Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Análise Preditiva Espaço-Temporal das Tendências Eleitorais Brasileiras com Base nos Dados do Twitter

Bruno Justino Garcia Praciano

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Orientador
Prof. Dr. -Ing João Paulo Lustosa da Costa

Coorientador
MSc. João Paulo de Abreu Maranhão

Brasília
2018

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Análise Preditiva Espaço-Temporal das Tendências Eleitorais Brasileiras com Base nos Dados do Twitter

Bruno Justino Garcia Praciano

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Prof. Dr. -Ing João Paulo Lustosa da Costa (Orientador)
ENE/UnB

Prof. Dr. Rafael Timóteo de Sousa Júnior
ENE/UnB

Dr. -Ing Ricardo Kerhle Miranda
ENM/UnB

Prof. Dr. José Edil Guimarães de Medeiros
Coordenador do Curso de Engenharia da Computação

Brasília, 13 de novembro de 2018

Dedicatória

Na *dedicatória* o autor presta homenagem a alguma pessoa (ou grupo de pessoas) que têm significado especial na vida pessoal ou profissional. Por exemplo (e citando o poeta): *Eu dedico essa música a primeira garota que tá sentada ali na fila. Brigado!*

Agradecimentos

Nos *agradecimentos*, o autor se dirige a pessoas ou instituições que contribuíram para elaboração do trabalho apresentado. Por exemplo: *Agradeço aos gigantes cujos ombros me permitiram enxergar mais longe. E a Google e Wikipédia.*

Resumo

O *resumo* é um texto inaugural para quem quer conhecer o trabalho, deve conter uma breve descrição de todo o trabalho (apenas um parágrafo). Portanto, só deve ser escrito após o texto estar pronto. Não é uma coletânea de frases recortadas do trabalho, mas uma apresentação concisa dos pontos relevantes, de modo que o leitor tenha uma ideia completa do que lhe espera. Uma sugestão é que seja composto por quatro pontos: 1) o que está sendo proposto, 2) qual o mérito da proposta, 3) como a proposta foi avaliada/validada, 4) quais as possibilidades para trabalhos futuros. É seguido de (geralmente) três palavras-chave que devem indicar claramente a que se refere o seu trabalho. Por exemplo: *Este trabalho apresenta informações úteis a produção de trabalhos científicos para descrever e exemplificar como utililzar a classe LaTeX do Departamento de Ciência da Computação da Universidade de Brasília para gerar documentos. A classe `UnB-CIC` define um padrão de formato para textos do CIC, facilitando a geração de textos e permitindo que os autores foquem apenas no conteúdo. O formato foi aprovado pelos professores do Departamento e utilizado para gerar este documento. Melhorias futuras incluem manutenção contínua da classe e aprimoramento do texto explicativo.*

Palavras-chave: Big Data, Aprendizado de Máquina Supervisionado, Análise de Sentimentos, Máquina de Vetor de Suporte

Abstract

O *abstract* é o resumo feito na língua Inglesa. Embora o conteúdo apresentado deva ser o mesmo, este texto não deve ser a tradução literal de cada palavra ou frase do resumo, muito menos feito em um tradutor automático. É uma língua diferente e o texto deveria ser escrito de acordo com suas nuances (aproveite para ler `http://dx.doi.org/10.6061%2Fclinics%2F2014(03)01`). Por exemplo: *This work presents useful information on how to create a scientific text to describe and provide examples of how to use the Computer Science Department's LaTeX class. The `UnB-CIC` class defines a standard format for texts, simplifying the process of generating CIC documents and enabling authors to focus only on content. The standard was approved by the Department's professors and used to create this document. Future work includes continued support for the class and improvements on the explanatory text.*

Keywords: Big Data, Supervised Machine Learning, Sentiment Analysis, Support Vector Machine

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

AM Aprendizado de Máquina.

API Aplication Programming Interface.

IBGE Instituto Brasileiro de Geografia e Estatística.

ICDM International Conference on Data Mining.

IEEE Instituto de Engenheiros Eletricistas e Eletrônicos.

PLN Processamento de Linguagem Natural.

SVM Support Vector Machine.

TM Text Mining.

TSE Tribunal Superior Eleitoral.

Capítulo 1

Introdução

A popularização da Internet tem revolucionado as sociedades com o passar do tempos, pois agora é possível conectar-se a diversas pessoas, ter acesso a informação de forma rápida e realizar grande parte das atividades em tempo real, e esse sucesso é devido ao seu baixo custo em relação aos veículos tradicionais de mídia [3].

A troca de dados tem acontecido maneira intensa, as redes sociais são as responsáveis por esse crescimento expressivo, pois as pessoas podem trocar ideias e opiniões acerca de determinado assunto e com isso facilitar o acesso de todos. Com o passar dos anos as redes sociais já fazem parte da vida de várias pessoas, e com isso as relações interpessoais modificaram-se e esse mundo tem gerado muitos dados de fácil e livre acesso [4].

Com as redes sociais é possível comunicar-se com pessoas de diversas nacionalidades e características, com a grande amplitude que essas alcançam, o volume de dados é algo imensurável e também é uma fonte de dados inesgotável. Com todo esse volume de informações, o ambiente torna-se atrativo para aplicar técnicas de aprendizado de máquina e outros tipos de análises.

1.1 Motivação

De acordo com o IBGE, no Brasil mais de 116 milhões de pessoas tem acesso a internet, ou seja, grande parte da população está expressando suas ideias de forma livre nas redes sociais. E como as eleições em um evento muito importante em qualquer democracia, realizar análise de sentimentos nos textos provenientes de redes sociais se tornam cada vez mais atrativos. O Brasil ocupa a 4ª no ranking de países com a quantidade de pessoas com acesso a internet [5].

Na Figura 1.1 é possível visualizar que grande parte do conteúdo disponível na internet está em inglês, ou seja, é por esse motivo que existem dicionários léxicos para atividades

de TM, como é o caso do WordNet [6], uma das maiores bases de dados do mundo para essa atividade.

As pesquisas utilizando TM com o idioma português ainda são recentes e poucos exploradas, pois a maioria das ferramentas são desenvolvidas para atender o idioma inglês. Mas a análise de sentimento em comentários de redes sociais, pode ser útil em diversas áreas como venda de um produto, avaliação de um estabelecimento, marketing e até mesmo realizar predições de eleições que é o objetivo desse trabalho.

