CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
CAMPUS TIMÓTEO

Jefferson Johnny Martins Gandra

DETECÇÃO DE SPAM EM REDES SOCIAIS UTILIZANDO APRENDIZAGEM DE MÁQUINA

Timóteo

2020

Jefferson Johnny Martins Gandra

DETECÇÃO DE SPAM EM REDES SOCIAIS UTILIZANDO APRENDIZAGEM DE MÁQUINA

Monografia apresentada à Coordenação de Engenharia de Computação do Campus Timóteo do Centro Federal de Educação Tecnológica de Minas Gerais para obtenção do grau de Bacharel em Engenharia de Computação.

Orientador: Lucas Pantuza Amorim

Timóteo

2020

Jefferson Johnny Martins Gandra

DETECÇÃO DE SPAM EM REDES SOCIAIS UTILIZANDO APRENDIZAGEM DE MÁQUINA

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia de Computação do Centro Federal de Educação Tecnológica de Minas Gerais, campus Timóteo, como requisito parcial para obtenção do título de Engenheiro de Computação.

Trabalho aprovado. Timóteo, 06 de novembro de 2020:

_____

Prof. Dr. Lucas Pantuza Amorim
Orientador

_____

Prof. Ms. Aléssio Miranda Júnior
Professor Convidado

_____

Prof. Ms. Adilson Mendes Ricardo
Professor Convidado

Timóteo
2020

Emitido em 06/11/2020

FOLHA DE ROSTO (PLATAFORMA BRASIL) Nº 1/2020 - DCCTM (11.63.05)

(Nº do Protocolo: NÃO PROTOCOLADO)

(Assinado digitalmente em 06/11/2020 18:21)
ADILSON MENDES RICARDO
PROFESSOR ENS BASICO TECN TECNOLOGICO
DCCTM (11.63.05)
Matrícula: 2849338

(Assinado digitalmente em 06/11/2020 17:07)
ALESSIO MIRANDA JUNIOR
PROFESSOR ENS BASICO TECN TECNOLOGICO
DCCTM (11.63.05)
Matrícula: 1713470

(Assinado digitalmente em 06/11/2020 16:47)
LUCAS PANTUZA AMORIM
PROFESSOR ENS BASICO TECN TECNOLOGICO
DCCTM (11.63.05)
Matrícula: 2897411

Para verificar a autenticidade deste documento entre em https://sig.cefetmg.br/documentos/ informando seu número: **1**, ano: **2020**, tipo: **FOLHA DE ROSTO (PLATAFORMA BRASIL)**, data de emissão: **06/11/2020** e o código de verificação: **4a02431196**

Agradecimentos

Agradeço primeiramente a Deus por ter me mantido com saúde e coragem para chegar até o final.

Agradeço a minha família pelo apoio, incentivo e atenção dedicados quando sempre precisei.

Agradeço aos meus amigos que fiz durante o curso, em especial, Arthur, Débora, Rafael, Gustavo e Lucas, pela companhia e parceria durante todos esses anos.

Agradeço a minha professora de TCC, Deisymar Botega Tavares, pelo incentivo e ensinamentos. Quando pensei em desistir, suas palavras me deram forças para prosseguir.

Agradeço especialmente ao meu professor orientador, Lucas Pantuza Amorim, que apesar da sua intensa rotina acadêmica disponibilizou a sua casa com boa vontade, paciência e dedicação, para a concretização deste trabalho. Obrigado por compartilhar seus conhecimentos e me manter motivado durante todo o processo.

Por fim, agradeço a todos os professores e servidores do Curso de Engenharia de Computação do CEFET-MG, campus Timóteo, pelo apoio durante toda a minha vida acadêmica.

“A vida não é sobre quão duro você é capaz de bater, mas sobre quão duro você é capaz de apanhar e continuar indo em frente.”

Rocky Balboa

Resumo

As redes sociais estão cada vez mais presentes no dia a dia das pessoas, o que expõe e as transformam em possíveis vítimas de pessoas de má índole no ambiente virtual. Estes ataques podem ser para disseminar dados mentirosos, roubar dinheiro ou informações sigilosas. Este trabalho apresenta um estudo de dois algoritmos, Naive Bayes e SVM, com o intuito de avaliar o desempenho na detecção de mensagens maliciosas (*spams*). Como ponto de partida foi usado o Twitter, uma das redes sociais mais acessadas no mundo. Devido a suas restrições de tamanho das mensagens, usuários mal intencionados podem compartilhar *links* curtos, sendo impossível para a administração da rede verificar previamente a URL antes de publicado. A metodologia empregada foi a coleta de dados, rotulação, extração das características, treinamento, classificação e avaliação. Após a análise dos resultados, verificou-se que a SVM apresentou melhores resultados que o Naive Bayes na precisão de detecção de postagens maliciosas.

Palavras-chave: Redes sociais, Twitter, *spam*, Naive Bayes, SVM.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

A popularização da Internet trouxe benefícios à humanidade, permitindo que qualquer pessoa com acesso à Internet possa buscar opções de entretenimento, jornalismo, ensino e comunicação interpessoal. Segundo FILHO, com o objetivo de aproximar pessoas no ambiente *online*, as redes sociais ganharam espaço, a comunicação interpessoal foi intensificada e, consequentemente, pessoas comuns se tornaram divulgadoras e até criadoras de conteúdo na grande rede.

Entre as redes sociais mais populares está o Twitter, com 330 milhões de usuários (Reuters, 2019). O Twitter pode ser definido como um *microblog* que permite aos usuários enviarem mensagens instantâneas com até 280 caracteres, e também permite o envio de imagens, vídeos e compartilhamento de outras páginas acrescentando o *hiperlink* de outros endereços.

Aproveitando o crescimento de usuários nas redes sociais e o uso de encurtadores de *links*, CHEN et al. relata que pessoas mal-intencionadas usam a rede social como forma de divulgar *links* maliciosos com intenção de roubar dados sigilosos dos usuários, que podem conter *softwares* maliciosos para *sites* falsos que praticam *phishing*, venda de drogas e trapaças.

O *phishing*, como descrito por ROSALES, é uma corruptela da palavra em inglês "*fishing*" (que significa pescar), o termo "phreak" (de "*freak*", ou aberração) e é utilizado para definir os ataques que emprega meios tecnológicos para induzir vítimas a entregarem ao atacante algo do seu interesse.

A tática de *phishing* é muito popular em *e-mails*, mas tem seu sucesso reduzido pois, quando um remetente envia muitos *e-mails* iguais para vários usuários, ele normalmente entra na caixa de *spam*. Isso não ocorre nas redes sociais, já que as mensagens instantâneas não são categorizadas como *spam*. Segundo o Retruster; Malwarebytes, os *sites* de redes sociais se tornaram o maior alvo dessa prática.

Apesar de ações como a criação de sistemas contendo uma lista de *sites* maliciosos conhecidos a serem bloqueados e o uso de encriptação HTTPS, o resultado não foi como esperado, pois a taxa de criação de novos *sites* maliciosos é grande. Segundo o *site* Retruster; Malwarebytes, 1.5 milhões de *sites* são criados a cada mês e somente em 2018 as tentativas de *phishing* subiram 65%.

As redes sociais contam com algumas técnicas anti-*spam* citadas em ROTH; HARVEY, mas a maioria delas precisa que o usuário reporte mensagens suspeitas. Isso pode ser ineficaz, pois antes de identificar corretamente o *spam*, outros usuários já podem ter sido vítimas.

1.1 Justificativa

Com o crescimento significativo do número de *sites*, torna-se difícil identificar aqueles que são maliciosos e evitar *phishing*. Assim, o combate ao *spam* surgiu como tentativa de se evitar o compartilhamento de *links* maliciosos na Internet.

O Twitter criou formas de diminuir o *spam* nas redes sociais investigando contas que apresentam comportamento malicioso. Apesar de apresentar bons resultados, o processo não age em tempo real e normalmente o resultado do processo é a suspensão da conta, o que pode ser facilmente burlado com a criação de novas contas por parte do usuário mal intencionado. Como reportado por ROTH; HARVEY, em março de 2017, o Twitter recebeu 17 mil denúncias de *tweets* considerados como *spam* por seus usuários, o que indica que mensagens maliciosas ainda são vistas e atingem seus usuários.

Devido ao grande número de usuários que ficam vulneráveis a esses ataques, faz-se necessária a investigação de métodos capazes de detectar *spam* antes que eles possam causar problemas. Entre os métodos possíveis, os algoritmos de aprendizagem de máquina se destacaram pela simplicidade e precisão. São usados principalmente por empresas para fazerem análises de mercado e analisarem a repercussão de campanhas de marketing e produtos nas redes sociais. A detecção de *spam* em *e-mails* evoluiu com o desenvolvimento de aprendizagem de máquina, conseguindo classificar corretamente *e-mails* maliciosos com 94% de precisão (OLIVO; OLIVEIRA, 2015). Todavia, a detecção de *spams* no Twitter traz novos desafios devido às propriedades do *tweet* (mensagem publicada na rede), e a não publicidade das soluções tecnológicas adotadas pelas empresas responsáveis.

1.2 Objetivos

Nesse contexto, o objetivo do presente trabalho é construir um método para detectar automaticamente as mensagens de *spam* utilizando algoritmos de aprendizagem de máquina, avaliando sua eficácia. Para isso são feitas as seguintes etapas:

1. Criação de uma ferramenta de coleta de *tweets*;
2. Criação de um extrator de características baseado em informações do *tweet*, perfil do usuário do Twitter e da URL contida no *tweet*;
3. Avaliação de diferentes classificadores no processo de detecção de *tweets*.

2 Fundamentação Teórica

2.1 Twitter

Twitter é uma rede social de *microblogging* criada em 2006 que conecta pessoas e funciona como um sistema de compartilhamento de informação. Nesta rede, a conexão entre os usuários é direcional, onde um usuário segue e é seguido por outros usuários, sem a necessidade deste vínculo ser recíproco. Os seguidores podem visualizar todos os conteúdos postados pelos seus usuários seguidos, podendo interagir através de compartilhamento ou respondendo as suas mensagens. O Brasil é o quarto país no mundo com mais acessos na rede social (CLEMENT, 2019).

Os *tweets*, como são chamadas as mensagens postadas no Twitter, possuem um limite de 280 caracteres e podem conter texto, imagens, vídeos e GIFs. Um *tweet* pode ser ou conter (Twitter, 2020a):

- ***Retweets***, que são *tweets* compartilhados por outros usuários para seus seguidores do Twitter. Sendo assim, os *retweets* serão visíveis para os usuários da rede social que seguem o usuário que retweetou;
- **Menções**, que são mensagens que contêm o "@" na frente de um nome de usuário, como por exemplo "Olá, @Mundo". O uso desse recurso é uma forma de direcionar a mensagem ao usuário mencionado, tornando-a visível tanto aos seguidores do usuário autor, quanto aos do mencionado;
- ***Reply***, que é um *tweet* que age como resposta a *tweets*;
- ***Tweets* promovidos**, que são mensagens pagas por anunciantes, com a intenção de promoverem sua marca ou produto, tornando os *tweets* visíveis pelo público como um anúncio publicitário;
- ***Trending Topics*** uma ferramenta popular do Twitter que permite capturar tendências e tópicos em evidência em determinado momento para uma localização geográfica específica.
- ***Hashtag*** de forma similar ao "@", que simboliza uma menção, o símbolo "#", simboliza uma *Hashtag* e é aplicada para fixar palavras-chave em um *tweet*. As *hashtags* e outras palavras-chave que forem muito comentadas em um curto período de tempo podem aparecer no *Trending Topics*,

Como diz SILVA, desde sua fundação o Twitter tem sido alvo de vários ataques à segurança e privacidade por parte de seus próprios usuários, que podem ter grande alcance devido a sua relevância no contexto social e cultural. A rede é também utilizada indevidamente

como canal de controle para *hackers* manipularem suas *botnets* (ROSALES, 2015), *phishing* através de URLs encurtadas e para propagação de *spam*.

2.2 Spam

O *spam* é uma mensagem não solicitada enviada em massa, com conteúdo indesejado, tentando induzir o usuário a clicar em um *link* malicioso. Ele está presente em diversos meios de comunicação eletrônica como *e-mail*, *blogs*, SMS e também nas redes sociais. Segundo ALBERTO et al., a popularização do *spam* é reflexo da dificuldade de detecção das intenções, mensagens e da facilidade de atingir maior número de usuários possível, aumentando assim as chances de encontrar uma vítima. Essa prática é usada para publicidade de produtos e serviços, aplicar golpes, disseminação de notícias falsas (*fake news*) e para espalhar *softwares* maliciosos.

