UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
FACULDADE DE CIÊNCIAS - CAMPUS BAURU
DEPARTAMENTO DE COMPUTAÇÃO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

MARCELO HIDEAKI IWATA KITO

UMA ABORDAGEM DE CIÊNCIA DE DADOS PARA IDENTIFICAR FAKE NEWS NO ÂMBITO POLÍTICO

BAURU
Novembro/2019

MARCELO HIDEAKI IWATA KITO

UMA ABORDAGEM DE CIÊNCIA DE DADOS PARA IDENTIFICAR FAKE NEWS NO ÂMBITO POLÍTICO

Trabalho de Conclusão de Curso do Curso de Bacharelado em Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Campus Bauru.
Orientador: Prof. Associado Dr. João Pedro Albino

BAURU
Novembro/2019

Marcelo Hideaki Iwata Kito Uma abordagem de Ciência de Dados para identificar Fake News no âmbito político/ Marcelo Hideaki Iwata Kito. – Bauru, Novembro/2019- 52 p. : il. (algumas color.) ; 30 cm.
Orientador: Prof. Associado Dr. João Pedro Albino
Trabalho de Conclusão de Curso – Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Bacharelado em Ciência da Computação, Novembro/2019.
1. Ciência de Dados 2. Processamento de Linguagem Natural 3. Aprendizado de Máquina 4. Detecção de Fake News 5. Política

Marcelo Hideaki Iwata Kito

Uma abordagem de Ciência de Dados para identificar Fake News no âmbito político

Trabalho de Conclusão de Curso do Curso de Bacharelado em Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Campus Bauru.

Banca Examinadora

Prof. Associado Dr. João Pedro Albino
Orientador
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Computação

Profa. Dra. Simone das Graças Domingues Prado
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Computação

Prof. Dr. Kelton Augusto Pontara da Costa
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Computação

Bauru, _____ de _____ de _____.

Dedico este trabalho a todos com quem aprendi e a todos que comigo aprenderão.

Agradecimentos

Agradeço primeiramente à minha família, que serviu como base para o meu desenvolvimento e nunca deixou de me apoiar.

Aos meus amigos, que estiveram comigo ao longo da minha formação, desfrutando de bons momentos juntos e se ajudando em momentos difíceis.

Aos meus professores, por todos os momentos prazerosos e por todo conhecimento que puderam nos passar.

Ao meu orientador Prof. Associado João Pedro Albino, que apoiou o desenvolvimento deste projeto em todos os momentos.

Aos meus colegas da Finch, com quem aprendi inúmeras coisas fundamentais para o meu desenvolvimento profissional e para a produção deste trabalho.

"If we have data, let's look at data.
If all we have are opinions, let's go with mine."
Jim Barksdale

Resumo

Com o advento da Internet, a quantidade de informações disponível cresceu muito rapidamente transformando-se, ao longo dos anos, em uma ampla rede de tecnologias e pessoas conectadas. Uma das muitas mudanças provocadas pela Internet é a facilidade com a qual um usuário consegue consumir informações, e até mesmo produzi-las. Entretanto, essa acessibilidade permitiu a veiculação de informações enganosas numa proporção maior e mais facilmente do que se faria em veículos convencionais. A essas informações falsas tem sido atribuído o termo "fake news", que procuram passar credibilidade ao leitor, mas sem seguir as diretrizes para garantir a precisão ou a veracidade das mesmas. Essas "fake news" têm obtido bastante atenção, principalmente no âmbito político, mas existem casos relacionados a outros tópicos, como vacinação, nutrição e mesmo no mercado de ações, podendo afetar diferentes esferas da sociedade. Com o intuito de atenuar a desinformação gerada pela propagação de "fake news", este projeto buscou desenvolver um aplicativo móvel com o intuito de classificar, utilizando-se de aprendizado de máquina, a veracidade das notícias fornecidas ao usuário. O aplicativo utilizou-se do fluxo de desenvolvimento de um modelo de aprendizado de máquina, da ferramenta Python e suas bibliotecas, assim como o *framework* de desenvolvimento híbrido Ionic. Na primeira etapa de coleta de dados (referenciado como "conjunto de dados primário") foram desenvolvidos rastreadores de rede (*web crawlers*) para obter dados de três sites de notícias escolhidos ao acaso. Em uma segunda etapa de aquisição de dados, utilizou-se o conjunto de dados "Fake.Br corpus", composto de notícias reais e falsas em português e recolhidas, catalogadas e disponibilizadas pela Universidade de São Paulo (USP). Por fim, são apresentados o aplicativo e os resultados obtidos com os modelos treinados.

Palavras-chave: Ciência de Dados, Processamento de Linguagem Natural, Aprendizado de Máquina, Detecção de Fake News, Política.

Abstract

With the advent of the Internet, the amount of available information has grown quickly, transforming itself over the years into a wide network of connected technologies and people. One of the many changes brought by the Internet is the ease with which a user can consume information, and even produce them. However, this accessibility has allowed deceptive information to be conveyed in a larger extent and more easily than it would be in the conventional media. This false information is being called "fake news", which aims to give the reader credibility, but without following the guidelines to ensure its accuracy or veracity. These "fake news" have been getting a lot of attention, specially in the political context, but there are cases related to other topics, such as vaccination, nutrition and even the stock market, which may affect different spheres of society. In order to reduce the misinformation generated by the spread of fake news, this project sought to develop a mobile application to classify, using machine learning, the veracity of news presented to the user. The application has made use of a machine learning model development pipeline, the Python programming language and its libraries, as well as the hybrid mobile application development framework Ionic. In the first step of data collection (referred to as the "primary data set") web crawlers were developed to obtain data from three randomly chosen news websites. In a second step of data acquisition, a data set named "Fake.Br corpus" was used, composed of real and fake news in Portuguese that were collected, cataloged and made available by the University of São Paulo (USP). Finally, both the application and the results obtained with the trained models are presented.

Keywords: Data Science, Natural Language Processing, Machine Learning, Fake News, Politics.

Lista de figuras

Lista de quadros

Lista de tabelas

Lista de abreviaturas e siglas

API	*Application Programming Interface*
CSS	*Cascading Style Sheets*
FN	*False Negative*
FP	*False Positive*
HTML	*Hypertext Markup Language*
HTTP	*Hypertext Transfer Protocol*
IDF	*Inverse Document Frequency*
JSON	*JavaScript Object Notation*
NLP	*Natural Language Processing*
REST	*Representational State Transfer*
SVM	*Support Vector Machine*
TF	*Term Frequency*
TN	*True Negative*
TP	*True Positive*
URI	*Uniform Resource Identifier*
USP	Universidade de São Paulo

Sumário

1 Introdução

Desde o advento da Internet, a quantidade de informações que a permeiam cresce muito rapidamente e está disponível na ponta dos dedos de seus usuários, tornando-se, ao longo dos anos, uma ampla rede de tecnologias e pessoas (SANTAELLA, 2012).

Miranda (2000) define conteúdo como sendo todos os recursos, produtos e serviços de informação operados dentro da Internet. Afirma ainda que uma das maiores mudanças provocadas pela Internet é a facilidade com a qual um usuário consegue consumir informações, e até mesmo produzi-las. Essa acessibilidade, porém, também permitiu a veiculação de informações enganosas numa proporção maior e mais facilmente do que se faria em veículos convencionais que não fazem uso da Internet. A essas informações falsas tem sido atribuído o termo *"fake news"*, que tentam passar credibilidade ao leitor, mas sem seguir as diretrizes para garantir a acurácia ou a veracidade das mesmas (LAZER et al., 2018).

Dentro das redes sociais, Pontes (2018) cita as "câmaras de ressonância", que são "bolhas" que concentram usuários com um mesmo pensamento. Nesses ambientes, geralmente há concordância de ideias e, raramente, uma contraposição, o que faz com que os usuários acreditem que suas opiniões são senso comum. Informações distorcidas que "escapam" dessas bolhas e caem no conhecimento público também podem ajudar a espalhar informações incorretas acerca de um determinado assunto.

Lazer et al. (2018) diz que as *fake news* têm ganhado bastante atenção num âmbito político, mas existem casos relacionados a outros tópicos, como vacinação, nutrição e mesmo no mercado de ações, podendo afetar diferentes esferas da sociedade.

1.1 A Internet nas eleições estadunidenses

A Internet se provou, durante as eleições estadunidenses de 2008, um ótimo meio de comunicação e uma poderosa ferramenta política, onde o então candidato à presidência Barack Obama postava vídeos se promovendo e arrecadava fundos para sua campanha eleitoral.

> Numa mensagem postada em 4 de junho de 2008 no blog Media & Politics, ainda durante as primárias do Partido Democrata, Michael Cornfield, cientista político americano e diretor de pesquisa do Projeto Democracia On-line da Universidade George Washington, declarou o seguinte: "Sem internet não haveria Obama. A diferença de compreensão, entre as campanhas de Obama e Clinton, sobre o que se pode realizar por meio da política on-line tem sido um fator decisivo nessa que é a maior reviravolta na história das primárias presidenciais. Há, naturalmente, outras diferenças importantes: a estratégia empregada no "caucus", o glamour, a oratória, os discursos enfocando diretamente o preconceito. Mas nenhuma delas teria sido decisiva sem o dinheiro que Obama arrecadou on-line, os vídeos que Obama postou on-line e, acima de tudo, os milhões de pessoas que aderiram on-line à campanha de Obama, em seus tempos e termos próprios"(GOMES et al., 2009 apud CORNFIELD, 2008)

Nas eleições de 2016, porém, houve uma certa preocupação com a "política online", mais especificamente se tratando das *fake news* que circularam nas redes sociais (ALLCOTT; GENTZKOW, 2017). Um fato notável é de que cerca de 62% dos adultos estadunidenses utilizam redes sociais para ficar a par de notícias, o que pode gerar uma inferência errônea devido às notícias falsas (GOTTFRIED; SHEARER, 2016).

Allcott e Gentzkow (2017) comentam a facilidade de disseminar informações relacionadas à política nas redes sociais, citando o baixo custo de se introduzir nelas e produzir conteúdo como um dos principais motivos. Outro fator determinante são as conexões de um usuário e as informações compartilhadas entre eles, o que o deixa mais propenso a ler e compartilhar informações que confirmem seu ponto de vista, ainda que estas sejam tendenciosas (BAKSHY; MESSING; ADAMIC, 2015).

1.2 *Fake News* no contexto brasileiro

> O problema da disseminação de *fake news* impacta ambas as motivações do voto democrático. Por um lado, a escolha racional de projetos e avaliação do histórico de candidatos fica prejudicada devido à inundação de informações possivelmente inverídicas. Por outro, fatos inverídicos deturpam os já mencionados traços dos candidatos - seja a favor ou em detrimento de sua imagem. Ademais, a disseminação de informações inverídicas também reforça enviesamentos ideológicos de cada eleitor, pois apresenta evidências que confirmam ideias, valores e concepções inerentes às suas visões de mundo (RUEDIGER et al., 2019).

A lei nº 9.504/97, que estabelece normas para eleições, regula a propaganda de caráter político na Internet. O artigo 57-B, § 2º da lei nº 9.504/97, por exemplo, proíbe a veiculação de conteúdos de cunho eleitoral mediante cadastro de usuário de aplicação de Internet com a intenção de falsear identidade (BRASIL, 1997). Ainda assim, existe o problema dos chamados "*bots* sociais", que são perfis falsos se passando por humanos que compartilham notícias enganosas de forma autônoma a fim de aumentar sua propagação (LAZER et al., 2018).

No âmbito brasileiro, o uso de ferramentas de mensagens instantâneas (com o WhatsApp se destacando na categoria) é bastante alto e elas permitem que a comunicação entre usuários seja feita de forma rápida e segura, pois os dados são criptografados. Essas características facilitam a difusão de informações, ao mesmo tempo que dificultam o trabalho de desmentir informações falsas (PONTES, 2018).

Desta maneira, este projeto, com o intuito de atenuar a desinformação gerada pela propagação de *fake news*, teve como base o desenvolvimento de uma aplicação *mobile* capaz de classificar, utilizando-se de aprendizado de máquina, a veracidade de uma notícia fornecida pelo usuário.

2 Fundamentação Teórica

2.1 Detecção automática de *Fake News*

Com cada vez mais pessoas utilizando a Internet e as redes sociais ativamente, uma maior massa de informações é compartilhada rapidamente. Isso infelizmente também se vale a notícias falsas, que num ambiente dinâmico como a Internet pode tomar proporções grandes e prejudicar alguém, já que muitas pessoas moldam suas opiniões a partir dela. Por causa disso, as *fake news* têm chamado bastante atenção ultimamente, causando até mesmo o surgimento de organizações cujo propósito é unicamente a checagem dos fatos, confrontando as notícias com outros dados e verificando sua veracidade (RASHKIN et al., 2017).

Com toda essa mobilização por parte de organizações e volume de dados, o empenho de criar uma ferramenta para a detecção automática de *fake news* se justifica. Tem havido um esforço por parte da academia para estudar como é feita a disseminação das notícias, o comportamento dos usuários que as criam e dos que as leem, além de estudos sobre as características de linguagem nelas utilizadas com o intuito de poder identificá-las (MONTEIRO et al., 2018).