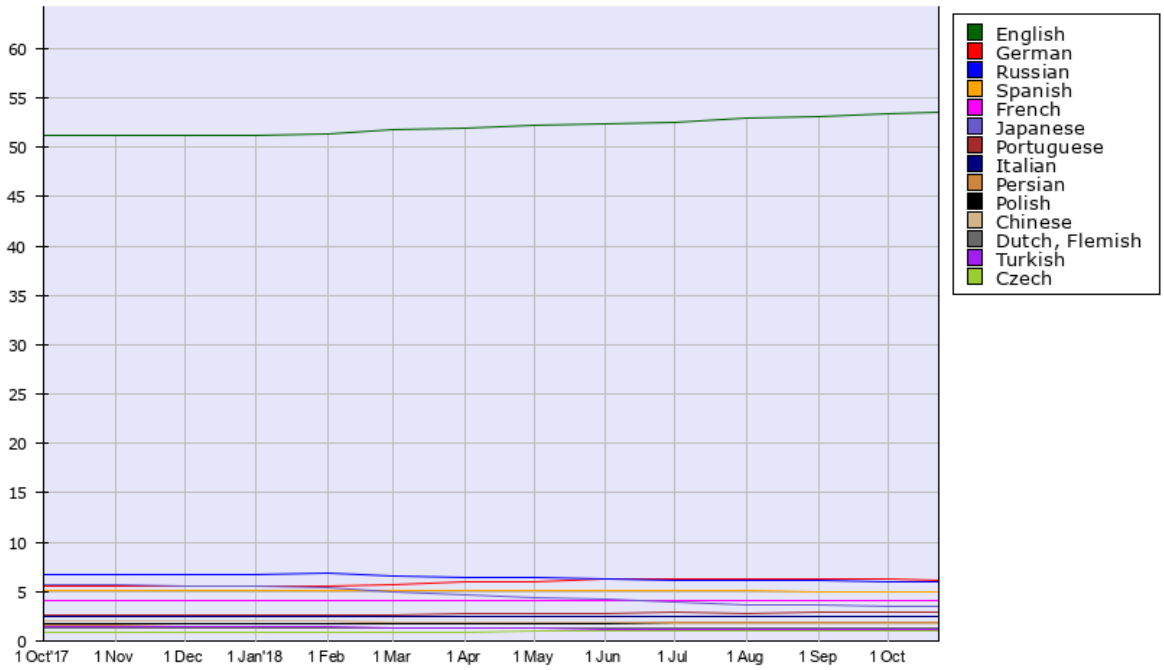

Figura 1.1: Idiomas utilizados em conteúdos disponíveis na internet [1].

1.2 Problemas

Existem várias redes sociais em funcionamento, e cada uma tem o foco diferente, por exemplo, o *Twitter* é uma rede social com o intuito de formação de opinião, pois grande parte dos usuários a utilizam para compartilhar texto pequenos de até 140 caracteres e também apresenta uma API aberta, mas essa possui limitação em relação ao número de requisições e também ao período que pode efetuar uma busca, que atualmente são de 14 dias [7].

A rede social mais utilizado no Brasil é o *Facebook*, que tem mais de 120 milhões de usuários ativos no Brasil [8], mas após os escândalos das eleições americanas que a envolveram [9] foram adotadas inúmeras medidas de segurança para evitar a extração de

dados da plataforma, atualmente para utilizar a API da empresa é necessário ser aprovado e também conta com o número limitado de requisições que podem ser feitas por cada token de segurança.

É importante citar que os dicionários em língua portuguesa para realizar esse tipo de atividades ainda são pouco desenvolvidos, para esse trabalho foi utilizado dois dicionários em conjunto para melhorar os resultados. O OpLexicon [10] foi combinado com o Sentilex [11], pois atualmente são os melhores dicionários abertos em português para realizar análise de sentimentos, também foi utilizada uma biblioteca chamada TextBlob [12] que é necessário realizar a tradução para o inglês, o que acaba ocasionando um viés na etapa de processamento dos dados.

1.3 Objetivos

O objetivo desse trabalho é criar uma forma de predição e um ambiente que expresse a opinião dos usuários da rede social *Twitter* aplicando em textos curtos técnicas de AM, o intuito principal desse trabalho é a utilização do framework proposto em textos que falam sobre políticos que estão concorrendo a cargos eletivos, mas com a inserção de outros textos na fase de treinamento do modelo de AM é possível ampliar as opções e realizar diversas análises para distintas áreas.

1.4 Trabalho Publicado

Durante o desenvolvimento desse trabalho de graduação, foi desenvolvido um artigo científico cujo o intuito era validar a ideia e verificar se apresentava algum tipo de inovação, onde foi apresentado os primeiros resultados do trabalho.

O artigo publicado é entitulado *Spatio-Temporal Trend Analysis of the Brazilian Elections based on Twitter Data* foi aceito em uma das maiores conferências de mineração de dados do mundo, o IEEE ICDM, e foi aceito para apresentação oral na seção de análise de sentimentos.

B. J. G. Praciano, J. P. C. L. da Costa, J. P. A. Maranhão, J. B. Prettz, R. T. de Sousa Jr. e F. Mendonça, "Spatio-Temporal Trend Analysis of the Brazilian Elections based on Twitter Data," 2018 IEEE International Conference on Data Mining (ICDM).

1.5 Trabalhos Relacionados

Em [13] os autores definiram o conceito de análise de sentimento em vários níveis e também utilizaram algoritmos que realizam o reconhecimento de entidades, que é a

detecção de nomes próprios e com isso remover esses substantivos da análise de sentimento para que o os resultados sejam melhores. O dicionário léxico utilizado para a classificação dos textos provenientes do *Twitter* foi o SentiWordNet, e as polaridades utilizadas nesse trabalho foram três: Positivo, Negativo e Neutro. Foi abordado duas paradigmas de AM, o supervisionado e o não-supervisionado e com o isso o melhor resultado foi de 90% na utilização de algoritmos supervisionados.

No artigo [14] foi utilizado um filtro baseado na mineração de opiniões, que é uma das áreas de análise de sentimentos. Foi utilizadas técnicas de decomposição tensorial para capturar interações intrínsecas, pois como o dado é multidimensional, o autor dividiu entre usuários, filmes e outros aspectos, com a aplicação dessas técnicas, houve uma grande redução no esforço computacional do computador na parte de análise de sentimentos, pois o *dataset* utilizado foi reduzido após a decomposição tensorial.

Em [3] os autores aplicaram TM nos dados provenientes do *Twitter* que citavam as eleições presidencias de 2012 da Coréia do Sul. Foram utilizadas distintas técnicas: *topic modeling* para acompanhar as mudanças nos assuntos mais falados do rede sociais, técnicas de análise de rede foram utilizadas para verificar quais pessoas eram citadas e por quem. Os resultados sugeriram que o *Twitter* pode ser um aliado para detectar as mudanças no contexto social enquanto são analisados o texto de quem escreveu.

Em [15] é proposto um sistema para acompanhar as eleições francesas através de tópicos escritos no *Twitter* através da análise de sentimentos. Os resultados obtidos convergiram com os resultados divulgados pelas autoridades da França e foram associados as mudanças de popularidade dos candidatos após a eleição.

Em [16], o dataset utilizado nesse trabalham foi o da eleição colombiana de 2014, técnicas de aprendizado supervisionado foram implementadas e também foram rotuladas previamente usuários que seriam spam. Foi implementado um sistema com o objetivo de investigar o potencial que uma rede social tem de interferir em uma votação, e de acordo com os resultados obtidos, foi possível afirmar que os dados utilizados não foram consistentes.

Em [17] foi usado um dicionário léxico e apenas o algoritmo Naïve Bayes para calcular o sentimento de *tweets* que foram coletados 100 dias antes da eleição americanas de 2016. Os autores classificaram manualmente os textos extraídos do *Twitter*. Os resultados obtids sugerem que essa rede social pode ser considerada ao realizar trabalhos com esse intuito.

1.6 Descrição dos capítulos

O presente trabalho é apresentado com a seguinte estrutura:

- Capítulo 2: Conceitos em Machine Learning e Mineração de Texto. Apresenta o conjunto de técnicas e metologias que foram necessárias para o desenvolvimento desse trabalho.
- Capítulo 3: Framework proposto. Discorre sobre a metodologia empregada no trabalho e ilustra todos os passos seguidos e necessários para entendidmento do modelo.
- Capítulo 4: Resultados. Nesse capítulo são apresentados os resultados obtidos com a utilização do modelo de aprendizado de máquina proposto e também uma breve justificativa a escolha das ferramentas.
- Capítulo 5: Conclusão. As conclusões sobre o tema são expostas.
- Capítulo 6: Trabalhos Futuros. Na última seção são discutidos quais temas que foram levantados que podem ser aprofundados.