O primeiro envio de *spam* por *e-mail* ocorreu em 1978, quando foi enviado para 393 usuários da ARPANET. Com o crescimento da Internet e dos recursos computacionais, o volume tornou-se maior do que os envios legítimos. De acordo com as Figuras 1 e 2, em 2018 o Brasil foi o quinto país dentre os que mais enviaram *spam* na Internet e ocupa a mesma colocação como alvo da prática no mundo. Isso evidencia a grande exposição do brasileiro a crimes cibernéticos e a importância da segurança digital no país.

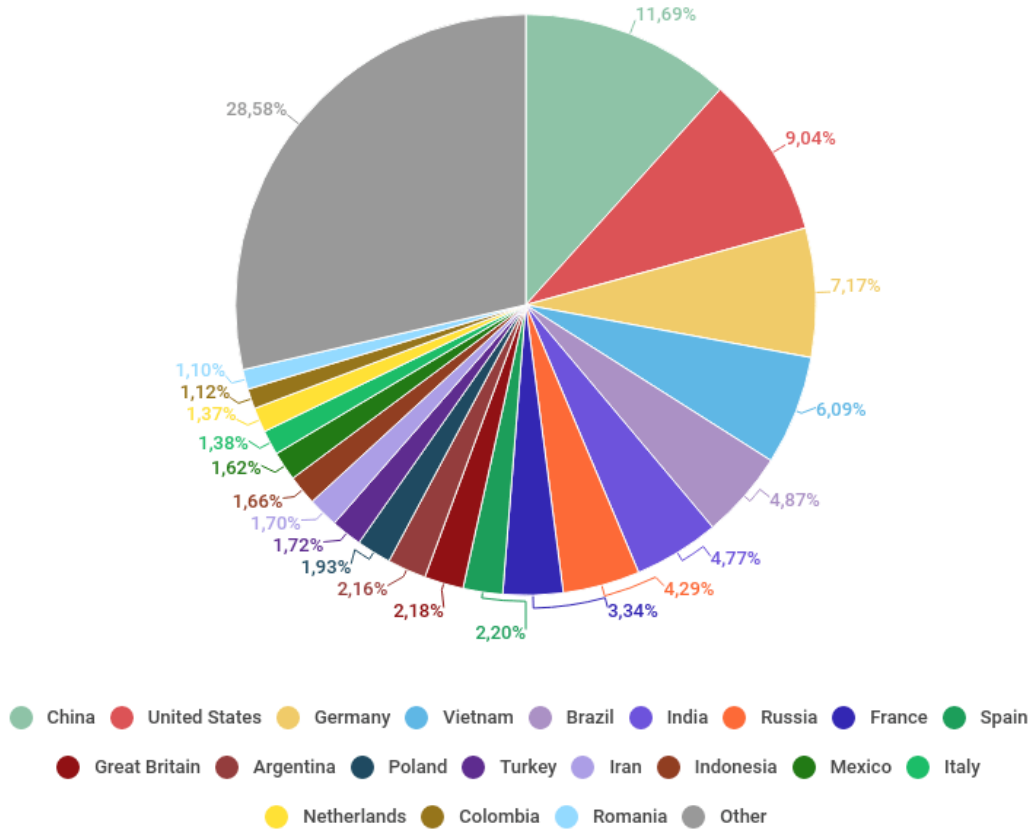

Figura 1 – Fontes de *spam* por país, CLEMENT

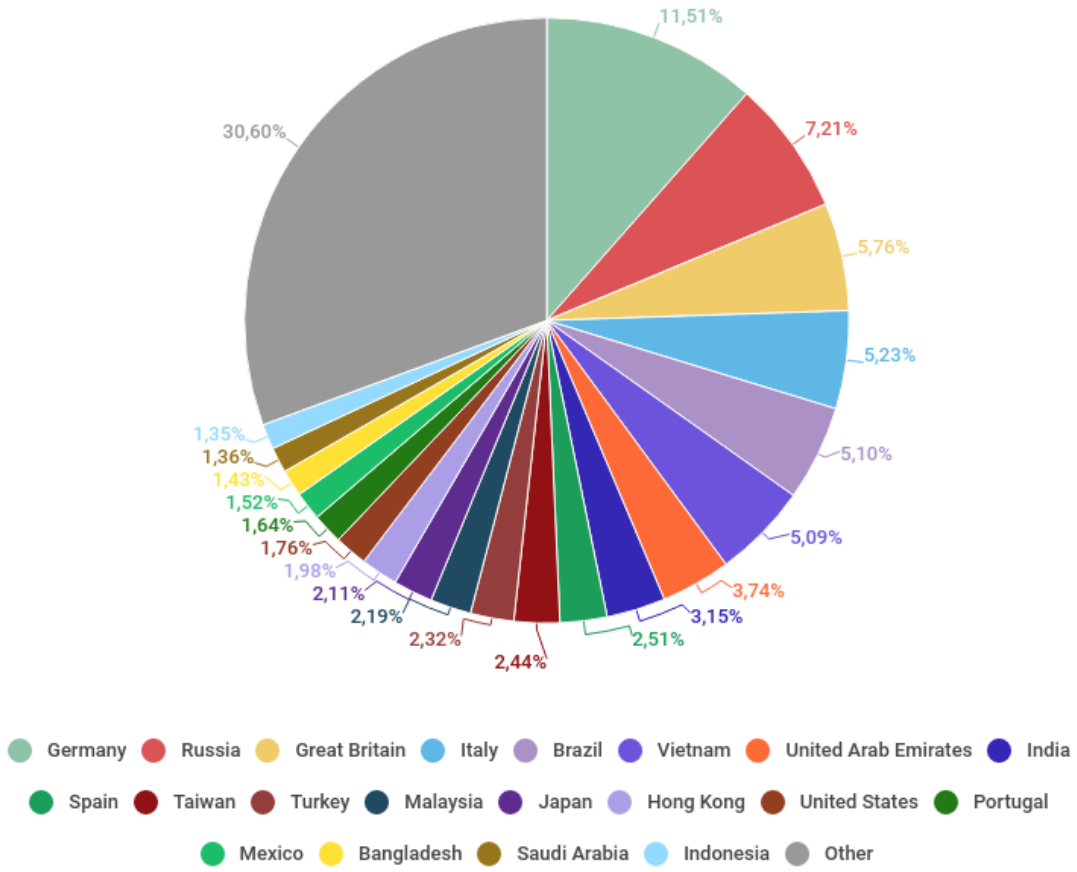

Figura 2 – Países alvos de *spam* por *e-mail*, CLEMENT.

OLIVO; OLIVEIRA afirma que o envio de *spam* pode causar vários problemas, desde o aborrecimento dos usuários à sobrecarga de servidores devido ao grande volume de mensagens recebidas mas, principalmente, o envio de *phishing*, onde a segurança financeira de dados particulares do usuário são colocadas em risco.

A tática consiste em atrair vítimas usando anúncios ou *sites* clonados, coletando informações pessoais ou instalando *softwares* maliciosos. Os objetivos dos *phishers*, como são chamados os atacantes, são:

- **Roubo**, quando o *phisher* faz a vítima pensar que está adquirindo um produto e recolhe informações sobre cartão de crédito, induz ao pagamento de boletos ou transferências bancárias;
- **Sabotagem**, quando computadores infectados por *malwares* agem em um ataque coordenado, executando milhares de requisições por hora a um determinado servidor ocasionando a sua queda por não suportar a demanda. Esse ataque é conhecido como ataque de negação de serviço (DDoS), e apesar de não haver roubo de dados nesse tipo de ataque, a vítima leva prejuízo pelo tempo que o sistema ficou inoperante (ROSALES, 2015);

- **Extorsão**, que acontece através de um *malware* denominados *ransomware*, um *software* criado para sequestrar informações (ROSALES, 2015). Em versões mais perigosas, os dados da vítima são parcialmente ou completamente criptografados e só podem ser recuperados após pago um "resgate". Esse golpe ficou famoso após o *malware* Wannacry ser usado contra hospitais americanos em 2017, chegando a atingir 250000 dispositivos (ROSALES, 2015);

- **Espionagem**, através de *malwares* instalados no computador, permitindo que sejam espionados por empresas ou outras pessoas;

- **Botnet**, que é um ataque *phishing*, podendo ser usado para construir uma rede com milhares de computadores infectados (ROSALES, 2015). A rede formada por computadores infectados pode ser usada posteriormente como um serviço de aluguel. Por exemplo, uma pessoa que não queria ou não tenha conhecimento em montar seu próprio *software* para *phishing*, pode contratar o serviço de um controlador de um *botnet* que a customiza para a finalidade desejada e a aluga por tempo determinado.

2.3 Aprendizagem de Máquina

A Aprendizagem de Máquina (AM) é um campo da Inteligência Artificial com finalidade de criar técnicas computacionais sobre aprendizado e desenvolvimento de sistemas inteligentes, capazes de tomar decisões através de experiências acumuladas (CAMILO et al., 2015). Trata-se de uma área utilizada para extração de conhecimento em grandes volumes, bastante aplicada em processos de Mineração de Dados (MONARD; BARANAUSKAS, 2003).

O sistema de aprendizado se modifica automaticamente para que consiga ficar mais eficiente cada vez que ele realiza a mesma tarefa sobre um mesmo domínio de conhecimento (SANCHES, 2003). Para aprender, o sistema utiliza estratégias baseadas nas que são utilizadas por seres humanos, onde o mais conhecido é o aprendizado indutivo. Nele, um conceito é aprendido efetuando-se inferência indutiva sobre exemplos apresentados. As hipóteses geradas através da inferência indutiva podem ou não preservar a verdade (MONARD; BARANAUSKAS, 2003). O aprendizado indutivo é dividido em duas categorias: aprendizado supervisionado e aprendizado não supervisionado, como é representado na Figura 3 (SANCHES, 2003).

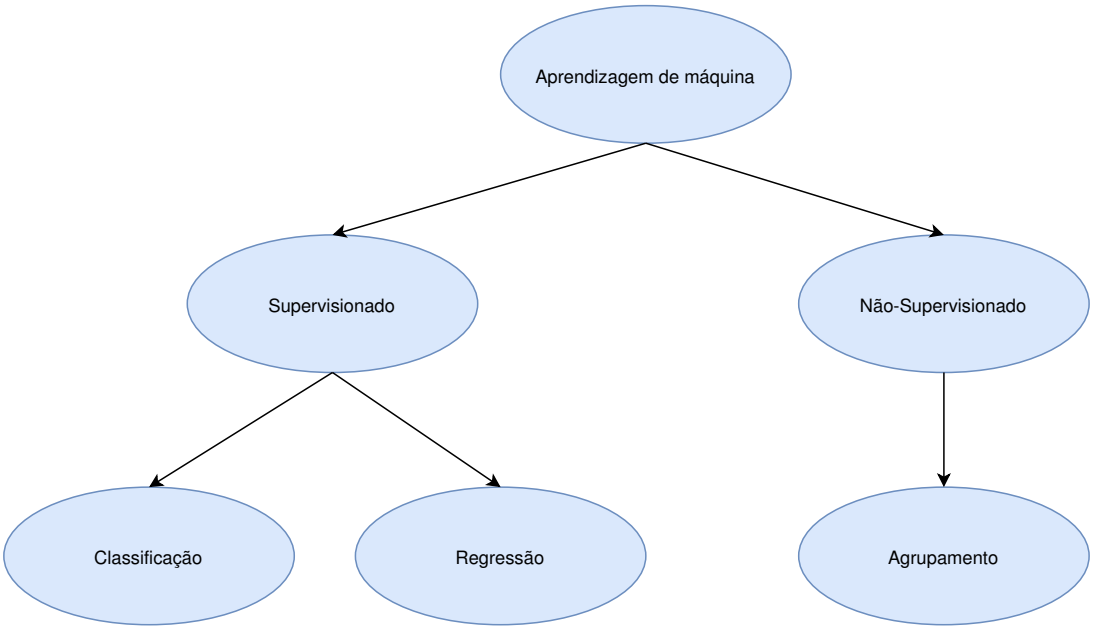

Figura 3 – Tipos de aprendizagem de máquina (do autor).

