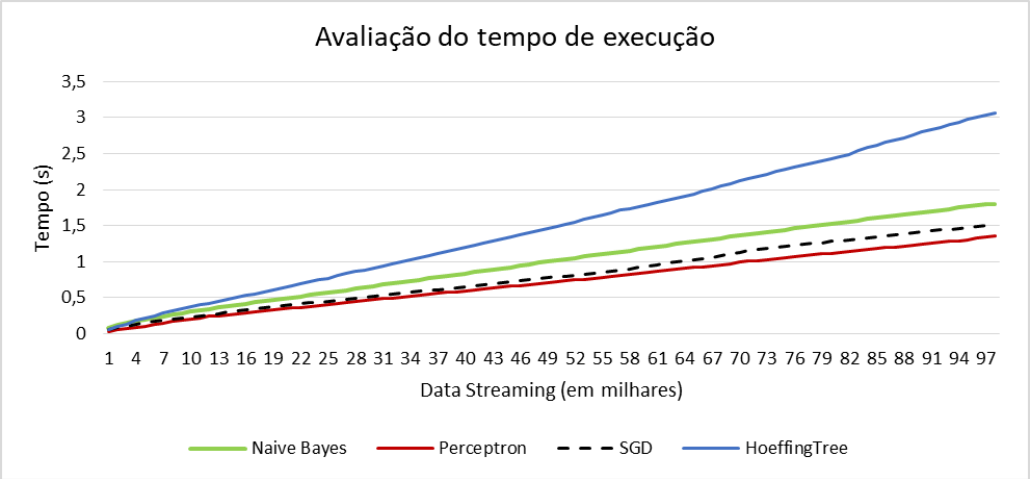

Figura 5.6 - Avaliação dos classificadores *online Naïve Bayes, Perceptron, Stochastic Gradient Descent e Hoeffding Tree* em relação ao tempo de aprendizagem (sem o ARF).

Os classificadores *Perceptron* e SGD apresentaram tempos médios de 1,35 e 1,5 segundos para gerar o modelo de classificação de *tweets*. O modelo baseado no *Naive bayes* apresentou um leve distanciamento do *Perceptron* em decorrência das suas atualizações da tabela probabilística.

A Figura 5.7 exibe o gráfico de avaliação prequential com a curva de aprendizagem dos classificadores. Para gerar o gráfico a entrada de dados foi a base unificada simulando *data streaming*. O eixo y exibe o percentual da acurácia prequential obtida pelos classificadores, enquanto o eixo x apresenta os intervalos de exemplos avaliados a cada mil *tweets*. No gráfico, avaliamos os classificadores e observamos que o *Perceptron* e o SGD apresentam desempenhos semelhantes. Isso ocorre devido ao fato de utilizarem a mesma função de ativação e, por isso apresentam as mesmas hipóteses de decisão.

O modelo gerado pelo classificador probabilístico *Naive Bayes* tem um comportamento abaixo em relação à eficiência dos classificadores *Adaptive Random Forest* e *Hoeffding Tree*s, ambos baseados na criação de árvores de decisão. O *Naive Bayes*, diferente dos supracitados, apresenta a necessidade de visualizar mais amostras para melhorar o seu desempenho.

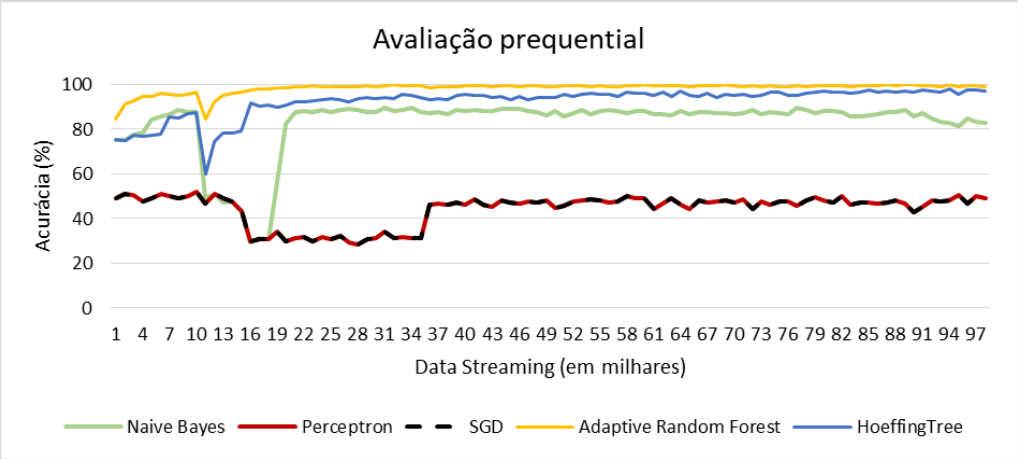

Figura 5.7 - Avaliação dos classificadores *online* em relação a métrica prequential (acurácia)

