UNIVERSIDADE FEDERAL DE ALAGOAS-UFAL
CAMPUS ARAPIRACA
CIÊNCIA DA COMPUTAÇÃO

ÉRICK RITIR OLIVEIRA

RECONHECIMENTO DE BOTS NO TWITTER: UMA ABORDAGEM UTILIZANDO APRENDIZAGEM DE MÁQUINA

ARAPIRACA
2019

Érick Ritir Oliveira

Reconhecimento de Bots no Twitter: Uma Abordagem Utilizando Aprendizagem de Máquina

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação da Universidade Federal de Alagoas - UFAL, Campus Arapiraca.

Orientador: Prof. Dr. Rodolfo Carneiro Cavalcante

Arapiraca
2019

Érick Ritir Oliveira

Reconhecimento de Bots no Twitter: Uma Abordagem Utilizando Aprendizagem de Máquina

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação da Universidade Federal de Alagoas - UFAL, Campus de Arapiraca.

Data de Aprovação: 30/08/2019

Banca Examinadora

Prof. Dr. Rodolfo Carneiro Cavalcante
Universidade Federal de Alagoas
Campus Arapiraca
Orientador

Prof. Dr. Tácito Trindade de Araújo Tiburtino Neves
Universidade Federal de Alagoas
Campus Arapiraca
Examinador

Prof. Dr. Elthon Allex da Silva Oliveira
Universidade Federal de Alagoas
Campus Arapiraca
Examinador

Aos meus pais.

AGRADECIMENTOS

Ao meu orientador, professor Rodolfo: obrigado pelo apoio e pela orientação neste trabalho. Tenho certeza que consegui finalizar este trabalho graças ao seu apoio e paciência.

Aos meus pais: obrigado por sempre terem me apoiado e cuidado de mim, pelos sacrifícios feitos para que eu e meus irmãos tivéssemos a vida que temos hoje e por sempre terem me incentivado a estudar.

Aos meus irmãos: obrigado por todo apoio e carinho que vocês me deram desde sempre e por sempre estarem ao meu lado.

Aos meus outros familiares: obrigado pelo suporte e pelo carinho.

Aos meus amigos: obrigado por sempre estarem ao meu lado e terem me ajudado nos momentos mais difíceis da graduação.

Aos meus professores da UFAL: obrigado pelo esforço e dedicação, que foram fundamentais para minha formação acadêmica.

We are the choices we make. And have to make.
We aren't anything else.

Patrick Ness

RESUMO

As redes sociais são ambientes na internet onde é possível compartilhar e consumir conteúdos e notícias em tempo real. Nesse contexto, contas falsas, ou bots, são contas que poluem esses espaços com objetivos maliciosos, como fazer spam ou promover notícias falsas. Logo, faz-se necessário detectá-las e removê-las das redes sociais. Este trabalho tem como objetivo classificar, utilizando três tipos de atributos (de comportamento, de perfil e de texto), contas do Twitter em conta falsa ou legítima. Foram utilizados três classificadores (Rede Neural, Naive Bayes e Random Forest) a partir de uma base de dados previamente classificada. A base de dados foi construída a partir de dois data sets encontrados na literatura. Para avaliar as classificações, foram utilizadas as métricas acurácia, recall, precisão e F1 Score. A partir dos experimentos, notou-se que os métodos propostos apresentaram resultados interessantes, e que, analisando os dados extraídos das contas, notou-se que há diferenças no comportamento e no modo como contas falsas utilizam a plataforma do Twitter em relação a usuários légitimos.

Palavras-chave: Aprendizagem de máquina. Twitter. Mineração de texto. Bots. Classificação.

ABSTRACT

Social networks are environments on the Internet where it is possible to share and consume content and news in real time. In this context, fake accounts, or bots, are accounts that pollute these environments with malicious purposes, such as spamming or promoting fake news. Therefore, it is necessary to detect and remove them from social networks.This study aims to classify, using three types of features (behavior, profile and text), Twitter accounts in false or legitimate account. Three classifiers (Artificial Neural Network, Naive Bayes and Random Forest) were used from a previously classified database. The database was built from two data sets found in the literature. To evaluate the classification, we used the accuracy, recall, precision and F1 score metrics. From the experiments, it was noted that the proposed methods presented interesting results, and that, analyzing the data extracted from the accounts, it was noted that there are differences in the behavior and the way fake accounts use the Twitter platform in relation to legitimate users.

Keywords: Machine learning. Twitter. Text mining. Bots. Classification.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

MLP	Multilayer Perceptron
NB	Naive Bayes
RF	Random Forest
API	Application Programming Interface
LSTM	Long Short-Term Memory
RNN	Recurrent neural network

SUMÁRIO

1 INTRODUÇÃO

Nas últimas décadas, as redes sociais emergiram como uma plataforma de comunicação em tempo real em um mundo cada vez mais conectado. É previsto que o número de pessoas utilizando redes sociais será de 3.1 bihões em 2021[1]. Redes sociais, como o *Twitter* e *Facebook*, permitem que qualquer pessoa crie uma conta, produza e compartilhe os conteúdos que elas quiserem. O sistema de amigos e seguir/ser seguido permite que várias pessoas se conectem entre elas, resultando num complexo grafo de conectividade. Contas conectadas são capazes de consumir, comentar e compartilhar informações produzidas por outras contas em tempo real. A liberdade de opinião e o alcance dessas plataformas fazem delas uma importante ferramenta capaz de influenciar a sociedade em diversos aspectos, como política, cultura, comércio, entre outros.

No entanto, essa mídia de comunicação de longo alcance é um ambiente perfeito para usuários maliciosos que desejam disseminar notícias falsas com o intuito de enganar ou manipular a opinião de outros usuários. É nesse contexto que contas automatizadas, conhecidas como *bots* ou contas falsas, são utilizadas para diversos fins maliciosos. Um dos problemas mais evidentes enfrentados pelas mídias sociais são os *bots* sociais, os quais produzem automaticamente conteúdo e interagem com humanos, imitando algum comportamento em algum sentido para enganar os outros [Ferrara et al. 2016], fazem spam [Zhang et al. 2012], disseminam notícias falsas [Howard et al. 2017], ou simplesmente seguem alguma outra conta a fim de aumentar o seu número de seguidores [Cresci et al. 2015]. Esse problema impõe um grande desafio, uma vez que *bots* podem facilmente ser confundidos com humanos [Everett et al. 2016].

Como contas falsas possuem diferentes intenções e objetivos, diversas tentativas de detectar contas falsas vem sido propostas na literatura, como analisar os dados de perfil dos usuários, os textos dos *tweets*, o modo e a frequência de postagem ou os relacionamentos da conta com seus seguidores e quem ele segue. A maior parte dessas abordagens geralmente utiliza dados coletados via API do *Twitter* [Makice 2009] para extrair informações das contas com o obetivo de treinar um algoritmo de Aprendizagem de Máquina supervisionada a fim de identificar se uma conta é falsa ou legítima.

O objetivo deste trabalho é tentar classificar contas do *Twitter* analisando três tipos de atributos investigados na literatura: atributos de perfil, atributos de comportamento e atributos textuais. Os dados dos atributos coletados foram analisados e posteriormente utilizados em

[1] https://www.statista.com/statistics/278414/number-of-worldwide-social-network-users/

algoritmos de Aprendizagem de Máquina para uma classificação em conta falsa ou legítima, sendo eles: *Naive Bayes*, MLP e *Random Forest*.

Os conjuntos de dados (*data sets*) utilizados neste trabalho foram formados a partir de dois *data sets* encontrados na literatura. A partir deles, foram coletados via API do *Twitter* os atributos das respectivas contas a fim de formar os *data sets* para este estudo. Para realizar os experimentos, os dados foram pré processados, de forma que para os dados textuais houve um processamento textual e para os dados numéricos houve um processo de normalização dos dados. Posteriormente, os dados foram classificados pelos algoritmos de Aprendizagem de Máquina citados anteriormente e as métricas utilizadas para avaliar essas classificações foram acurácia, *F1 Score*, precisão e *recall*.

A seguir, é descrito como está organizado o restante deste trabalho. Capítulo 2 descreve os algoritmos e os métodos utilizados neste trabalho bem como seus conceitos e definições. Capítulo 3 descreve alguns trabalhos propostos na literatura. Capítulo 4 apresenta os métodos deste trabalho. Capítulo 5 descreve os *data sets* utilizados, o processo de extração dos atributos das contas, como os algoritmos de classificação foram configurados e os resultados obtidos. Capítulo 6 apresenta conclusões acerca dos experimentos realizados e possíveis trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Para realizar os experimentos guiados neste trabalho, foram utilizados alguns algoritmos e técnicas. A seguir, são descritos os principais conceitos, técnicas e algoritmos utilizados neste trabalho.

2.1 APRENDIZAGEM DE MÁQUINA

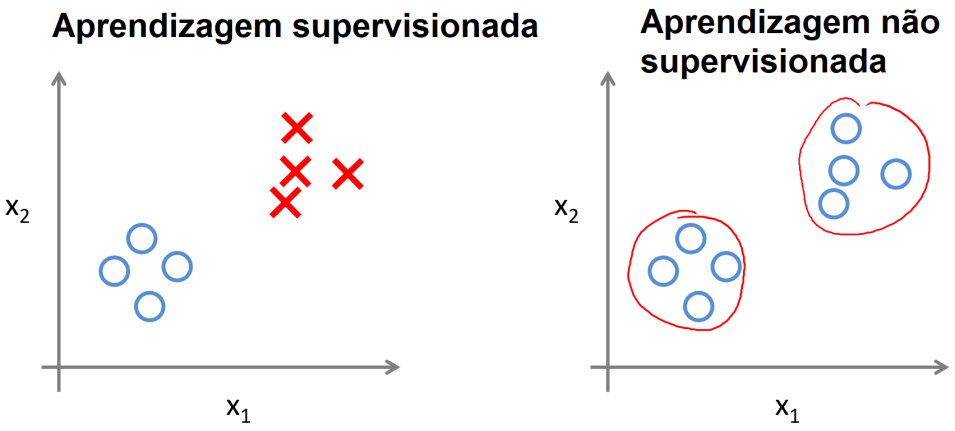
Aprendizagem de Máquina é uma área da Inteligência Artificial a qual permite um computador aprender automaticamente sem intervenção humana a partir da observação de dados. O aprendizado se dá de forma indutiva, onde é feita uma hipótese geral a partir de observações particulares. Por exemplo: É jogado uma pedra no rio e ela afunda. Novamente, é jogado uma pedra no rio e ela afunda. Dessa forma, é induzido que toda vez que será jogada uma pedra no rio ela irá afundar. Algoritmos de Aprendizagem de Máquina são categorizados como supervisionados ou não supervisionados.

No Aprendizado supervisionado, o algoritmo de aprendizado vai receber um conjunto de exemplos (instâncias) previamente classificados (onde sua classe é conhecida) descritos por um vetor de atributos e pelo rótulo da classe associada. O objetivo do algoritmo é que, dado um novo conjunto de dados como entrada e que não é conhecido a sua classe, ele consiga determinar corretamente a classse de cada instância. Quando o valores das classes são discretos, esse problema é considerado um problema de classificação, e quando os valores das classes são contínuos, é um problema de regressão. Usando este trabalho como exemplo, as classes das instâncias (contas do *Twitter*) são classificadas como *bot* ou não *bot*, ou seja, é um problema de classificação.

No Aprendizado não supervisionado, os dados de entrada não possuem suas classes conhecidas, logo, cabe ao algoritmo entender a estrutura e como os dados estão organizados a fim de encontrar padrões em que possam ser agrupados de alguma forma. Esse agrupamento é chamado de *clustering* e a ideia é que as instâncias pertencentes a um *cluster* possuam características similares. Após o agrupamento, é necessário entender o critério que levou a agrupar cada grupo a fim de entender o que cada grupo representa.