2.4 Aprendizado Supervisionado

Segundo MONARD; BARANAUSKAS, no aprendizado supervisionado o algoritmo recebe exemplos rotulados como pertencentes a alguma classe conhecida, com o objetivo de treiná-los, para que futuros exemplos sejam classificados. Cada exemplo é representado por um vetor de valores de características, denominados atributos, e os rótulos das classes associadas. Caso o tipo da classe seja contínuo, o problema de indução é chamado de regressão. Caso seja discreto, o problema é chamado de classificação (SANCHES, 2003).

Neste trabalho foram utilizados os algoritmos de aprendizagem supervisionados Naive Bayes (PRADO et al., 2002) e *Support Vector Machine* (SVM) (BOSER; GUYON; VAPNIK, 1992), explicados em mais detalhes nas subseções seguintes.

2.5 Aprendizado Não Supervisionado

No aprendizado não supervisionado, o algoritmo recebe os exemplos não rotulados e procura padrões através de alguma caracterização de regularidade para fazer um agrupamento dos dados da melhor forma possível. Este processo é chamado de *clustering* (SANCHES, 2003), e é normalmente seguido de uma análise para estabelecer o que cada agrupamento representa no contexto do problema que está analisando (MONARD; BARANAUSKAS, 2003).

2.6 Paradigmas de Aprendizagem de Máquina

Os algoritmos de aprendizagem de máquina podem ser classificados pelos paradigmas abaixo (MONARD; BARANAUSKAS, 2003):

- **Simbólico**: o sistema é representado em uma estrutura simbólica e é alimentado com exemplos e contraexemplos para obter o aprendizado. Árvore de decisão e agentes inteligentes são exemplos de aplicação desta técnica;

- **Estatístico**: o sistema estatístico emprega modelos estatísticos para obter uma solução aproximada ao conceito induzido; ele possui a vantagem de poder embutir nas probabilidades calculadas o conhecimento de domínio que se tem, e a desvantagem de exigir alto custo computacional (PRADO et al., 2002). Um exemplo de algoritmo estatístico é o modelo Naive Bayes, utilizado neste trabalho. (PRADO et al., 2002), que apesar de ser um método estatístico, não possui alto custo computacional e é simples de ser implementado;

- **Baseado em exemplos**: o sistema usa as amostras anteriores para classificar uma amostra ainda desconhecida. Após a classificação, esta nova amostra também é salva na memória e usada em futuras classificações. Esse tipo de aprendizagem é chamada de *Lazy* (MONARD; BARANAUSKAS, 2003). Alguns exemplo de sistema baseado em exemplos são Nearest Neighbours (GOLDBERGER et al., 2000) e Raciocínio Baseado em Casos (RBC) (WANGENHEIM; GRESSE; WANGENHEIM, 2013);

- **Conexionista**: o modelo conexionista é baseado no funcionamento do cérebro humano e é o paradigma objeto de estudo de redes neurais artificiais. As redes neurais aprendem através de casos conhecidos e em seguida podem generalizar. Exemplos de algoritmos são as redes *Multi Layer Perceptron* (MLP) (KOHONEN, 1990) e *Self Organing Map* (SOM) (KOHONEN, 1990);

- **Genético**: os algoritmos genéticos funcionam semelhantemente à evolução das espécies. Uma população de elementos de classificação inicial é criada, os melhores indivíduos são selecionados a partir de uma eurística definida e então se realiza um *crossover*, com indivíduos novos ou criados através de um processo de mutação.

2.7 Naive Bayes

O algoritmo Naive Bayes é um modelo probabilístico e supervisionado de classificação, baseado na aplicação do Teorema de Bayes. Segundo ZHANG, Naive Bayes é um dos mais eficientes e eficazes algoritmos de aprendizagem indutiva para aprendizado de máquina e mineração de dados. Seu desempenho competitivo na classificação é surpreendente, pois a suposição de independência condicional em que se baseia raramente é verdadeira em aplicações do mundo real.

No teorema de Bayes, para cada nova instância $A = a_1, a_2 \ldots a_n$, deseja-se predizer sua classe, como mostra a Equação 2.1.

$$P(classe|A) = \frac{\Big(P(A|classe) \times P(classe)\Big)}{\Big((P(A)\Big)} \tag{2.1}$$

A instância A pode ser substituída por $a_1, \ldots, a_n$ obtendo a Equação 2.2.

$$P(classe|a_1 \ldots a_n) = \frac{P(a_1 \ldots a_n|classe) \times P(classe)}{P(a_1 \ldots a_n)} \tag{2.2}$$

O cálculo da provável classe que rotula a nova instância é estatisticamente equivalente a maximizar a $P(classe|a_1 \ldots a_n)$ (PRADO et al., 2002). Para isso, é necessário maximizar o numerador e minimizar o denominador da Equação 2.2. O denominador $P(a_1 \ldots a_n)$ é anulado por ser uma constante, já que não depende da variável $classe$, obtendo a Equação 2.3.

$$\arg\max P(Classe|a_1 \ldots a_n) = \arg\max P(a_1 \ldots a_n|classe) \times P(classe) \tag{2.3}$$

A palavra *Naive* (ingênuo em inglês) no nome é dado pela suposição ingênua que o algoritmo faz de independência entre os atributos para cada instância, tornando o cálculo do valor de $P(a_1 \ldots a_n|classe)$ mais simples ao reduzi-la. Com isso, o classificador Naive-Bayes torna-se como a Equação 2.4. (PRADO et al., 2002):

$$\arg\max (P(classe|a_1 \ldots a_n)) = \arg\max P(a_1 \ldots a_n|classe) \times P(classe) \tag{2.4}$$

Este algoritmo é naturalmente incremental, significando que ao receber uma nova instância (ou um conjunto de novas instâncias), simplesmente incrementa as contagens relevantes. As predições podem ser feitas a qualquer momento a partir das contagens atuais.

O classificador Naive Bayes produz uma solução ótima quando os atributos são de fato independentes (PRADO et al., 2002). Apesar de na maioria dos casos a suposição de independência dos atributos ser falsa, o classificador ainda produz resultados satisfatórios.

2.8 Support Vector Machine

A Máquina de Suporte Vetorial ou *Suport Vector Machine* (SVM) foi elaborada a partir do estudo proposto por BOSER; GUYON; VAPNIK em 1992. Esse classificador tem a finalidade de reduzir o erro no conjunto de treinamento além de reduzir o erro em relação à um novo conjunto de dados que não foram utilizadas no processo de treinamento (SILVA, 2015). Em outras palavras, objetiva-se equilibrar o risco empírico com o risco de generalização, minimizando a quantidade de ajustes com respeito às amostras de treinamento.

Os resultados dos métodos SVM vem apresentando desempenho satisfatórios em diversas situações de classificação, como o reconhecimento de caracteres manuscritos, detecção de faces e imagens, categorização de texto e análise de bio-sequência. Por isso tornou-se uma referência na pesquisa de aprendizagem de máquina (LIMA, 2014).

No SVM, a ideia principal é mapear o espaço original (x), em um espaço de características (f), de dimensão proporcional à quantidade de características através de uma função de mapeamento não linear, conforme mostrado na Figura 4. Assim, os dados são mapeados em um espaço de características contendo várias dimensões em um plano de n-dimensões,

mesmo que os dados não sejam linearmente separáveis. Vários planos de dimensão $(n-1)$, chamados de hiperplanos são construídos, possuindo o melhor poder de generalização. Intuitivamente, realiza-se uma separação ótima entre as classes de um problema, ou seja, a distância para os pontos mais perto de cada classe, chamada de margem, é máxima.

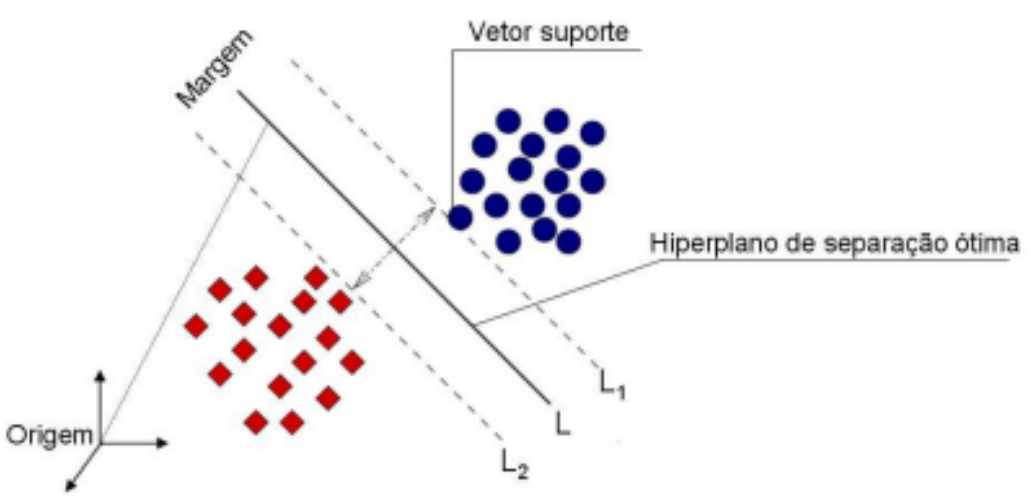

Figura 4 – Hiperplano com margem maximizado (LIMA, 2014).

Em casos reais, raramente os atributos são linearmente separáveis, necessitando de uma função *kernel* que é utilizada para mapeá-los em um espaço com uma dimensão adicional onde eles podem ser novamente separados de forma linear. Esse processo é ilustrado na Figura 5. Essa função *kernel* existe em várias formas, como Polinomial, Função de Base Radial (RBF) e curva sigmoide (SANTOS, 2010).

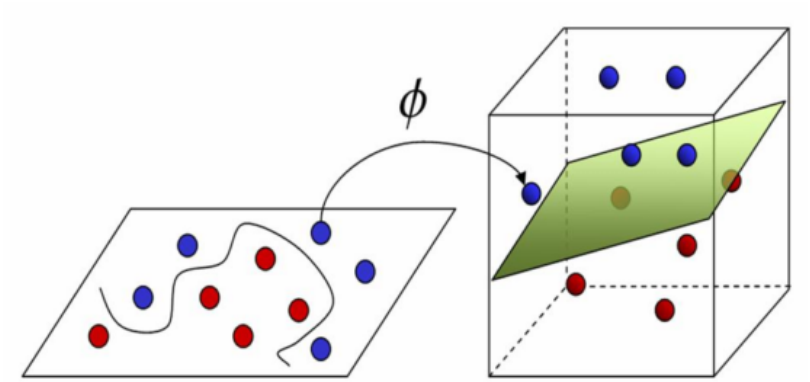

Figura 5 – *Kernel* separando classes em uma dimensão de ordem superior (LIMA, 2014).

Algumas vezes, mesmo com a utilização do *kernel*, poderão existir pontos que estejam fora do padrão esperado (*outliers*) como na Figura 6, que podem ser tratados com uso de parametrizações. Porém, o modelo resultante corre o risco de não conseguir generalizar bem novas amostras por estar muito bem ajustado aos dados treinados (*overfitting*). O ideal é buscar um balanço entre os resultados da base de teste e a generalização para novos dados,

de forma que o SVM possa produzir erros para evitar *overfitting* e tentar minimizar as chances de um resultado errado (SANTOS, 2010).

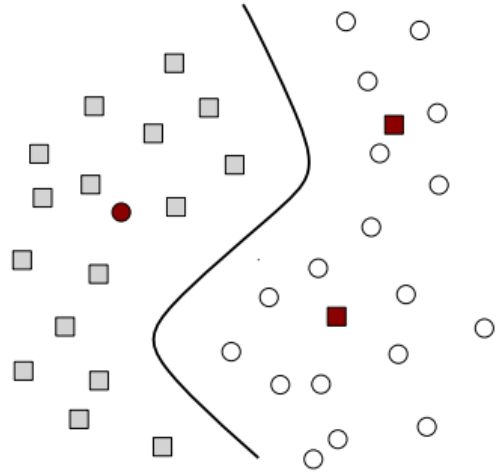

Figura 6 – Exemplo de classificação com *outliers* (LIMA, 2014).

2.9 Avaliação da Qualidade dos Classificadores

Para o desenvolvimento de classificadores, o uso de algumas medidas é necessário para se ter certeza que o treinamento do classificador foi eficaz.