Os algoritmos *Naive Bayes*, *Perceptron* e SGD apresentam uma curva de aprendizagem abaixo de 30% durante o processamento de aproximadamente 27 mil amostras e os dois classificadores baseados em funções de ativação demoram muito mais que o necessário ao Naives Bayes para processar a atualização do modelo de classificação em decorrência da mudança das suas hipóteses. Em relação aos classificadores *online Hoeffding Tree* e *Adaptive Random Forest*, observamos que ambos apresentam uma "recuperação" mais rápida quando percebem uma mudança de contexto. Nesse contexto, o classificador ensemble *online Adaptive Random Forest* destaca-se como o mais eficiente, iniciando com uma acurácia de quase 85% e rapidamente se mantem próximo aos 100%. Isso se deve ao fato do seu processo de criação de árvores em segundo plano ser eficiente. Sempre que uma mudança é detectada, há a substituição da árvore de decisão atual pela que estava sendo treinada.

O classificador *Hoeffding Tree* necessita de mais amostras, em relação ao ARF, porém de muito menos amostras que os demais classificadores *online*, apresentando um desempenho eficiente, ou seja, alta acurácia, baixo custo computacional e baixo tempo de atualização do modelo de classificação.

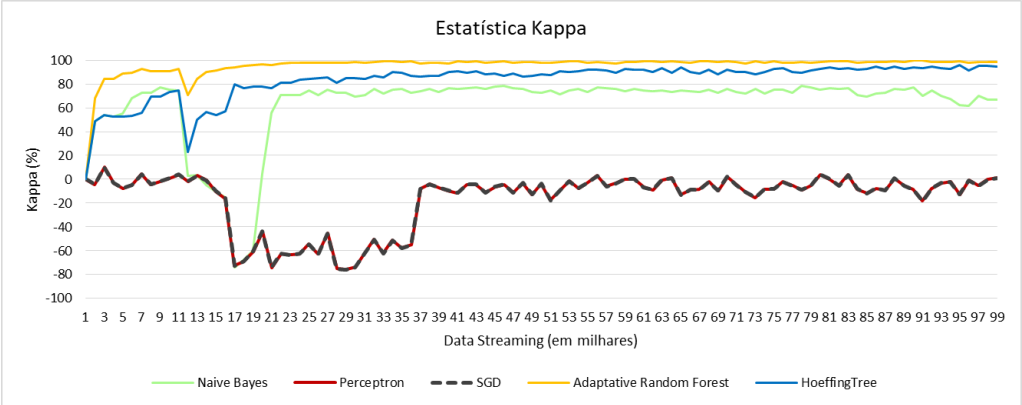

Figura 5.8 - Avaliação dos classificadores *online* em relação à métrica Kappa

O gráfico exibido na Figura 5.8 apresenta a avaliação dos classificadores quanto à métrica Kappa, que avalia o nível de concordância da tarefa de classificar *tweets* corretamente, avaliando também a coesão e o nível de confiança do classificador. No gráfico observamos que o nível de confiança de classificação de *tweets*, dentre os classificadores *online* avaliados é de 9% para o *Perceptron* e para o SGD, já para o ARF, *Hoeffding Tree* e para o *Naive bayes*, apresenta um grau de confiança e coesão acima de 89%.

A Tabela 5.3 apresenta os valores mais expressivos dos classificadores *online* avaliados em relação às métricas de avaliação de desempenho já discutidos nos gráficos apresentados neste capítulo.

Tabela 5.3 – Valores de desempenho dos classificadores *online* avaliados

Classificador	Tempo de Aprendizagem (s)	Custo de Memória (RAM-Hours)	Kappa (%)	Prequential (%)
Adaptive Random Forest	51,57	8.24×10^{-5}	99,59	99,8
Hoeffding Tree	3,06	6.84×10^{-8}	96,08	98,2
Naive Bayes	1,8	8.88×10^{-8}	78,42	89,4
Perceptron	1,35	9.89×10^{-9}	9,96	52,1
SGD	1,5	9.82×10^{-8}	9,96	52,1

Capítulo 6

6 Conclusões e Trabalhos Futuros

Neste trabalho propomos e avaliamos um *framework* de detecção de *phishing* no Twitter. Esse *framework* se propõe a utilizar algoritmos de aprendizagem *online*, ou seja, aprendem a cada amostra, no processo de classificação de *tweets* em *phishing* ou legítimos, podendo ser aplicado no processamento de grandes volumes de dados como *data streaming*.