Uma das soluções remete a uma tarefa da área de processamento de linguagem natural, que será a abordagem utilizada neste trabalho. Neste sentido, busca-se, a partir de algoritmos de aprendizado de máquina, realizar a detecção de padrões textuais utilizados por autores de *fake news* que possam caracterizá-la como tal (CONROY; RUBIN; CHEN, 2015).

2.2 *Big Data*

Big data trata da grande quantidade de dados em diversos formatos e estruturas, de forma que sistemas de gerenciamento de banco de dados tradicionais não conseguem gerenciá-los (KHANNA; AWAD, 2015).

Laney (2001) define três características que vieram a ser conhecidos como os "3 V's do *Big data*":

- Volume: grandes quantidades de dados são geradas à medida que estes são vistos como ativos importantes para organizações;
- Velocidade: trata-se da velocidade em que dados são gerados e recebidos, sendo necessário tratá-los rapidamente;

- Variedade: os dados podem estar em uma ampla variedade de formatos, em alguns casos necessitando um tratamento antes de seu uso efetivo.

2.3 Ciência de Dados

> A chamada "Ciência de Dados" incorpora elementos variados e se baseia em técnicas e teorias oriundas de muitos campos básicos em engenharia e ciências básicas, sendo assim intimamente ligada com muitas das disciplinas tradicionais bem estabelecidas, porém viabilizando uma nova área altamente interdisciplinar (ZIVIANI; PORTO, 2014).

Dhar (2012) afirma que a ciência de dados implica um foco nos dados e, por extensão, na estatística, que é um estudo sistemático das características dos dados e de seu papel nas inferências. Faz também a diferenciação das duas ciências no sentido que a ciência de dados trata dados heterogêneos e não estruturados (como textos e imagens).

O computador cada vez mais tem assumido o papel de apoiador nas tomadas de decisão, permitindo sua escalabilidade diante do grande volume de dados gerados a cada momento e reduzindo a interferência humana sobre tal processo (DHAR, 2012). Para Provost e Fawcett (2013) a recente inundação de dados somada aos avanços no poder dos computadores e redes impulsionaram a aplicação da ciência de dados para obtenção de vantagens competitivas.

2.4 Aprendizado de Máquina

Segundo Khanna e Awad (2015) o aprendizado de máquina é um ramo da inteligência artificial que aplica algoritmos para sintetizar relações implícitas num conjunto de dados e informações. Tem sido útil em uma série de aplicações como buscas na Internet, direcionamento de anúncios, análise de crédito, entre muitas outras.

Problemas de aprendizado de máquina podem ser categorizados em (SCIKIT-LEARN, 2019):

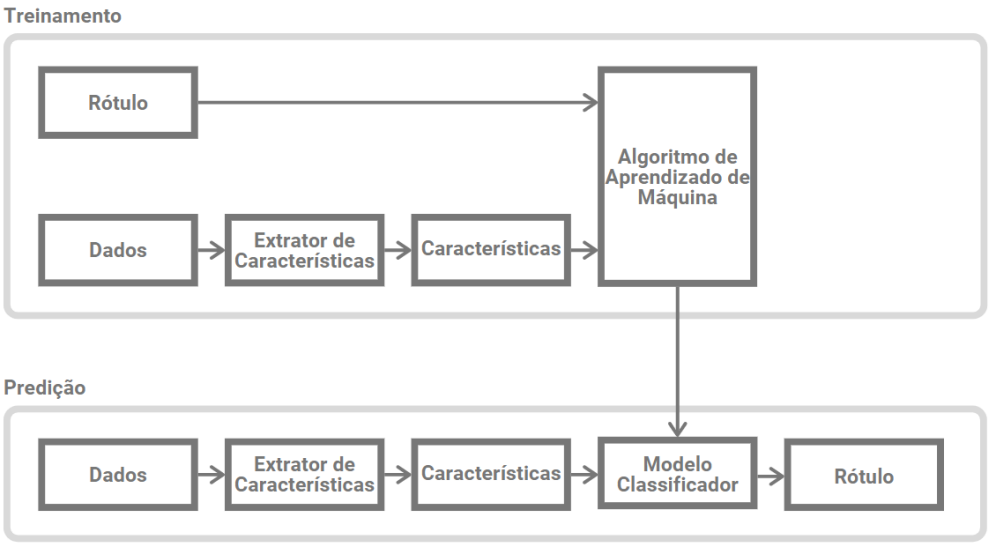
- Aprendizado supervisionado: caso em que os dados possuem atributos adicionais associados às entradas e que desejam ser preditos. Este tipo de problema se subdivide em:
  - Classificação: há amostras de dados já rotuladas pertencentes a duas ou mais classes e deseja-se prever amostras não rotuladas;
  - Regressão: caso onde a saída desejada consiste de uma ou mais variáveis contínuas;
- Aprendizado não-supervisionado: é o tipo de aprendizado em que se é fornecido uma série de vetores sem um valor correspondente. O objetivo neste tipo de problema pode

ser descobrir grupos com amostras similares (*clustering*), redução de dimensionalidade, etc.

Este trabalho consistirá num esforço de classificação e, portanto, haverá uma aprendizagem supervisionada com duas classes, cada qual determinando se uma notícia é falsa ou verdadeira.

No fluxo de classificação, durante a fase de treinamento será feita uma extração de características em cima dos textos. Esses atributos serão passados ao algoritmo de aprendizado de máquina juntamente a um rótulo que explicita a qual classe o texto (de onde estes foram extraídas) pertence. Já para a predição, os textos inseridos devem passar pelo mesmo processo de extração de características e inseridas para que o modelo treinado as consuma, dando como saída a classe predita. Este processo é ilustrado na Figura 1.

Figura 1 – Processo de treinamento e predição para aprendizado supervisionado.



Fonte: Adaptado de Bird, Klein e Loper (2009).

2.4.1 Algoritmos

Aqui são apresentados alguns dos algoritmos de aprendizado de máquina (também chamados de modelos) utilizados no decorrer do trabalho.

2.4.1.1 Regressão Logística

A regressão logística é um modelo de classificação que prevê a probabilidade de ocorrência de um determinado evento y dado um conjunto de variáveis de entrada $X =$

$x_1, x_2, ..., x_n$. Assim, a função pode ser modelada como mostra a Equação 2.1.

$$P(y|X) = \frac{e^{\beta_0+\beta_1 x_1+...+\beta_n x_n}}{1 + e^{\beta_0+\beta_1 x_1+...+\beta_n x_n}} \tag{2.1}$$

A função logística pode ser reescrita e transformada na sua inversa, chamada função logit (Equação 2.2).

$$logit(P(y|X)) = ln\left(\frac{P(y|X)}{1 - P(y|X)}\right) = \beta_0 + \beta_1 x_1 + ... + \beta_n x_n \tag{2.2}$$

Com a transformação logit, o lado direito da equação é linearizado, permitindo que se execute o ajuste mais facilmente (KHANNA; AWAD, 2015).

Os coeficientes $\beta_0, \beta_1, ..., \beta_n$ são estimados de forma que se tenha um modelo que prediz um valor bastante próximo a 1 para uma das classes e 0 para a outra.

2.4.1.2 Naive Bayes

Naive Bayes é um algoritmo de aprendizado de máquina baseado no Teorema de Bayes (Equação 2.3). Segundo o teorema, pode-se prever a probabilidade de um evento y acontecer dado que um evento X aconteceu. O termo *naive* (ingênuo) vem da suposição de que todas as variáveis do problema são independentes e a presença de uma não afeta as demais (GANDHI, 2018).

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)} \tag{2.3}$$

Para o contexto de classificação, pode-se traduzir y como sendo a classe a qual um documento contendo as características $X = x_1, x_2, ..., x_n$ pertence. Isso é mostrado pela Equação 2.4.

$$P(y|x_1, x_2, ..., x_n) = \frac{P(x_1|y)P(x_2|y)...P(x_n|y)P(y)}{P(x_1)P(x_2)...P(x_n)} \tag{2.4}$$

Como o denominador se manterá o mesmo durante todo o problema, pode-se removê-lo e estabelecer uma proporcionalidade, como mostra a Equação 2.5.

$$P(y|x_1, x_2, ..., x_n) \propto P(y) \prod_{i=1}^{n} P(x_i|y) \tag{2.5}$$

Assim, pode-se definir para um problema de K classes que o resultado da classificação será y_r, onde este será a classe para a qual a probabilidade calculada é a maior. Isso pode ser observado na Equação 2.6.

$$y_r = \underset{k \in \{1,2,...,K\}}{argmax}\ P(y_k) \prod_{i=1}^{n} P(x_i|y_k) \tag{2.6}$$

A partir disso, variações do algoritmo *Naive Bayes* surgiram, como o Multinomial, Gaussiano e de Bernoulli. Uma das diferenças entre estes modelos é como $P(x_i|y)$ será calculado (METSIS; ANDROUTSOPOULOS; PALIOURAS, 2006).

Outra diferença é a forma como os dados são inseridos no algoritmo. Num problema de classificação textual, por exemplo, o modelo de Bernoulli requer os dados na forma de um vetor de valores binários, onde cada entrada representa a presença ou não de um termo no texto. Já para o modelo Multinomial, o documento é representado por um vetor cujos valores representam a frequência de um termo no texto (SHIMODAIRA, 2014).

2.4.1.3 Florestas Aleatórias

O modelo de florestas aleatórias é um algoritmo do tipo *ensemble*. Esta classe de algoritmos envolve o uso de diversos modelos em conjunto a fim de fornecer classificações melhores (BROWNLEE, 2016).

A unidade básica deste algoritmo é a árvore de decisão. Uma árvore de decisão consiste em uma árvore binária onde cada nó recebe uma entrada e pode se dividir em duas sub-árvores. A construção da árvore é feita a partir de características aleatórias de um conjunto de dados e a predição é feita de forma que o valor inserido na árvore a atravessa realizando as comparações até chegar nos nós folhas, que contêm as classes do problema.

O treinamento do algoritmo de florestas aleatórias consiste na construção de n árvores de decisão independentes a partir de um conjunto de dados. Usualmente um número $\sqrt{p}$ de características escolhidas aleatoriamente são utilizadas para cada árvore, onde p é a quantidade total de variáveis de entrada. Segundo Koehrsen (2017) o uso de um subconjunto das características aumenta a diversidade na floresta, levando a predições mais robustas.

A predição do modelo de floresta aleatória consiste numa votação, onde uma entrada será passada ao conjunto de árvores de decisão geradas e cada árvore realizará a predição para estes dados. Ao fim deste processo serão contabilizados os resultados de cada árvore e a classe com mais votos será o resultado da inferência.

2.4.1.4 *Support Vector Machine*

Segundo Brownlee (2016), SVM (*Support Vector Machines*) é um dos algoritmos mais populares quando se trata de aprendizado de máquina. Foram extremamente populares quando foram criados na década de 90 e ainda são muito utilizados com um alto desempenho.

O algoritmo SVM tem como objetivo definir um hiperplano que melhor separa o espaço n-dimensional gerado pelas variáveis de entrada, estratificando os dados em dois grupos. A equação geral de um hiperplano em um espaço n-dimensional é dada pela Equação 2.7.

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n = 0 \tag{2.7}$$

Os coeficientes $\beta_0, \beta_1, \beta_2, ..., \beta_n$ representam a inclinação do hiperplano e as variáveis $x_1, x_2, ..., x_n$ são as entradas.

A classificação pode ser feita colocando os valores da variável de entrada na equação e verificando onde o ponto se localiza com relação ao hiperplano, sendo que a classe corresponderá ao agrupamento no qual o ponto estiver contido. O melhor hiperplano considerado é aquele que separa os grupos com a maior margem, sendo esta a distância perpendicular entre o hiperplano e o ponto mais próximo de cada grupo. Este hiperplano é chamado de *Maximal-Margin Hyperplane* (BROWNLEE, 2016).

Na prática, porém, os dados dificilmente conseguem ser separados linearmente. Assim, uma técnica denominada *Soft Margin Classifiers* é utilizada. Nesta, permite-se que alguns dos pontos violem a divisão criada pelo hiperplano, ficando contidos no agrupamento errado.

Ainda assim, existem alguns casos onde o algoritmo não consegue realizar uma separação linear. Para contornar isso, pode-se utilizar um *kernel* para mapear os dados em um espaço dimensional maior, onde estes podem ser separados linearmente. Este espaço é chamado de *kernel space* e seu uso junto ao ajuste de hiperplano é mais vantajoso do que buscar hipersuperfícies que estratifiquem bem o espaço original (KHANNA; AWAD, 2015).

2.4.2 Métricas

Após a etapa de aprendizado, é importante que se faça uma avaliação do algoritmo para testar se seu comportamento está de acordo com o esperado.

Para tal, são necessárias métricas que quantifiquem sua eficácia nas predições. Uma ferramenta bastante utilizada para realizar essas mensurações é a matriz de confusão. A matriz de confusão (Quadro 1) é uma matriz quadrada que compara as classes preditas com as classes reais, utilizando as seguintes nomenclaturas (KOHAVI; PROVOST, 1998):

- *True Positive* (TP): Saída corretamente definida como positiva;

- *True Negative* (TN): Saída corretamente definida como negativa;
- *False Positive* (FP): Saída incorretamente definida como positiva;
- *False Negative* (FN): Saída incorretamente definida como negativa.