A Figura 1 mostra um exemplo de representação dos dois tipos de aprendizagem. Na esquerda, o Aprendizado supervisionado, onde há instâncias rotuladas com dois tipos de classe. Na direita, o Aprendizado não supervisionado, onde encontra-se instâncias não rotuladas e onde é necessário agrupá-las por algum ou alguns critérios.

Figura 1 – Tipos de aprendizagem.

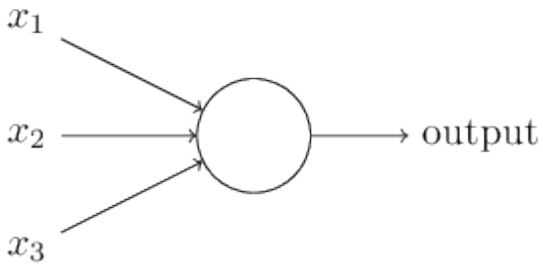


Fonte: Autor (2019).

2.1.1 *Multilayer Perceptron*

Redes Neurais Artificiais (RNAs) são, basicamente, modelos de computação inspirados no funcionamneto do cérebro humano. Para entender o que é uma Rede Neural, antes é necessário entender a ideia que deu início ao modelo de Redes neurais, que é o *Perceptron*. A ideia de *Perceptron* foi desenvolvida nas décadas de 1950 e 1960 por Frank Rosenblatt [Rosenblatt 1958] cujo objetivo era simular o comportamento de um neurônio. Um *Perceptron* recebe um conjunto de entradas $(X_1, X_2, X_3, ..., X_n)$ e produz uma saída binária.

Figura 2 – Perceptron.



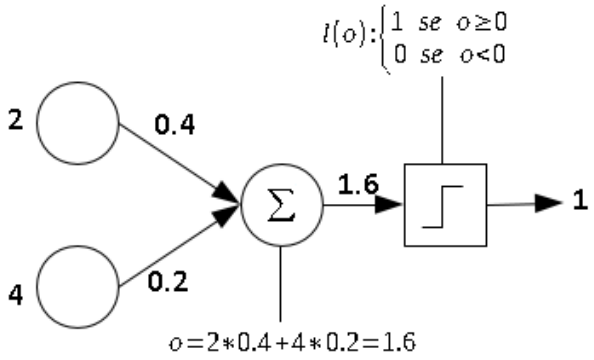
Fonte: Disponível em: <http://neuralnetworksanddeeplearning.com/chap1.html>. Acesso em: 10 jul. 2019.

Na Figura 2, é apresentado um *Perceptron* com três entradas, *x1*, *x2* e *x3*, gerando posteriormente uma saída. Essa saída é realizada por uma função de ativação. A Função de ativação processa o conjunto de entradas recebidas e o transforma em um estado de ativação. Dentre as Funções de ativação, pode-se citar a Função Linear, Função Limiar e Função Sigmoide Logística. Rosenblatt propôs uma regra para calcular a saída. Ele introduziu os pesos $(w_1, w_2, w_3, ..., w_n)$, que são números reais que expressam a importância de cada dado de entrada para a saída. Desse

modo, a saída do perceptron é baseada na combinação linear dos dados de entrada ponderados pelos pesos.

Assim, um *Perceptron* toma decisões colocando pesos em suas evidências. A Figura 3 mostra um exemplo de perceptron. Neste exemplo, os dados de entrada são o 2 e o 4 e eles então são multiplicados pelos seus respectivos pesos e depois somados. Desse modo, a saída dessa operação é avaliada por uma função de perda, a qual resulta em 1 se o resultado da soma for igual ou maior a 0, e 0 se o resultado for menor que 0. Variando esse valor limite resulta em diferentes modelos.

Figura 3 – Exemplo de perceptron

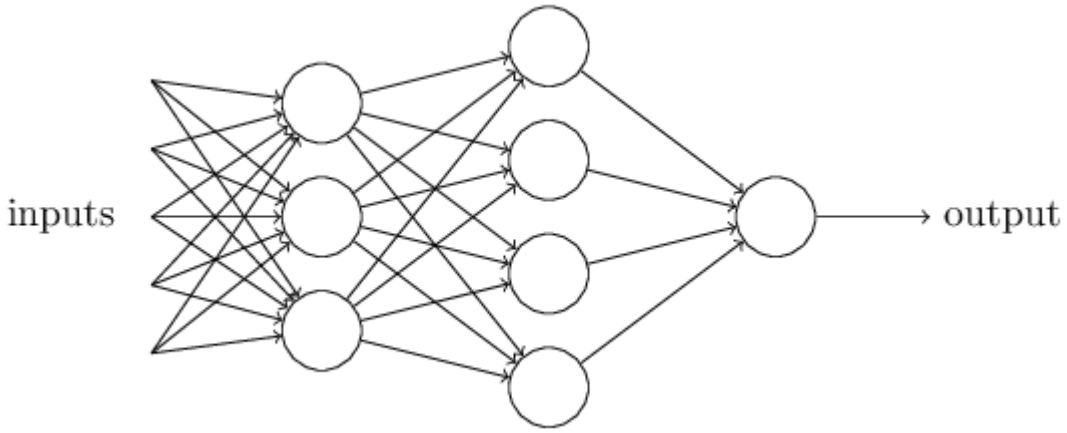


Fonte: Disponível em: <https://juliocprocha.blog/2017/07/27/perceptron-para-classificacao-passo-a-passo/>. Acesso em: 10 jul. 2019.

Explicado então o que é um *Perceptron*, uma MLP (*Multilayer Perceptron*) é uma Rede Neural formada por vários grupos de neurônios, organizados em várias camadas. Cada neurônio na primeira camada envia sinais para todos os neurônios na segunda camada, e assim por diante. Um MLP contém uma camada de entrada, pelo menos uma camada oculta e uma camada de saída. A Figura 4 mostra uma representação de uma MLP.

Uma MLP se baseia na ideia de sinapses, onde os sinais se propagam atráves dos neurônios. Dessa forma, existe categorias de Redes Neurais onde as quais os sinais se propagam de uma maneira diferente. São elas: Redes *Feedforward*, Redes recorrentes e redes que utilizam o método *backpropagation*. As Redes *Feedforward* é o tipo mais comum e o sinais nela seguem em uma única direção. Nas Redes recorrentes, há conexões ligando a saída da rede e a sua entrada. Dessa forma, elas podem lembrar entradas passadas e são úteis para lidar com processamento de informações sequenciais. O *backpropagation* é um algoritmo de treinamento o qual a rede é

Figura 4 – Exemplo de MLP



Fonte: Disponível em: <http://neuralnetworksanddeeplearning.com/chap1.html>. Acesso em: 10 jul. 2019.

treinada com pares entrada-saída. Para cada entrada de treinamento é associado uma saída e o treinamento é feito percorrendo a rede no sentido para frente (*forward*) e para trás (*backward*). Primeiramente o algoritmo inicia todas as conexões com valores aleatórios. Para cada par de treinamento, é calculado os sinais de saída em cada neurônio, verificado o erro, e depois os pesos são atualizados em cada neurônio de cada camada. Esse processo é repetido até que o erro seja menor que um critério definido. O algoritmo de treinamento *backpropagation* é o mais utilizados em MLPs, contudo possui algumas desvantagens, como ser considerado lento ao tratar problemas complexos e que muitas etapas de treinamento podem levar o modelo a memorizar os padrões de treinamento (o chamado *overfitting*).

2.1.2 *Random Forest*

Random Forest [Breiman 2001] é um algoritmo de aprendizagem supervisionada formado por um conjunto de Árvores de Decisão (*Decision Trees*). Logo, para entender o algoritmo *Random Forest* é necessário entender o que são as Árvores de Decisão.

Árvores de Decisão (*Decision Trees*) é um algoritmo de Aprendizagem de Máquina supervisionado. O algoritmo tenta resolver um dado problema utilizando a representação de uma árvore, sendo essa dividida em raiz, nós internos e folhas. A árvore é construída da raiz para as folhas, ou seja, num método *top-down*. Cada nó interno da árvore corresponde a um teste sobre os valores de um atributo e cada folha (nó terminal) corresponde a uma classe. Para realizar a classificação, a árvore percorre da raiz até as folhas, sendo a passagem de um nó para outro uma etapa de classificação. Basicamente, a construção de uma árvore de decisão se dá da seguinte forma:

1. Escolher um atributo

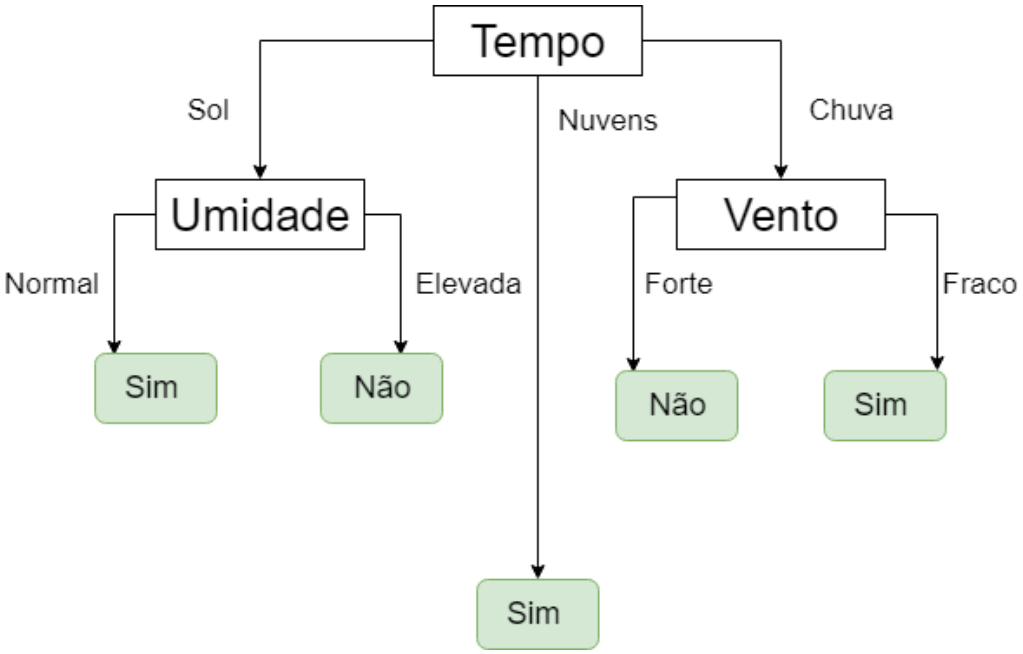
2. Expandir a árvore alocando um ramo pra cada valor do atributo

3. Passar os exemplos para as folhas

4. Para cada folha: Se todos os exemplos são da mesma classe, associar essa classe à respectiva folha. Do contrário, repetir os passos de 1 a 4.

Nas árvores de decisão, para prever a classe de um objeto, começamos pela raiz da árvore. A seguir, comparamos os valores do atributo da raiz com os valores do atributo do objeto. Com base na comparação, seguimos para o galho correspondente àquele valor e pulamos para o próximo nó. Continuamos comparando os valores dos atributos do objeto com os valores dos atributos dos nós internos até que se chegue nos nós de folha. Assim, a folha a qual o objeto se encontra no final, vai predizer a classe do objeto. A Figura 5 mostra um exemplo de representação do modelo para um problema de avaliar se o dia está favorável para um jogo de golf ou não.

Figura 5 – Exemplo de representação de uma Árvore de Decisão.



Fonte: Autor (2019).