O *framework* consiste em quatro fases: (i) pré-processamento; (ii) rotulagem; (iii) extração de características; e, (iv) classificação. O pré-processamento é subdividido em duas etapas, de coleta e filtragem de dados. A rotulagem também é subdividida em duas etapas, de desencurtamento e rotulagem dos *tweets* em *phishing* ou legítimo. Na fase de extração de características os vetores são gerados e um novo conjunto de características foi proposto, com o acréscimo de características não utilizadas pelas pesquisas anteriores avaliadas. Por fim, a última fase de classificação na qual os classificadores *online* são avaliados em relação a eficiência na classificação de *tweets* em legítimos ou *phishing*.

Em cada etapa observamos pontos que poderiam ser melhorados, como por exemplo, durante a coleta e filtragem dos *tweets*, na qual a forma de armazenamento dos *tweets* para a posterior geração das bases, poderia ser otimizada simplesmente inserindo um índice único para cada ID de *tweet* coletado evitando a sua coleta em duplicidade.

Na fase de rotulagem, o processo de obtenção de reputação das URLs é custoso em relação ao tempo e requer controle efetivo de dados já consultados e atualização da base de *tweets*. Além da restrição dos serviços de consulta que limitam a quantidade de requisições diárias. Diante desse cenário, chegamos ao entendimento que uma abordagem não supervisionada pode ser explorada nos trabalhos futuros, ainda com algoritmos de

aprendizagem *online*. Ainda na fase de rotulagem, durante a etapa de desencurtamento, observamos que não há a necessidade de consultar as URLs intermediárias das cadeias de redirecionamento, uma vez que essas URLs não são apresentadas ao usuário final. Porém, os valores dos saltos contidos nessas cadeias são informações valiosas para o processo de classificação de URLs maliciosas.

Durante a fase de extração de característica para criação dos vetores, muitas das características, necessitam de requisições HTTP, o que dificulta a tarefa de classificação dos *tweets* em tempo real, uma vez que é necessário aguardar a resposta dos servidores dos serviços consultados. Por exemplo, servidores responsáveis por fornecer a informação sobre a data de criação do domínio não respondiam as requisições automatizadas por *scripts* sendo necessária a consulta manual.

Para avaliar o framework proposto, duas bases de dados foram geradas: a base ICC_P, a partir de dados pré-existentes e a base BW a partir da coleta de *tweets*. Apesar das limitações foi possível rotular mais de 20 milhões de URLs, que consequentemente rotularam mais de 45 mil *tweets* de *phishing* coletados durante um evento relacionado a vendas *online*.

Na fase de classificação, os algoritmos foram avaliados a partir da unificação das bases para simular *data streaming*. Dentre os algoritmos avaliados destaca-se o classificador *online* probabilístico, *Naive Bayes*, que foi o algoritmo que se manteve estável em relação às métricas utilizadas para avaliação dos classificadores. O *Naive Bayes* não foi o melhor classificador em termos de acurácia, ele obteve 89% de acurácia média, porém é o classificador que apresentou regularidade no uso da memória, no tempo de geração do modelo e no índice de confiabilidade Kappa, com 78%.

Os classificadores *Adaptive Random Forest* e *Hoeffding Tree*, baseados em árvores de decisão, apresentaram acurácia média acima dos 98%. No entanto, o processo de criação dos modelos de classificação demostram baixa eficiência quanto ao uso de memória e tempo de geração dos modelos elevado em relação aos demais.

Dessa forma, concluímos que o framework proposto é viável e pode ser utilizado em aplicações reais que processem *data streaming* visando o aumento da segurança dos usuários da rede social Twitter.

Como trabalhos futuros, identificamos a necessidade de avaliar outros classificadores *online* de outras naturezas, principalmente, os de abordagem de aprendizagem *online* não supervisionada no processamento de data streaming para suprimir o processo de rotulagem dos dados. Outro ponto potencial a ser explorado é realizar uma seleção otimizada de características baseadas em suas correlações, para aperfeiçoar o desempenho dos classificadores *online*.

Referências Bibliográficas

AGGARWAL, A.; RAJADESINGAN, A.; KUMARAGURU, P. **PhishAri: Automatic realtime phishing detection on twitter**. 2012 eCrime Researchers Summit. **Anais**.IEEE, out. 2012.