Quadro 1 – Matriz de confusão.

Real/Previsto	**Positivo**	**Negativo**
Positivo	TP	FN
Negativo	FP	TN

Fonte: Elaborado pelo autor.

A partir disso, são derivadas medidas que dão uma noção do desempenho do modelo. Algumas delas são (SCIKIT-LEARN, 2019):

- Accuracy $= \frac{TP+TN}{TP+TN+FP+FN}$
- Precision (P) $= \frac{TP}{TP+FP}$
- Recall (R) $= \frac{TP}{TP+FN}$
- F1-score $= \frac{2.P.R}{P+R}$

As métricas acima, em geral, são apresentadas com números entre 0 e 1, podendo variar dependendo da implementação, sendo que quanto maior este valor, melhor seu desempenho.

Uma técnica para avaliar a capacidade de generalização de um modelo treinado é a validação cruzada. Esta técnica prevê a separação de um conjunto de testes para a etapa de treinamento do modelo, de forma que se avalie seu desempenho com um conjunto não visto antes pelo algoritmo (KHANNA; AWAD, 2015).

Na validação cruzada de *k-folds*, o conjunto de dados de treinamento é dividido em k subconjuntos mutuamente exclusivos de tamanhos iguais. Um modelo é então treinado utilizando $k-1$ partições para o treinamento e 1 partição para o teste, calculando, para cada iteração, uma métrica que representa o desempenho em cima deste subconjunto de dados. Isso se repete k vezes, de forma que todas as partições sejam utilizadas para o teste.

2.5 Processamento de Linguagem Natural

Segundo Fernquist (2016), a área de *Natural Language Processing* (NLP) se refere ao uso de Inteligência Artificial e Ciência da Computação para o estudo de línguas humanas naturais, como o português e o inglês.

2.5.1 Representação dos dados

Um dos grandes desafios na área de NLP é a representação dos dados, que são textos. Isso se dá principalmente pela dificuldade de o computador compreender palavras em linguagens naturais.

Para solucionar isso os textos são transformados em uma representação numérica, normalmente levantando probabilidades ou frequências referentes a uma palavra (também chamado de *token*) ou uma sequência de palavras, numa forma compreensível pelos computadores (esta transformação também é conhecido de vetorização) (KANA, 2019).

2.5.1.1 *Bag-of-words*

Esta é uma das representações mais simples e representa cada termo como uma unidade igualmente significativa para a resolução do problema. Nessa abordagem, levantam-se as frequências de uma palavra ou de um N-grama, que representa um conjunto de N palavras seguidas (CONROY; RUBIN; CHEN, 2015).

Apesar de simples e fácil de implementar, este modelo de representação possui duas principais desvantagens (SINGHAL, 2019):

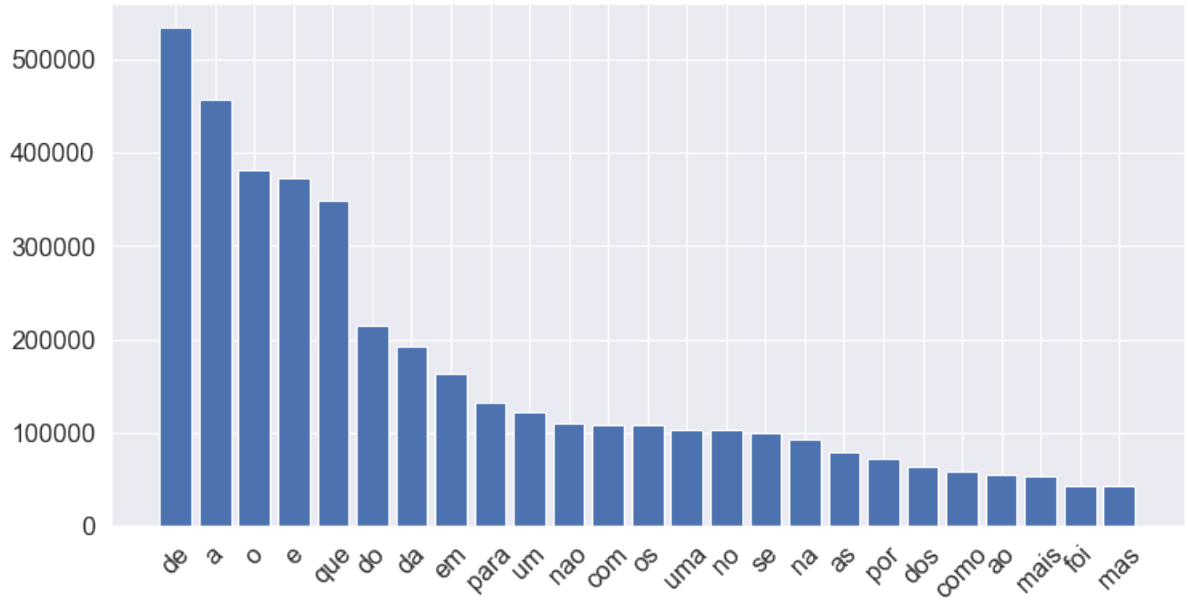
- Ele se preocupa com as ocorrências de uma palavra, e não onde ela está posicionada. Isso faz com que se perca qualquer informação contextual presente nos textos;
- O vocabulário gerado consistirá em um vetor com todos os termos únicos presentes nos dados, o que gera uma representação esparsa na transformação de alguns textos, já que nem todos os termos estarão presentes.

2.5.1.2 *Term Frequency - Inverse Document Frequency* (TF-IDF)

Às vezes utilizar somente a frequência como característica para a transformação não é o ideal (KANA, 2019). A Figura 2 mostra as palavras mais comuns dos textos do conjunto primário (que será abordado mais adiante) com algumas alterações (com todos os caracteres minúsculos, sem acentos e números). Observa-se que as palavras mais ocorrentes são palavras bastante comuns, como "de", "a" e "o", e que normalmente não ajudam na distinção de uma classe para outra.

Assim, a ideia do TF-IDF é aumentar o peso de uma palavra ou N-grama à medida que ela aparece mais dentro de um documento, porém diminuir seu peso conforme ela aparece em mais documentos.

Figura 2 – Frequência das palavras mais comuns nos textos.



Fonte: Elaborada pelo autor.

O *Term Frequency* (TF) é simplesmente a frequência de um termo t dentro de um documento d. Esse valor é dado pela Equação 2.8

$$TF(t,d) = count(t,d) \tag{2.8}$$

Sua implementação pode variar (alterando a forma que este é calculado a fim de normalizar o valor, por exemplo), mas a ideia por trás deve ser a mesma.

Já o *Inverse Document Frequency* (IDF) é calculado para cada *token* t a partir de todo o conjunto de textos como descreve a Equação 2.9.

$$IDF(t) = ln\left(\frac{1+n}{1+DF(t)}\right) + 1 \tag{2.9}$$

onde n é o total de documentos no conjunto de dados e $DF(t)$ é a quantidade de documentos onde o termo t está presente. De forma análoga ao TF, sua implementação pode variar.

O peso final de um termo é dado pelo produto entre TF e IDF.

2.6 REST API

REST é um acrônimo para *Representational State Transfer* e trata-se de uma arquitetura cliente-servidor onde o cliente envia requisições ao servidor, o qual realiza o processamento e

retorna uma resposta (MUMBAIKAR; PADIYA, 2013).

A arquitetura trabalha em cima de recursos, onde cada um é identificado por uma *Uniform Resource Identifier* (URI). Este, por sua vez, é um identificador de recurso uniforme e consiste numa sequência de caracteres que permite o endereçamento de recursos através de uma nomenclatura extensível (BERNERS-LEE; FIELDING; MASINTER, 2005).

A arquitetura REST tem sido bastante utilizada no desenvolvimento *web* e móvel no mercado, sendo um requisito para muitas empresas de tecnologia modernas, seja para o desenvolvimento delas ou para criar aplicações que interagem com elas. (HILLAR, 2016).

Os princípios de desenvolvimento REST são (MUMBAIKAR; PADIYA, 2013):

- Endereçamento: modelos REST operam em recursos endereçados de uma maneira padronizada, cada um sendo identificado por uma URI;
- Ausência de estado: toda transação é independente e não relacionada a transações anteriores. Assim, todos os dados necessários para realizar o processamento devem estar contidos na requisição;
- Interface uniforme: utiliza-se de uma interface padronizada para acessar recursos, como uma série definida de métodos *Hypertext Transfer Protocol* (HTTP).

Segundo Masse (2011) uma *Application Programming Interface* (API) é uma interface de programação de aplicações que disponibiliza uma série funções que facilita a comunicação entre programas e permite a troca de informações. Uma API *web* é a interface de um serviço *web* que recebe requisições e responde à aplicação cliente. Uma API *web* em conformidade com a arquitetura REST é chamada de REST API.

3 Metodologia

Neste capítulo é descrita a estrutura de desenvolvimento de um projeto de aprendizado de máquina, assim como as principais ferramentas utilizadas para tal.

3.1 Etapas

Segundo Guo (2017), um fluxo de desenvolvimento de um modelo de aprendizado de máquina segue as seguintes etapas:

1. Aquisição dos dados: realizar a coleta dos dados que podem ser úteis na resolução do problema, podendo estes ser de uma ou mais fontes;
2. Preparar os dados: aqui os dados serão carregados e pré-processados. A visualização dos dados pode ser de grande ajuda nesta etapa, permitindo a compreensão da relação entre os dados ou mesmo a detecção de fatores que podem ser prejudiciais ao desempenho do modelo, como por exemplo o desbalanceamento, que é a diferença de amostragem entre as classes do problema e que pode enviesar o modelo. Também serão realizados quaisquer pré-processamentos necessários, assim como uma divisão dos dados em um conjunto de treino (utilizado para treinar o algoritmo) e um conjunto de teste (para testar e avaliar o algoritmo);
3. Escolher o modelo: escolher o algoritmo que será utilizado para treinamento e teste;
4. Treinamento: nesta etapa será realizado o aprendizado em si. Num problema de classificação, o algoritmo será exposto às amostras de treino e associará características dos dados às classes apresentadas;
5. Avaliação: serão realizados testes com o conjunto anteriormente separado. Nesta etapa pode-se visualizar o comportamento do algoritmo quando exposto a dados não vistos previamente. Usualmente, cerca de 20% dos dados são utilizados para teste;
6. Ajuste de hiperparâmetros: esta fase consiste no ajuste de parâmetros do modelo e esta regulagem consiste principalmente numa experimentação, não há um conjunto de passos bem definidos para tal, já que geralmente depende dos dados utilizados;
7. Predições: o último passo é a disponibilização do modelo para realizar as predições.

3.2 Ferramentas

3.2.1 Scrapy

O Scrapy é um *framework* de *web crawling* e *web scraping* de alto nível, utilizado para navegar através de páginas web e extrair dados estruturados delas (SCRAPY, 2019).

3.2.2 Jupyter Notebook

O Jupyter Notebook é uma plataforma que possibilita o desenvolvimento, a documentação, a execução de códigos e a visualização de resultados num ambiente interativo baseado em *web* (JUPYTER NOTEBOOK, 2019).

A ferramenta permite que o usuário edite códigos diretamente pelo *browser*, além de proporcionar uma fácil verificação dos resultados atrelados à execução no mesmo. Outra característica é a documentação facilitada dentro do ambiente, pois este também aceita linguagem *Markdown*, a qual realiza a formatação de textos a partir de uma sintaxe, não se limitando a apenas texto puro.

3.2.3 NumPy

NumPy é uma biblioteca Python dedicada a computação científica, implementando poderosas estruturas para manipulação de dados. O pacote contém, dentre outras funcionalidades (NUMPY, 2019):

- *Arrays* N-dimensionais;
- Funções de *broadcasting* (como trata *arrays* de diferentes formas em operações aritméticas);
- Operações de álgebra linear.

Bressert (2012) cita também uma melhoria significativa ao utilizar-se da biblioteca em termos de tempo de processamento em comparação a uma operação análoga escrita puramente em Python (executando aproximadamente 25 vezes mais rápido).

Ainda que não se utilize o NumPy de forma explícita, este é usado em diversas outras bibliotecas de análise de dados, como o Pandas, que é construído em cima do pacote e possui diversas funções que preveem a integração entre as duas (MCKINNEY, 2012).

3.2.4 Pandas

Pandas é um pacote Python que fornece estruturas de dados rápidas, flexíveis e expressivas para tornar o trabalho com dados "relacionais" ou "rotulados" fácil e intuitivo (PANDAS, 2019).

Para McKinney (2011) o pacote foi criado com o intuito de preencher uma lacuna existente entre a linguagem Python e outras plataformas de computação estatística específicas no que tange a disponibilidade de ferramentas de análise de dados.

A ferramenta possui duas estruturas de dados principais que são *Series* e *DataFrames* que podem ser facilmente manipuladas de diversas maneiras. Algumas das funcionalidades do pacote incluem (PANDAS, 2019):

- Carregar e salvar a partir de diversos formatos de arquivo;
- Manipular dados nulos;
- Adicionar e remover colunas;
- Unir e redimensionar os conjuntos de dados;
- Operações semelhantes a operações realizadas em bancos de dados relacionais, como por exemplo *group by*.