Capítulo 2

Conceitos em Aprendizado de Máquina e Mineração de Texto

Neste capítulo é feita uma introdução sobre o AM, tema principal dessa monografia. E está dividido da seguinte maneira na seção 2.1 são apresentados os conceitos básicos para o entendimento inicial do tema. Na seção 2.2 são apresentados os conceitos de mineração de texto. Na seção 2.3 é apresentado o conceito de análise de sentimentos e a sua aplicação.

2.1 Aprendizado de Máquina

2.1.1 Conceitos Básicos

Com o alto volume de informações geradas no dia a dia, a utilização de algoritmos que sejam capazes de identificar padrões tornam-se cada vez mais necessários, pois em diversas empresas e órgãos do governo não são capazes de analisar todas as informações existentes ou entregar de maneira célere [18].

Portanto para realizar esse tipo de atividade é necessário entender o problema existente e também ter um volume de dados razoável para realizar o treinamento do modelo, e com a técnica de AM é possível automatizar a construção de sistemas inteligentes que podem ser ajustados de acordo com a necessidade de cada tarefa [19].

Em outras palavras o AM é um conjunto de regras, que possibilitam uma máquina a tomar decisões baseadas em experiências passadas ao invés de um software que teve que ser definido anteriormente como tratar cada tipo de regra, também existe a possibilidade desses modelos serem desenvolvidos e melhorarem quando expostos a novos dados [20].

O conceito de AM pode ser sintetizado como a capacidade de um programa de computador aprender com a experiencia (E) relacionada a alguma classe de tarefas (T), baseada

em uma medida de desempenho (P). Dessa forma, o desempenho em tarefas (T), quando medido por (P), melhora com a experiencia em (E) [21]

Essas técnicas tem sido utilizadas amplamente para resolução de diversos problemas, atualmente o assunto está em alta e existem diversas oportunidade para colocar em prática a matemática que foi desenvolvida para a criação desses algoritmos. Gigantes da indústria tem investido severamente para o desenvolvimento de novas tecnologias, como os veículos autônomos, robôs advogados, veículos não tripulados e na identificação de doenças.

Existem distintos paradigmas para ensinar uma máquina, esse trabalho irá focar apenas nos tipos de AM que estão sendo utilizados no desenvolvimento do framework, portanto não será abordados temas como aprendizado estatístico ou redes bayesianas.

2.1.2 Paradigmas de Aprendizado de Máquina

Os principais tipos de AM utilizados atualmente serão detalhados nas 3 subseções a seguir:

2.1.2.1 Aprendizado Supervisionado

O aprendizado supervisionado é uma das formas em ensinar uma tarefa para a máquina, onde existe uma entrada e uma saída desejada que já foi anteriormente rotulada, esse processo pode ser feito de forma manual ou utilizar de dicionários, quando a informação de entrada é um texto. Tendo os dados detalhados a máquina é capaz de classificar novas entradas a partir de experiências antigas [21]. Na Figura 2.1 é possível visualizar a forma que é realizada etapa de treinamento do modelo, onde é fornecida uma informação, juntamente com o que ela significa, pois esse tipo de aprendizado é necessário rotular todas as informações que serão utilizadas para que a máquina consiga distinguir o que é gato e o que é cachorro.

Na Figura 2.2 é possível visualizar o modelo preditivo em funcionamento, onde o usuário fornece uma informação de entrada e o sistema o responde com a classe que foi identificada através da entrada.

Algum dos algoritmos mais utilizados para esse paradigma serão detalhados durante essa subseção.

2.1.2.1.1 SVM A máquina de vetor de suportes, do inglês support vector machine (SVM), faz parte do aprendizado supervisionado, com ele é possível classificar grupos de dados separando as suas margens. Essas margens são delineadas pela fração dos dados de treinamento, são chamadas de vetores de suporte [22].

As vantagens de utilizar SVM são [23]:

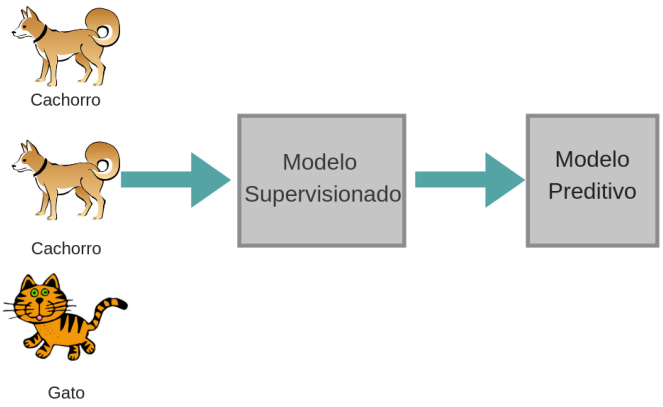

Figura 2.1: Treinamento de um modelo de aprenzidado de máquina supervisionado.

Figura 2.2: Utilização do modelo para realizar predição a partir de classes previamente treinadas.

- Efetivos em espaços multidimensionais
- Continua eficaz em casos onde o número de dimensões é maior que o número de amostras.
- Utiliza os vetores de suporte, que otimiza o uso de memória do computador.
- É possível utilizar diversos kernels para a função de decisão.

Como desvantagens, temos [23]:

The disadvantages of support vector machines include:

- Se o número de entradas for muito maior que o número de amostras, pois existe a possibilidade de overfitting.

O SVM é construido em um hiper-plano ou em vários hiper-planos em um espaço de infinitas dimensões, que podem ser usadas classificação ou regressão. Na Figura 2.3 é detalhado um hiper-plano que mostra uma boa separação das variáveis que possui a maior distância até os pontos dos dados de treinamento mais próximos de qualquer classe, pois é conhecido no SVM que quanto maior for a margem, menor será o erro do classificador [24].

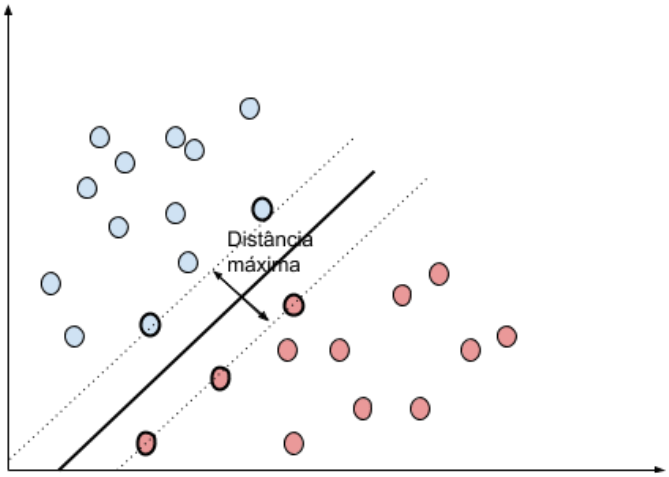

Figura 2.3: Exemplo de vetores de suporte com dimensão 2.

Nesse trabalho foi utilizado esse algoritmo para classificar dados multi-classes, pois foi utilizadas três classes distintas, para desenvolvimento do modelo: Positivo, Negativo e Neutro.