Como a ordem dos atributos é importante, um dos objetivos na implementação de uma Árvore de Decisão é escolher qual o atributo que ficará na raiz e em quais nós de cada nível ficarão os atributos restantes. Para identificar e elencar os atributos, o critério é baseado em Ganho de informação e Índice de Gini.

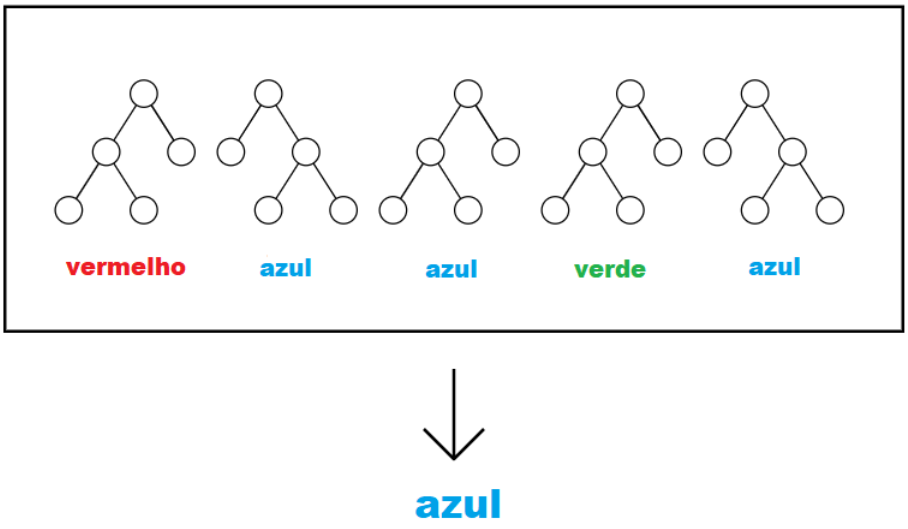
Um dos problemas do algoritmo é que a árvore pode se ajustar perfeitamente aos dados de treino, crescendo de um forma que consiga classificar corretamente todo os dados de treino (*overfitting*). Uma solução para isso é o método da "poda", onde ao percorrer a árvore em

profundidade, para cada nó de decisão é calculado seu erro e a soma dos erros dos nós "filhos". Caso o erro do nó seja menor ou igual a essa soma, o nó é transformado em folha.

Desse modo, o algoritmo de Árvores de Decisão é fácil de se entender e de ser interpretado visualmente, pois se baseia num esquema similar ao qual nós humanos realizamos ao chegar a uma decisão. Como desvantagens, muitas vezes ocorre o *overfitting* e certas classificações necessitam de uma abordagem mais complexa quando possuem várias classes. Desse modo, é proposto o *Random Forest*, sendo uma combinação de várias árvores de decisão. Na etapa de treinamento, cada árvore aprende a partir de uma amostra dos dados, o que resulta em vários modelos treinados diferentemente.

Para o *Random Forest* realizar uma classificação, cada árvore individual apresenta uma previsão de classe e a classe com mais votos torna-se a previsão final do modelo. *Random Forest* também pode ser considerado um algoritmo de *ensemble* (mistura de classificadores), visto que cada árvore (*Decision Tree*) da floresta (*forest*) dá seu voto e o voto majoritário é escolhido como voto final. A figura 6 mostra um exemplo de *Random Forest*, onde o problema é predizer uma cor e cada árvore de decisão realiza sua classificação. *Random Forest*, apesar de exigir um maior custo computacional quando implementado muitas árvores, é um algortimo robusto e eficiente.

Figura 6 – Exemplo de Random Forest



Fonte: Disponível em: <https://victorzhou.com/blog/intro-to-random-forests/>. Acesso em: 12 jul. 2019.

2.1.3 *Naive Bayes*

Naive Bayes [Rish et al. 2001] é um algoritmo de classificação baseado no Teorema de *Bayes*, sendo um dos algoritmos de aprendizagem supervisionada mais simples.

O *Naive Bayes* assume que os atributos analisados são condicionalmente independentes entre si. Por exemplo, um solicitante de empréstimo é aceitável ou não dependendo de sua renda, histórico de empréstimos e transações anteriores, idade e localização. Mesmo que esses atributos sejam interdependentes, eles serão considerados independentes. Devido a essa lógica, o algoritmo é considerado *naive* (ingênuo). O Teorema nos permite encontrar a probabilidade de um evento ocorrer dado a probabilidade de que outro evento já ocorreu. A equação 1 apresenta a fórmula do Teorema de *Bayes*.

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)} \tag{1}$$

Nessa fórmula, é calculado a probabilidade de um evento A acontecer dado que um evento B já aconteceu. P(B|A) é a probilidade de acontecer um evento B dado que A já aconteceu, P(A) é a probabilidade de A ocorrer e P(B) é a probabilidade de B ocorrer.

Naive Bayes, apesar de ser um algoritmo fácil e rápido, supõe que os dados são independentes, o que é bastante raro acontecer num problema real.

2.1.4 LSTM

As Redes de Memória de Longo Prazo (*Long Short-Term Memory*) foram propostas por [Hochreiter e Schmidhuber 1997] e são um tipo mais sofisticado de RNN (Redes Neurais Recorrentes). A vantagem das LSTMs sobre as RNNs é de que elas retêm informações por longos períodos de tempo, permitindo que informações importantes aprendidas no início de uma sequência de dados tenham um impacto maior nas decisões do modelo feitas no final da sequência. Portanto, LSTMs se adequam a problemas onde há dados sequenciais, como textos e atividades de processamento de linguagem natural. [Kudugunta e Ferrara 2018] utilizou uma rede LSTM num modelo de rede neural para uma tarefa de classificação, alcançando uma acurácia de 95%.

2.2 PROCESSAMENTO DE TEXTO

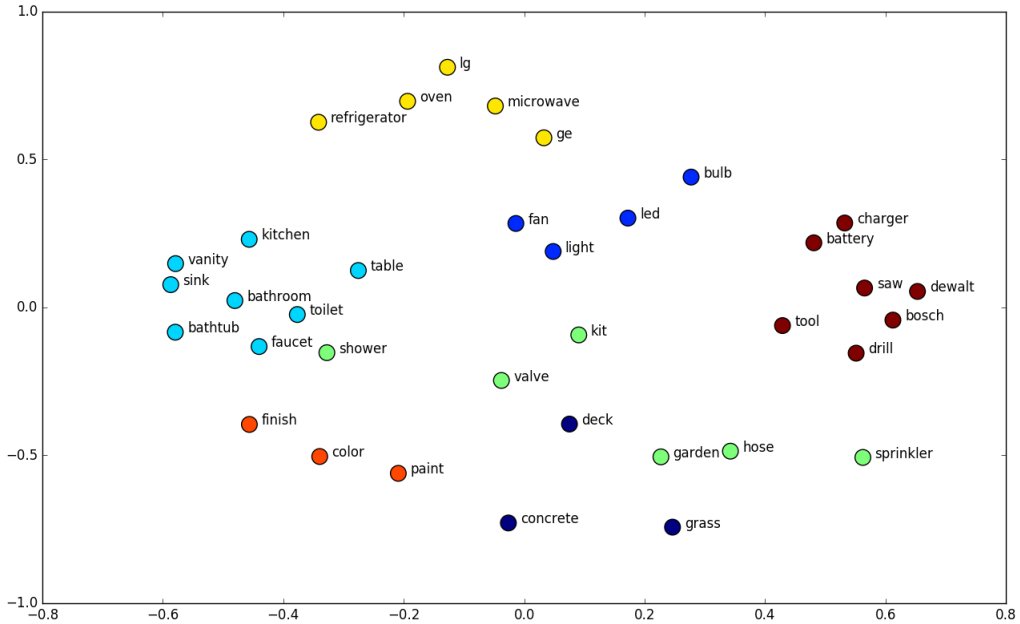
Neste experimento, para analisar os dados textuais produzidos pelas contas coletadas, foi feito um processamento de texto com os dados a fim de serem utilizados por uma Rede Neural. A etapa de processamento de texto é necessária pois palavras representam atributos categóricos. Dessa forma, é necessário mapear dados textuais para vetores numéricos, uma vez

que Redes Neurais são projetadas para aprender a partir de dados numéricos. Posteriormente ao processamento de texto, foi utilizado neste trabalho a técnica de *Embedding*, descrita a seguir.

2.2.1 *Word Embeddings*

Word Embedding é uma técnica de representação de texto onde palavras que possuem significados semelhantes possuem representações similares. As palavras são representadas como vetores de valores reais em um espaço vetorial pré definido, chamado de *Embedding Space*. A Figura 7 mostra um exemplo desse espaço. Cada palavra é mapeada para um vetor e os valores vetoriais são aprendidos de uma forma que se assemelha a uma Rede Neural. As representações das palavras são aprendidas a partir do uso das palavras. Logo, isso permite que palavras utilizadas em contexto similares tenham representações similares, o que faz ser possível capturar o significado das palavras. Na Figura 7 é possível notar que palavras que utilizamos em contextos e frases similares estão próximas, por exemplo: carregador (*charger*) e bateria (*battery*), banheiro (*bathroom*) e toalete (*toilet*), cor (color) e pintura (*paint*), entre outros.

Figura 7 – Exemplo de um Embedding Space com duas dimensões.



Fonte: Disponível em: <https://www.shanelynn.ie/get-busy-with-word-embeddings-introduction/>. Acesso em: 13 jul. 2019.

3 TRABALHOS RELACIONADOS

Contas falsas, ou *bots*, são um grande problema para as redes sociais e foram criadas e utilizadas para diversos propósitos. Inicialmente, o objetivo era seguir outro perfil com o propósito de inflar seu número de seguidores com seguidores falsos [Cresci et al. 2015]. Naturalmente, essas contas falsas não realizam nenhuma atividade na rede social além de seguir outras contas. Outro objetivo dos *bots* é disseminar notícias falsas. *Bots* são usados para publicar e compartilhar alguma notícia [Howard et al. 2017] ou *spam* [Zhang et al. 2012] a fim de estender seu alcance na rede social. Esses *bots* normalmente apresentam atividades incomuns nas redes sociais quando comparados com usuários legítimos. Na literatura, vários trabalhos foram realizados para identificar *bots* baseados em seus objetivos e suas atividades. Seguindo a taxonomia proposta por [Adewole et al. 2017], este trabalho foi dividido em quatro categorias, sendo estas baseadas no tipo de dado em que cada abordagem utiliza para classificar contas: (i) dados de perfil do usuário, (ii) comportamento de postagem, (iii) dados textuais e (iv) baseado em grafos.

3.1 DADOS DE PERFIL DO USUÁRIO

Dados de perfil são as informações básicas atribuídas a uma conta quando criada no *Twitter*, como o nome de usuário, sua descrição, foto de perfil, imagem de fundo, sua localização onde foi criada e o momento em que foi criada. Abordagens baseadas nos dados de perfil são bastante úteis para detectar seguidores falsos ou outros tipos de *bots* que possuem pouca ou nenhuma atividade na plataforma. Em [Azab et al. 2016], os autores selecionaram 22 atributos binários de perfil para descrever as contas levando em consideração a presença ou ausência desses atributos. O objetivo dessa abordagem foi de achar o menor conjunto de atributos que fornece a melhor distinção entre contas reais e falsas. Nesse trabalho, o ganho de informação foi usado para atribuir pesos a cada atributo e, posteriormente, os atributos que possuíam mais de 50% de peso foram selecionados para compor o mínimo conjunto de atributos. Os atributos selecionados foram: (i) se a localização foi informada no momento em que a conta foi criada, (ii) se a conta possui pelo menos 30 seguidores, (iii) se a conta foi incluída como favorita por outro usuário, (iv) se a conta usou *hashtag* em pelo menos um tweet, (v) se a conta fez login no *Twitter* utilizando um iphone, entre outros. Cinco algoritmos de classificação foram utilizados para modelar essses dados: *Random Forest*, *Naive Bayes*, *Artificial Neural Network*, *Support Vector Machine* e *Decision Tree*. Os resultados mostraram que os métodos propostos alcançaram mais de 90% de acurácia na classificação.