3.2.5 Matplotlib

A visualização dos dados é uma parte importante para o entendimento dos mesmos, já que, visualmente, pode-se compreender suas distribuições, realizar comparações, explicitar relações entre variáveis, entre outras análises.

O Matplotlib é uma ferramenta que pode auxiliar grandemente esta etapa de análise. Trata-se de uma biblioteca de plotagem de imagens 2D baseada em Python que fornece diversas funções que permitem, em poucas linhas de código, a criação de gráficos de barra, histogramas, gráficos de dispersão, entre outros. O pacote busca simplificar o que é fácil e tornar possível o que é difícil (MATPLOTLIB, 2019).

3.2.6 Seaborn

O Seaborn é uma biblioteca para plotagem de gráficos estatísticos em Python. Foi construída em cima da biblioteca Matplotlib e está fortemente atrelada às estruturas de dados do pacote pandas (SEABORN, 2019).

Muitas tarefas podem ser realizadas apenas utilizando funções do pacote, mas algumas personalizações podem requerer o uso da biblioteca Matplotlib.

3.2.7 Scikit Learn

O Scikit-learn fornece a implementação de diversos algoritmos de aprendizado de máquina no estado da arte, mantendo uma interface fácil de se utilizar integrada à linguagem Python (PEDREGOSA et al., 2011).

No pacote, o objeto central é chamado *estimator*, que implementa um método *fit* para treinamento do modelo e um método *predict* para realizar a inferência a partir de um treinamento prévio (SCIKIT-LEARN, 2019). Isso demonstra o uso de um conjunto de convenções a nível de código para fornecer uma interface consistente (PEDREGOSA et al., 2011).

Além de algoritmos de aprendizado de máquina, o Scikit-learn também provê uma série de métodos auxiliares, como algoritmos para extração de atributos, avaliação e seleção de modelos, dentre outros.

3.2.8 Flask

Flask é um *micro-framework* escrito em Python e voltado ao desenvolvimento web. Possui um núcleo bastante simples, mas extensível para que possa agregar funcionalidades mais complexas a partir de outras bibliotecas (FLASK, 2019).

A ideia deste projeto é disponibilizar uma API REST de forma que esta possa ser consumida por uma aplicação móvel. O Flask-RESTful é uma extensão do Flask que permite a construção de uma API REST a partir do mesmo.

3.2.9 Ionic

O Ionic é um *framework* de desenvolvimento de aplicações *mobile* híbrido, que permite que o desenvolvimento e a aprendizagem sejam feitas de forma rápida, já que se utiliza de tecnologias *web* como *Hypertext Markup Language* (HTML), *Cascading Style Sheets* (CSS) e JavaScript (IONIC, 2019).

Além disso, o *framework* possui uma ampla variedade de componentes prontos que podem ser facilmente inseridos na aplicação e utilizados, permitindo que o desenvolvedor se foque nas funcionalidades do aplicativo de fato.

4 Desenvolvimento

Aqui são detalhados os passos de desenvolvimento seguidos para a realização do projeto.

4.1 Aquisição dos dados

Nesta primeira etapa foram realizadas duas abordagens para a coleta de dados. A primeira consistiu no desenvolvimento de *web crawlers* para adquirir dados de três sites de notícias. Este primeiro conjunto será referenciado como "conjunto de dados primário" no decorrer deste trabalho. A segunda foi a aquisição de um conjunto de dados contendo notícias verdadeiras e falsas levantadas pela Universidade de São Paulo (USP) chamado "Fake.Br Corpus".

4.1.1 Conjunto de dados primário

Nesta primeira etapa foram definidos sites de onde foram extraídos notícias verdadeiras e notícias falsas. Para as notícias verdadeiras, foram escolhidos as seções de política dos sites G1 e El País e, para a extração das notícias falsas, foi escolhido a seção de política do site boatos.org.

Aqui foi utilizada o pacote Python Scrapy para cada um dos sites, onde foram extraídos todas as notícias relacionadas a política existentes até a data de extração. Para cada site foi criado um arquivo no formato *JavaScript Object Notation* (JSON), armazenando os seguintes campos das notícias:

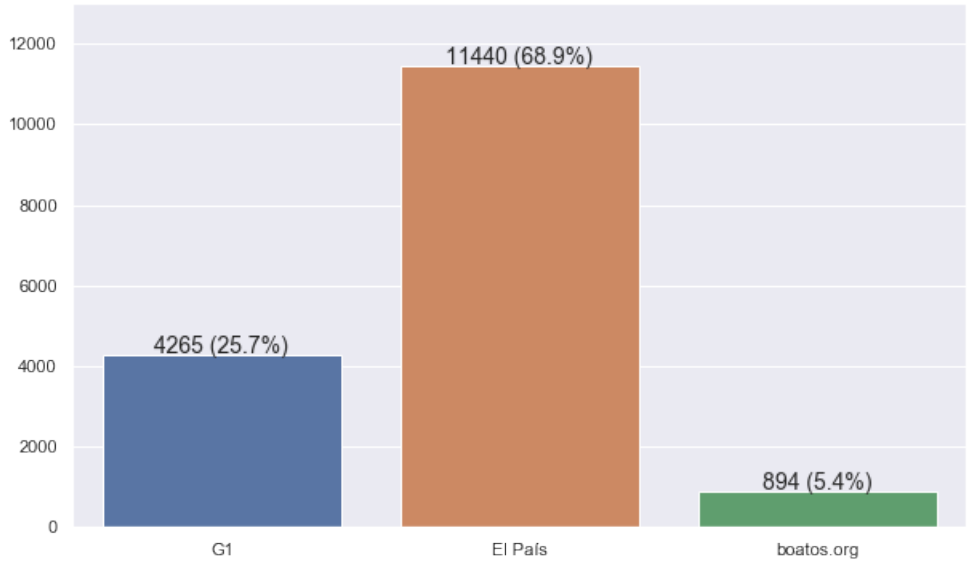
- *title*: título da notícia;
- *date*: data de publicação;
- *text*: corpo da notícia.

Ao todo, foram extraídas 17327 notícias, sendo elas distribuídas, de acordo com suas fontes, conforme indica a Figura 3.

4.1.2 Fake.Br corpus

O conjunto de dados Fake.Br corpus é composto por notícias reais e falsas escritas em português levantadas, rotuladas e disponibilizadas pela USP.

Figura 3 – Distribuição de notícias do conjunto primário com relação à fonte.



Fonte: Elaborada pelo autor.

Os dados estão disponíveis no *Github* (SANTOS, 2018) e contêm assuntos de 6 categorias: política, celebridades e TV, sociedade e dia a dia, ciência e tecnologia, economia e religião.

Neste trabalho o foco são as notícias de cunho político, assim, estas serão as únicas aproveitadas deste conjunto de dados.

4.2 Análise e preparação dos dados

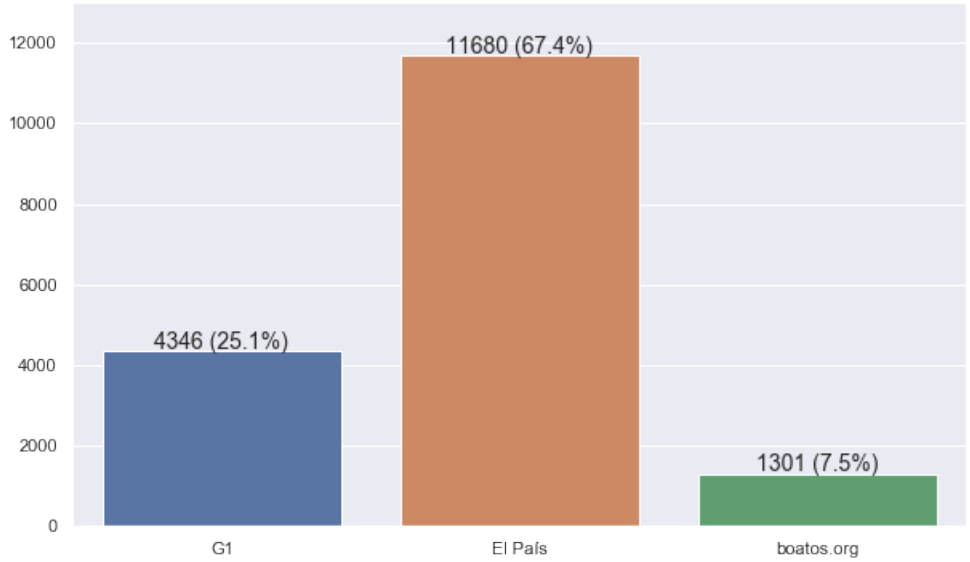
Esta etapa consistiu em análises dos dados utilizados assim como a execução de métodos para transformar os dados num formato adequado para ser utilizado pelo algoritmo de aprendizado de máquina.

4.2.1 Análise do conjunto de dados primário

Num primeiro momento de análise, notou-se que algumas das notícias levantadas não possuíam um texto, muitas vezes por se tratarem apenas imagens ou vídeos. Assim, viu-se a necessidade de eliminar tais entradas. Após este corte inicial, foram eliminadas 407 notícias do site boatos.org, 240 do El País e 81 do G1. A distribuição das notícias com relação à fonte após o corte é apresentada pela Figura 4.

Após isso, realizou-se uma análise do balanceamento dos dados adquiridos dentro do conjunto primário, verificando como estão distribuídas as amostras coletadas com relação a cada classe do problema. A Figura 5 ilustra a quantidade de notícias adquiridas para cada classe

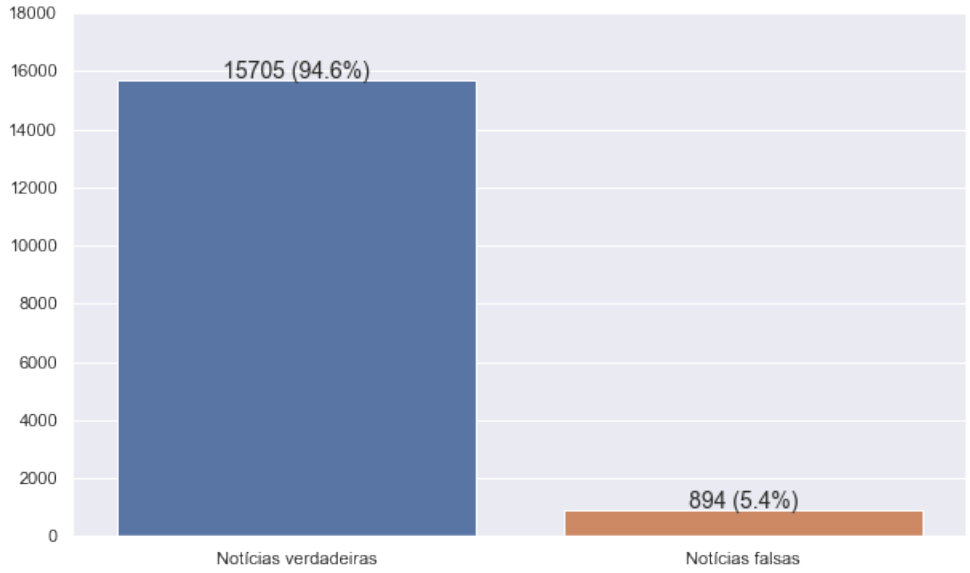
Figura 4 – Distribuição de notícias do conjunto primário com relação à fonte após eliminar entradas sem texto.



Fonte: Elaborada pelo autor.

e mostra um grande desbalanceamento dentro deste conjunto, já que existem aproximadamente 17 vezes mais notícias verdadeiras do que falsas. Isso pode acarretar no enviesamento do modelo, já que existe muito mais amostras de uma classe do que de outra.

Figura 5 – Distribuição de classes no conjunto primário.

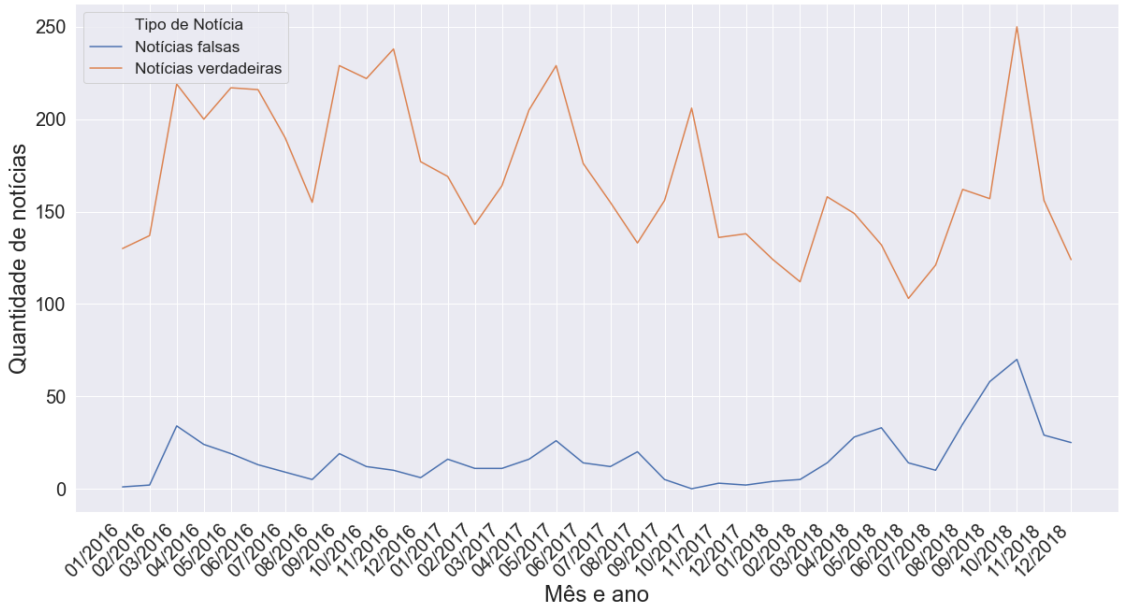


Fonte: Elaborada pelo autor.