ALPAYDIN, E. **Introduction to machine learning**. Second Edi ed. Massachusetts: The MIT Press, 2010. v. 1107

ANTI-PHISHING WORKING GROUP. **Phishing Activity Trends Report 1st to 3rd Quarters 2015.**

ANTONIADES, D. et al. **we.b: The web of short URLs**. International Word Wide Web Conference Comittee (IW3C2). **Anais**.2011

ARACHCHILAGE, N. A. G.; LOVE, S. A game design framework for avoiding phishing attacks. **Computers in Human Behavior**, v. 29, n. 3, p. 706–714, maio 2013.

ASSOLINI, F. **Beaches, Carnivals and Cybercrime: A Look Inside The Brazilian Underground**.

ATEFEH, F.; KHREICH, W. A Survey of Techniques for Event Detection in Twitter. **Computational Intelligence**, v. 31, n. 1, p. 133–164, 2015.

BASNET, R. B.; DOLECK, T. Towards Developing a Tool to Detect Phishing URLs: A Machine Learning Approach. **2015 IEEE International Conference on Computational Intelligence & Communication Technology**, p. 220–223, 2015.

BECK, K. Analyzing *tweet*s to identify malicious messages. **2011 Ieee International Conference on Electro/Information Technology**, p. 1–5, maio 2011.

BENCZÚR, András A.; KOCSIS, Levente; PÁLOVICS, Róbert. *Online* Machine Learning in Big Data Streams. **arXiv preprint arXiv:1802.05872**, 2018.

BEZERRA, M. A. **Investigando o uso de Características na Detecção de URLs**. [s.l.] UFAM, 2015.

BIFET, Albert; KIRKBY, Richard. Data stream mining a practical approach. 2011.

BLUM, A. et al. Lexical feature based phishing URL detection using *online* learning. **Proceedings of the 3rd ACM workshop on Artificial intelligence and security - AISec '10**, p. 54, 2010.

BOTTOU, Léon. *Online* learning and stochastic approximations. ***Online* learning in neural networks**, v. 17, n. 9, p. 142, 1998.

BOTTOU, Léon. Stochastic gradient descent tricks. In: **Neural networks: Tricks of the trade**. Springer, Berlin, Heidelberg, 2012. p. 421-436.

BOYD, D. M.; ELLISON, N. B. Social Network Sites: Definition, History, and Scholarship. **Journal of Computer-Mediated Communication**, v. 13, n. 3, p. 210–230, 2007.

CHEN, Chia-Mei; GUAN, D. J.; SU, Qun-Kai. Feature set identification for detecting suspicious URLs using Bayesian classification in social networks. Information Sciences, v. 289, p. 133-147, 2014.

CHEN, C. et al. **6 Million Spam Tweets: A Large Ground for Timely Twitter Spam**LondonICC 2015, , 2015.

CHEN, W. et al. Real-time twitter content polluter detection based on direct features. **2015 IEEE 2nd International Conference on InformationScience and Security, ICISS 2015**, 2016.

CHHABRA, S. et al. **Phi.sh/$oCiaL: The Phishing Landscape through Short URLs**. Proceedings of the 8th Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference on - CEAS '11. **Anais**. New York, New York, USA: ACM Press, 2011.

CRAMMER, K. et al. *Online* Passive-Aggressive Algorithms. **Journal of Machine Learning Research**, v. 7, p. 551–585, 2006.

CUNHA, Francisco F. R.; SANTOS, Eulanda M; SOUTO, Eduardo J.P. Detecção de Phishing em Páginas Web Utilizando Técnicas de Aprendizagem de Máquina. **XII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais**. p. 491-500. 2012.

DOMINGOS, Pedro; HULTEN, Geoff. Mining high-speed data streams. In: **Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining**. ACM, 2000. p. 71-80.

DREDZE, M.; CRAMMER, K.; PEREIRA, F. Confidence-Weighted Linear Classification. **Proceedings of the 25th international conference on Machine learning - ICML '08**, p. 264–271, 2008.

ELLISON, N. B.; STEINFIELD, C.; LAMPE, C. The benefits of facebook "friends:" Social capital and college students' use of *online* social network sites. **Journal of Computer-Mediated Communication**, v. 12, p. 1143–1168, 2007.

FEROZ, M.; MENGEL, S. **Examination of Data, Rule Generation and Detection of Phishing URLs using *Online* Logistic Regression**. IEEE International Conference on Big Data. **Anais**...2014

FREITAS, P. P. S. **BeShort: Um algoritmo para encurtamento de URLs**. [s.l.] Universidade Federal de Outro Preto, 2012.