A abordagem proposta por [Gurajala et al. 2016] considerou alguns atributos de perfil como: id, número de seguidores, número de pessoas seguidas, se a conta foi verificada, o momento de criação, descrição, localização, se a conta foi atualizada depois de criada, a foto de perfil e o *screen-name*. Inicialmente 62 milhões de contas foram coletadas e foi utilizado um algoritmo de reconhecimento de padrões em que consistia em agrupar contas que possuíam dados de perfil similares, a fim de obter grupos contendo pelo menos duas contas. Eles utilizaram entropia de *Shannon* no atributo *screen-name* com objetivo de agrupar perfis onde ele possivelmente foi gerado automaticamente (exemplo: "freefollow1", "freefollow3"). Adicionalmente, foi utilizado distribuição de tempo de atualização das contas para reduzir os falsos positivos no processo de *clustering*. Como resultado, foi obtido um conjunto de contas falsas que continha 8873 grupos com aproximadamente 56.000 contas.

Em [Chu et al. 2012], um processo de classificação baseado em componentes foi proposto, onde um dos componentes, chamado de propriedades da conta, era formado por atributos de perfil do usuário. Em [Kudugunta e Ferrara 2018], foram utilizados atributos de perfil chamados de metadados em nível de conta: número de status, número de seguidores e perfis seguidos, número de favoritos, se a conta foi verificada, foto de perfil padrão, habilitação da geolocalização, se utiliza imagem de fundo e se a conta é bloqueada. Um simples processo de classificação foi aplicado utilizando alguns algoritmos como: *Random Forest*, *Artificial Neural Network*, *AdaBoost Classifier*, *SGD Classifier* e *Logistic Regression*. Resultados alcaçaram 98.39% de acurácia e, adicionando técnicas de *synthetic oversampling*, a acurácia atingiu 99.81%.

3.2 COMPORTAMENTO DE POSTAGEM

O comportamento de um usuário numa rede social pode ser descrito como qualquer tipo de atividade que o usuário realiza na plataforma. No contexto de classificação de contas em reais ou falsas, alguns autores acreditam que as atividades dos usuários são aquelas que produzem conteúdo, como *tweets*, *retweets* e *replies*. A principal hipótese é a de que usuários legítimos não apresentam um comportamento regular. Contas falsas, por outro lado, podem apresentar um padrão pré definido em seu comportamento. Logo, o objetivo principal é encontrar características no comportamento de *bots* e usuários as quais possam distingui-los. Atributos de comportamento de postagem podem ser bastante úteis para identificar *spammers* ou contas que espalham notícias falsas.

Na literatura alguns trabalhos tentam detectar *bots* oberservando suas atividades de postagem. [Gurajala et al. 2016] propôs uma abordagem que utiliza tempo de atualização para

agrupar contas que possuem padrões similares. Esse método analisa a distribuição dos dados relacionados ao momento de postagem durante o dia e a frequência de postagem nos dias da semana. Em [Cai et al. 2017], os autores propuseram um modelo que utiliza a composição de atributos de comportamento e de conteúdo, como o tempo entre as postagens e seu tipo (*tweet* ou *retweet*), utilizando algoritmos de aprendizagem profunda como CNN e LSTM para realizar a classificação. O melhor resultado foi 87.32% de *F1 score*. Em [Varol et al. 2017], os autores utilizaram atributos temporais para descrever os dados de comportamento como um subconjunto de um número mais robusto de atributos de diferentes origens. O melhor resultado atingido foi de 95% na métrica AUC utulizando o algoritmo *Random Forest*.

3.3 DADOS TEXTUAIS

Métodos baseados em dados textuais são aqueles onde é aplicado alguma análise textual nos *tweets* com o objetivo de identificar padrões que distinguem *bots* de usuários legítimos. Esses métodos também são muito úteis para reconhecer *spammers*. Em [Clark et al. 2016], os autores apresentam um esquema de classificação o qual utiliza apenas processamento de linguagem natural. Para conseguir isso, os autores desenvolveram um algoritmo de classificação que opera utilizando três atributos linguísticos do texto: (i) média de URL por *tweet*, (ii) média de dissimilaridade entre pares de *tweets* e (iii) decadência de introdução de palavras. Em [Kudugunta e Ferrara 2018], alguns atributos em nível de tweet como o número de *retweets*, número de *hashtags*, número de URLs, número de menções a outros usuários, entre outros, foram investigados. Eles aplicaram uma rede LSTM a fim de classificar contas utilizando apenas os textos dos *tweets*. Resultados mostraram uma acurácia de 95%. Adicionalmente, foi aplicado LSTM na combinação dos textos dos *tweets* com seus metadados, onde posteriormente foi alcançado uma acruácia de 96%. Outros atributos de texto foram investigados em [Inuwa-Dutse et al. 2018], os quais são baseados em três conceitos: (i) distribuição de co-ocorrência de palavras chaves, onde o objetivo é identificar *tweets* de *spammers* utilizando palavras amplamente usadas por eles, como "*free*"e "*follow*", (ii) riqueza e densidade léxica, onde é extraído a riqueza do vocabulário dos textos presentes nos *tweets*, (iii) menções a usuários, o qual analisa o número de menções presentes no *tweet*, considerando que perfis falsos tendem a ter uma grande quantidade pois mencionam outros usuários a fim de aumentar os acessos a seus *tweets*.

3.4 BASEADO EM GRAFOS

Alguns trabalhos na literatura focam em detectar contas falsas analisando os relacionamentos entre contas. Esses relacionamentos podem ser representados como uma estrutura de grafo, o qual é chamado de grafo social e onde os vértices são os usuários e as arestas são as relações, as quais são: seguindo, ser seguido, *like*, *retweet*, entre outros. Relacionamentos atípicos podem indicar atividades maliciosas na plataforma, como "*fake following*", onde há contas falsas seguindo algum perfil com o objetivo apenas de inflar o número de seguidores desse perfil. [Mehrotra et al. 2016] evitou usar métodos convencionais de perfil e utilizou atributos focados em medidas de centralidade do grafo social. [Xue et al. 2013] propôs um método para identificar contas falsas analisando o relacionamento entre os convites de amizade entre usuários. Nesse metódo, é criado um grafo de convites de amizade onde é presumido que contas falsas possuem mais convites rejeitados do que usuários legítimos. Em [Wang 2010], para realizar a classificação, foi combinado dados textuais com atributos baseados em grafos, como número de seguidos/seguidores e a reputação de um perfil. Nesse caso, a reputação de um perfil é a relação entre o seu número de seguidos e seguidores. A hipótese é de que contas falsas seguem muitos usuários porém possuem poucos seguidores.

3.5 OUTROS MÉTODOS

Alguns métodos propostos na literatura são considerados híbridos. [Chu et al. 2012] tenta classificar perfis em humanos, *bots* ou *cyborgs* usando três componentes: (i) medida de entropia, o qual tenta detectar periodicidade ou regularidade no tempo de postagem com o objetivo de identificar um comportamento automatizado, (ii) detecção de *spam*, o qual utiliza uma variante do modelo *Bayesiano* para detectar padrões nos textos anteriormente classificados como *spam* e (iii) propriedades das contas, que são os atributos de perfil. Em [Lee et al. 2011], os atributos investigados foram agrupados em: (i) demografia do usuário, que são os atributos de informação descritiva sobre o usuário (*scree-name*, descrição etc), (ii) rede de amizades do usuário, que são os atributos relacionados aos relacionamentos do perfil (iii) conteúdo do usuário, que são as informações de conteúdo dos *tweets* e (iv) histórico do usuário, que são os dados temporais do que foi postado pelo usuário. Trinta algoritmos de Aprendizagem de Máquina foram utilizados para classificar as contas em usuários ou *bots*. Os resultados mostraram que o *Random Forest* apresentou a melhor eficácia, com 98.42% de acurácia.

4 MÉTODOS

O objetivo deste estudo é utilizar três métodos distintos, analisando diferente atributos, a fim de classificar uma conta do *Twitter* como falsa ou legítima. Tais métodos utilizam Algoritmos de Aprendizagem de máquina para classificar contas obtidas a partir de dois *data sets* disponibilizados na literatura: Social Spambots e Honeypot. No primeiro método, analisa-se os atributos de comportamento de um usuário na plataforma a fim de encontrar algum padrão de postagem. No segundo, utilizando-se dos atributos de perfil, é analisado as informações de perfil de um usuário, como o número de pessoas que ele segue e por quantas ele é seguido. Além disso, um terceiro método é proposto utilizando-se do conteúdo dos *tweets* postados por um usuário, com o objetivo de identificar padrões nos textos que possam diferenciar contas automatizadas de contas autênticas.

4.1 ANÁLISE DE COMPORTAMENTO

Partindo do presuposto de que *bots* possuem uma atividade robótica, talvez seja possível extrair algum padrão automático a partir de seu comportamento. Usuários legítimos, por outro lado, geralmente se comportam de uma maneira irregular e desordenada. A fim de validar essa hipótese, foi selecionado um conjunto de atributos que descrevem as atividades de uma conta no *Twitter*, como o tempo de intervalos entre *tweets* e o padrão de postagem nos dias da semana.

4.1.1 Atributos

Neste método, foram selecionados os seguintes atributos que descreveriam um comportamento automático: (i) o intervalo entre os *tweets* subsequentes, (ii) a frequência de postagem nos dias da semana e (iii) o desvio padrão do número de postagens em cada dia da semana. Considerando esses atributos, cada conta é então representada por um vetor de 10 atributos numéricos representados na Tabela 1.

Como já discutido em estudos anteriores [Chu et al. 2012], a frequência de postagem de usuário no *Twitter* é uma das características que diferem uma conta falsa de uma real. Usuários legítimos tendem a se comportar de uma maneira irregular, enquanto contas falsas tendem a se comportar em períodos regulares. Neste trabalho foi considerado a média dos intervalos entre os *tweets* (em segundos). Para uma sequência de n *tweets* $(X_1, X_2, X_3, ..., X_n)$ coletados de uma conta, a média dos intervalos de postagens é extraída conforme descrito na Equação 2, onde T_{X_i} é um momento de postagem do *tweet* X_i. Também foi considerado o desvio padrão dos

Tabela 1 – Atributos de comportamento.

Atributo	Descrição
mean_update_time	Média dos intervalos entre *tweets* (segundos.)
std_update_time	Desvio padrão dos intervalos entre *tweets*
monday_rf	Frequência relativa das postagens às Segundas
tuesday_rf	Frequência relativa das postagens às Terças
wednesday_rf	Frequência relativa das postagens às Quartas
thursday_rf	Frequência relativa das postagens às Quintas
friday_rf	Frequência relativa das postagens às Sextas
sartuday_rf	Frequência relativa das postagens aos Sábados
sunday_rf	Frequência relativa das postagens aos Domingos
std_day_week	Desvio padrão das postagens ao longo dos dias da semana

Fonte: Autor (2019).

intervalos entre postagens subsquentes. A hipótese é de que um grande desvio padrão indica uma grande aleatoriedade no comportamento de postagem, o que pode ser uma caracterísitica de uma conta legítima.

$$\overline{I} = \frac{1}{n-1}\sum_{i=1}^{n-1}(T_{X_{i+1}} - T_{X_i}) \tag{2}$$

A fim de reconhecer um comportamento automatizado, foi coletado a frequência com que o usuário posta em cada dia da semana em relação ao número total de *tweets* e seu respectivo desvio padrão. Logo, para cada conta foi extraído a frequência $Fday$ para cada dia da semana, como mostrado na Equação 3, onde n é o número de *tweets* postados pelo usuário e P_{day} é o número de *tweets* postados em um respectivo dia. Também foi extraído o desvio padrão do número absoluto de *tweets* postados em cada dia da semana, como descrito na Equação 4, onde $\overline{P}$ é a média de *tweets* postados em cada dia da semana.

$$F_{day} = \frac{P_{day}}{n} \tag{3}$$

$$S_w = \sqrt{\frac{1}{6}\sum_{i=1}^{7}(\overline{P} - P_{day})^2} \tag{4}$$

4.1.2 Classificação

Nesse método, para realizar a classificação de contas como falsas ou legítimas, foi utilizado três algoritmos de aprendizagem de máquina: *Naive Bayes*, *Multilayer Perceptron* e *Random Forest*.