Outro ponto observado foi a distribuição das notícias com relação à data de sua publicação. Optou-se por realizar um recorte no conjunto de dados, de forma a pegar apenas notícias datando entre 01/01/2016 e 01/12/2018. Com este recorte, objetiva-se ter um conjunto

reduzido de notícias que tratam de assuntos similares. As notícias, dentro deste período se distribuíram como mostra a Figura 6.

Figura 6 – Quantidade de amostras no conjunto primário por classe ao longo do tempo.



Fonte: Elaborada pelo autor.

Deste recorte, sobraram 6088 amostras verdadeiras e 615 falsas.

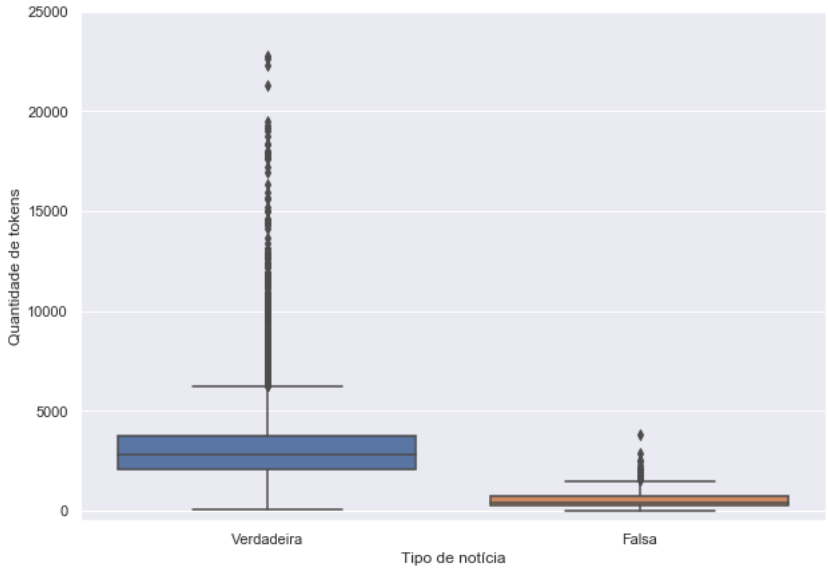
A Figura 7 apresenta um diagrama de caixa com o tamanho dos textos em quantidade de palavras. É notável a diferença entre as duas classes. Analogamente ao desbalanceamento de documentos, uma discrepância no tamanho dos textos nessa escala pode ser prejudicial para o treinamento do algoritmo de classificação, pois o processo de vetorização é realizado a nível de palavra e a diferença pode acabar enviesando o modelo também.

Com o intuito de parear o tamanho dos textos, realizou-se outro recorte. Para tal, optou-se por ordenar as entradas de notícias verdadeiras por tamanho e selecionar as 615 primeiras notícias com mais de 20 palavras. Deste processo, resultou um conjunto com 1230 notícias e com uma distribuição de tamanho ilustrada pela Figura 8.

Ainda existe um certo desbalanceamento, porém é visível que este foi bastante aliviado com relação ao que se havia antes.

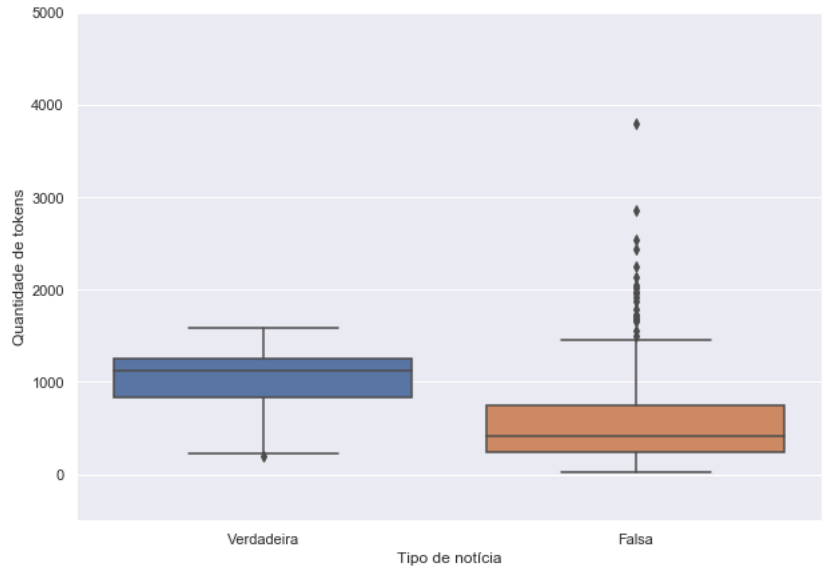
A partir desses dados foi feito um levantamento das palavras que mais ocorrem dentro de textos de cada classe. Em casos onde se utiliza a ocorrência de alguns termos como variáveis de entrada para o treinamento do modelo, essa análise pode se mostrar interessante para embasar seu uso. A Figura 9 mostra esses números.

Figura 7 – Diagrama de caixa do tamanho dos textos do conjunto primário em número de palavras.



Fonte: Elaborada pelo autor.

Figura 8 – Diagrama de caixa do tamanho dos textos do conjunto primário em número de palavras após balanceamento.



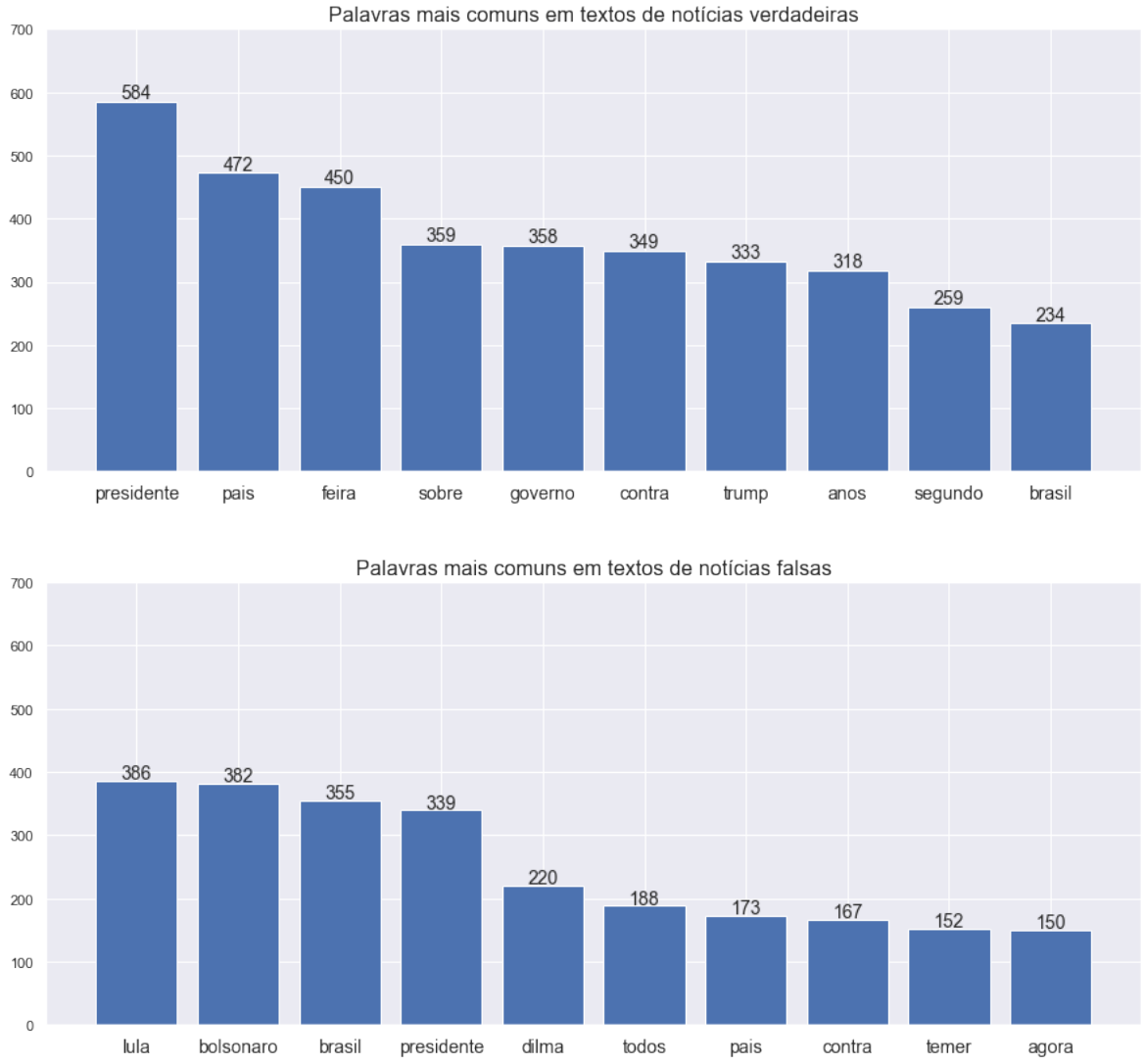
Fonte: Elaborada pelo autor.

4.2.2 Análise do conjunto de dados Fake.Br corpus

Segundo Monteiro et al. (2018), o conjunto de dados Fake.Br corpus foi montado cuidadosamente, buscando, para cada notícia falsa, levantar uma notícia verdadeira tratando do mesmo assunto ou até explicitamente negando a mesma.

Para a construção deste conjunto de dados foram coletadas 7200 notícias (3600

Figura 9 – Palavras mais comuns em textos do conjunto primário para cada classe.



Fonte: Elaborada pelo autor.

verdadeiras e 3600 falsas) que foram publicadas entre janeiro de 2016 e janeiro de 2018. As notícias falsas foram recolhidas de 4 sites distintos: Diário do Brasil, A Folha do Brasil, The Jornal Brasil e Top Five TV (MONTEIRO et al., 2018).

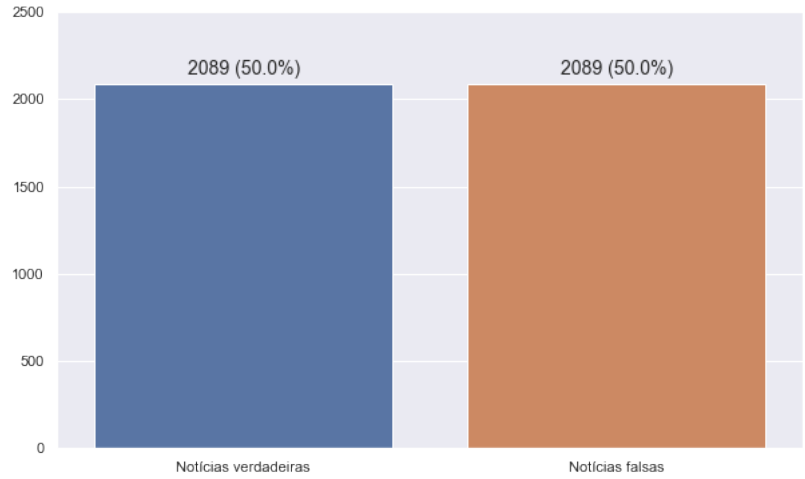
As notícias levantadas foram divididas em 6 categorias: política, celebridades e TV, sociedade e notícias diárias, ciência e tecnologia,economia e religião.

Para este trabalho foram utilizadas apenas as notícias do conjunto de dados referentes à política, totalizando 4178 notícias (58% do conjunto).

Como dito, os dados foram coletados de forma a haver uma notícia verdadeira para cada notícia falsa, assim, verifica-se que os dados estão balanceados, como é apresentado pela Figura 10.