Tendo como os dados de treinamento $x_i \in R^p$, onde i=1,$\cdots$, $n.Onde$ y $\in \{1, -1\}^n$, na Equação 2.1 é possível visualizar a solução matemática para esse algoritmo.

$$\begin{aligned} &\min_{\omega,\beta,\zeta} \frac{1}{2}\omega^T\omega + C\sum_{i=1}^{n}\zeta_i \\ &\text{sujeito a } y_i\left(\omega^T\phi\left(x_i\right) + b\right) \geq 1 - \zeta_i, \\ &\zeta_i \geq 0, i = 1, \cdots, n \end{aligned} \tag{2.1}$$

A partir do espaço de Hilbert, temos que o dual de um espaço de Hilbert é um espaço de Hilbert [25],

na Equação 2.2, temos:

$$\min_{\alpha} \frac{1}{2}\alpha^T Q\alpha - e^T\alpha$$

$$\text{sujeito } ay^T\alpha = 0 \tag{2.2}$$

$$0 \leq \alpha_i \leq C, i = 1 \cdots, n$$

onde e é o vetor com todos os valores, $C > 0$ é o limite superior, Q é uma matriz semi-definida positiva de tamanho n por n, $Q_{ij} \equiv y_i y_j K(x_i, x_j)$, onde $K(x_i, x_j) = \phi(x_i)^T\phi(x_j)$ é a função kernel do SVM. Onde os vetores de treinamento são definidos em um espaço dimensional maior pela função ϕ. E por fim na Equação 2.3, temos a função de decisão:

$$sgn(\sum_{i=1}^{n} y_i\alpha_i K(x_i, x) + \rho) \tag{2.3}$$

Onde o termo $y_i\alpha_i$ representa o coeficiente dual, $K(x_i, x)$ são os vetores de suporte e ρ é um termo independente.

2.1.2.1.2 Naive Bayes O Naive Bayes é um outro algoritmo AM supervisionado, a sua formulação matemática é sustentada pelo teorema estatístico de Bayes com um pouco de ingenuidade, pois assume-se que a suposição de independência condicional entre cada par de recursos, dado o valor da variável de classe [26]. O teorema de Bayes clássico segue a seguinte relação, onde um termo independente y e o vetor x_1 até x_n, na Equação 2.4

$$P\left(y \mid x_1, \cdots, x_n\right) = \frac{P(y)P(x_1, \cdots, x_n \mid y)}{P(x_1, \cdots, x_n)} \tag{2.4}$$

Na Equação 2.5 é desconsiderado completamente a correlação entre as variáveis,

$$P\left(x_i \mid y, x_1, \cdots, x_{i-1}, x_{i+1}, \cdots, x_n\right) = P(x_i|y), \tag{2.5}$$

Na Equação 2.6, para todo i, é possível simplificar através da relação:

$$P(y|x_1, \cdots, x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1, \cdots, x_n)}, \tag{2.6}$$

Sabendo que $P(x_1, \cdots, x_n)$ é constante dada a entrada, podemos usar a seguinte regra de classificação, na Equação 2.7:

$$\widehat{y} = \arg\max_{y} P(y)\prod_{i=1}^{n} P(x_i|y), \tag{2.7}$$

O classificador Naive Bayes pode ser extremamente rápido em comparação a outros algoritmos mais sofisticados, pois existe o desacoplamento das distribuições de características condicionais de classe que significa que cada uma pode ser estimada independentemente com uma distribuição unidimensional [27].

2.1.2.1.3 Árvores de Decisão A árvore de decisão funciona quando as classes presentes no conjunto de dados de treinamento se dividem, ou seja, esse algoritmo seleciona um problema mais complexo e divide em subproblemas mais simples e de forma recursiva a mesma estratégia é a aplicada a cada subproblema [28].

Na Figura 2.4, é um exemplo de como uma árvore de decisão poderia realizar uma análise de crédito de forma rápida.

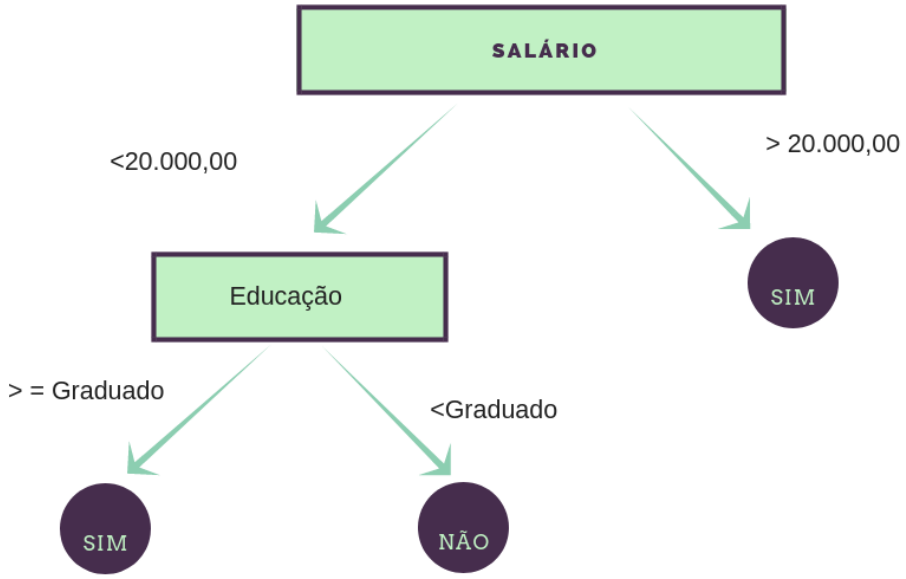

Figura 2.4: Árvore de decisão para análise de crédito de um cliente que deseja empréstimo alto no banco, levando em consideração a educação e faixa salarial.

Tendo como entrada os vetores de treinamento $x_i \in R^n$, $i = 1, \cdots, n$, e o vetor com os rótulos de saída $y \in R^n$, a árvore de decisão divide recursivamente o espaço de forma que as amostras com os mesmos rótulos sejam agrupadas [29].

Considerando os dados no nó m que pode ser representado pela letra Q. Para cada candidato é dividido $\theta = (j, t_m)$ consistindo a entrada como j e o *threshold* t_m, dividindo os dados entre $Q_{left}(\theta)$ e $Q_{right}(\theta)$ e esses subconjuntos pode ser vistos na Equação 2.8

$$\begin{aligned} Q_{left}(\theta) &= (x, y)|x_j \leq t_m \\ Q_{right}(\theta) &= Q \backslash Q_{left}(\theta) \end{aligned} \tag{2.8}$$

A impureza no nó apresentado é calculada utilizando uma função de impureza denotada de $K()$, que pode ser escolhida dependendo se o problema for de classificação ou regressão. Na Equação 2.9 é calculada essa impureza:

$$G(Q, \theta) = \frac{n_{left}}{N_m} K(Q_{left}(\theta)) + \frac{n_{right}}{N_m} K(Q_{right}(\theta)) \tag{2.9}$$

Por fim, na Equação 2.10 são selecionados os parâmetros para diminuir a impureza,

$$\theta = \arg\min_{\theta} G(Q, \theta) \tag{2.10}$$

É possível gerar outros nós a partir dos subconjuntos de dados gerados para direita e para esquerda, até que a profundidade máxima permitida seja atingida, $N_m < \min_{samples}$ ou $N_m = 1$.