4.2 ANÁLISE DE PERFIL

Como citado anteriormente na Seção 3, alguns autores utilizam os dados de perfil como atributos a fim de distinguir usuários falsos de legítimos. Os dados de perfil são as informações básicas atribuídas a uma conta no momento em que é criada na rede social e também diz respeito a como o usuário utiliza a plataforma do *Twitter*. Com isso, esses atributos foram coletados das respectivas contas e compuseram o *data set* utilizado neste método.

4.2.1 Atributos

Para compor essa segundo experimento de classificação, foram escolhidos os seguintes atributos: (i) número de seguidores, (ii) número de perfis seguidos pela conta e (iii) se a conta ativou a geolocalização. A hipótese é a de que perfis falsos possuem poucos seguidores mas seguem muitos perfis [Wang 2010], assim como tendem a possuir a geolocalização desativada [Azab et al. 2016].

4.2.2 Classificação

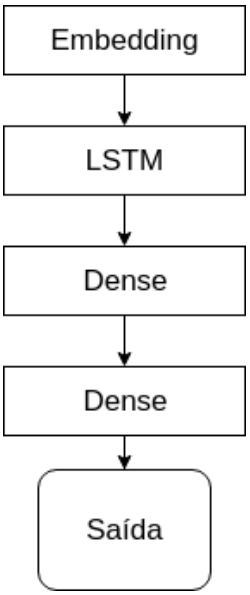
Para realizar a classificação, foram utilizados os mesmos algoritmos da análise de comportamento, sendo eles: *Naive Bayes*, *Multilayer Perceptron* e *Random Forest*.

4.3 ANÁLISE TEXTUAL

Além de analisar contas no *Twitter* por meio de seu comportamento e informações de perfil, também é possível extrair informações a partir do conteúdo que o usuário produz, ou seja, o que está escrito em seus *tweets*. Analisar computacionalmente a linguagem natural é uma tarefa complexa, principalmente muma plataforma onde a linguagem é flexível e onde há espaço para metáforas, ironia etc. A Mineração de Texto é o campo da computação que possui o objetivo de extrair informações a partir de dados textuais utilizando técnicas de processamento de linguagem natural. Podemos, por exemplo, usar técnicas de mineração de texto para classificar textos como *spam* ou não *spam*, analisar sentimentos de frases etc.

Neste trabalho, foi utilizado a bibilioteca para Python *Keras* [2]. *Keras* permite implementar um modelo de Rede Neural onde é possível utilizar uma grande variedade de camadas disponíveis. A primeira camada adicionada ao modelo é a camada *Embedding*, onde é recebido de entrada os dados textuais processados utilizando os métodos da classe *Tokenizer*, do *Keras*. *Tokenizer* permite vetorizar um texto em uma lista de inteiros. Cada inteiro é mapeado para um valor em um dicionário que codifica o texto inteiro, com as chaves no dicionário sendo os próprios termos do vocabulário. No vocabulário a indexação é ordenada de modo que as palavas mais utilizadas fiquem nas primeiras posições. É possível definir o tamanho do vocabulário com o parâmetro *num_words*, e quando surge uma nova palavra que não pertença ao vocabulário, o tamanho do vocabulário expande em mais uma posição para a respectiva palavra. Como sentenças podem possuir um número de palavras diferente, é possível utilizar o método *pad_sequence*, onde é preenchida a sequência com zeros. Para especificar o tamanho das sequências pode-se utilizar o parâmetro *maxlen*, assim, quando uma sentença recebe o tamanho definido e caso não possua palavras suficientes, o resto do vetor é preenchido por zeros. Processado os dados textuais de entrada, foi criado uma Rede Neural, no *Keras*, onde a primeira camada é a *Embedding*, seguida de uma LSTM e duas *Dense*. A Figura 8 apresenta o modelo formado pelas camadas adicionadas.

Figura 8 – Rede Neural Keras



Fonte: Autor (2019).

[2] https://keras.io/

4.3.1 Atributos

Na análise textual há um único atributo, que são os textos dos *tweets*. Como cada perfil produz uma quantidade considerável de *tweet*s ao longo do tempo, foi coletado os últimos 50 *tweets* de cada perfil.

5 EXPERIMENTOS E RESULTADOS

Neste capítulo é descrito os conjuntos de dados utilizados nesta pesquisa e como esses dados foram coletados no *Twitter*, como os algoritmos de classificação foram configurados e os resultados obtidos nesses experimentos.

5.1 DATA SETS

Para compor os *data sets* utilizados nesse trabalho, foram coletados dados de várias contas por meio da biblioteca *Tweepy* [3], uma biblioteca Python que se comunica coma API do *Twitter*. Como o problema a ser investigado é um problema de aprendizagem supervisionada, também as classes das contas foram coletadas (*bot* ou usuário legítimo). Para realizar essa etapa, então, foi usado dois *data sets* disponíveis na literatura, os quais continham o id do *Twitter* de cada perfil assim como sua respectiva classe. Foi usado os ids dos perfis para executar um *script* a fim de coletar os atributos usados neste trabalho.

O primeiro *data set* utilizado para coleta de dados foi utilizado nos experimentos feitos em [Kudugunta e Ferrara 2018], onde um sub conjunto de um *data set* mais completo foi originalmente proposto por [Cresci et al. 2017]. Esse *data set* contém contas de quatro classes, chamadas (i) *legitimate user accounts*, (ii) *social spambots 1*, (iii) *social spambots 2* e (iii) *social spambots 3*. O primeiro grupo é de usuários legítimos e o grupos restantes são de contas falsas. De acordo com [Cresci et al. 2017], o grupo "*legitimate user accounts*"é composto por contas de usuários legítimos coletadas e verificadas manualmente. "*Social spambots 1*"são contas usadas para retweetar posts de um candidato político da Itália. Essas contas se comportam de maneira similar à dos usuários legítimos, porém quando o candidato postava um novo *tweet*, essas contas retweetavam poucos minutos depois. "*Social spambots 2*"são contas que foram usadas para promover uma respectiva *hashtag*, a qual se referia a uma aplicação de smartphone. "*Social spambots 3*"são perfis do *Twitter* que interagiam com usuários legítimos porém também usadas para anunciar produtos que estavam a venda em uma plataforma de vendas online.

A fim de avaliar a robustez dos métodos propostos neste trabalho com contas de diferentes características, foi construído um segundo data set coletando as contas presentes no data set proposto em [Lee et al. 2011]. Nesse trabalho, os autores criaram 60 contas falsas, denominadas "*honeypot*", e quando usuários seguissem ou interagissem com essas contas, todas as suas informações e *tweets* anteriores seriam coletadas por um sistema de observação. A cada hora, o

[3] https://www.tweepy.org/

Tabela 2 – Número de instâncias nos *data sets* coletados.

Dataset	Usuários legítimos	Bots		
		Spam1	Spam2	Spam 3
Spambot	2517	488	3319	380
Honeypot	9023		5735	

Fonte: Autor (2019).

sistema de observação checava o status do usuário para determinar se algum *tweet* foi postado. Com esse processo, 23.869 contas foram coletadas. A hipótese dos autores é a de que como as contas *honeypot* possuem um comportamento de um *bot*, não haveria motivos para um usuário legítimo seguir ou interagir com elas. Logo, os perfis que interagiam ou seguiam as contas *honeypot* também seriam perfis falsos. Para confirmar essa hipótese, foi verificado que parte das contas coletadas violavam os termos de uso do *Twitter*. Após isso, os autores analisaram via *clustering* as contas restantes, dividindo-as em quatro grupos, posteriormente classificadas como: (i) *duplicate spammers*, contas que postavam *tweets* quase idênticos, (ii) *duplicated @ spammers*, contas similares às anteriores, porém que mencionavam usuários legítimos, (iii) *malicious promoter*, que postavam *tweets* relacionados a negócios e marketing e possuíam vários seguidores (*followers*) e seguidos (*following*) e (iv) *friend infiltrator*, que seguia qualquer conta que a seguisse e, posteriormente, começava a ter um comportamento de *spam*. Finalmente, para compor o *data set*, junto com o número original de contas falsas, os autores coletaram aleatoriamente 19.297 usuários legítimos. Para confirmar que essas contas eram legítimas, os autores as monitoraram por três meses a fim de checar se elas continuariam ativas e não suspenas. Após isso, 19.296 perfis ainda estavam ativos e foram classificados como usuários legítimos.

Foi utilizado os ids e as classes desses dois data sets para criar dois data sets com as contas e atributos investigados neste trabalho. Foi implementado um *script* em Python usando a biblioteca *Tweepy* para requisitar as informações de *timeline* e metadados de cada usuário utilizando o parâmetro id. A API do *Twitter* provê o texto de cada *tweet* e alguns metadados, como as *hashtags* utilizadas, menções a outros usuários e o momento de postagem. O *data set* composto pelas contas do *data set* utilizado em [Kudugunta e Ferrara 2018] é referido neste trabalho como *data set* Spambot, e o *data set* composto pelas contas do *data set* utilizado em [Lee et al. 2011] é referido como *data set* Honeypot. Como o *Twitter* vem mantendo um esforço contínuo em eliminar contas falsas, algumas das contas presentes no *data set* original não estão mais ativas. Tabela 2 mostra o número resultante das instâncias coletadas em cada *data set*.

5.2 EXPERIMENTOS COM SPAMBOT

Nesta seção, é tratado dos experimentos utilizados no *data set* Spambot, assim como a análise dos dados coletados para os respectivos atributos. Como o *data set* Spambot possui três grupos de *bots*, foi unificado os três grupos e classificados como uma só classe, similarmente a [Kudugunta e Ferrara 2018], a fim de realizar uma classificação binária. Logo, o *data set* ficou composto por 4187 instâncias da classe "*bot*"e 2517 instâncias da classe "usuário". O objetivo dos experimentos realizados é validar se a partir dos atributos investigados é possível distinguir contas falsas de contas legítimas. Com isso, foi comparada a capacidade de classificão dos algoritmos MLP, *Random Forest* e *Naive Bayes* implementados com a biblioteca para Python *scikit-learn* [Pedregosa et al. 2011]. Para o método que utiliza atributos textuais, foi usado um modelo de rede neural da biblioteca *Keras* junto com uma rede LSTM em uma de suas camadas.

Para a MLP, a função de ativação utilizada foi a *logistic* e o *solver* foi lbfgs. Foi realizado uma *grid search* para otimizar os parâmetros para a MLP, variando o seguinte: *max iterations* = [1000,1100,...,1900,2000], *alpha* = [0.1,0.01,...,1^{-10}], *hidden_layer_sizes* = [10,15]. A melhor configuração foi *max iterations* = 1000, *alpha* = 0.1, *hidden_layer_sizes* = 12. Para o *Random Forest*, também foi executado uma *grid search* para otimizar os parâmetros variando o seguinte: *number of estimators* = [200,500], *max features* = [auto,sqrt,$\log_2$], *max depth* = [4,5,6,7,8], *criterion* = [gini, entropy]. Os melhores resultados foram *max features* = auto, *max depth* = 8, *number of estimators* = 200 e o *criterion* = entropy. Parâmetros do *Naive Bayes* não foram otimizados. Uma validação cruzada com 10 *folds* foi utilizada para avaliar a performance de classificação desses algoritmos.