GENG, G. G. et al. **Favicon - a clue to phishing sites detection**. 2013 APWG eCrime Researchers Summit. **Anais**...IEEE, set. 2013.

GOMES, Heitor M. et al. Adaptive random forests for evolving data stream classification. **Machine Learning**, v. 106, n. 9-10, p. 1469-1495, 2017.

GUPTA, N.; AGGARWAL, A.; KUMARAGURU, P. bit.ly/malicious: Deep dive into short URL based e-crime detection. **2014 APWG Symposium on Electronic Crime Research (eCrime)**, p. 14–24, set. 2014.

JEONG, S. Y.; B, Y. S. K.; DOBBIE, G. **Phishing Detection on Twitter Streams**. Springer International Publishing Switzerland. **Anais**...Switzerland: Springer International Publishing, 2016.

KANDASAMY, K.; KOROTH, P. An Integrated Approach to Spam Classification on Twitter Using URL Analysis, Natural Language Processing and Machine Learning Techniques. **2014 IEEE Students' Conference on Electrical, Electronics and Computer Science**, p. 1–5, mar. 2014.

KHONJI, M.; IRAQI, Y.; JONES, A. Phishing Detection: A Literature Survey. **IEEE**

Communications Surveys & Tutorials, v. 15, n. 4, p. 2091–2121, 2013.

KWAK, H. et al. **What is Twitter , a Social Network or a News Media?** The International World Wide Web Conference Committee (IW3C2). **Anais**...2010

LE, A.; MARKOPOULOU, A.; FALOUTSOS, M. **PhishDef: URL Names Say It All**. INFOCOM, 2011 Proceedings IEEE. **Anais**...12 set. 2011.

LEE-BERNERS, T.; MASINTER, M. **Request for Comments: 1738**. Minnesota: [s.n.].

LEE, S.; KIM, J. **WarningBird: Detecting Suspicious URLs in Twitter Stream ∗**. Network and Distributed System Security Symposium (NDSS). **Anais**...2012

LIN, M.-S. et al. Malicious URL filtering — A big data application. **2013 IEEE International Conference on Big Data**, p. 589–596, out. 2013.

LOSING, Viktor; HAMMER, Barbara; WERSING, Heiko. Incremental *online* learning: A review and comparison of state of the art algorithms. **Neurocomputing**, v. 275, p. 1261-1274, 2018.

MA, J. et al. Identifying Suspicious URLs : An Application of Large-Scale *Online* Learning. ***Online***, p. 681–688, 2009.

MITCHELL, T. M. **Machine Learning**. [s.l.] MacGraw-Hill Science, 1997.

NAIR, M. C.; PREMMA, S. **A Distributed System for Detecting Phishing in Twitter Stream**. International Journal of Engineering Science and Innoavtive Technology (IJESIT). **Anais**...2014

NIKIFORAKIS, Nick et al. Stranger danger: exploring the ecosystem of ad-based url shortening services. In: Proceedings of the 23rd international conference on World wide web. ACM, 2014. p. 51-62.

REPÚBLICA, S. DA C. S. DA P. DA. **Pesquisa brasileira de mídia 2016 : hábitos de consumo de mídia pela população brasileira**. Brasília: SECOM, 2016.

SHALEV-SHWARTZ, Shai; SINGER, Yoram. ***Online* learning: Theory, algorithms, and applications**. 2007.

SHARMA, N. et al. **Real-Time Detection of Phishing Tweets**. Fourth International Conference on Computer Science, Engineering and Applications. **Anais**...2014

STRONKMAN, R. **Exploiting Twitter to fulfill information needs during incidents**. [s.l.] Delft University of Technology, 2011.

SYMANTEC, H. C. W. **Internet Security Threat Report Volume 23**. Mountain View, CA: [s.n.].

THOMAS, K. et al. **Design and Evaluation of a Real-Time URL Spam Filtering Service**. 2011 IEEE Symposium on Security and Privacy. **Anais**...IEEE, maio 2011.

WHITTAKER, C.; RYNER, B. Large-Scale Automatic Classification of Phishing Pages. **Design**, 2008.

XIAOLING et al. Scam Detection in Twitter. **Data Mining for Service**, v. 3, n. Springer Berlin Heidelberg, p. 133–150, 2014.

ZHANG, J.; WANG, Y. A real-time automatic detection of phishing URLs. **Proceedings of 2nd International Conference on Computer Science and Network Technology, ICCSNT 2012**, p. 1212–1216, 2012.

ZHU, X.; GOLDBERG, A. B. **Introduction to Semi-Supervised Learning**. [s.l.] Morgan & Claypool, 2009. v. 3