2.1.2.1.4 Regressão Logística Apesar do nome, é um modelo de classificação linear e não de regressão como sugere o seu nome. Neste modelo, os dados são descritos, através de um único teste e são modeladas a partir de uma função logística [18].

Seja a probabilidade P(X), na Equação 2.11

$$\begin{aligned} P(x) &= \frac{1}{1 + e^{-(B_0 + B_1 x)}} = \frac{e^{B_0 + B_1} x}{1 + e^{B_0 + B_1} x} \\ \\ \frac{p(x)}{1 - p(x)} &= e^{B_0 + B_1 x} \end{aligned} \tag{2.11}$$

Onde, $_i$ são calculados através da função de máxima verossimilhança.

2.1.2.2 Aprendizado não-supervisionado

Nesse tipo de aprendizado não existe a necessidade de um vetor contendo os rótulos de cada medida, ou seja, não é necessário avisar a máquina o tipo de cada dado. A essa não inserção de rótulos faz com que o algoritmo aprenda de acordo com as características dos dados de entrada. Esse tipo de paradigma não tem certos benefícios que existem

no aprendizado supervisionado, para o seu funcionamento, o modelo propõe hipóteses a partir do que foi inserido na entrada [19]. Esse tipo de algoritmo é muito interessante em base de dados que ainda não foram exploradas e que necessita-se visualizar os dados agrupados.

A Figura 2.5 é ilustrado o exemplo que foi exposto ao falar sobre aprendizado supervisionado, mas agora levando em consideração o não-supervisionado. E é possível ver que o modelo foi capaz de agrupar as características semelhantes que os cachorros têm e agrupá-los em um único cluster e também foi formado um outro cluster contendo apenas o gato, pois não tem características semelhantes aos cães.

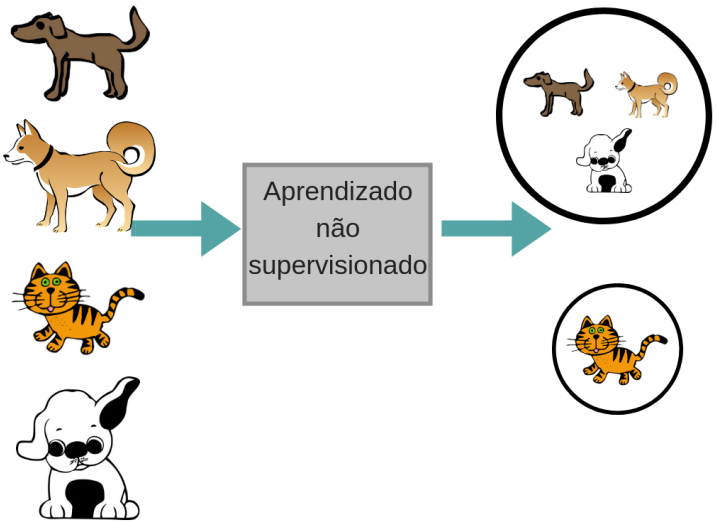

Figura 2.5: Treinamento de um modelo de aprendizado de máquina não-supervisionado.

2.1.2.2.1 K-means O algoritmo mais utilizado no aprendizado não-supervisionado é o *k-means*, pois a sua implementação é muito simples, pois basta comprovar o processo de minimizacão da distância quadrática total de cada ponto de um grupo, em relação ao centróide de referência [30]. Na Equação 2.12 é possível visualizar como é realizada a seleção de clusters através do número de amostras X, descrito pela média μ_j de cada amostra nos clusters.

$$\sum_{i=0}^{n} \min_{\mu_j \in C}(||x_i - \mu_j||^2) \tag{2.12}$$

2.1.2.3 Aprendizado por reforço

O último paradigma é baseado na forma em que o modelo reage de acordo com o que o ambiente passa para ele, é possível aprender continuamente. Esse aprendizado é

realizado através das ações que são executadas, caso seja a decisão correta o modelo ganha uma recompensa, caso contrário pode ser imposta algum tipo de penalidade [31]. Esse paradigma de aprendizado é necessário ajustar os parâmetros de forma contínua com o intuito de maximar ou minimizar um determinado índice, e sempre há um "treinador"nessa fase verificando se os estímulos proporcionados estão tendo a saída correta, pois caso esses dados não estejam balanceados, eles podem alterar a performance do algoritmo [32].

Na aprendizagem por reforço, o modelo só aprende com uma série de estímulos, que podem ser recompensas ou punições no decorrer da aprendizagem, um exemplo do mundo real seria, avaliar um motorista com nota baixa em um aplicativo de locomoção urbana, nesse caso ele irá saber que existiu alguma condunta ruim. Um caso em que recebe um estímulo positivo pode ser dar um biscoito a um cachorro após fazer de forma correta o que foi pedido.

Na Figura 2.6 é possível ver um fluxograma de treinamento de um modelo utilizando o reforço.

Figura 2.6: Fluxograma de treinamento de um modelo de aprendizado de máquina por reforço.

2.1.3 Medidas de avaliação

Ao utilizar um modelo AM é necessário que existam parâmetros para demonstrar se o modelo está correto ou não. Pois existem possibilidade do modelo ter um *overfitting* que é quando o modelo se ajusta perfeitamente ao conjunto de dados de teste, mas não é capaz de prever novos resultados, já no *underfitting* é quando não há dados suficientes ou com baixa qualidade e não é possível treinar o modelo da maneira correta [33].

Na validação cruzada, os dados são dividos de forma aleatória em n grupos de tamanho aproximadamente iguais. E este processo é realizado por várias vezes, e em cada rodada do teste é considerado um novo valor. O seu desempenho é calculado a partir da média dos desempenhos calculados em cada uma das divisões, o número que n pode assumir é variável, depende da situação, os valores mais utilizados são 5 e 10 [34]

A matriz de confusão é outra forma de verificar a qualidade do modelo, pois ela fornece métricas importantes para medir o desempenho. Essa é baseada em classes relacionadas à assertividade da previsão das classes. Os positivos verdadeiros (VP) e verdadeiros negativos (VN) que são os valores corretos que o sistema previu de forma correta, e também existe outras duas medidas que são os falsos negativos (FN) e falsos positivos (FP) que são as entradas que foram classificadas de forma errada e tiveram sua saída trocada [2]. Na Figura 2.7 é possível visualizar as classes de uma matriz de confusão.

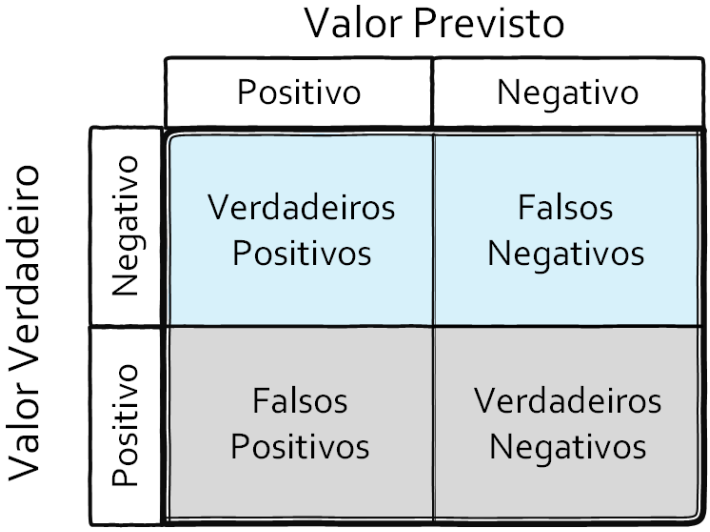

Figura 2.7: Matriz de confusão [2].