Para os atributos de perfil, foi executado uma *grid search* para os mesmos algoritmos de aprendizagem de máquina e a mesma variância de parâmetro. A melhor configuração para a MLP foi: *max iterations* = 1000, *alpha* = 0.0001, *hidden_layer_sizes* = 13. E para o *Random Forest*, os melhores resultados encontrados foram: *max features* = sqrt, *max depth* = 8, *number of estimators* = 500 e *criterion* = entropy. Parâmetros do *Naive Bayes* também não foram otimizados e uma validação cruzada de 10 *folds* foi utilizada para avaliar a performance de classificação dos algortimos.

Para o atributo de texto, foi utilizado um modelo de Rede Neural *Sequential* da biblioteca *Keras*. *Tweets* em geral não são sempre um texto plano, podendo conter menções a outros usuários (composto pelo arroba seguido do *username* do usuário), *emojis*, *hashtags* e URLs. Com isso, foi realizada uma etapa de pré processamento onde qualquer menção a outro usuário seria substituída

pela tag <user>, URLs seriam substituídas pela tag <url> e *hashtags* seriam substuídas pela tag <hashtag>. Para a etapa de *word embedding*, os dados precisam ser tokenizados num formato compatível, ou seja, transformados de texto para dados numéricos. A biblioteca *Keras* oferece alguns métodos para processamento de texto. Para este trabalho, foi utilizado os métodos da classe *Tokenizer* a fim vetorizar os textos em uma lista de inteiros. Posteriormente, os dados foram utilizados pela rede neural da biblioteca *Keras* para realizar o treinamento e a classificação. Foi realizada uma validação cruzada de 10 *folds* para avaliar a performance do modelo.

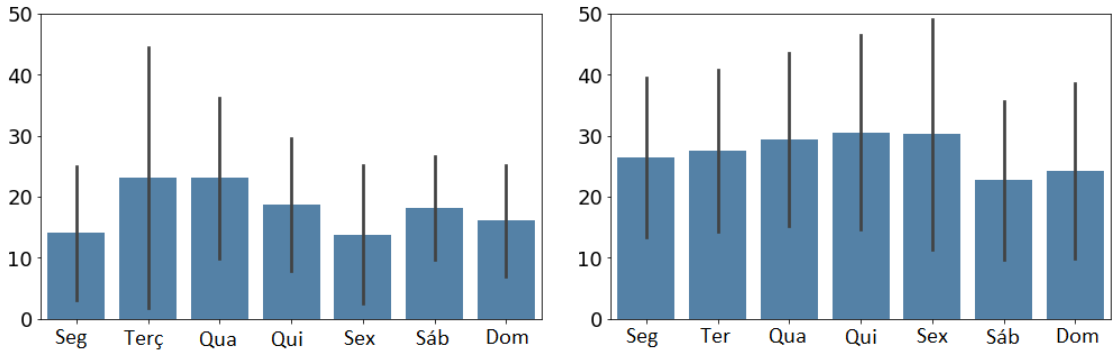
5.2.1 Análise de dados

Nesta subseção, é analisada a capacidade de discriminação promovida pelos atributos investigados no *data set* Spambot. É analisado, então, os dados a partir dos atributos de comportamento e de perfil. Vários trabalhos na literatura propuseram o uso de intervalos entre posts como atributo para reconhecimento de automação [Cai et al. 2017] [Varol et al. 2017]. Adicionalmente ao intervalo entre posts, é possível identificar um padrão que discrimina *bots* de usuários comuns analisando as postagens ao longo dos dias da semana. A Figura 9 mostra o comportamento de postagem das contas do *data set* Spambot ao longo da semana. Nesta figura, podemos ver uma diferença de padrões de postagem entre as contas falsas e as contas legítimas. Também podemos ver que um usuário legítimo (Figura. 9(b)) geralmente posta mais durante os dias da semana. Isso pode ser explicado pelo fato de que, em finais de semana, pessoas tendem a realizar atividades offline. Na figura também podemos ver que o número de posts produzidos por usuários reais é quase o mesmo ao longo dos dias da semana, o que aponta um comportamento espontâneo. Esse comportamento espontâneo é evidenciado também pelo alto desvio padrão do número de posts. *Bots*, por outro lado, apresentam um específico padrão de comportamento, mostrando algumas preferências em termos de dias para postagem, como as terças e as quartas.

Além disso, é analisado a função de densidade de probabilidde dos valores dos atributos de comportamento para o *data set* Spambot (Figura. 10). Esses resultados mostram que os valores dos atributos são bastante discriminantes entre *bots* e usuários legítimos. Em termos de média e desvio padrão dos intervalos entre posts subsequentes, usuários legítimos não apresentam algum padrão, enquanto *bots* tendem a ter um intervalo bastante pequeno entre postagens subsequentes. Essa Figura também mostra que *bots* apresentam uma pequena variabilidade de postagens durante os dias da semana, o que sugere um comportamento programado.

Como dito anteriormente, os atributos de perfil contém as informações básicas da conta de um usuário e com isso pode-se extrair algum padrão a partir dessas informações. Para compor

Figura 9 – Número de postagem em cada dia da semana para o *data set* Spambot.



(a) Bots (b) Usuários legítimos

Fonte: Autor (2019).

os atributos de perfil, foram escolhidos os seguintes: número de seguidores, número de perfis seguidos pelo usuário e a geolocalização do usuário (se ativada ou não). Analisando o número de seguidores (*followers*) e o número de usuários que a conta segue (*following*), percebe-se que contas falsas possuem um menor número de seguidores, como mostra na Figura 11, visto que não haveria motivos para uma conta legítima seguir uma conta falsa. Além disso, *bots* seguem mais perfis do que usuários legítimos, indicando intenção de expandir o acesso a seus *tweets* e ganhar mais visibilidade para fins maliciosos, como *spam*, notícias falsas e propaganda.

Ao analisar a geolocalização, resultados mostram que quase todas as contas falsas possuem a geolocalização desativada, como mostra na Figura 12, tornando um atributo discriminante. Já para o grupo de usuários legítimos, a maioria desses possui a geolocalização ativada (Figura. 12(b)).

5.2.2 Resultados com Spambot

Considerando os três tipos de *bots* como apenas uma classe, foi selecionado aleatoriamente 380 instâncias de cada grupo para compor a classe *bot*, totalizando 1140 instâncias. Foi selecionado também, para balancear o conjunto de dados, 1140 instâncias do grupo de usuários legítimos. A Tabela 3 apresenta os resultados da classificação utilizando atributos de comportamento em termos de acurácia, *recall*, precisão e *F1 score*. Resultados mostraram que a acurácia dos modelos estão perto de 100% no caso da MLP e *Random Forest*, o que indica que os atributos de comportamento propostos nesta pesquisa são relevantes para diferenciar *bots* sociais de usuários legítimos. MLP apresentou a melhor acurácia (99.5%) comparado aos outros métodos. Esse resultado é bastante similar ao atingido por [Kudugunta e Ferrara 2018], o qual investigou vários atributos, como dados de perfil, de *tweets* e textuais.

Tabela 3 – Classificação utilizando atributos de comportamento.

Algoritmo	Métrica	Acurácia	Recall	Precisão	F-Score
Radom Forest	média	0.9727	0.9710	0.9747	0.9727
	desvio padrão	0.0109	0.0111	0.0167	0.0107
MLP	**média**	**0.9951**	**0.9938**	**0.9965**	**0.9951**
	desvio padrão	0.0049	0.0079	0.0069	0.0049
Naive Bayes	média	0.7105	0.4960	0.8675	0.6296
	desvio padrão	0.0347	0.0616	0.0346	0.0580

Fonte: Autor (2019).

Tabela 4 – Classificação utilizando atributos de perfil

Algoritmo	Métrica	Acurácia	Recall	Precisão	F-Score
Radom Forest	**média**	**0.9421**	**0.9543**	**0.9318**	**0.9426**
	desvio padrão	0.0254	0.0359	0.0259	0.0259
MLP	média	0.8460	0.9798	0.7743	0.8645
	desvio padrão	0.0241	0.0184	0.0302	0.0185
Naive Bayes	média	0.8442	0.9780	0.7730	0.8630
	desvio padrão	0.0241	0.0197	0.0309	0.0183

Fonte: Autor (2019).

Para os atributos de perfil, a Tabela 4 mostra que os resultados também foram satisfatórios, tendo o *Random Forest* com mais de 94% de acurácia.

Já para a análise textual, obteve-se uma acurácia de 94% com o modelo de rede neural da bibliteca *Keras*. Os resultados obtidos até então mostram que para o *data set* Spambot, analisando os três tipos de atributos investigados neste trabalho, foi possível construir modelos de Aprendizagem de Máquina capazes de diferenciar *bots* de usuários legítimos com uma boa acurácia.

5.3 EXPERIMENTOS COM HONEYPOT

A fim de avaliar a robustez dos métodos propostos, foi utilizado um segundo *data set*, Honeypot. Diferentemente do *data set* Spambot, o Honeypot só possui dois grupos: *bots* e usuários legítmos. O objetivo dos experimentos realizados a seguir é validar se, analisando os atributos investigados até então, é possível distinguir contas falsas de contas legítimas baseando-se em um *data set* diferente. Com isso, também foi utilizado os mesmos algoritmos de Aprendizagem de Máquina: MLP, *Random Forest* e *Naive Bayes*, todos implementados com a biblioteca para Python *scikit-learn* [Pedregosa et al. 2011]. Para o método que utiliza atributos textuais, também

foi utilizado uma Rede Neural com as mesmas camadas.

Para a MLP, a função de ativação utilizada foi a *logistic* e o *solver* foi lbfgs. Foi realizado uma *grid searh* para otimizar os parâmetros para a MLP, variando o seguinte: *max iterations* = [1000,1100,...,1900,2000], *alpha* = [0.1,0.01,...,1^{-10}], *hidden_layer_sizes* = [10,15]. A melhor configuração foi *max iterations* = 1000, *alpha* = 0.1, *hidden_layer_sizes* = 12. Para o *Random Forest*, também foi executado uma *grid search* para otimizar os parâmetros variando o seguinte: *number of estimators* = [200,500], *max features* = [auto,sqrt,$\log_2$], *max depth* = [4,5,6,7,8], *criterion* = [gini, entropy]. Os melhores resultados foram *max features* = auto, *max depth* = 8, *number of estimators* = 200 e *criterion* = entropy. Parâmetros do *Naive Bayes* não foram otimizados. Uma validação cruzada com 10 *folds* foi utilizada para avaliar a performance de classificação desses algoritmos.

Para os atributos de perfil, foi executado uma *grid search* para os mesmos algoritmos e a mesma variância de parâmetro. A melhor configuração para a MLP foi: *max iterations* = 1000, *alpha* = 0.0001, *hidden_layer_sizes* = 11. E para o *Random Forest*, os melhores resultados encontrados foram: *max features* = log2, *max depth* = 8, *number of estimators* = 500 e *criterion* = gini. Parâmetros do *Naive Bayes* também não foram otimizados e uma validação cruzada de 10 *folds* foi utilizada para avaliar a performance de classificação dos algortimos.

Para a análise textual, foram executadas as mesmas etapas de pré processamento realizadas no data set anterior.