- **Verdadeiro-positivo (VP)**: representa a proporção de resultados em que o classificador previu corretamente o elemento como pertencente a classe x entre todos os exemplos cujo o valor pertence à classe x;
- **Falso-positivo (FP)**: representa a proporção de resultados em que o classificador previu um elemento pertencente de outra classe como classe x;
- **Falso-negativo (FN)**: número de elementos que foram classificados como negativos quando deveriam ser positivo;
- **Verdadeiro-negativo (VN)**: número de elementos que foram classificados como negativos e deveriam ser da classe de negativos;
- **Acurácia**: é uma medida para avaliar o modelo de classificação que analisa a porcentagem de acerto do classificador (Equação 2.5);

$$Acurácia = \frac{VP + VN}{VP + FP + FN + VN} \tag{2.5}$$

- **Precisão**: representa a proporção de verdadeiro-positivos que foram previstos de forma correta (Equação 2.6);

$$Precisão = \frac{VP}{VP + FP} \tag{2.6}$$

- **Sensibilidade**: também chamada de *Recall*, representa a proporção de instâncias positivas que realmente foram previstas de forma correta: (Equação 2.7);

$$Sensibilidade = \frac{VP}{P} \tag{2.7}$$

- ***F-Measure***: usa as medidas da acurácia e precisão para definir uma medida que possa avaliar a harmonia entre precisão e sensibilidade fazendo uso de uma variável β para definir qual dos dois é o mais importante para a classificação. Para trabalhos em que precisão e sensibilidade tenha a mesma importância, β é definido como 1.

$$F_\beta = \frac{(1+\beta)^2 \times Precisão \times Recall}{\beta^2 \times Precisão \times Recall} \tag{2.8}$$

Um classificador não pode ser analisado somente pelo número de acertos pois, em alguns problemas, podem acontecer casos em que ele tende a acertar mais uma classe do que outra.

2.10 Weka

O Weka, abreviação para Waikato Environment fo Knowledge Analysis, é um *framework* que apresenta uma coletânea de algoritmos de aprendizagem de máquina para atividades de mineração de dados, contendo ferramentas para análise e preparação dos dados, classificação, regressão, clusterização e visualização, dispondo de uma interface gráfica e uma biblioteca em Java (SHEN; LIU; SHEN, 2010). Além disso, ele também oferece muitos plugins adicionais que são desenvolvidos e mantidos por uma comunidade de código aberto, além da Universidade de Waikato.

A ferramenta WEKA incorpora as seguintes etapas (PATIL; TOSHNIWAL; JOSHI, 2009):

- Análise e pré-processamento das funcionalidades da base de dados e acesso à correção dos dados;
- Definição dos atributos de classe que dividem o conjunto de instâncias nas classes apropriadas;
- Extração dos recursos potenciais a serem usados para classificação;
- Seleção de um subconjunto de recursos a serem usados no processo de aprendizagem;
- Investigação de um possível desequilíbrio nos dados selecionados e como ele pode ser neutralizado;
- Seleção de um subconjunto de instâncias, ou seja, os registros nos quais a aprendizagem deve se basear;
- Aplicação de um algoritmo de classificação para o processo de aprendizagem;
- Decisão sobre um método de teste para estimar o desempenho do algoritmo selecionado.

A forma padrão de representar conjuntos de dados que o Weka usa para receber os dados utilizados para a classificação é em formato de arquivo chamado ARFF (Attribute-Relation File Format). A representação do conjunto de dados em ARFF de um exemplo é mostrada na Figura 7.

```
@relation weather

@attribute outlook {sunny, overcast, rainy}
@attribute temperature real
@attribute humidity real
@attribute windy {TRUE, FALSE}
@attribute play {yes, no}

@data
sunny,85,85,FALSE,no
sunny,80,90,TRUE,no
overcast,83,86,FALSE,yes
rainy,70,96,FALSE,yes
rainy,68,80,FALSE,yes
rainy,65,70,TRUE,no
overcast,64,65,TRUE,yes
sunny,72,95,FALSE,no
sunny,69,70,FALSE,yes
rainy,75,80,FALSE,yes
sunny,75,70,TRUE,yes
overcast,72,90,TRUE,yes
overcast,81,75,FALSE,yes
rainy,71,91,TRUE,no
```

Figura 7 – Exemplo de arquivo Arff. (WEISS, 2010)

Os arquivos ARFF têm duas seções distintas: (i) cabeçalho, contém o nome da relação, uma lista dos atributos, que representam respectivamente cada coluna, e seus tipos, (ii) dados, em que cada linha representa uma instância e com seus seus respectivos atributos separados por vírgula. Esses dados devem ser escritos após o uso do @data para a sua declaração. (PAYNTER et al., 2008).

3 Estado da Arte

As pesquisas que abordam atividades suspeitas de usuários são geralmente divididas em duas classes: (i) identificação de contas maliciosas e (ii) identificação de mensagens maliciosas. A identificação de contas maliciosas visa relacionar padrões que admitam definir se o usuário da conta é ou não um *spammer*. A identificação de mensagens maliciosas procura relacionar o conteúdo propagado como sendo ou não atividade maliciosa (SILVA, 2015).

Nesta sessão, serão analisadas e discutidas atividades maliciosas e também o comportamento automatizado no Twitter.

3.1 Detecção de Contas Maliciosas

Os estudos que compreendem o reconhecimento de atividade maliciosa e/ou automatizada em contas de usuários do Twitter podem ser separados em dois grupos: (i) investigação das propriedades da conta e (ii) exame do domínio (AMLESHWARAM et al., 2013).

O primeiro grupo pretende investigar as propriedades das contas dos usuários, bem como o número de seguidores, número de URLs nos *tweets* e a distância em número de nós entre as vítimas e os *bots* em um grafo social. Esses trabalhos utilizam, na maioria das vezes, técnicas de aprendizagem de máquina para classificar contas como *spammers* ou legítimas, conforme um conjunto de características.

O segundo grupo, por sua vez, baseia-se na verificação das URLs, com a finalidade de identificar URLs maliciosas a partir do exame de contas capturadas em sistemas de detecção de hackers (*honeypot*), e na análise das URLs publicadas em *tweets* em comparação com listas públicas de bloqueio.

3.2 Propriedade das Contas

No trabalho de WANG, foi proposto um sistema de detecção de *spam* para identificar usuários suspeitos no Twitter utilizando um modelo de grafo social para explorar as relações entre seguidor e amigo entre os usuários. Também foram usados outros recursos baseados em conteúdo e em gráficos para facilitar a detecção de *spam*. Após, foram empregados algoritmos de classificação tradicionais para detectar comportamentos suspeitos de contas de *spam*. Um rastreador da Web usando métodos de API do Twitter também foi desenvolvido para coletar conjuntos de dados reais de informações públicas disponíveis no Twitter.

O autor analisou o conjunto de dados e avaliou que, entre os recursos baseados em gráficos, o recurso de reputação proposto possui o melhor desempenho na detecção de comportamentos anormais. Não há muitas contas de *spam* que seguem uma grande quantidade de usuários e alguns *spammers* possuem muitos seguidores.

O trabalho de BENEVENUTO et al. et al. utilizou *tweets* relacionados a três assuntos

que ficaram em alta no Twitter em 2009 e coletaram informações de 54 milhões de usuários. A partir disso, classificaram manualmente em *spammers* e não *spammers*. Em seguida, identificou uma série de características relacionadas ao conteúdo de seus *tweets* e ao comportamento social do usuário, que poderiam ser usadas para detectar *spammers*. Foram utilizadas essas características como atributos do processo de aprendizado de máquina para classificar os usuários como *spammers* ou não *spammers*. Nessa estratégia, o autor conseguiu destacar atributos importantes para a classificação de usuários no Twitter, além de classificar corretamente 70% dos *spammers* e 96% dos não *spammers*.

O trabalho de KOKUBUN; NAKAMURA analisou o uso das URLs maliciosas nas mensagens postadas no Twitter e verificou que 0.1% das URLs divulgadas no serviço possuem encurtadores de *links*, representando um risco para a rede social. Em seu estudo, foi verificado que mais de 65% das URLs que direcionavam a *sites* maliciosos durante os três primeiros meses de 2010 eram *links* encurtados. Desse percentual, 73% foram clicadas mais de 11 vezes e somente 12% das URLs não foram clicadas nenhuma vez.

Em seu estudo, o autor coletou 420 mil URLs únicas e usou a API do Google Safe Browsing para julgar se uma URL era maliciosa. O autor também verificou que praticamente todos os *spams* não estavam mais presentes para consulta após um mês da postagem, possivelmente por uma tentativa de destruição de alguma evidência pelo usuário malicioso ou por uma ação do Twitter para aplicar uma punição aos usuários que infringissem as suas regras de uso.

3.3 Detecção de Mensagens Maliciosas

As pesquisas sobre a detecção de contas maliciosas no Twitter necessitam de dados das contas, incluindo o número de seguidores e amigos, taxa de URLs, data de criação da conta, entre outros. Entretanto, grande parte dessas informações podem ser manipuladas por *spammers*. Dessa maneira, a detecção de *tweets* de *spam* isoladamente e sem informação prévia do usuário tem surgido como uma nova forma de identificar *spam*, através da aplicação de Processamento de Linguagem Natural, para extrair atributos (SILVA, 2015).

O trabalho de GHARGE; CHAVAN trata da detecção de *tweets* maliciosos, utilizando recursos baseados em modelos de linguagem que ajudam a melhorar a identificação de *spam*, sem empregar detalhes das contas dos usuários. A ideia subjacente ao estudo é a seguinte: foi realizada a análise do uso de linguagens nos *tweets* e a página onde o URL está a ele vinculado, para exame da razão de divergência entre os modelos de linguagem para classificá-los como *spam* ou não *spam*. No cenário de *spam*, os modelos de linguagem diferem-se uns dos outros. Consequentemente, a função da pessoa que postou o *spam* é desviar as pessoas para outro conteúdo totalmente irrelevante.

4 Procedimentos Metodológicos

Este trabalho é um estudo de caso e, como tal, visa investigar características peculiares em uma entidade bem definida como um programa, uma instituição, um sistema educativo, uma pessoa ou uma unidade social (FONSECA, 2002). Para isso, uma aplicação foi desenvolvida e pode ser dividida em 6 partes: coleta de dados, rotulação, extração de características, treinamento, classificação e avaliação. A Figura 8 mostra a sequência destas tarefas, assim como o aplicativo usado para realizá-las, quando disponível e não codificado pelo autor.

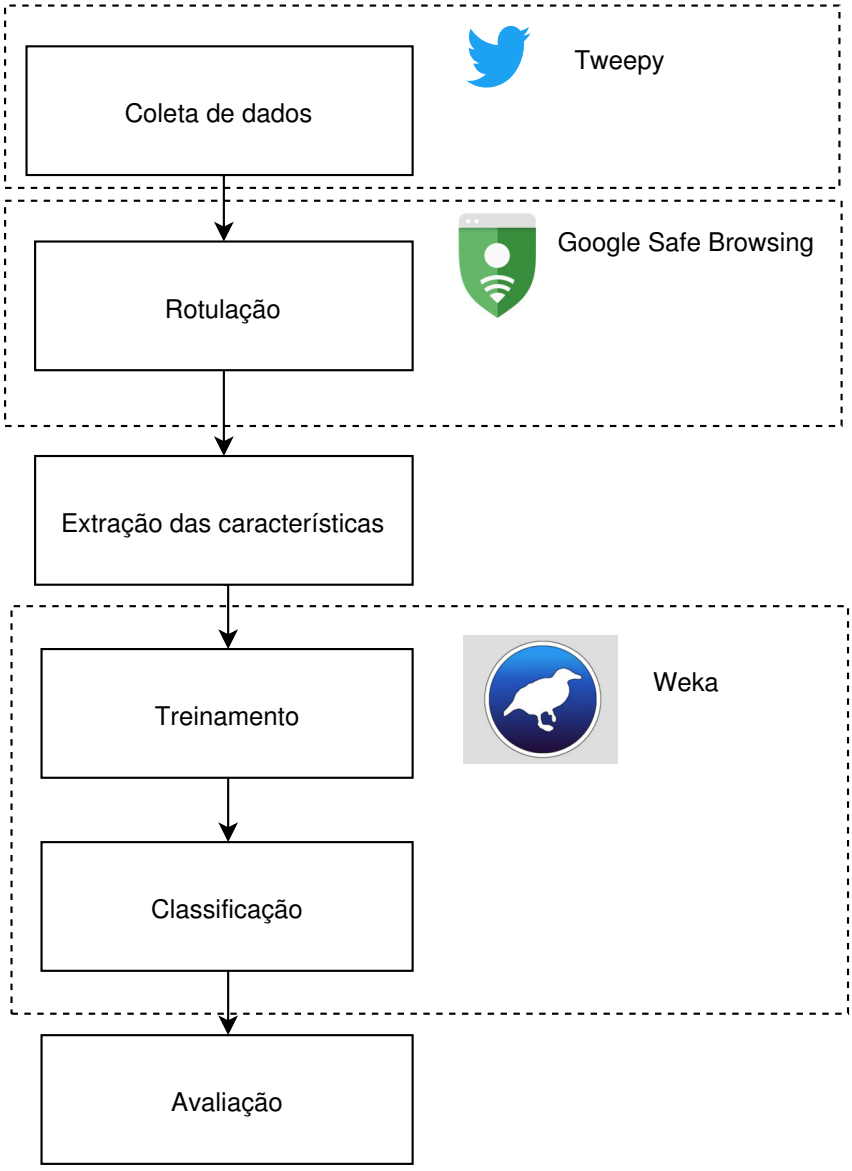

Figura 8 – Etapas para a classificação dos *tweets* (do autor).