Assim como no conjunto primário, é importante verificar como estão os tamanhos dos

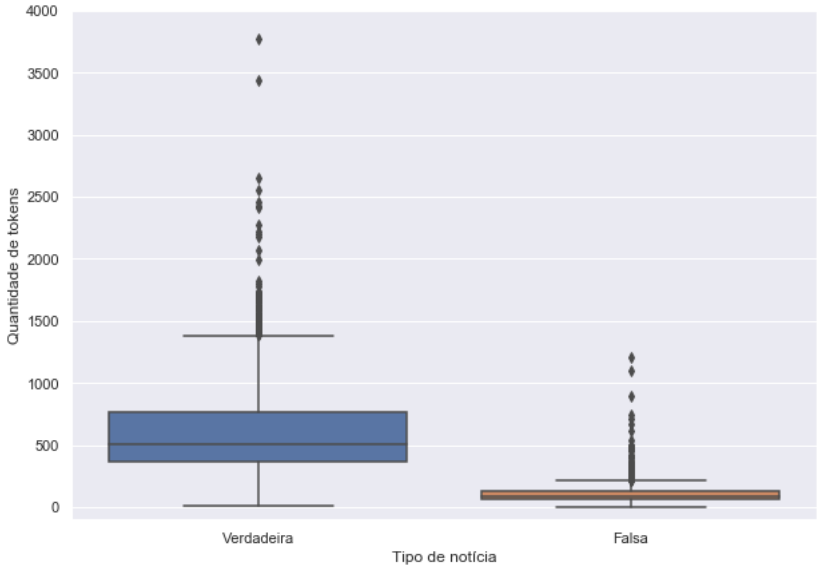
Figura 10 – Distribuição de classes no conjunto Fake.Br corpus.



Fonte: Elaborada pelo autor.

textos com relação a seus *tokens*. Esta distribuição é apresentada pela Figura 11.

Figura 11 – Diagrama de caixa do tamanho dos textos do conjunto Fake.Br corpus em número de palavras.

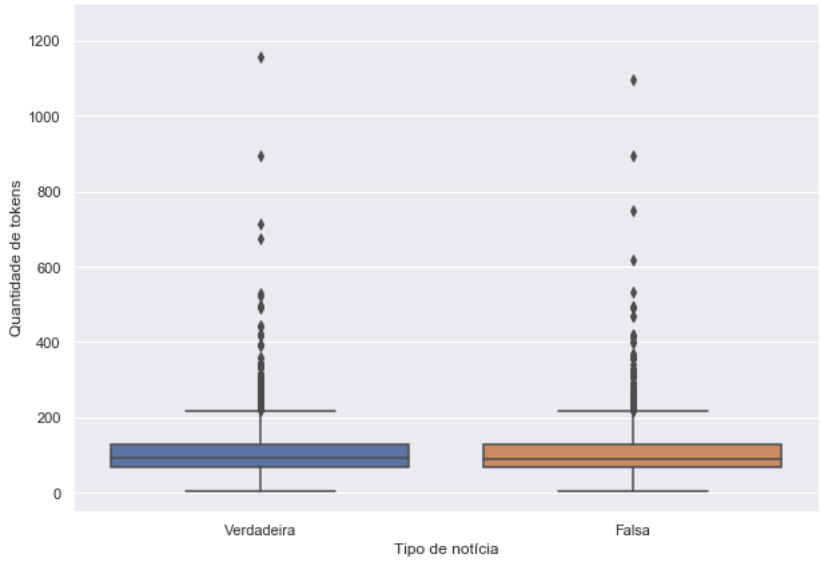


Fonte: Elaborada pelo autor.

Pode-se perceber que as notícias verdadeiras possuem um tamanho de texto significativamente maior do que as notícias falsas, assim como se observou no conjunto primário. Buscando contornar o problema de tamanhos de texto discrepantes, o conjunto de dados Fake.Br corpus possui também uma versão com os textos truncados (em número de palavras) de forma a ficarem com os tamanhos alinhados com suas contrapartes (MONTEIRO et al., 2018).

A Figura 12 mostra um diagrama de caixa com o tamanho dos textos normalizados em quantidade de palavras.

Figura 12 – Diagrama de caixa do tamanho dos textos do conjunto Fake.Br corpus em número de palavras após balanceamento.



Fonte: Elaborada pelo autor.

Finalmente, de forma similar ao conjunto de dados primário, realizou-se um levantamento das palavras mais comuns. O resultado é apresentado na Figura 13.

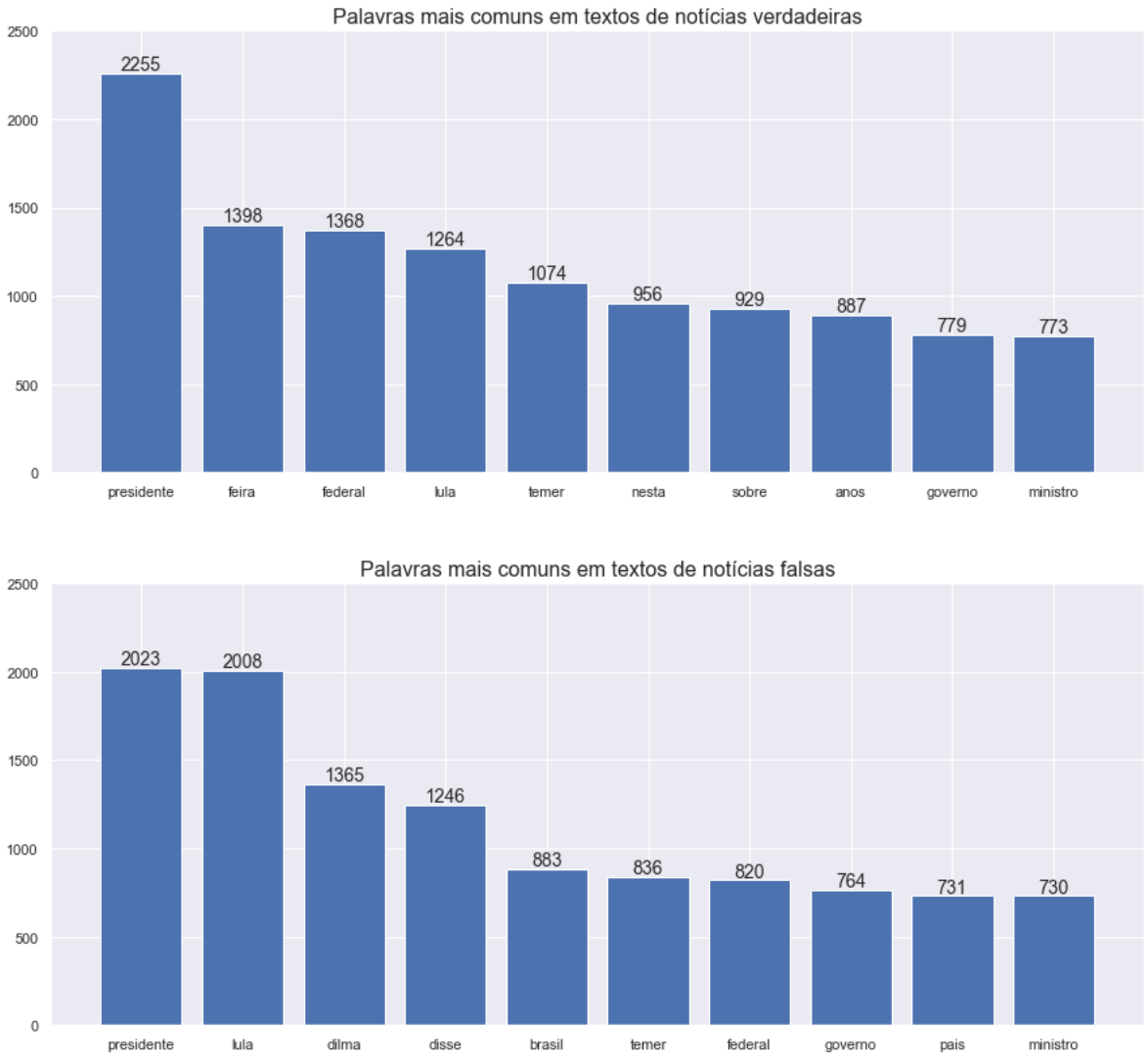
4.2.3 Pré-processamento

Dentro do contexto de processamento de linguagem natural, o pré-processamento (ou limpeza) dos textos é uma etapa fundamental. Aqui é necessário que haja uma padronização dos *tokens* dentro dos textos para que todos sejam interpretados corretamente. A ausência de um pré-processamento pode resultar, por exemplo, na interpretação incorreta de duas palavras que são iguais mas que se diferenciam por uma letra maiúscula, gerando duas entradas separadas para o que deveria ser um só *token*.

Nesta etapa foi realizado um pré-processamento que foi executado sobre todos os textos. Esta consistiu em alguns procedimentos comuns dentro da área de NLP:

- Transformar todos os caracteres em minúsculos;
- Realizar o *stemming* (reduzir palavras flexionadas à sua raíz) das palavras;
- Remover acentos;
- Remover caracteres não alfabéticos;

Figura 13 – Palavras mais comuns em textos do conjunto Fake.Br corpus para cada classe.



Fonte: Elaborada pelo autor.

- Remover espaços extras;
- Remover palavras com 2 ou menos caracteres;
- Remover *stopwords* (palavras "vazias", que são muito comuns na linguagem e não agregam valor à resolução do problema).

Dois outros procedimentos menos comuns foram implementados buscando garantir uma maior coerência nos textos:

- Remover *tokens* que possuem intercalações entre letras e números;
- Remover *tokens* que repetem um caractere mais de duas vezes seguidamente.

Além disso, foram aplicados algoritmos de vetorização nos textos. Estes algoritmos transformam textos em representações que podem ser compreendidos pelo computador.

Para este trabalho foram utilizados os algoritmos CountVectorizer e TfidfVectorizer da biblioteca Scikit-Learn, que transformam os textos nas representações *bag-of-words* e TF-IDF respectivamente. Optou-se por limitar o número de características dos algoritmos de vetorização a 10000 para não tornar o treinamento muito custoso computacionalmente, além de configurá-los para considerar palavras individuais e 2-gramas.

Aqui os conjuntos foram divididos utilizando o método train_test_split implementado pelo Scikit-Learn, de forma a dividi-los em partições de 70% para o treino e 30% para teste.

Este pré-processamento foi executado para ambos os conjuntos de dados.

4.3 Escolha do modelo

Para este trabalho foram utilizados 4 modelos implementados pela biblioteca Scikit-Learn:

- LogisticRegression (Regressão Logística);
- RandomForestClassifier (Florestas Aleatórias);
- GaussianNB (*Naive Bayes*);
- SVC com *kernel* linear (SVM).

4.4 Treinamento

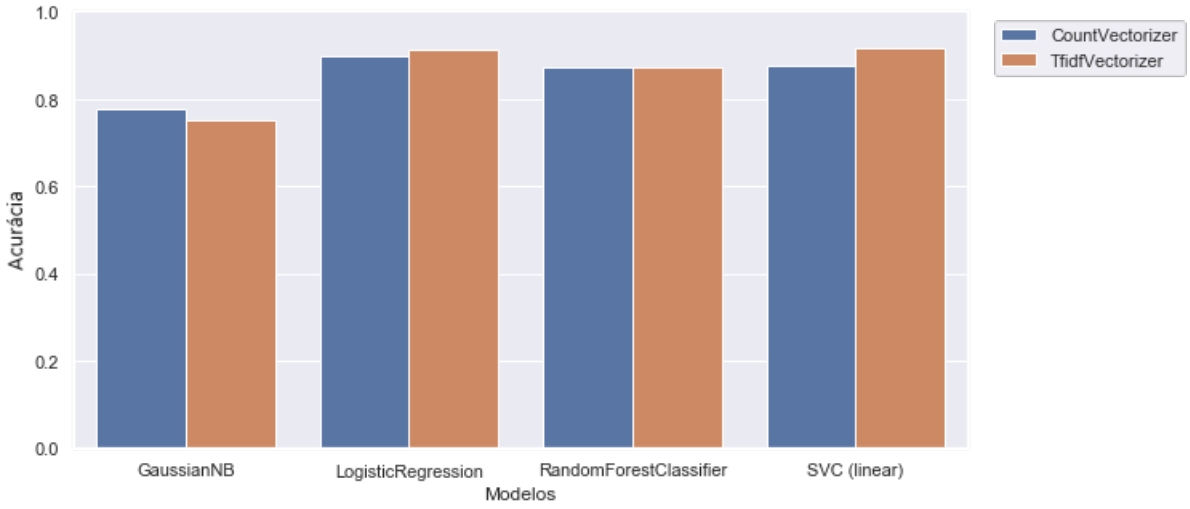
Inicialmente, resolveu-se executar uma validação cruzada de *k-folds* para verificar o comportamento de cada modelo para cada vetorização. Para tal, com cada conjunto de dados utilizou-se a função de avaliação por validação cruzada da biblioteca Scikit-Learn, $k = 5$ e usando como métrica o F1-score médio dos 5 *folds*.

As Figuras 14 e 15 apresentam os resultados das validações cruzadas para os conjuntos primário e Fake.Br corpus respectivamente.

Para a etapa de treinamento foram utilizados 70% dos dados de cada conjunto, aplicando os textos pré-processados aos modelos supracitados.