5.3.1 Análise de Dados

Uma abordagem similar a do *data set* Spambot foi relizada para o *data set* Honeypot. Figura 13 mostra que o comportamento de postagem dos *bots* e usuários é bastante similar. Isso pode ser explicado devido ao fato de que os *bots* do Honeypot são mais sofisticados que os *bots* do *data set Spambot*. A similaridade entre usuários e *bots* é ainda mais evidente quando analisado as funções de densidade de probabilidade para os valores dos atributos dos usuários e *bots* (Figura. 14).

Além disso, nota-se que os atributos de perfil não são tão discriminantes para as contas falsas e legítimas do *data set* Honeypot, como mostra na Figura 15. Nesse caso, como no *data set* Spambot, usuários também seguem menos perfis do que *bots*, porém, ao analisar o número de seguidores, usuários legítimos possuem menos seguidores do que contas falsas.

Para o atributo de geolocalização, também houve uma diferença entre as contas, porém não tão perceptível como no Spambot (Figura 16).

Tabela 5 – Resultados da classificação para os atributos de comportamento

Algoritmo	Métrica	Acurácia	Recall	Precisão	F-Score
Radom Forest	média	0.8982	0.8930	0.9020	0.8974
	desvio padrão	0.0071	0.0124	0.0082	0.0074
MLP	**média**	**0.9880**	**0.9891**	**0.9868**	**0.9880**
	desvio padrão	0.0041	0.0046	0.0071	0.0041
Naive Bayes	média	0.5379	0.1785	0.6298	0.2779
	desvio padrão	0.0165	0.0217	0.0597	0.0313

Fonte: Autor (2019).

Tabela 6 – Resultados da classificação para os atributos de perfil

Algoritmo	Métrica	Acurácia	Recall	Precisão	F-Score
Radom Forest	**média**	**0.7714**	**0.8357**	**0.7399**	**0.7842**
	desvio padrão	0.0257	0.0573	0.0111	0.0317
MLP	média	0.6826	0.7557	0.6590	0.7029
	desvio padrão	0.0263	0.0671	0.0177	0.0354
Naive Bayes	média	0.6089	0.8113	0.5817	0.6746
	desvio padrão	0.0431	0.0782	0.0338	0.0246

Fonte: Autor (2019).

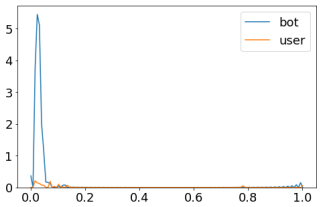
5.3.2 Resultados com o Honeypot

Para os atributos de comportamento, foi utilizado o mesmo esquema de avaliação do *data set* Spambot. A Tabela 5 apresenta os resultados obtidos para os respectivos algoritmos. A MLP alcançou o melhor resultado, com uma acurácia de 98.8%.

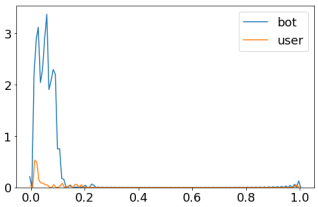
Utilizando os atributos de perfil, a Tabela 6 apresenta os resultados alcançados, onde o algoritmo *Random Forest* foi o que atingiu o melhor resultado, com 77.14% de acurácia.

Para a análise textual, foi alcançado 75.83% de acurácia. Um resultado similar ao método anterior o qual analisa os atributos de perfil.

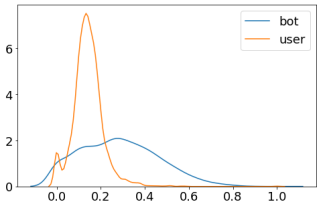
Figura 10 – Função Densidade de Probabilidade dos valores dos atributos do *data set* Spambot.



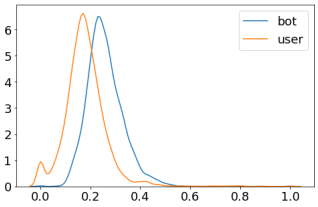
(a) Média dos intervalos entre *tweets*.



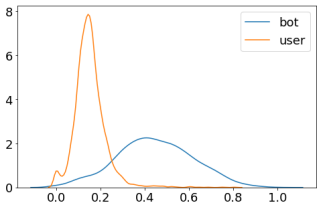
(b) Desvio padrão do intervalo entre os *tweets*



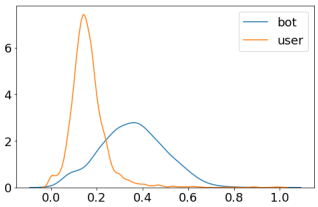
(c) Frequência de postagens nas segundas.



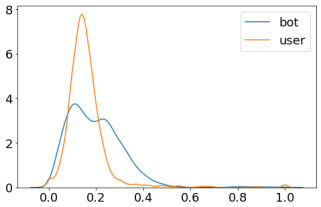
(d) Frequência de postagens nas terças.



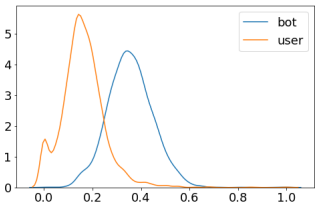
(e) Frequência de postagens nas quartas.



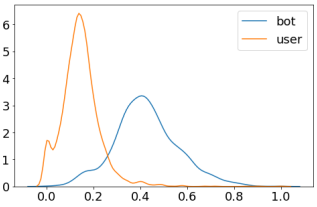
(f) Frequência de postagens nas quintas.



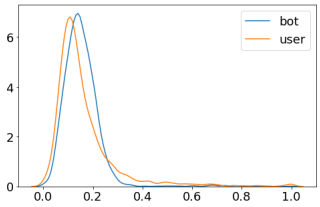
(g) Frequência de postagens nas sextas.



(h) Frequência de postagens nos sábados.



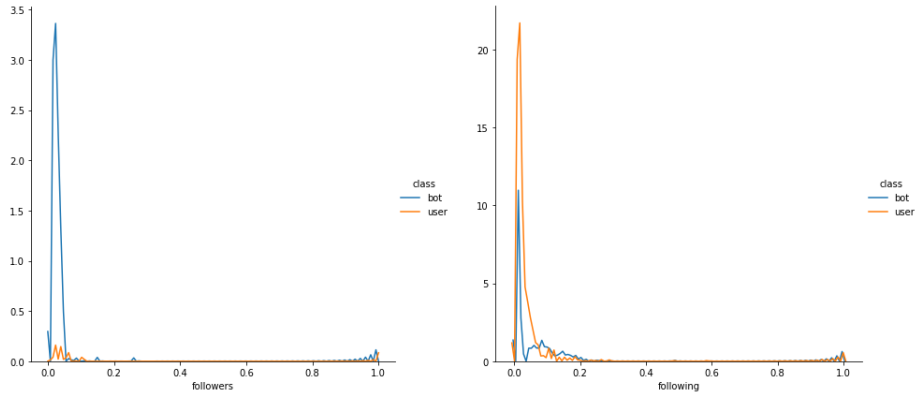
(i) Frequência de postagens nos domingos.



(j) Desvio padrão de postagem ao longo da semana.

Fonte: Autor (2019).

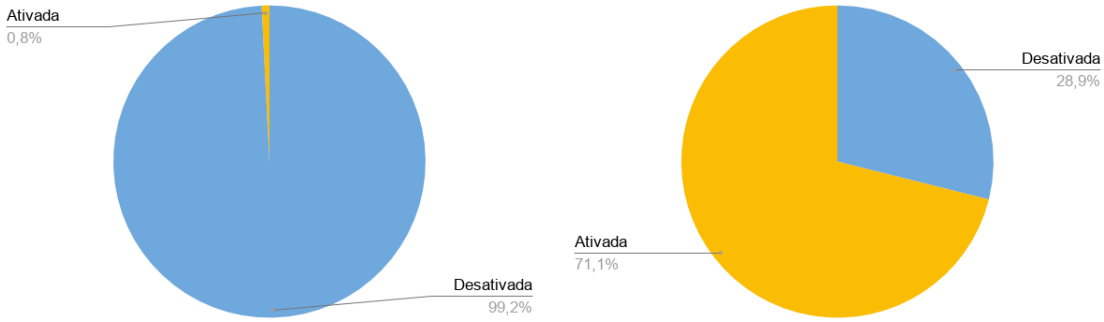
Figura 11 – Following e Followers para o *data set* Spambot.



(a) Followers (b) Following

Fonte: Autor (2019).

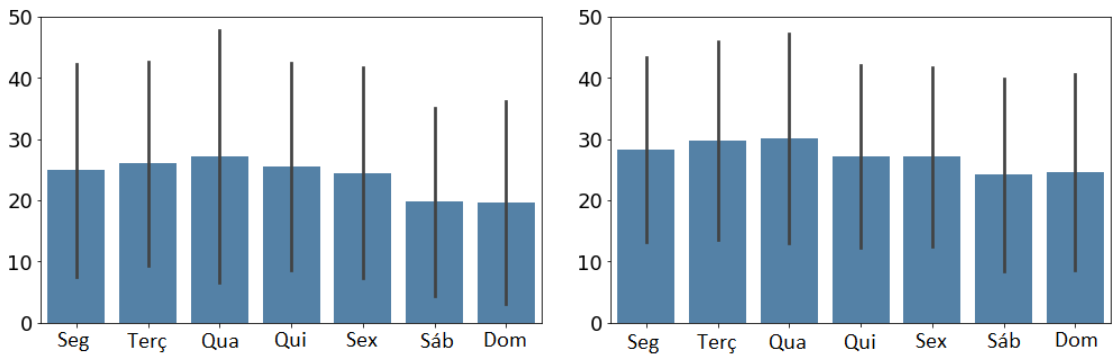
Figura 12 – Geolocalização para o *data set* Spambot.



(a) Bots (b) Usuários legítimos

Fonte: Autor (2019).

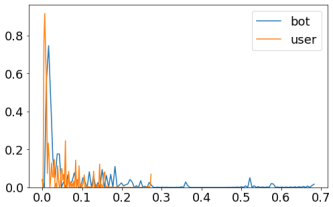
Figura 13 – Número de postagem para cada dia da semana do *data set* Honeypot.



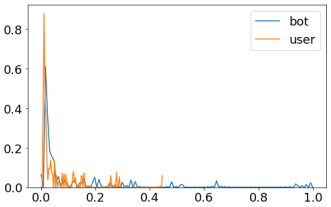
(a) Bots (b) Usuários legítimos

Fonte: Autor (2019).

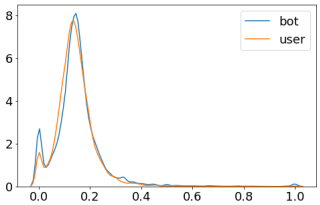
Figura 14 – Função Densidade de Probabilidade para os valores dos atributos do *data set* Honeypot.



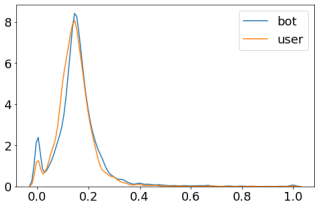
(a) Média dos intervalos entre *tweets*.



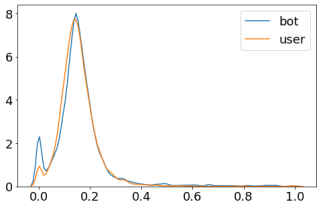
(b) Desvio padrão dos intervalos entre os *tweets*



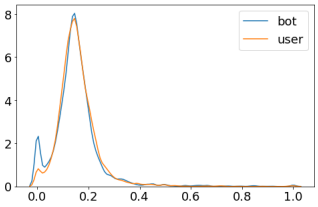
(c) Frequência de postagens nas segundas.



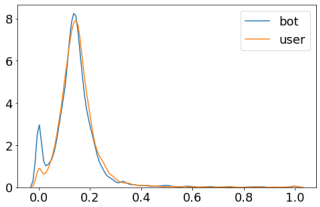
(d) Frequência de postagens nas terças.