Com os resultados extraídos da matriz de confusão é possível calcular outras métricas para verificar o desempenho de cada algoritmo, na Equação 2.13 é calculada a acurácia do modelo, na Equação 2.14 é calculada a precisão do modelo, na Equação 2.15 é calculada a frequência que o modelo encontra os exemplos de uma determinada classe e por fim na Equação 2.16, temos o F1-score que é uma métrica que indica a qualidade geral do modelo.

$$\text{Acurácia} = \frac{VP + VN}{VP + VN + FP + FN} \tag{2.13}$$

$$\text{Precisão} = \frac{VP}{VP + VN + FP + FN} \tag{2.14}$$

$$\text{Recall} = \frac{VP}{VP + FN} \tag{2.15}$$

$$\text{F1-Score} = \frac{2 * \text{Precisão} * \text{Recall}}{\text{Precisão} + \text{Recall}} \tag{2.16}$$

2.2 Mineração de Texto

A TM representa uma parcela da mineração de dados, também pode ser descrita como a forma de descobrir padrões em textos, que vão desde identificar palavras até mesmo sumarizar o texto ou realizar análise de sentimentos de determinada informação. A sua análise tenta recolher o maior número de informações possíveis para encontrar padrões úteis e também pode ser utilizada juntamente com algoritmos de AM. Os textos geralmente são mais difíceis de obter análises, pois em grande parte não são apresentados de forma estruturada, portanto para que o resultado seja satisfatório é necessário uma etapa de pré-processamento dos dados, para que de alguma forma seja possível estruturar o texto.

O corpus é o conjunto de frases que foram rotulados com a sua saída que será utilizada em um modelo de AM, ou seja ele é a base de dados que contém todos os textos disponíveis de forma rotulada.

2.2.1 Processamento de Linguagem Natural

O PLN é uma área de pesquisa que preocupa-se em explorar como um computador pode entender um texto escrito por humanos, com esse tipo de ferramenta é possível realizar análises morfológicas, sintáticas ou léxicas no texto e também extrair informações de um determinado documento [35].

2.2.1.1 Análise de Sentimentos

A análise de sentimentos é um campo que envolve outras ciências além da computação, como a psicologia e linguística, pois é necessário a construção de dicionários para facilitar a classificação de textos, quando por exemplo a intenção é apenas identificar a polaridade de uma palavras, ela poderá ter três saídas conhecidas: Positivo, Negativo e Neutro [15].

Existem diversas formas de fazer o uso da análise de sentimentos, a realização dessas tarefas no idioma em português ainda é muito díficil devido não ter dicionários grandes para realizar o tratamento com um grande volume de dados, também é possível verificar a polaridade de cada frase através da bibliotca [12], mas para isso é necessário traduzir

toda a frase para o idioma inglês através da API aberta da *Google*, o que pode implicar um viés durante o processo de análise de sentimentos.

É possível utilizar um modelo de AM para realizar a análise de sentimentos, onde faz o uso do modelo previamente treinado para realizar a predição da polaridade de novas frases que não existiam durante a fase de treinamento [36].

2.2.1.2 Pré-Processamento

O pré-processamento dos dados é a parte mais importante da análise, pois é a etapa onde irá ser tratada as informações e com isso facilitará o processo de aprendizado da máquina. Onde será indentificado padrões para facilitar a limpeza dos dados, onde pode ser definido que todo o texto deverá estar escrito em letra minúscula, remoção de pontuação e acentos, nessa fase também é possível retirar os textos que não estão no padrão desejado, por exemplo apresentam *emotions* ao invés de texto, remoção de números, ou seja, o interesse dessa etapa é filtrar o texto de forma que a máquina obtenha apenas informações que irão ser utilizadas para treinamento [37].

2.2.1.3 Tokenização

Acontece na etapa de pré-processamento e a sua utilização tem como finalidade extrair pequenas informações de um texto livre. Recebe esse nome, pois cada divisão é nomeada de *token* e que na maioria das vezes é uma palavra que está inserida no texto, mas em outros casos podem ser separados em letras ou duplas ou trincas de palavras [38]. Nesse trabalho foi utilizado o *token* em sua forma clássica utilizando a linguagem de programação *Python* e a biblioteca *NLTK* [39].

Por exemplo, caso tenhamos a frase *"Obina é melhor do que Neymar!"*, teremos sete tokens, conforme exemplo abaixo.

[*Obina*] [é] [*melhor*] [*do*] [*que*] [*Neymar*] [!]

2.2.1.4 Stemming

Essa técnica serve para evitar que palavras que tenham o mesmo radical sejam classificadas de maneira distintas, ou seja, essa técnica tem por objetivo reduzir até a menor parte com significado da palavra que no caso é o radical, portanto no processo de *steeming*, palavras como caminhar, caminhada, caminharam, caminharão resultem na mesma palavra final: caminh. A necessário atentar-se na escolha do algoritmo correto para realizar *steeming*, pois essa etapa está diretamente ligada com o idioma em que o texto está escrito.

2.2.1.5 Stop Words

São os termos não relevantes para a análise do texto, são palavras que na maioria das vezes são conectivos de frases ou preposições, que podem ser consideradas como palavras vazias, não existe uma lista universal de *Stop Words*, mas por padrão é utilizada a lista presente na biblioteca NLTK [39]. Na Tabela 2.1 encontra-se algumas das *Stop Words* utilizadas nesse trabalho.

de	os	tua	tem	estão	da	lhes	essas
e	é	foi	nossas	muito	o	se	tuas
tu	por	as	sua	aquele	entre	não	ele
delas	minhas	às	nos	pela	havia	me	como
ser	aqueles	nossa	vocês	eu	ter	tenho	suas
está	isso	pelos	estes	tinha	depois	foram	este
para	só	quem	deles	isto	um	eles	do
vos	mais	mesmo	num	dele	será	minha	a
no	teus	à	você	em	meus	esses	pelas
com	ao	dela	há	que	na	nosso	te
aos	dos	ou	aquela	era	uma	das	esta
teu	nem	já	até	seja	esse	mas	quando
aquelas	nossos	têm	também	seus	lhe	meu	seu
ela	elas	estas	nós	sem	essa	fosse	qual
		pelo	nas	numa	aquilo		

Tabela 2.1: *Stop Words* utilizadas nesse trabalho

2.2.2 Bag of Words

Os dados utilizados nesse trabalho são textos, ou seja, é um formato que não existe um padrão conhecido, o que dificulta a extração de informações. Por isso é necessário alguma técnica que organize esse conjunto de dados e extraia informações relevantes deles [40].

Portanto é necessário utilizar o *bag of words* para realizar a estruturação do texto que será analisado. No caso mais simples, seria uma matriz que apenas indica se determinado termo existe ou não naquela frase, dessa forma seria uma abordagem booleana. Onde temos duas frases que serão transformadas na matriz a ser utilizada nas próximas etapas.

D_1 = José gosta de assistir os jogos do Flamengo.
D_2 = Maria gosta de assistir filmes

$$\begin{bmatrix} \text{José} & \text{gosta} & \text{assistir} & \text{jogos} & \text{Flamengo} & \text{Maria} & \text{filmes} \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

2.2.3 Term Frequency

A medida de *Term Frequency* é uma outra forma de representar os dados na matriz do modelo estudado, ele considera a quantidade de ocorrência que um determinado termo n_t, o que difere drásticamente do primeiro tipo de *bag of words* [41]. Esse modelo tem uma performance superior ao booleano que foi apresentado na subseção anterior, a sua fórmula matemática é descrita na Equação 2.17.

$$TF_t = \frac{n_t}{N}, \tag{2.17}$$

onde n_t é a frequência que o termo n apareceu dentro do *Corpus* e N é a quantidade total de termos.