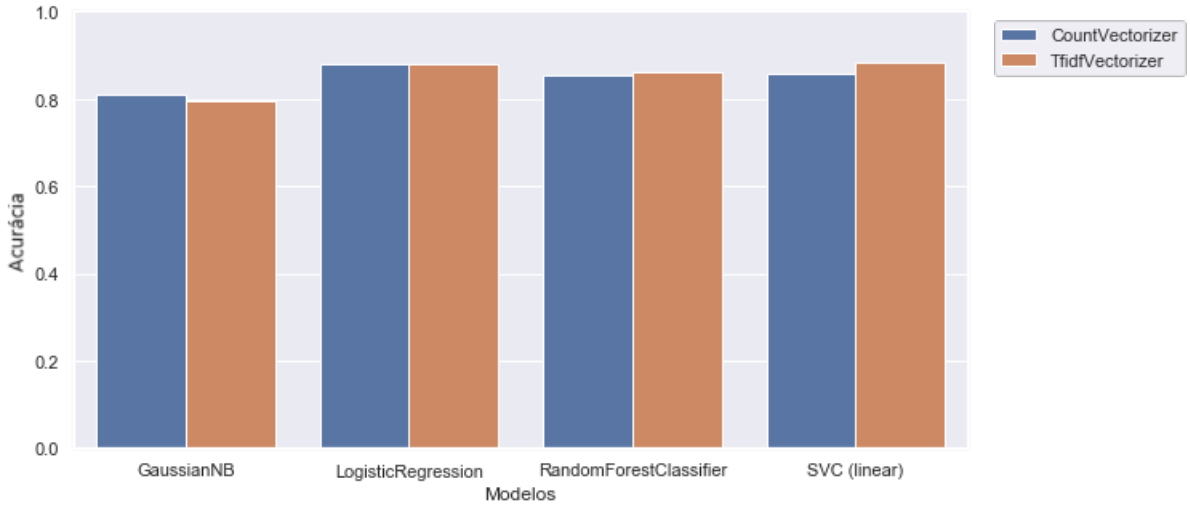
Ao fim desta fase, foram gerados 8 modelos para cada conjunto de dados, um para cada combinação de vetorizador e modelo.

Figura 14 – Validação cruzada *k-fold* para o conjunto primário.



Fonte: Elaborada pelo autor.

Figura 15 – Validação cruzada *k-fold* para o conjunto Fake.Br corpus.



Fonte: Elaborada pelo autor.

4.5 Teste e resultados

Para a etapa de teste, utilizou-se os dados não utilizados na etapa de treinamento, ou seja, o equivalente a 30% dos conjuntos originais.

Num primeiro momento, os modelos treinados com os dados do conjunto primário foram testados com as partições de treino do conjunto primário e os modelos treinados com os dados do conjunto Fake.Br corpus com as partições de treino do conjunto Fake.Br corpus.

As Tabelas 1 e 2 mostram os resultados obtidos a partir de cada vetorizador com cada modelo nos conjuntos de dados primário e Fake.Br corpus, respectivamente.

Tabela 1 – Resultados dos testes dos modelos treinados com o conjunto primário e partição de teste do mesmo.

Vetorizador	Modelo	Acurácia
CountVectorizer	GaussianNB	0.799458
CountVectorizer	**LogisticRegression**	**0.915989**
CountVectorizer	RandomForestClassifier	0.888889
CountVectorizer	SVC (kernel linear)	0.905149
TfidfVectorizer	GaussianNB	0.769648
TfidfVectorizer	LogisticRegression	0.899729
TfidfVectorizer	RandomForestClassifier	0.886179
TfidfVectorizer	SVC (kernel linear)	0.891599

Fonte: Elaborada pelo autor.

Tabela 2 – Resultados dos testes dos modelos treinados com o conjunto Fake.Br corpus e partição de teste do mesmo.

Vetorizador	Modelo	Acurácia
CountVectorizer	GaussianNB	0.802233
CountVectorizer	LogisticRegression	0.880383
CountVectorizer	RandomForestClassifier	0.860447
CountVectorizer	SVC (kernel linear)	0.870813
TfidfVectorizer	GaussianNB	0.794258
TfidfVectorizer	**LogisticRegression**	**0.881180**
TfidfVectorizer	RandomForestClassifier	0.855662
TfidfVectorizer	SVC (kernel linear)	0.855662

Fonte: Elaborada pelo autor.

Observa-se que o modelo de regressão logística obteve o melhor desempenho em ambos os conjuntos, porém, no conjunto primário teve melhor acurácia quando pareado à representação *bag-of-words*, alcançando 0.915989 de acurácia, e no conjunto Fake.Br corpus performou melhor junto à vetorização TF-IDF, com 0.881180 de acurácia.

Apesar disso, buscando comparar os modelos gerados com cada conjunto de dados de forma mais justa, resolveu-se criar um lote de dados a partir das partições de teste de cada grupo. Este lote possuía 1623 entradas, sendo 369 do conjunto primário e 1254 do conjunto Fake.Br corpus. No intuito de igualar a quantidade de amostras de cada conjunto, as 1254 do conjunto Fake.Br corpus foram randomicamente removidas até haver 369 amostras. Este conjunto com 738 amostras será referido como partição de teste geral.

Os modelos então foram testados com esse conjunto. Os resultados para os modelos treinados com os conjuntos primário e Fake.Br corpus são apresentados nas Tabelas 3 e 4 respectivamente.

A partir das tabelas com os resultados, verifica-se que o conjunto de dados Fake.Br

Tabela 3 – Resultados dos testes dos modelos treinados com o conjunto primário e partição de teste geral.

Vetorizador	Modelo	Acurácia
CountVectorizer	GaussianNB	0.714092
CountVectorizer	LogisticRegression	0.800813
CountVectorizer	RandomForestClassifier	0.785908
CountVectorizer	SVC (kernel linear)	0.791328
TfidfVectorizer	GaussianNB	0.708672
TfidfVectorizer	**LogisticRegression**	**0.810298**
TfidfVectorizer	RandomForestClassifier	0.791328
TfidfVectorizer	SVC (kernel linear)	0.798103

Fonte: Elaborada pelo autor.

Tabela 4 – Resultados dos testes dos modelos treinados com o conjunto Fake.Br corpus e partição de teste geral.

Vetorizador	Modelo	Acurácia
CountVectorizer	GaussianNB	0.742547
CountVectorizer	LogisticRegression	0.838753
CountVectorizer	RandomForestClassifier	0.846883
CountVectorizer	SVC (kernel linear)	0.822493
TfidfVectorizer	GaussianNB	0.726287
TfidfVectorizer	**LogisticRegression**	**0.861789**
TfidfVectorizer	RandomForestClassifier	0.837398
TfidfVectorizer	SVC (kernel linear)	0.833333

Fonte: Elaborada pelo autor.

corpus obteve a maior acurácia sobre o conjunto de testes geral, alcançando 0.861789 de acurácia, enquanto que a melhor acurácia atingida com os modelos treinados a partir do conjunto primário foi de 0.810298. Também é observável, em ambos os conjuntos, que a representação com TF-IDF e o modelo de regressão logística foram os que melhor performaram com uma margem razoável dos demais. Portanto, buscando verificar se este resultado pode ser melhorado, o conjunto e o modelo foram levados à etapa de otimização de hiper-parâmetros.

4.6 Otimização de hiper-parâmetros

Esta fase não possui uma sequência de passos bem definida, consistindo principalmente na experimentação de parâmetros. Assim, buscou-se realizar uma busca exaustiva sobre um conjunto de parâmetros do modelo de regressão logística implementado pelo Scikit-Learn.

A própria biblioteca Scikit-Learn fornece um método chamado *Grid Search*, o qual recebe um dicionário com uma série de parâmetros que serão experimentados em cima de um

estimador fornecido.

Essa experimentação foi realizada com os dados do conjunto Fake.Br corpus em cima do estimador LogisticRegression, variando os seguintes parâmetros:

- C: 0.001, 0.01, 0.1, 1, 10, 100, 1000;
- solver: lbfgs, saga;
- tol: 0.000001, 0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100;

onde:

- C é um parâmetro de regularização que determinará a complexidade do modelo;
- solver corresponde ao algoritmo que será utilizado na otimização do problema;
- tol é a tolerância para o critério de parada.

O conjunto de hiper-parâmetros resultante da execução do *grid search* é apresentado na Tabela 5.

Tabela 5 – Melhor parametrização apontada pelo *grid search*.

Hiper-parâmetro	**Valor**
C	100
solver	saga
tol	0.0001

Fonte: Elaborada pelo autor.

Realizou-se mais um treinamento com a partição de treino do conjunto Fake.Br corpus e os componentes TfidfVectorizer e LogisticRegression, com este último ajustado aos hiper-parâmetros encontrados com o *grid search*. A partir disso, foi feito um último teste com o conjunto de teste geral.

Com esse teste, foi obtida uma acurácia de 0.876693. A matriz de confusão gerada por esse teste é apresentada no Quadro 2.

Quadro 2 – Matriz de confusão com os resultados da classificação.

Real/Previsto	**Notícia falsa**	**Notícia verdadeira**
Notícia falsa	326	45
Notícia verdadeira	46	321

Fonte: Elaborado pelo autor.

Para finalizar essa etapa, criou-se um objeto da biblioteca Scikit-Learn chamado *Pipeline*. Este descreve uma sequência de transformações que serão aplicadas sequencialmente nos dados de entrada e que possuem um estimador no final. O *Pipeline* foi criado com o transformador de dados TfidfVectorizer e o estimador LogisticRegression com os hiper-parâmetros ajustados.

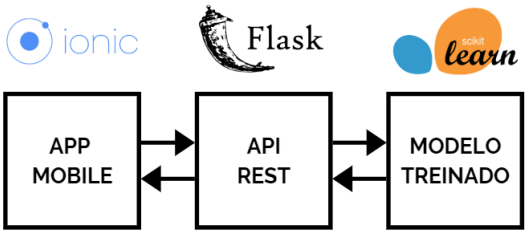
Esse *Pipeline* foi então treinado com todo o conjunto de dados do conjunto Fake.Br corpus e exportado com auxílio da biblioteca Python Pickle. Este arquivo será utilizado na próxima etapa.

4.7 Disponibilização do modelo

Esta última fase consistiu na definição da arquitetura de disponibilização do modelo para que seja consumido por um aplicativo de celular.

Para tal, optou-se por criar uma REST API que recebe requisições da aplicação móvel. Essa API fará uso do modelo treinado para realizar a classificação da notícia. Os resultados serão retornados à aplicação e mostrados ao usuário. A Figura 16 mostra, de forma simplificada, como é realizado esse fluxo de processamento, assim como as principais tecnologias envolvidas na criação de cada parte.

Figura 16 – Arquitetura de disponibilização do modelo.



Fonte: Elaborada pelo autor.

4.7.1 Desenvolvimento da API REST

A API foi desenvolvida utilizando o *micro-framework* Flask. Foi criado um *endpoint* do tipo POST com a rota "*/classification*" que recebe um JSON contendo o campo "*text*", que corresponde ao texto da notícia a ser verificada.

A API faz o consumo do modelo escolhido, o qual é carregado a partir de um arquivo serializado.

Para realizar a predição, serão feitas a limpeza e a vetorização do texto que em seguida será enviado ao modelo para que seja classificado. O resultado é então retornado num formato JSON juntamente à probabilidade associada à predição do algoritmo de classificação.

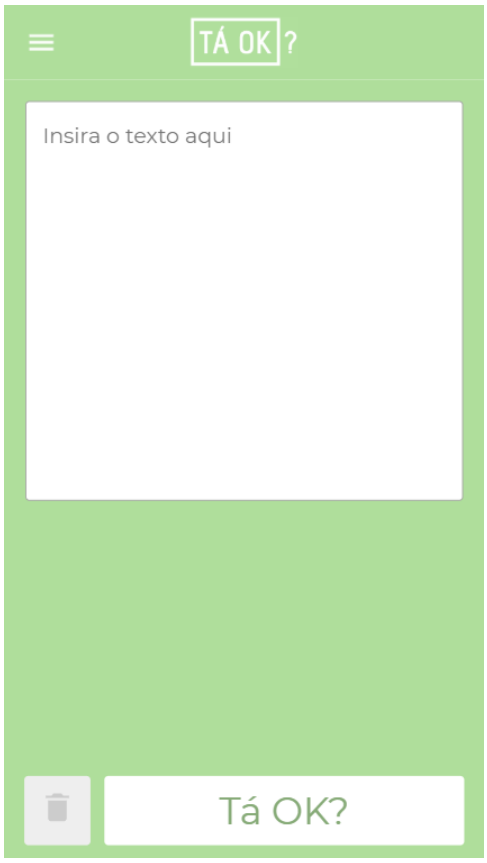
Finalmente, esta API foi disponibilizada na plataforma PythonAnywhere, que é um ambiente de desenvolvimento e hospedagem baseado em Python (PYTHON ANYWHERE, 2019).

4.7.2 Desenvolvimento da aplicação

A aplicação móvel que consome o modelo treinado a partir da API foi chamada de "TÁ OK?" e foi desenvolvida utilizando o *framework* de desenvolvimento híbrido Ionic.

O aplicativo consiste em uma tela com uma área de texto onde o usuário pode inserir uma notícia de cunho político contendo pelo menos 20 palavras, um botão para limpar a área de texto e um botão para realizar a verificação da mesma. A tela principal do aplicativo é mostrada na Figura 17.

Figura 17 – Imagem da página principal do aplicativo.