4.1 Coleta de Dados

O acesso aos dados do Twitter é possível através de uma Interface de Programação de Aplicações (API) fornecida pela empresa, disponibilizada depois da realização de cadastro no programa de desenvolvedores, após informados os motivos de requisição de um acesso. Ao ser liberado o acesso do desenvolvedor, é necessário criar uma aplicação. Isso porque a autenticação utilizada pelo Twitter, chamada de OAuth (Twitter, 2020b), requer chaves únicas para autorizar a aplicação a ter acesso às informações.

A coleta de dados foi desenvolvida usando um *script* de captação de dados (*crawler*) em Python chamado `twitterScrapper.py` (Seção 7.1). A escolha da linguagem se deve pela existência da biblioteca `tweepy`, uma ferramenta que facilita estabelecer conexão com a API oficial do Twitter para coletar dados de acordo com um termo de busca. Essa busca pode ser feita para coletar dados em tempo real ou que foram postados em um intervalo de dados especificado. O Algoritmo 1 apresenta o pseudocódigo de como funciona a etapa de coleta dos dados.

Algorithm 1 twitterScrapper.py

```
1: consumer_key ← consumer key para Twitter
2: consumer_Secret ← consumer secret para Twitter
3: Access_token ← token key para Twitter
4: token_secret ← token secret para Twitter
5: auth ← tweepy.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
6: db ← conexão com o banco de dados
7: streamer ← tweets recebidos em tempo real atraves do tweepy
8: streamer.filter(track=WORDS)
9: função onData(data)                    //sempre acionada para cada dado adicionado à stream
10:     se data[entities][url] > 0 então
11:         a =db.tweets.insert(datajson)
12:     fim se
13: fim função
```

O retorno da API é um *tweet* com seus dados estruturados em formato JSON (JSON, 2020). Essa estrutura tem o conteúdo em formato de texto codificado segundo um esquema definido, é leve e mais fácil de computadores gerenciarem e converterem em outras estruturas. Foi escolhido MongoDB como banco de dados utilizado nesse trabalho para armazenar as informações, uma vez que já salva os dados na estrutura JSON. Os dados do Twitter coletados e armazenados são exemplificados na Figura 9.

```
1    created_at  : "Sun Jul 07 18:15:48 +0000 2019    "                String
2    id : 1147932241978281984                                            Int64
3    id_str  : "1147932241978281984   "                                  String
4    text : "RT @Amy_Siskind: Real president (remember wh…"              String
5    source  : "<a href="http://twitter.com/download/iphone"…"           String
6    truncated  : false                                                  Boolean
7    in_reply_to_status_id  : null                                       Null
8    in_reply_to_status_id… : null                                       Null
9    in_reply_to_user_id   : null                                        Null
10   in_reply_to_user_id_s… : null                                       Null
11   in_reply_to_screen_na… : null                                       Null
12 › user  : Object                                                      Object
13   geo  : null                                                         Null
14   coordinates  : null                                                 Null
15   place  : null                                                       Null
16   contributors  : null                                                Null
17 › retweeted_status  : Object                                          Object
18   quoted_status_id  : 1147918213952360448                             Int64
19   quoted_status_id_str  : "1147918213952360448   "                    String
20 › quoted_status  : Object                                             Object
21 › quoted_status_permali… : Object                                     Object
22   is_quote_status  : true                                             Boolean
23   quote_count  : 0                                                    Int32
24   reply_count  : 0                                                    Int32
25   retweet_count  : 0                                                  Int32
26   favorite_count  : 0                                                 Int32
27 › entities  : Object                                                  Object
28   favorited  : false                                                  Boolean
29   retweeted  : false                                                  Boolean
30   possibly_sensitive  : false                                         Boolean
31   filter_level  : "low  "                                             String
32   lang : "en "                                                        String
33   timestamp_ms  : "1562523348694   "                                  String
```

Figura 9 – Estrutura de um *tweet* salvo no banco de dados MongoDB (do autor).

Os *tweets* foram coletados no período de 05 a 12 de setembro de 2019, filtrados por assuntos presentes nos *trending topics* dos respectivos dias. Essa escolha se deu por serem assuntos mais populares no momento e onde os usuários estão mais engajados, o que além de permitir uma base de dados maior devido à grande concentração de *tweets* em torno desses assuntos, ainda é onde os usuários estão mais vulneráveis a ataques. A coleta trouxe 343202 mensagens para o banco de dados, e não houve limite de região e idioma.

4.2 Rotulagem

A rotulação das mensagens foi realizada após a verificação de cada URL presente em um *tweet*, avaliados como *spam* ou não pelo Google Safe Browsing. A conexão com essa API foi realizada através de um *script* escrito em Node.js durante a extração de características.

O Google Safe Browsing é uma API gratuita para uso não comercial que permite a verificação de confiabilidade de um *site* através de uma conferência em uma lista de *sites* não confiáveis mantida pela empresa. Para se obter acesso à API é necessário ter um projeto no Google Developer Console e retirar uma credencial (`API_KEY`) usada na criação das requisições. Apesar da facilidade para verificação, existe a limitação de 10000 requisições diárias.

Com a rotulagem foram detectados 5000 *tweets* com URLs maliciosas e, para simplificar a base e deixar os dados simétricos, foram usados também 5000 *tweets* que não continham *spam*.

4.3 Extração de Características

Nesta etapa, os vetores de características dos dados rotulados são gerados a partir da extração de informações dos *tweets* processados nas etapas anteriores. 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 padrões no seu comportamento.

As características utilizadas neste trabalho seguem os apresentados pelo BARBOSA, sendo eles:

Idade da conta: essa caracterização refere-se ao tempo total entre a criação da conta e a postagem da mensagem no Twitter. Usuários maliciosos costumam ter contas que não têm uma vida útil longa, pois assim que são denunciados por outros usuários ou notados pelo Twitter, têm sua conta excluída. É comum também que *phishers* abandonem a conta assim que considerarem o ataque como concluído;

Quantidade de seguidores: é uma caracterização relacionada ao comportamento dos usuários em relação a sua rede social. Usuários maliciosos buscam popularidade para ter mais alvos de seus ataques. Quanto mais seguidores eles possuem, mais visíveis são suas mensagens;

Quantidade de usuários que estão seguindo: essa caracterização tem o propósito de popularizar um perfil e aumentar a visibilidade das mensagens divulgadas;

Listas do usuário: é possível criar e/ou participar de grupos de contatos, o que no Twitter são conhecidos como listas. Essa caracterização 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;

Quantidade de *tweets* do usuário: um comportamento característico de usuários maliciosos é a alta frequência de postagens. O uso de robôs para disseminação de *tweets* maliciosos aumenta a quantidade média de *tweets* enviados, pois o fazem de forma automatizada;

Quantidade de *hashtags*: *phishers* ou *bots* fazem uso de *hashtags* e de *trend topics* no texto do *tweet* para aumentar as chances do *tweet* malicioso alcançar um número maior de usuários;

Quantidade de usuários mencionados: essa caracterização é utilizada pelos *phishers* para chamar a atenção de determinados usuários, uma vez que os *tweets* aparecem nas linhas

do tempo dos perfis marcados;

Quantidade de *retweets*: o uso de contas para redirecionarem mensagens maliciosas é comum. Contas com o objetivo de compartilhar mensagens são criadas pelos *phishers* e tem por objetivo aumentar a sua rede de contatos e consequentemente, sua propagação;

Quantidade de URLs: essa caracterização contabiliza a quantidade de URLs contidas no texto do *tweet*. Uma das características mais importantes na detecção de *tweets* maliciosos é o padrão de uso de robôs na criação de mensagens para disseminação de *links* maliciosos. Essa caracterização analisa o texto da mensagem e é obtida por meio da contagem da quantidade de URLs curtas presentes. Seu resultado é um valor numérico positivo;

Quantidade de caracteres: essa caracterização está relacionada ao tamanho da mensagem e é calculada a partir da contagem de caracteres do *tweet*. Os *tweets* criados por robôs apresentam padrões de escrita e similaridade no tamanho e conteúdo;

Quantidade de dígitos: essa caracterização conta a quantidade de números na mensagem. Ela é importante porque algumas mensagens maliciosas tentam trocar letras por números a fim de atrair os usuários para *sites* que têm nomes similares aos originais, diferenciando poucos caracteres, como por exemplo `paypa1` ou `p4yp4l`.

A extração é feita utilizando um *script* em Node.js que coleta, processa e grava o resultado em arquivo. Cada coluna representa um *tweet*, e cada linha representa uma das 13 características extraídas, como exemplificado na Tabela 1:

Característica	*Tweet* 1	*Tweet* 2	*Tweet* 3	*Tweet* 4
Idade da Conta	177	708	847	18
Quantidade de Seguidores	112	103	2	0
Quantidade de Seguindo	248	107	365	5
Quantidade de Usuários Favoritos	3055	1	0	0
Listas	0	1	0	0
Quantidade de *Tweets*	14899	1080	409	27
Quantidade de *Retweets*	0	0	0	2901
Quantidade de *Hashtags*	1	0	0	0
Quantidade de menção de usuários	0	0	0	1
Quantidade de URLs	1	1	2	1
Quantidade de Caracteres	115	63	65	75
Quantidade de Dígitos	9	0	20	0
Classe	Não *Spam*	Não *Spam*	*Spam*	*Spam*

Tabela 1 – Exemplo da estrutura dos dados extraídos.

4.4 Treinamento e Classificação

Após realizada a extração de características, foi criada uma base de treino contendo 40% dos *tweets* de cada classe, ou seja, 2000 *tweets* para cada classe foram separados para serem utilizados como base de treinamento para os algoritmos. Os outros 60% que representam 3000 *tweets* de cada classe, foram utilizados para realização do teste.

O processo de treinamento da base foi realizado com os algoritmos de aprendizagem de máquina Naive Bayes e SVM, resultando em um modelo que é aplicado na base de treinamento. Para essa etapa foi desenvolvido um algoritmo em Java utilizando o *framework* Weka (Seção 7.2).

Após o treinamento, foi realizada a classificação de mensagens de teste utilizando o modelo gerado pela fase anterior. Foi verificado manualmente se a precisão do classificador para as classes existentes era alta e se a matriz de confusão gerada para a base de teste apresentou baixo número de falso-positivos, demonstrando um resultado satisfatório. Cada posição na matriz representou a quantidade de instâncias em sua classe original e como os classificadores as previram.

4.4.1 Escolha dos Algoritmos

A escolha dos algoritmos citados se deve ao fato de apresentarem resultados com acurácia aceitável e por construírem modelos mais rápidos em relação a algoritmos de redes neurais (Capítulo 3), mesmo em computadores pessoais.

5 Resultados Obtidos

Este capítulo apresenta os principais resultados obtidos durante o desenvolvimento da pesquisa com a aplicação da abordagem proposta na base selecionada. Além disso, serão feitas discussões e análises dos dados levantados.