2.2.4 Term Frequency - Inverse Document Frequency (TF-IDF)

Uma abordagem mais eficaz é a TF-IDF, pois quando se utiliza a *Term Frequency* ele considera todas as palavras tendo a mesma importância, portanto é necessário normalizar esses valores, para diminuir a influência que esses termos irão ter no processo de análise textual [42]. Para isso é necessário utilizar a equação descrita por [42], onde é apresentada uma função inversa que irá calcular a importância de cada termo. É possível visualizar a Equação 2.18 que é descrita por [42].

$$IDF = \log \frac{N}{d}, \tag{2.18}$$

Onde N é o número total de frases e d é a quantidade de documentos nos quais o termo aparece.

A fórmula final do TF-IDF é obtida a partir do produto da Equação 2.17, que é a frequência de cada termo com a Equação 2.18, que é o a taxa de normalização dos valores. Na Equação 2.19 é possível ver que esse tipo de normalização irá desconsiderar palavras que aparecem em todas as frases do *Corpus*, pois considera que esses termos não são úteis para as etapas seguintes.

$$Tf - IDF = TF * IDF, \tag{2.19}$$

Capítulo 3

Framework Proposto

Neste capítulo, é detalhado o framework proposto para análise preditiva das tendências das eleições presidenciais no Brasil com base na análise de sentimentos dos dados do Twitter. Conforme mostrado na Figura 3.1, o framework é dividido em cinco blocos, que são descritos nas subseções de 3.1 a 3.5.

3.1 Extração de Dados

O primeiro bloco da Figura 3.1, corresponde ao rastreamento e extração de *tweets*. Neste bloco, os tweets são extraídos do banco de dados do *Twiter* disponível na *Internet*. Como a nova versão da API dessa rede social não permite extrair tweets para datas anteriores a uma semana, foi necessário desenvolver um *Web Crawler*, ou seja, um aplicativo usando a linguagem *Python* com a biblioteca *Scrapy*. As datas usadas para a extração de tweets foi uma semana antes do primeiro dia da eleição e no próprio dia da eleição. Este procedimento foi replicado para o segundo turno das eleições. Os dados utilizados nesse trabalho são referentes as eleições de 2014, onde o segundo turno foi disputado entre o candidato Aécio Neves contra a candidata Dilma Roussef.

Todos os tweets foram coletados em um intervalo pré-definido como o argumento do mecanismo de busca de tweets. Como é necessário aplicar as técnicas de Processamento de Linguagem Natural (PLN) em português, a preferência foi dada aos conteúdos escritos naquela língua. Como mencionado no capítulo I, o objetivo deste trabalho é prever as tendências das eleições presidenciais brasileiras com base no *Twitter* para isso que usamos mais de 100.000 tweets para desenvolver o modelo de aprendizado de máquina e quatro algoritmos serão usados para validar o modelo e escolher o melhor a ser utilizado em trabalhos futuros.

Além do texto do *tweet*, algumas outras informações coletadas incluem autor, data, contagem de retweets, contagem de favoritos, localização, menções, hashtags, ID de pu-

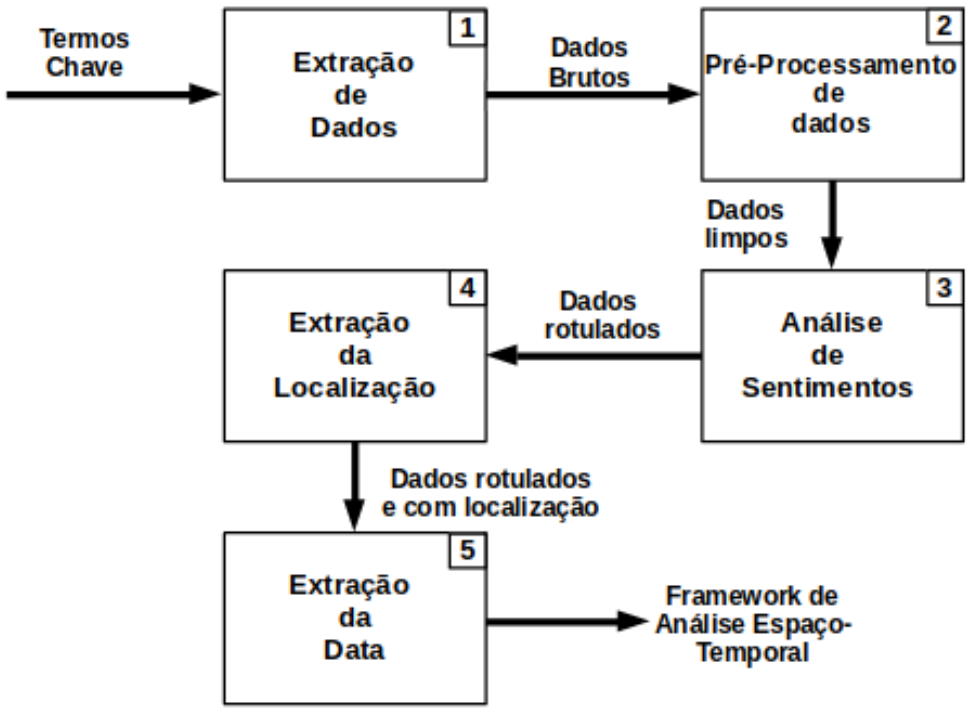

Figura 3.1: Diagrama em blocos do framework proposto para análise preditiva espaço-temporal com base nos dados do Twitter.

blicação e link de publicação. Essas informações em uma estrutura de dados denominada *dataframe*, que é semelhante a uma matriz, mas contém o nome de cada coluna.

Para manipular os dados em formato de *dataframe*, foi utilziada a biblioteca *Pandas*, que é facilita todo o processo de manipulação de dados [43]. Na Figura 3.2 é possível visualizar essa estrutura.

3.2 Pré-processamento de dados

O segundo bloco da Figura 3.1 corresponde ao pré-processamento de dados. As redes sociais apresentam múltiplos públicos e a intensidade da emoção é o fator que diferencia um tweet de outro, onde intensidade se refere ao grau ou quantidade de uma emoção, como positivo, negativo ou neutro. Uma estratégia usual é considerar que todas as mensagens coletadas têm a mesma importância [44]. Não há padrão de escrita definido para ser usado em redes sociais. Consequentemente, foi necessário realizar a limpeza de dados

	username	date	retweets	favorites	text	geo	mentions	hashtags	id
0	RaulDudek	05/10/2014 19:59	0	0	geraldo alckmin reeleito governador sao paulo ...	NaN	NaN	#AECIOdeVirada	5,19E+17
1	wtshatboy	05/10/2014 19:59	1	0	turno dilma aecio dois bosta	NaN	NaN	NaN	5,19E+17
2	RNemitz	05/10/2014 19:59	0	2	dilma rousseff pt aecio neves psdb vao dis...	NaN	NaN	NaN	5,19E+17
3	GeiciSantiago	05/10/2014 19:59	0	0	dilma novo nao ne	NaN	NaN	NaN	5,19E+17
4	michaelserra	05/10/2014 19:59	0	0	vai ser tao apertado votos luciana genro podem...	NaN	NaN	NaN	5,19E+17
5	potterhalder	05/10/2014 19:59	1	0	voces idolatrando aecio minimo nao	NaN	NaN	NaN	5,19E+17

Figura 3.2: Dataframe com os dados utilizados para realizar as análises.

para padronizar as sentenças. Além disso, todos os *emoticons* foram desconsiderados, uma vez que pretendemos analisar o léxico da língua portuguesa e como estamos estamos analisando um processo sério, que são as eleições considerando questões, é necessário que tenhamos um certo padrão a seguir para evitar viés nas análises. O fluxo de limpeza de dados pode ser visto na Figura 3.3.