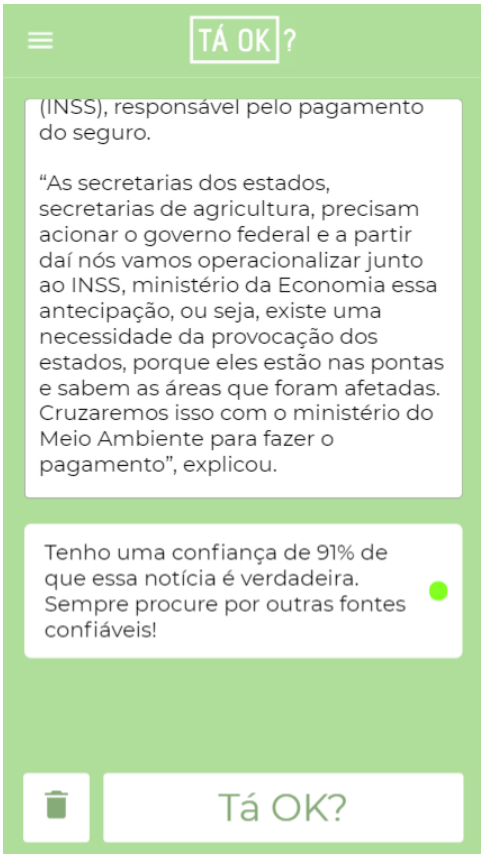


Fonte: Elaborada pelo autor.

A aplicação fará uma requisição à API hospedada na plataforma PythonAnywhere, a qual realizará o processamento da notícia. O resultado será retornado ao aplicativo, onde a predição e a confiança associada serão formatadas dentro de uma frase randomicamente escolhida de um conjunto de frases pré-definidas. A frase então será exibida ao usuário. A

Figura 18 mostra um exemplo de resposta referente a uma notícia inserida pelo usuário e enviada à API para o processamento.

Figura 18 – Imagem do aplicativo após retorno da predição.



Fonte: Elaborada pelo autor.

5 Conclusão

No decorrer deste trabalho, desenvolveu-se um fluxo de ciência de dados com o intuito de se obter um classificador de notícias que consiga discernir notícias verdadeiras e falsas no âmbito político a partir de seus textos. O tema detecção de *fake news* tem sido bastante tratado nos últimos anos devido à facilidade de interação num ambiente como a Internet, fator que gera uma grande massa de informação e desinformação que pode afetar várias esferas da sociedade.

Este projeto envolveu a aquisição e análise de dois conjuntos de dados diferentes, assim como a aplicação dos algoritmos de vetorização e classificação mais utilizados da área. Desta experimentação, destacou-se o uso do algoritmo de vetorização TF-IDF juntamente ao modelo de regressão logística, que tiveram um bom desempenho em ambos os conjuntos de dados. Mais especificamente, com o conjunto de dados Fake.Br corpus, essa combinação gerou um resultado bastante satisfatório em cima do conjunto de teste geral, atingindo as acurácias de 0.861789 sem a otimização de hiper-parâmetros e de 0.876693 com a otimização de hiper-parâmetros. Vale mencionar o desempenho com o conjunto de dados primário, que atingiu uma acurácia de 0.810298 em cima do conjunto de teste geral sem ajustes de hiper-parâmetros.

Com relação à aplicação móvel, esta foi desenvolvida de forma a permitir que o usuário consiga usá-la facilmente e também objetiva educá-lo para ter uma maior atenção às notícias recebidas antes de compartilhá-las.

Além disso, houve o estudo de uma grande variedade de bibliotecas – principalmente da linguagem Python – referentes ao desenvolvimento do fluxo de ciência de dados e à disponibilização de um produto que consome o modelo treinado. Isso possibilitou um entendimento mais amplo dessas tecnologias, as quais são bastante requisitadas no mercado de trabalho.

Algumas das dificuldades enfrentadas envolveram os dados do conjunto de dados primário, os quais estavam desbalanceados e demandaram uma análise mais aprofundada para que estes pudessem ser utilizados propriamente.

5.1 Trabalhos futuros

Como sugestão para trabalhos futuros, considera-se a melhoria dos resultados obtidos neste trabalho. Para tal, pode-se verificar o uso de outros modelos de aprendizado de máquina, outras características ou mesmo diferentes algoritmos de vetorização.

Outro ponto que pode ser melhorado se refere à coleta dos dados, abrindo a possibilidade de incrementar o conjunto de dados primário a fim de montar um conjunto com o qual se pode treinar um modelo que melhor generaliza o problema de classificação.

Com relação à aplicação móvel, esta futuramente poderá ser disponibilizada em um serviço de distribuição digital de aplicativos, e ser baixada diretamente de um dispositivo com o sistema *Android* através de uma plataforma como o *Google Play*.

Referências

ALLCOTT, H.; GENTZKOW, M. *Social Media and Fake News in the 2016 Election*. [S.l.], 2017. (Working Paper Series, 23089).

BAKSHY, E.; MESSING, S.; ADAMIC, L. A. Political science. exposure to ideologically diverse news and opinion on facebook. *Science (New York, N.Y.)*, v. 348, Maio 2015.

BERNERS-LEE, T.; FIELDING, R. T.; MASINTER, L. *Uniform Resource Identifier (URI): Generic Syntax*. [S.l.], 2005. Disponível em: <http://www.rfc-editor.org/rfc/rfc3986.txt>. Acesso em: 13 Out. 2019.

BIRD, S.; KLEIN, E.; LOPER, E. *Natural language processing with Python: analyzing text with the natural language toolkit*. [S.l.]: "O'Reilly Media, Inc.", 2009.

BRASIL. *Lei Nº 9504 de 30 de setembro de 1997. Estabelece normas para as eleições*. Brasília, DF, 1997.

BRESSERT, E. *SciPy and NumPy: an overview for developers*. [S.l.]: O'Reilly, 2012.

BROWNLEE, J. *Master Machine Learning Algorithms: Discover How They Work and Implement Them From Scratch*. [S.l.]: Jason Brownlee, 2016.

CONROY, N. J.; RUBIN, V. L.; CHEN, Y. Automatic deception detection: Methods for finding fake news. *Proceedings of the Association for Information Science and Technology*, Wiley Online Library, v. 52, n. 1, p. 1–4, 2015.

DHAR, V. Data Science and Prediction. Maio 2012.

FERNQUIST, J. *Detection of deceptive reviews : using classification and natural language processing features*. 2016. 53 p. (UPTEC F, 16056).

FLASK. *Flask*. 2019. Disponível em: <https://palletsprojects.com/p/flask/>. Acesso em: 08 Out. 2019.

GANDHI, R. Naive bayes classifier. Maio 2018. Disponível em: <https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c>. Acesso em: 19 Out. 2019.

GOMES, W.; FERNANDES, B.; REIS, L.; SILVA, T. "Politics 2.0": a campanha online de Barack Obama em 2008. *Revista de Sociologia e Política*, scielo, v. 17, p. 29 – 43, Outubro 2009. ISSN 0104-4478.

GOTTFRIED, J.; SHEARER, E. *News Use Across Social Media Platforms 2016*. 2016.

GUO, Y. The 7 steps of machine learning. Agosto 2017. Disponível em: <https://towardsdatascience.com/the-7-steps-of-machine-learning-2877d7e5548e>. Acesso em: 13 Out. 2019.

HILLAR, G. C. *Building RESTful Python Web Services*. [S.l.]: Packt Publishing Ltd, 2016.

IONIC. *Ionic*. 2019. Disponível em: <https://ionicframework.com/>. Acesso em: 08 Out. 2019.

JUPYTER NOTEBOOK. *Jupyter Notebook*. 2019. Disponível em: <https://jupyter-notebook.readthedocs.io/en/stable/index.html>. Acesso em: 07 Out. 2019.

KANA, M. Representing text in natural language processing. Julho 2019. Disponível em: <https://towardsdatascience.com/representing-text-in-natural-language-processing-1eead30e57d8>. Acesso em: 27 Out. 2019.

KHANNA, R.; AWAD, M. *Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers*. [S.l.: s.n.], 2015. ISBN 1430259892.

KOEHRSEN, W. Random forest simple explanation. Dezembro 2017. Disponível em: <https://medium.com/@williamkoehrsen/random-forest-simple-explanation-377895a60d2d>. Acesso em: 20 Out. 2019.

KOHAVI, R.; PROVOST, F. *Glossary of Terms*. 1998. Disponível em: <http://ai.stanford.edu/~ronnyk/glossary.html>. Acesso em: 11 Out. 2019.

LANEY, D. *3D Data Management: Controlling Data Volume, Velocity, and Variety*. [S.l.], 2001.

LAZER, D. M. J.; BAUM, M.; BENKLER, Y.; BERINSKY, A. J.; GREENHILL, K. M.; MENCZER, F.; METZGER, M. J.; NYHAN, B.; PENNYCOOK, G.; ROTHSCHILD, D.; SCHUDSON, M.; SLOMAN, S.; SUNSTEIN, C.; THORSON, E. A.; WATTS, D. J.; ZITTRAIN, J. L. The science of fake news. *Science*, v. 359, p. 1094–1096, Março 2018.

MASSE, M. *REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces*. [S.l.]: "O'Reilly Media, Inc.", 2011.

MATPLOTLIB. *Matplotlib*. 2019. Disponível em: <https://matplotlib.org/>. Acesso em: 07 Out. 2019.

MCKINNEY, W. pandas: a foundational python library for data analysis and statistics. 2011.

MCKINNEY, W. *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython*. [S.l.]: "O'Reilly Media, Inc.", 2012.

METSIS, V.; ANDROUTSOPOULOS, I.; PALIOURAS, G. Spam filtering with naive bayes - which naive bayes? In: MOUNTAIN VIEW, CA. *CEAS*. [S.l.], 2006. v. 17, p. 28–69.

MIRANDA, A. Sociedade da informação: globalização, identidade cultural e conteúdos. *Cia da Informação*, scielo, v. 29, p. 78 – 88, Agosto 2000. ISSN 0100-1965.

MONTEIRO, R. A.; SANTOS, R. L. S.; PARDO, T. A. S.; ALMEIDA, T. A. de; RUIZ, E. E. S.; VALE, O. A. Contributions to the study of fake news in portuguese: New corpus and automatic detection results. In: *Computational Processing of the Portuguese Language*. [S.l.]: Springer International Publishing, 2018. p. 324–334. ISBN 978-3-319-99722-3.

MUMBAIKAR, S.; PADIYA, P. Web services based on soap and rest principles. *International Journal of Scientific and Research Publications*, v. 3, n. 5, p. 1–4, 2013.

NUMPY. *NumPy*. 2019. Disponível em: <https://numpy.org>. Acesso em: 07 Out. 2019.

PANDAS. *pandas*. 2019. Disponível em: <https://pandas.pydata.org/>. Acesso em: 29 Set. 2019.

PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V.; VANDERPLAS, J.; PASSOS, A.; COURNAPEAU, D.; BRUCHER, M.; PERROT, M.; DUCHESNAY, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011.

PONTES, C. H. F. Fake news e o desafio da justiça eleitoral nas eleições de 2018. f. 41, 2018.

PROVOST, F.; FAWCETT, T. Data science and its relationship to big data and data-driven decision making. *Big Data*, v. 1, Março 2013.

PYTHON ANYWHERE. *Python Anywhere*. 2019. Disponível em: <https://www.pythonanywhere.com>. Acesso em: 26 Out. 2019.

RASHKIN, H.; CHOI, E.; JANG, J. Y.; VOLKOVA, S.; CHOI, Y. Truth of varying shades: Analyzing language in fake news and political fact-checking. In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, 2017.

RUEDIGER, M. A.; FILHO, C. A. L.; SANTOS, E. F.; SANTOS, G. K.; SALVADOR, J. P. F.; KAROLCZAK, R. M.; GUIMARÃES, T.; AQUINO, T. M.; SILVEIRA, V. D. Bots e o direito eleitoral brasileiro: eleições 2018. *Policy paper 3*, Rio de Janeiro: FGV DAPP, 2019.

SANTAELLA, L. A tecnocultura atual e suas tendências futuras. *Signo y Pensamiento*, scieloco, v. 31, p. 30 – 43, Junho 2012. ISSN 0120-4823.

SANTOS, R. L. S. *Fake.Br Corpus*. 2018. Disponível em: <https://github.com/roneysco/Fake.br-Corpus>. Acesso em: 24 Out. 2019.

SCIKIT-LEARN. *Scikit-Learn*. 2019. Disponível em: <https://scikit-learn.org/stable/>. Acesso em: 29 Set. 2019.

SCRAPY. *Scrapy*. 2019. Disponível em: <https://scrapy.org/>. Acesso em: 29 Set. 2019.

SEABORN. *Seaborn*. 2019. Disponível em: <https://seaborn.pydata.org/index.html>. Acesso em: 08 Out. 2019.

SHIMODAIRA, H. Text classification using naive bayes. *Learning and Data Note*, v. 7, p. 1–9, 2014.

SINGHAL, S. Data representation in nlp. Junho 2019. Disponível em: <https://medium.com/@shiivangii/data-representation-in-nlp-7bb6a771599a>. Acesso em: 27 Out. 2019.

ZIVIANI, A.; PORTO, F. Ciência de Dados. In: *Grandes Desafios da Computação no Brasil - Relatos do 3º seminário*. Rio de Janeiro, Brasil: [s.n.], 2014. p. 50–71.