5.1 Análise Empírica

Analisando o comportamento entre todos os 10 mil *tweets* coletados, sendo 5 mil de cada classe, os usuários comuns ou legítimos, aqueles que não utilizam o serviço para praticar *phishing*, costumam usar as *hashtags* uma vez por mensagem enquanto comentam sobre o assunto relativo a *hashtag*. Já as contas com suspeita de *phishing*, abusavam da quantidade de *hashtags* para conseguir atingir o maior número de *trending topics* e, consequentemente, alcançar mais pessoas. É comum ver também o uso de *hashtags* diferentes para o mesmo *link* após um curto espaço de tempo.

Na Figura 10 é possível ver que quase 50% dos *tweets* considerados não maliciosos coletados contêm somente uma *hashtag*, e que a proporção reduz drasticamente em quantidades maiores. Entre os *tweets* maliciosos, aproximadamente 25% possuem somente uma *hashtag*, e a variação é bem mais sutil.

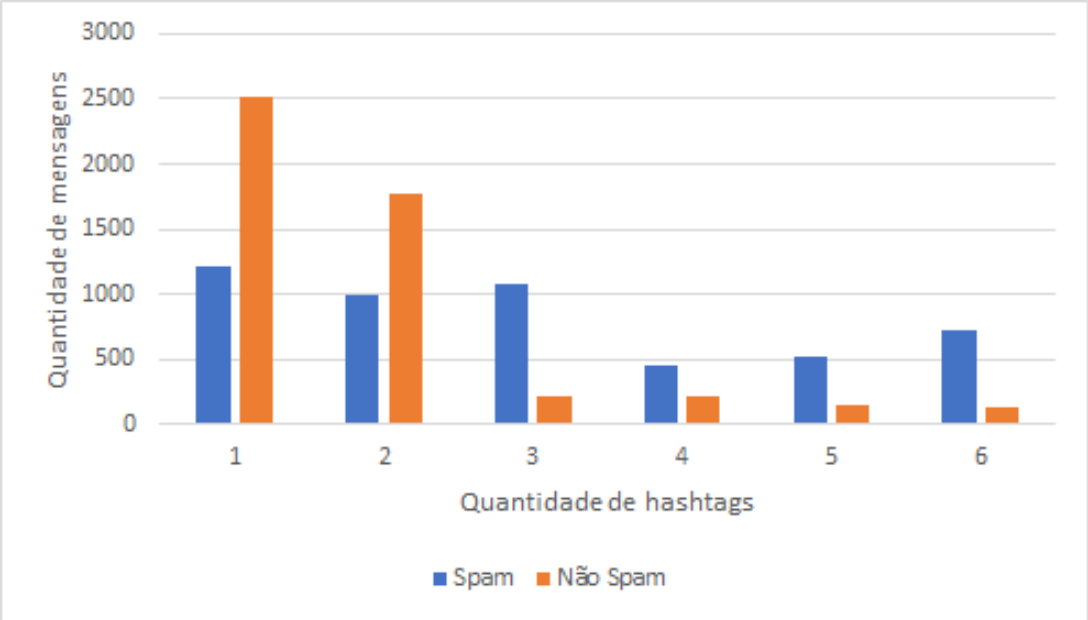

Figura 10 – Quantidade de *hashtags*.

Quanto aos *links* presentes nas mensagens não *spam*, verifica-se que estes são comumente usados para compartilhar uma notícia, imagens ou vídeos que complementam a informação que o autor deseja expressar. Com isso, mais de 90% das mensagens coleta-

das dos usuários não *spam* do serviço contêm somente um *link*. Já em *tweets* maliciosos, é possível ver o uso de URL aproximadamente 30% maior (Figura 11).

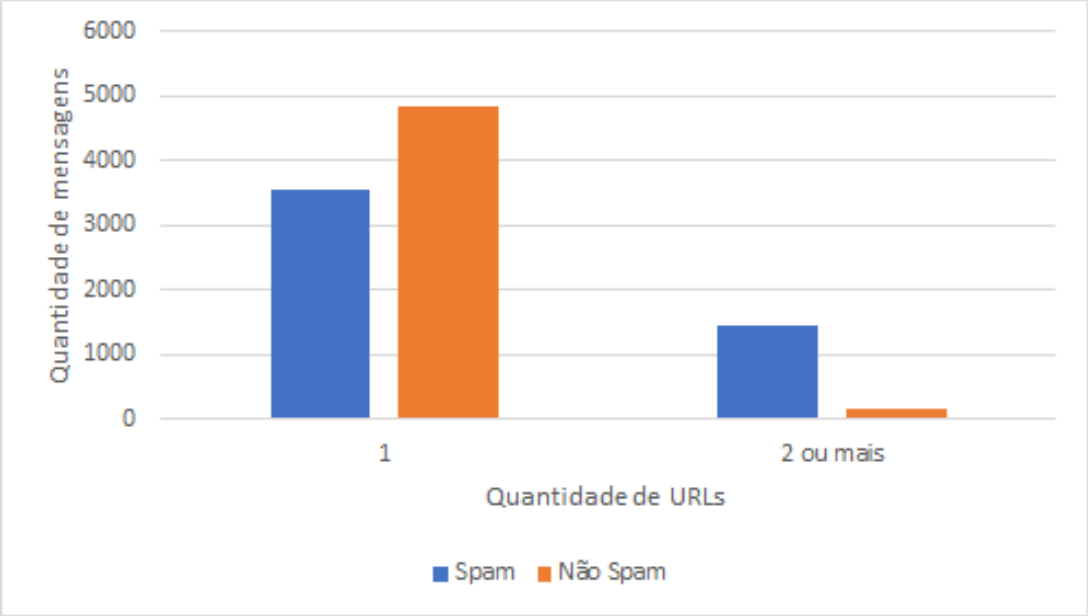

Figura 11 – Quantidade de URLs.

Os seguidores de contas de usuários maliciosos são, em maioria, contas pouco seguidas. Isto mostra que a maioria dos *spams* capturados pelo Google Safe Browsing (Google Safe Browsing Project, 2011) não estava usando técnicas para se obter mais usuários os seguindo. Como os *phishers* estavam usando estratégia de usar as *hashtags* para popularizar suas mensagens, é esperado que não tenham muitos seguidores (Figura 12).

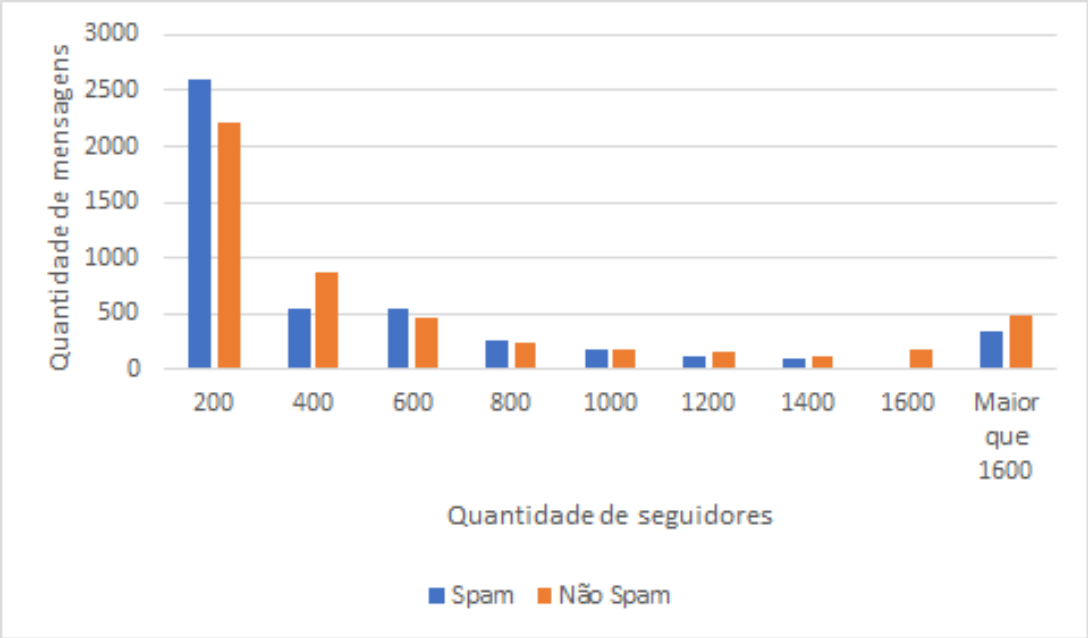

Figura 12 – Quantidade de seguidores.

Entre todas as características analisadas, a que mais evidencia uma conta *spammer* é o tempo desde sua criação, em sua maioria com menos de 60 dias. Isso é esperado, pois usuários maliciosos usam várias contas novas para realizar o ataque e, assim que não são mais úteis, as descartam (Figura 13).

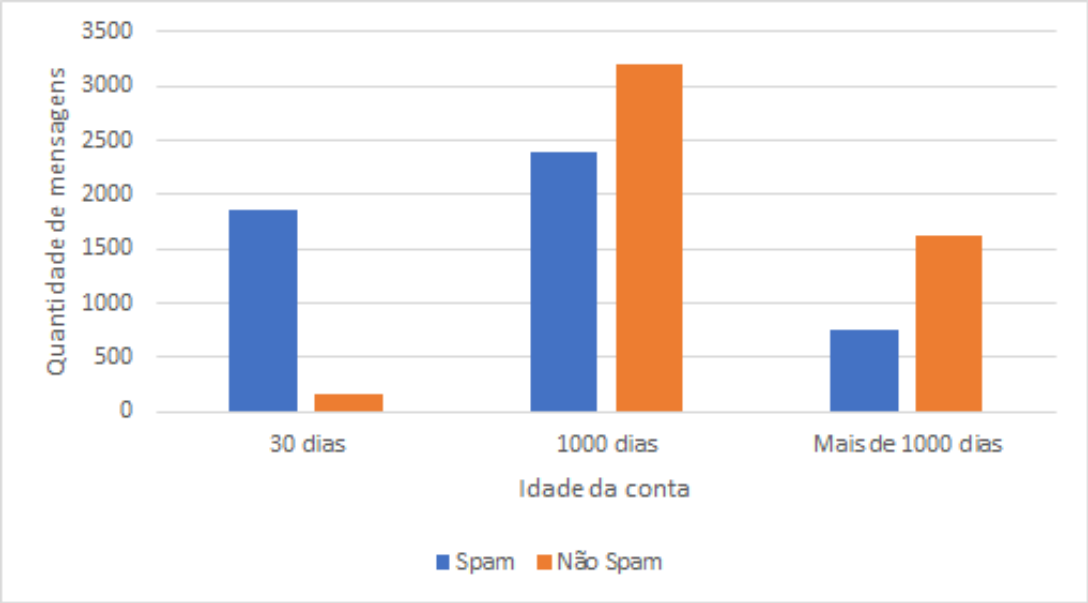

Figura 13 – Idade da conta do Twitter.

5.2 Correlação das Características

Na Figura 14 são exibidas as classificações de algumas dessas mensagens obtidas. É possível observar que as mensagens dos usuários maliciosos são mais simples e contêm mais *links* encurtados. Nela é apresentada a taxa de correlação de Pearson dos dados usados no trabalho após a normalização e retirada dos dados duplicados. Nota-se também que, quanto mais escura uma célula, mais os valores da respectiva linha e coluna estão correlacionados.

Ainda na Figura 14, é possível visualizar que nem todas as características são correlacionadas, mas um exemplo de grande correlação é o número de listas com o número de seguidores e o número de listas com o número de perfis que o seguem. A correlação é justificável, pois um usuário que tende a ter vários seguidores ou vários o seguindo, faz uma lista de “melhores amigos” para compartilhamento de dados futuros.

A não linearidade entre os valores é muito importante para os algoritmos SVM e Naive Bayes, pois elas assumem que todas as instâncias são independentes entre si, ou seja, não podem ser correlacionadas.