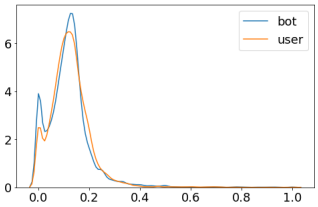
(e) Frequência de postagens nas quartas.



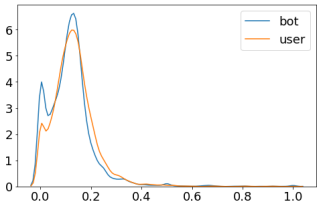
(f) Frequência de postagens nas quintas.



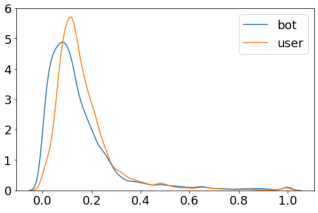
(g) Frequência de postagens nas sextas.



(h) Frequência de postagens nos sábados.



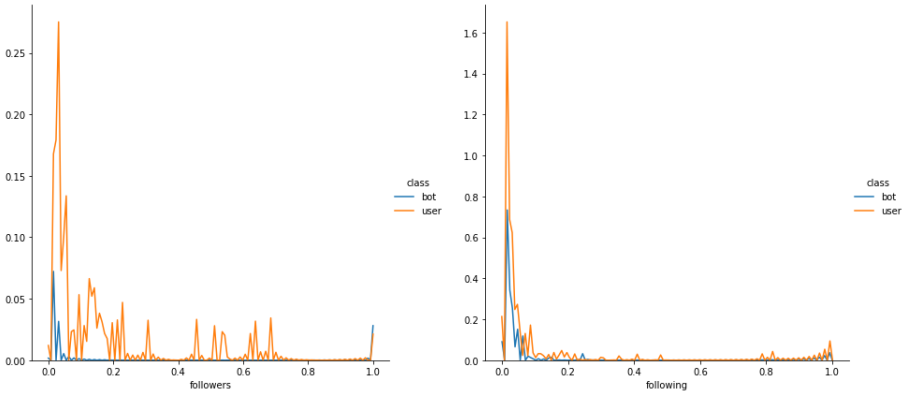
(i) Frequência de postagens nos domingos.



(j) Desvio padrão de postagem ao longo da semana

Fonte: Autor (2019).

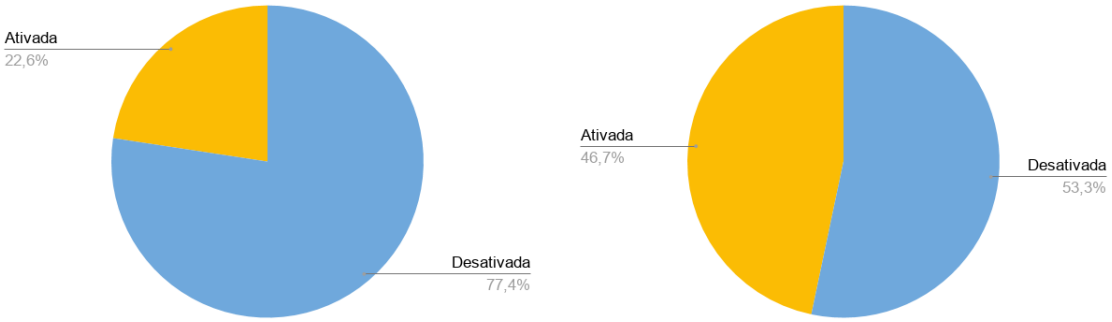
Figura 15 – Following e Followers



(a) Bots (b) Usuários legítimos

Fonte: Autor (2019).

Figura 16 – Geolocalização para o *data set* Honeypot.



(a) Bots (b) Usuários legítimos

Fonte: Autor (2019).

6 CONCLUSÃO

Neste trabalho foi apresentado uma breve descrição sobre *bots*, ou contas falsas, e como elas impactam negativamente as redes sociais com propósitos maliciosos. Contas falsas, além de poluírem a plataforma, influenciam a opinião pública publicando e compartilhando notícias falsas. Faz-se necessário, então, detectá-las e combatê-las a fim de criar um ambiente harmônico para um ambiente digital onde vem sendo cada vez mais utilizado.

Dessa forma, o objetivo deste trabalho foi aplicar algoritmos de Aprendizagem de Máquina em dois *data sets* a fim de verificar se os atributos investigados possuem relevância e poder de discriminação ao classificar uma conta do *Twitter* como falsa ou legítima. Para o primeiro *data set*, chamado de Spambot, os resultados foram satisfatórios analisando os três tipos de atributos (de comportamento, de perfil e textuais). Já para o segundo *data set*, chamado de Honeypot, com exceção da análise de comportamento, os resultados alcançados não se mostraram tão satisfatórios. Isso pode indicar que as contas que compuseram esse *data set* são mais sofisticadas e possuem características e comportamentos mais próximos de usuários legítimos.

Ao analisar os atributos coletados das contas pôde-se verificar alguns padrões que diferem contas automatizadas de contas legítimas, como por exemplo: *bots* tendem a possuir um comportamento mais regularizado e possuir a geolocalização desativada. Como as análises textuais tiveram um resultado interessante, isso indica que usuários legítimos talvez possuam uma linguagem mais complexa, enquanto que *bots*, devido a sua natureza robótica, talvez possuam limitações ao se tratar de linguagem natural.

6.1 TRABALHOS FUTUROS

Para trabalhos futuros, há várias maneiras de avançar nesta pesquisa. Um método de combinação de classificadores (*ensemble*) poderia aumentar o poder de classificação deste trabalho. Nesse método, cada modelo estudado aqui atribuiria uma classe a cada instância e o modelo de comitê escolheria o voto majoritário como voto final. Outra investigação mais desafiadora e sofisticada seria monitorar contas falsas a fim detectar mudanças de comportamento em uma certa mudança de período. Uma conta falsa poderia se comportar de maneira similar a um humano e, de repente, num período de eleições políticas por exemplo, se comportaria de uma maneira mais automática a fim de fazer *spam*. Como os *data sets* utilizados nesse trabalho não são tão atuais e o comportamento de contas falsas talvez esteja mais sofisticado, também

seria interessante utilizar um *data set* composto por contas mais recentes.

REFERÊNCIAS

ADEWOLE, K. S.; ANUAR, N. B.; KAMSIN, A.; VARATHAN, K. D.; RAZAK, S. A. Malicious accounts: dark of the social networks. **Journal of Network and Computer Applications**, Elsevier, v. 79, p. 41–67, 2017.

AZAB, A. E.; IDREES, A. M.; MAHMOUD, M. A.; HEFNY, H. Fake account detection in twitter based on minimum weighted feature set. **Int. Sch. Sci. Res. Innov**, v. 10, n. 1, p. 13–18, 2016.

BREIMAN, L. Random forests. **Machine learning**, Springer, v. 45, n. 1, p. 5–32, 2001.

CAI, C.; LI, L.; ZENGI, D. Behavior enhanced deep bot detection in social media. In: IEEE. **Proceedings [...]**. [S.l.], 2017. p. 128–130.

CHU, Z.; GIANVECCHIO, S.; WANG, H.; JAJODIA, S. Detecting automation of twitter accounts: Are you a human, bot, or cyborg? **IEEE Transactions on Dependable and Secure Computing**, IEEE, v. 9, n. 6, p. 811–824, 2012.

CLARK, E. M.; WILLIAMS, J. R.; JONES, C. A.; GALBRAITH, R. A.; DANFORTH, C. M.; DODDS, P. S. Sifting robotic from organic text: a natural language approach for detecting automation on twitter. **Journal of Computational Science**, Elsevier, v. 16, p. 1–7, 2016.

CRESCI, S.; PIETRO, R. D.; PETROCCHI, M.; SPOGNARDI, A.; TESCONI, M. Fame for sale: Efficient detection of fake twitter followers. **Decision Support Systems**, Elsevier, v. 80, p. 56–71, 2015.

CRESCI, S.; PIETRO, R. D.; PETROCCHI, M.; SPOGNARDI, A.; TESCONI, M. The paradigm-shift of social spambots: Evidence, theories, and tools for the arms race. In: INTERNATIONAL WORLD WIDE WEB CONFERENCES STEERING COMMITTEE. **Proceedings [...]**. [S.l.], 2017. p. 963–972.

EVERETT, R. M.; NURSE, J. R.; EROLA, A. The anatomy of online deception: What makes automated text convincing? In: ACM. **Proceedings [...]**. [S.l.], 2016. p. 1115–1120.

FERRARA, E.; VAROL, O.; DAVIS, C.; MENCZER, F.; FLAMMINI, A. The rise of social bots. **Communications of the ACM**, ACM, v. 59, n. 7, p. 96–104, 2016.

GURAJALA, S.; WHITE, J. S.; HUDSON, B.; VOTER, B. R.; MATTHEWS, J. N. Profile characteristics of fake twitter accounts. **Big Data & Society**, SAGE Publications Sage UK: London, England, v. 3, n. 2, p. 1–13, 2016.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural computation**, MIT Press, v. 9, n. 8, p. 1735–1780, 1997.

HOWARD, P. N.; BOLSOVER, G.; KOLLANYI, B.; BRADSHAW, S.; NEUDERT, L.-M. Junk news and bots during the us election: What were michigan voters sharing over twitter. **Computational Propaganda Research Project, Oxford Internet Institute, Data Memo**, n. 2017.1, 2017.

INUWA-DUTSE, I.; LIPTROTT, M.; KORKONTZELOS, I. Detection of spam-posting accounts on twitter. **Neurocomputing**, Elsevier, v. 315, p. 496–511, 2018.

KUDUGUNTA, S.; FERRARA, E. Deep neural networks for bot detection. **Information Sciences**, Elsevier, v. 467, p. 312–322, 2018.

LEE, K.; EOFF, B. D.; CAVERLEE, J. Seven months with the devils: A long-term study of content polluters on twitter. In: AAAI. **Proceedings [...]**. [S.l.], 2011.

MAKICE, K. **Twitter API: Up and running: Learn how to build applications with the Twitter API**. [S.l.]: "O'Reilly Media, Inc.", 2009.

MEHROTRA, A.; SARREDDY, M.; SINGH, S. Detection of fake twitter followers using graph centrality measures. In: IEEE. **Proceedings [...]**. [S.l.], 2016. p. 499–504.

PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V. et al. Scikit-learn: Machine learning in python. **Journal of machine learning research**, v. 12, n. Oct, p. 2825–2830, 2011.

RISH, I. et al. An empirical study of the naive bayes classifier. In: IJCAI. **Proceedings [...]**. [S.l.], 2001. v. 3, n. 22, p. 41–46.

ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. **Psychological review**, American Psychological Association, v. 65, n. 6, p. 386, 1958.

VAROL, O.; FERRARA, E.; DAVIS, C. A.; MENCZER, F.; FLAMMINI, A. Online human-bot interactions: Detection, estimation, and characterization. In: AAAI. **Proceedings [...]**. [S.l.], 2017.

WANG, A. H. Don't follow me: Spam detection in twitter. In: IEEE. **Proceedings [...]**. [S.l.], 2010. p. 1–10.

XUE, J.; YANG, Z.; YANG, X.; WANG, X.; CHEN, L.; DAI, Y. Votetrust: Leveraging friend invitation graph to defend against social network sybils. In: IEEE. **Proceedings [...]**. [S.l.], 2013. p. 2400–2408.

ZHANG, X.; ZHU, S.; LIANG, W. Detecting spam and promoting campaigns in the twitter social network. In: IEEE. **Proceedings [...]**. [S.l.], 2012. p. 1194–1199.