5.3 Avaliação dos Classificadores

Na fase de classificação, foi usada a base de dados de treinamento com 2000 *tweets* de cada classe para treinar os algoritmos de classificação Naive Bayes e SVM utilizando

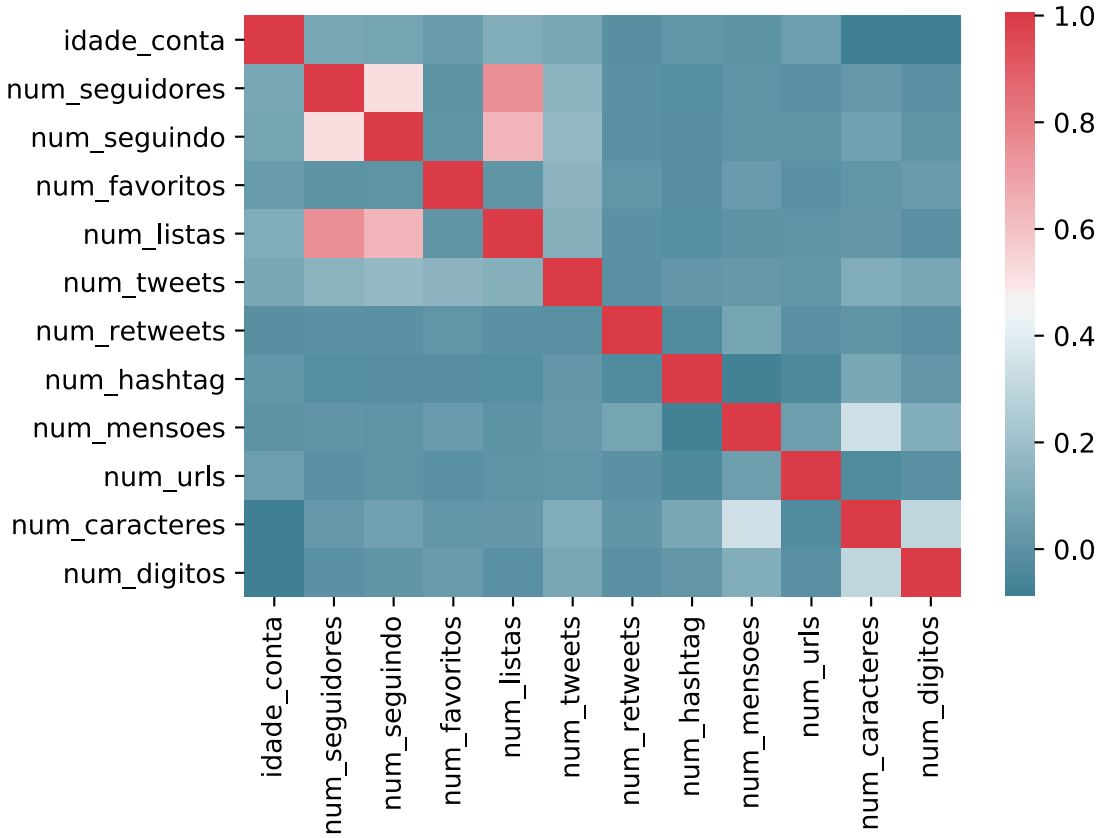

Figura 14 – Matriz de correlação.

o *framework* Weka em Java e, após isso, é usado o modelo usado para treinar a base de treinamento composto pelos 6000 *tweets* restantes.

5.3.1 Naive Bayes

Na classificação com o algoritmo Naive Bayes, o Weka permite duas formas de parametrização para a geração de um modelo: multinominal e gaussiano. Apesar disso, somente o modelo Gaussiano é aplicável ao problema, pois o modelo Multinominal é aplicável apenas para casos em que as instâncias não tem a mesma quantidade de características.

Na Figura 15 observa-se que o algoritmo teve uma precisão de 90% em mensagens *spam*, ou seja, mensagens classificadas como *spam* que foram classificadas corretamente. Entretanto, nota-se que o nível de acerto das previsões em relação às mensagens classificadas como não *spam* é baixo. A deficiência do classificador pode ser observada no número elevado de falso-positivo, classificando erroneamente aproximadamente 78% das mensagens como não *spam*.

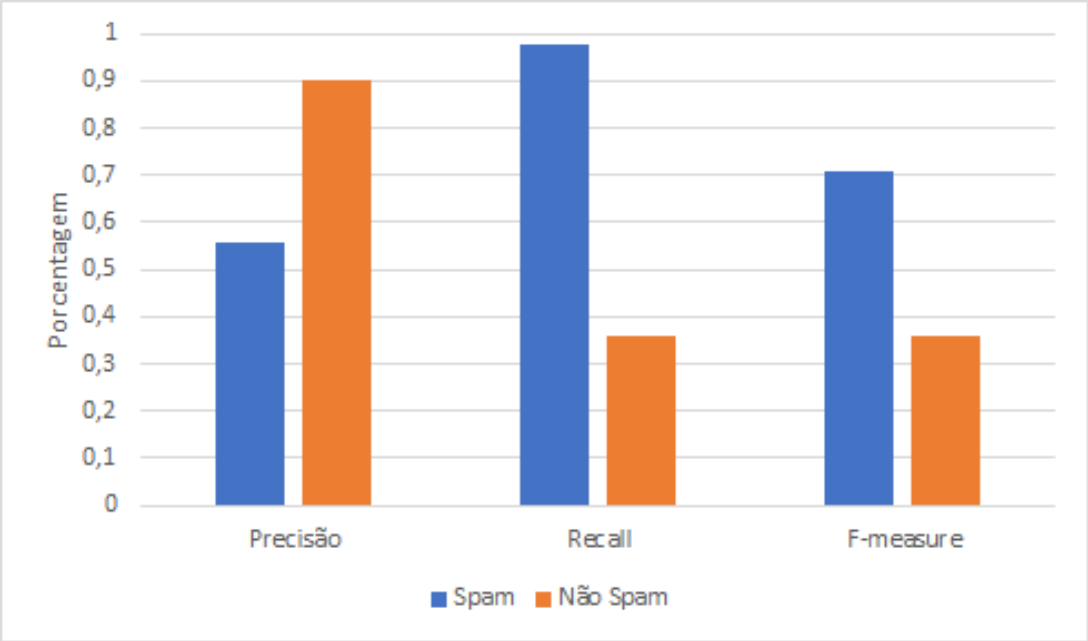

Figura 15 – Valores de desempenho do classificador Naive Bayes.

Apesar de o Naive Bayes conseguir identificar os usuários maliciosos, é perceptível através da Tabela 2 a dificuldade de conseguir classificar corretamente os usuários não *spam*, tornando o modelo não utilizável em uma situação real.

		Classificação predita		
		Spam	Não Spam	Total
Classificação correta	Spam	2929	71	3000
	Não Spam	2330	670	3000
	Total	5259	741	6000

Tabela 2 – Matriz de confusão do classificador Naive Bayes.

5.3.2 SVM

Para o classificador SVM, o Weka (seção 2.10) permite usar como parâmetro diferentes configurações do *kernel*, que alteram a forma como as instâncias são mapeadas em um hiperplano. Também é possível alterar o parâmetro C, responsável por controlar a rigidez da classificação em relação à tolerância a erros, o qual varia de 0 a 1. Quanto maior o seu valor, menos tolerante a erros será o modelo gerado. Neste trabalho, a classificação foi testada com diferentes variações de *kernel* e parâmetro C. Verificou-se que o algoritmo apresentou melhor resultado com o *kernel* linear e com C igual a 1, ou seja, com maior tolerância a erro.

Na Tabela 3, observa-se que o SVM teve um número muito baixo de falso-positivo, evidenciando que este algoritmo teve um desempenho melhor que o Naive Bayes para classificação, tanto de *spam* quanto de não *spam*. A Figura 16 reforça este entendimento, mostrando que o F-*measure* do algoritmo, ou seja, a taxa entre a precisão e o *recall*, foi superior a 90% nos dois algoritmos.

		Classificação predita		
		Spam	Não Spam	Total
Classificação correta	Spam	2590	410	3000
	Não Spam	836	2164	3000
	Total	3426	2574	6000

Tabela 3 – Matriz de confusão do classificador SVM.

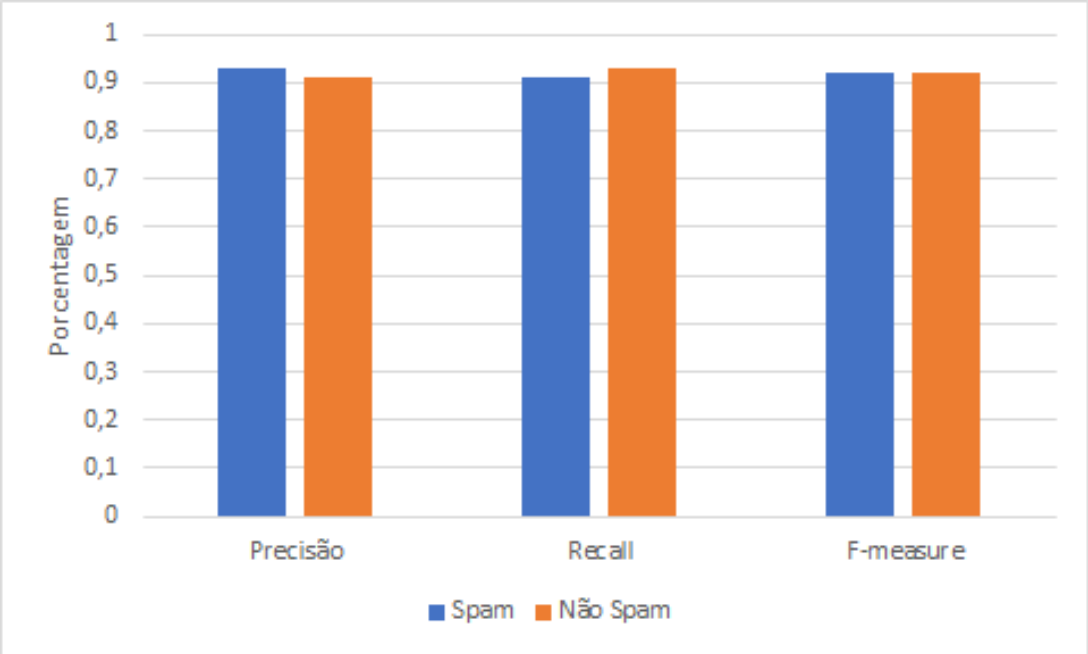

Figura 16 – Valores de desempenho do classificador SVM.

Ao analisar a Figura 16, é possível observar que a classificação das mensagens através do SVM tem precisão de 93% para detecção de *spam* e 91% para detecção de não *spam*. Além disso, a taxa de acerto das duas classes (*recall*) foi acima de 90%.

6 Conclusão

A busca automática por mensagens maliciosas é um dos grandes problemas na era digital. Um exemplo importante disso são os ataques de *phishing* contra os usuários da Internet. O presente trabalho buscou caracterizar *tweets* presentes nos *trending topics* e automatizar a identificação daqueles com potencial de serem mal intencionados. Foram comparados os desempenhos de dois algoritmos de aprendizagem de máquina: Naive Bayes e SVM.

Para isso, empregou-se um *crawler* feito em Python e utilizou-se o *framework tweepy* para a extração e coleta de *tweets* presentes em *trending topics* do Twitter, sendo rotuladas 5000 mensagens de cada classe com a aplicação do Google Safe Browsing. Após isso, foram extraídas as características do Twitter para a utilização na classificação.

Com as características extraídas dos *tweets*, foi observado que os classificados como *spam* possuem algumas propriedades evidentes, tais como maior quantidade de *hashtag*, menor número de seguidores e os seus usuários possuem contas de curta duração em relação às contas de usuários legítimos.

As características foram utilizadas nos algoritmos Naive Bayes e SVM para a classificação de forma automática. Observou-se que o algoritmo SVM apresentou maior precisão na detecção de *tweets* maliciosos, com poucos falso-positivos. Já o algoritmo Naive Bayes não teve uma boa precisão, apresentando muitos falso-positivos.

A maior dificuldade enfrentada neste trabalho foi encontrar *spams* no meio de inúmeros *tweets* que estão presentes nos *trending topics* devido ao limite de consultas da API do Google Safe Browsing. Isto prejudicou a tarefa de rotulagem em razão da dificuldade de coleta de uma base de dados grande o suficiente para a classificação e teste que foram aplicados.

6.1 Trabalhos Futuros

Para a continuidade deste trabalho, sugere-se o uso de algoritmos classificadores não explorados nesse trabalho. Outra sugestão é a análise de consumo de recursos computacionais empregados pelos classificadores, tais como memória e tempo gasto para classificação, com a finalidade de analisar o uso desses *softwares* em computadores pessoais em tempo real.

7 Apêndice

7.1 Apêndice A - Crawler para *tweets*

```
class StreamListener(tweepy.StreamListener):

    def on_connect(self):

        print("Conctado a API")

    def on_error(self, status_code):
        print('Ocorreu um erro: ' + repr(status_code))
        return False

    def on_data(self, data):

        try:
            client = MongoClient(MONGO_HOST)

            db = client.tabela

            # Decodifica o JSON do Twitter
            datajson = json.loads(data)
            url = datajson['entities']['urls']

            # Verifica se o tweet tem URL
            if(len(url) > 0):

                created_at = datajson['created_at']

                print("Tweet collected at " + str(created_at))

                # Guarda os dados no banco
                a = db.tweets.insert(datajson)
                print(a)
        except Exception as e:
           print(e)

auth = tweepy.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
auth.set_access_token(ACCESS_TOKEN, ACCESS_TOKEN_SECRET)


listener = StreamListener(api=tweepy.API(wait_on_rate_limit=True))
streamer = tweepy.Stream(auth=auth, listener=listener)
print("Tracking: " + str(WORDS))
```

```
streamer.filter(track=WORDS)
```

7.2 Apêndice B - Código do classificador Weka

```
package tweet_analise;


import weka.core.Instances;
import weka.core.converters.ArffLoader;
import weka.classifiers.bayes.*;
import weka.classifiers.meta.FilteredClassifier;
import weka.filters.unsupervised.attribute.StringToWordVector;
import weka.classifiers.functions.*;
import java.io.File;
import weka.classifiers.Evaluation;
import weka.classifiers.functions.LibSVM;


public class Tweet_analise {

  public static FilteredClassifier naive_bayes(Instances train) throws Exception{
      NaiveBayes nb = new NaiveBayes();

      String[] options = weka.core.Utils.splitOptions("-D");
      nb.setOptions(options);



      FilteredClassifier fc = new FilteredClassifier();

      fc.setClassifier(nb);
      fc.buildClassifier(train);

      return fc;
  }

  public static FilteredClassifier SVM(Instances train) throws Exception{
      LibSVM svm = new LibSVM();
      String[] options = weka.core.Utils.splitOptions("-K 0 -C 1 -W 1");


      FilteredClassifier fc = new FilteredClassifier();

      svm.setOptions(options);
      fc.setClassifier(svm);
      fc.buildClassifier(train);
```

```
    return fc;
}


public static void main(String[] args) throws Exception {
  // Inicia o carregamento do arquivo
  ArffLoader loader = new ArffLoader();
  ArffLoader testBase = new ArffLoader();

  //treinamento: varivel que guarda o endereo para o arquivo Arff para treinar o
      algoritmo de aprendizagem de mquina
  // teste: varivel que guarda o endereo para o arquivo Arff para testar o
      algoritmo de aprendizagem de mquina

  loader.setFile(new File(treinamento));
  testBase.setFile(new File(teste));

  //Estrutura os atributos para as bases de teste e treinamento
  Instances structure = loader.getDataSet();
  structure.setClassIndex(structure.numAttributes() - 1);

  Instances structureTest = testBase.getDataSet();
  structureTest.setClassIndex(structure.numAttributes() - 1);

  //Randomiza a ordem da base de treinamento
  structure.randomize(new java.util.Random());


  int trainSize = (int) Math.round(structure.numInstances());
  int testSize = (int) Math.round(structureTest.numInstances());

  Instances train = new Instances(structure, 0, trainSize);
  Instances test = new Instances(structureTest, 0, testSize);

  FilteredClassifier nb = naive_bayes(train);
  FilteredClassifier svm = SVM(train);


  Evaluation eval_NB = new Evaluation(test);
  eval_NB.evaluateModel(nb, test);


  Evaluation eval_SVM = new Evaluation(test);
  eval_SVM.evaluateModel(svm, test);

  System.out.println("Naive Bayes");
  System.out.println(eval_NB.toSummaryString(true));
  System.out.println(eval_NB.toClassDetailsString());
```

```
    System.out.println("SVM");
    System.out.println(eval_SVM.toSummaryString(true));
    System.out.println(eval_SVM.toClassDetailsString());
  }
}
```

Referências

ALBERTO, C. et al. MDLText e Indexação Semântica aplicados na Detecção de Spam nos Comentários do YouTube. v. 10, n. 3, p. 49–73, 2017. Citado na página 14.

AMLESHWARAM, A. A. et al. CATS: Characterizing automation of Twitter spammers. *2013 5th International Conference on Communication Systems and Networks, COMSNETS 2013*, IEEE, p. 1–10, 2013. Citado na página 24.

BARBOSA, H. P. d. O. Detecção De Phishing No Twitter Baseada Em Algoritmos De Aprendizagem Online. 2018. Citado na página 29.

BENEVENUTO, F. et al. Detecting Spammers on Twitter. *In Collaboration, electronic messaging, anti-abuse and spam conference*, v. 6, p. 1–9, 2010. Citado na página 24.

BOSER, B. E.; GUYON, I. M.; VAPNIK, V. N. A training algorithm for optimal margin classifiers. *Proceedings of the fifth annual workshop on Computational learning theory - COLT '92*, p. 144–152, 1992. ISSN 0-89791-497-X. Disponível em: <http://portal.acm.org/citation.cfm?doid=130385.130401>. Citado nas páginas 17 e 19.

CAMILO, C. et al. Mining association rules in graphs based on frequent cohesive itemsets. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, v. 9078, n. 3, p. 637–648, 2015. ISSN 16113349. Disponível em: <http://www.aaai.org/ojs/index.php/aimagazine/article/view/1230%5Cnhttp://www.rbc.lsie.unb.br/index.php/rbc/article/viewFile/594/577%5Cnhttp://portal.acm.org/citation.cfm?doid=1900008.1900067%5Cnhttp://www.inf.ufg.br/sites/default/files/uploads/relatorios->. Citado na página 16.

CHEN, C. et al. 6 million spam tweets: A large ground truth for timely Twitter spam detection. *IEEE International Conference on Communications*, IEEE, v. 2015-Septe, p. 7065–7070, 2015. ISSN 15503607. Citado na página 11.

CLEMENT, J. *Facebook: most users by country | Statista*. 2019. Disponível em: <https://www.statista.com/statistics/268136/top-15-countries-based-on-number-of-facebook-users/>. Citado nas páginas 13, 14 e 15.

FILHO, R. M. Um arcabouço para pesquisas de opinião em redes sociais. p. 118, 2014. Citado na página 11.

FONSECA, J. J. S. d. Metodologia da Pesquisa Científica. *UECE - Universidade Estadual do Ceará*, 2002. Citado na página 26.

GHARGE, S.; CHAVAN, M. An Integrated approach for Malicious Tweets detection using NLP . *ICICCT*, 2017. Disponível em: <https://ieeexplore-ieee-org.ez107.periodicos.capes.gov.br/stamp/stamp.jsp?tp=&arnumber=7975235>. Citado na página 25.

GOLDBERGER, J. et al. *Neighbourhood Components Analysis*. [S.l.], 2000. Citado na página 18.

Google Safe Browsing Project. *Google safe browsing api homepage*. 2011. Disponível em: <http://code.google.com/apis/safebrowsing/>. Citado na página 33.

JSON. *JSON*. 2020. Disponível em: <https://www.json.org/json-pt.html>. Citado na página 27.

KOHONEN, T. The Self-Organizing Map. *Proceedings of the IEEE*, v. 78, n. 9, p. 1464–1480, 1990. ISSN 15582256. Citado na página 18.

KOKUBUN, Y.; NAKAMURA, A. Analysis of Malicious URLs on Twitter. In: *2018 International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE, 2018. p. 1285–1288. ISBN 978-1-7281-1360-9. Disponível em: <https://ieeexplore.ieee.org/document/8947647/>. Citado na página 25.

LIMA, R. Avaliação Do Algorítimo Svm Na Detecção De Comportamentos Suspeitos Em Cenas De Vídeo. 2014. Disponível em: <http://repositorio.roca.utfpr.edu.br/jspui/bitstream/1/6469/1/PG_COADS_2014_2_04.pdf>. Citado nas páginas 19, 20 e 21.

MONARD, M. C.; BARANAUSKAS, J. A. Conceitos sobre Aprendizado de Máquina. *Sistemas inteligentes: fundamentos e aplicações*, p. 89–114, 2003. Citado nas páginas 16, 17 e 18.

OLIVO, C. K.; OLIVEIRA, A. O. S. L. E. S. Capítulo 4 Abordagens para Detecção de Spam de E-mail. p. 141–182, 2015. Citado nas páginas 12 e 15.

PATIL, B. M.; TOSHNIWAL, D.; JOSHI, R. C. Predicting burn patient survivability using decision tree in WEKA environment. In: *2009 IEEE International Advance Computing Conference, IACC 2009*. [S.l.: s.n.], 2009. p. 1353–1356. ISBN 9781424429288. Citado na página 22.

PAYNTER, G. et al. *Attribute-Relation File Format (ARFF)*. 2008. Disponível em: <https://www.cs.waikato.ac.nz/~ml/weka/arff.html>. Citado na página 23.

PRADO, T. A. S. et al. Aprendizado Bayesiano Aplicado ao Processamento de Línguas Naturais. *Série de Relatórios do Núcleo Interinstitucional de Linguiística Computacional*, p. 26, 2002. Disponível em: <http://www2.icmc.usp.br/~taspardo/NILCTR0225-PardoNunes.pdf>. Citado nas páginas 17, 18 e 19.

Retruster; Malwarebytes. *2019 Phishing Statistics and Email Fraud Statistics*. Disponível em: <we>. Citado na página 11.

Reuters. *Número de usuários do Twitter cresce após limpeza na rede social - Link - Estadão*. 2019. Disponível em: <https://link.estadao.com.br/noticias/empresas,numero-de-usuarios-do-twitter-cresce-apos-limpeza-na-rede-social,70002800800>. Citado na página 11.

ROSALES, F. *Ataques de phishing*. [s.n.], 2015. Disponível em: <https://www.ocu.org/tecnologia/internet-telefonia/consejos/evitar-ataque-phishing#>. Citado nas páginas 11, 14, 15 e 16.

ROTH, Y.; HARVEY, D. *Como o Twitter está combatendo spam e automação mal-intencionada*. Disponível em: <https://blog.twitter.com/pt_br/topics/company/2018/como-o-twitter-esta-combatendo-spam-e-automacao-mal-intencionada.html>. Citado nas páginas 11 e 12.

SANCHES, M. K. Aprendizado de máquina semi-supervisionado: proposta de um algoritmo para rotular exemplos a partir de poucos exemplos rotulados. 2003. Citado nas páginas 16 e 17.

SANTOS, L. M. Em Redes Sociais : Estudo De Casos Selecionados Usando O Twitter Lavras - Mg. 2010. Citado nas páginas 20 e 21.

SHEN, Y.; LIU, J.; SHEN, J. The Further Development of Weka Base on Positive and Negative Association Rules. In: *2010 International Conference on Intelligent Computation Technology and Automation*. IEEE, 2010. v. 3, p. 811–814. ISBN 978-1-4244-7279-6. Disponível em: <http://ieeexplore.ieee.org/document/5523114/>. Citado na página 22.

SILVA, A. S. d. Detectando Comportamento Automatizado nos Tópicos de Tendência do Twitter no Brasil. 2015. Disponível em: <https://sucupira.capes.gov.br/sucupira/public/consultas/coleta/trabalhoConclusao/viewTrabalhoConclusao.jsf?popup=true&id_trabalho=3008474#>. Citado nas páginas 13, 19, 24 e 25.

Twitter. *About different types of Tweets*. 2020. 1 p. Disponível em: <https://help.twitter.com/en/using-twitter/types-of-tweets>. Citado na página 13.

Twitter. *OAuth 2.0 Bearer Token | Docs | Twitter Developer*. 2020. Disponível em: <https://developer.twitter.com/en/docs/authentication/oauth-2-0>. Citado na página 27.

WANG, A. H. DON ' T FOLLOW ME Spam Detection in Twitter. *2010 International Conference on Security and Cryptography (SECRYPT)*, IEEE, p. 1–10, 2010. Citado na página 24.

WANGENHEIM, A. V.; GRESSE, C.; WANGENHEIM, V. *Raciocínio Baseado em Casos-2ª ed. Revisada e Atualizada Evaluation of Educational Games for Computing Education View project*. [S.l.], 2013. Disponível em: <https://www.researchgate.net/publication/262374659>. Citado na página 18.

WEISS, G. M. *weather.arff*. 2010. Disponível em: <https://storm.cis.fordham.edu/~gweiss/data-mining/weka-data/weather.arff>. Citado na página 23.

ZHANG, H. *The Optimality of Naive Bayes*. [S.l.]. Disponível em: <www.aaai.org>. Citado na página 18.