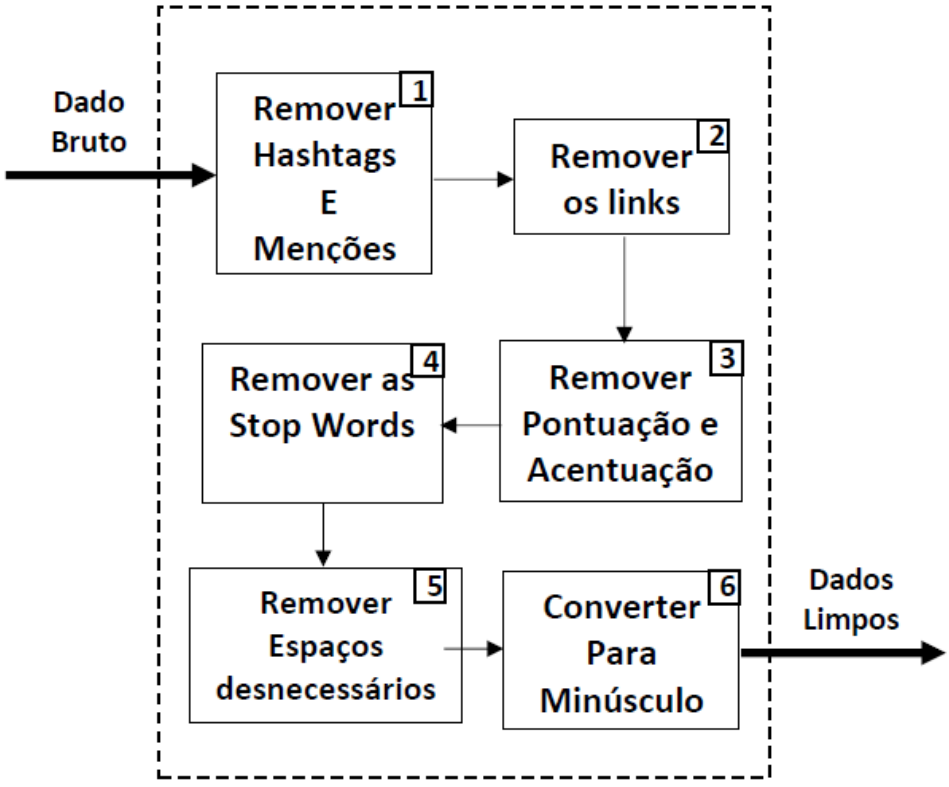

Figura 3.3: Fluxo de pré-processamento de dados.

Tabela 3.1: Limpeza dos textos extraídos do *Twitter*

Etapa Inicial	Mais uma vez @AecioNeves perde em Minas. Não seria a hora de ele se desculpar com o estado, ao invés de continuar. #Eleições https:/bit.ly/twitter
Remoção de Hashtags e Menções a Perfis	Mais uma vez perde em Minas. Não seria a hora de ele se desculpar com o estado, ao invés de continuar. https:/bit.ly/twitter
Remoção de Links	Mais uma vez perde em Minas. Não seria a hora de ele se desculpar com o estado, ao invés de continuar.
Remoção de Pontuação e Acentuação	Mais uma vez perde em Minas Nao seria a hora de ele se desculpar com o estado ao inves de continuar
Remoção de Stopwords	perde em Minas seria hora desculpar estado inves continuar
Remoção de Espaços Desnecessários	perde em Minas seria hora desculpar estado invés continuar
Converter para minúsculo	perde em minas seria hora desculpar estado invés continuar

O tweet bruto foi usado neste processo de limpeza, definimos 6 etapas para sanitizar os dados, todos os blocos descritos nesse parágrafo correspondem ao diagrama de blocos da Figura 3.3 primeiro bloco é responsável por remover tags de retweet e menções de profile, no segundo precisa remover links de sites, o terceiro remover acentos e pontuações, o quarto remover todas as stopwords portuguesas, o quinto remover espaço desnecessário e por último deve-se transformar o tweet para letra minúscula.

As *Stop Words* são freqüentemente usadas em sentenças que desempenham um papel muito pequeno na análise de sentimento e, consequentemente, devem ser removidas [45]. Eles não contribuem para o processo de análise de sentimento e apenas retardam o processo. Além disso, os dados devem ser submetidos a um processo de stemming em que as palavras são reduzidas ao seu radical [44].

Na Tabela 3.1, mostra como foi feita a limpeza de dados para um *tweet* selecionado aleatoriamente do *dataframe* utilizado para análises. Todas as etapas de pré-processamento estão descritas na Figura 3.3

No apêndice A.1 código utilizado para a limpeza e estruturação dos dados é apresentado, foi utilizada expressões regulares para substituir os carácteres especiais que o idioma português tem, foi feito isso para que seja possível realizar o processo de *steeming* na seção 3.5.

Tabela 3.2: Classificação do tweets através da polaridade usando a biblioteca textblob e os dicionários Oplexicon/Sentilex.

	Positivo	Neutro	Negativo
TextBlob	42.67%	24.01%	33.27%
OpLexicon/Sentilex	25.12%	26.51%	48.35%

Tabela 3.3: Análise de sentimentos usando três tweets selecionados de forma aleatória

Tweet	TextBlob	Oplexicon/Sentilex
1	Negativo	Positivo
2	Neutro	Neutro
3	Negativo	Negativo

3.3 Análise de Sentimentos

Neste trabalho usamos as bibliotecas *TextBlob* [12] e OpLexicon [10] para processamento de texto. A Tabela 3.2 mostra as porcentagens de polaridade para cada biblioteca obtida para os tweets extraídos após o pré-processamento. Após a classificação dos dados, é necessário criar dois bancos de dados distintos para que o *TextBlob* e o OpLexicon possam ser comparados usando modelos de aprendizado de máquina.

Na Tabela 3.3 mostra um exemplo da análise de sentimento de uma amostra aleatória com três tweets para comparar os resultados obtidos com as duas bibliotecas.

Em primeiro lugar, tentamos usar a biblioteca *TextBlob* na linguagem Python como a principal fonte de classificação, mas como ela usa um dicionário léxico de palavras inglesas [6]. Então foi necessário traduzir tweets que não estavam em inglês com um método presente na biblioteca e depois verificar a polaridade da frase. Nesse processo, as palavras podem perder seu significado, porque o processo de tradução pode aprensentar um alto viés, visto que o português tem várias maneiras de expressar a mesma ideia.

O OpLexicon em conjunto com o Sentilex com os melhores resultados foi constituído por 30.322 palavras (23.433 adjetivos e 6.889 verbos) e foi baseado no português brasileiro. Foi classificada por sua categoria morfológica marcada com polaridades positivas, negativas e neutras [46].

A automação desse processo é necessária, pois torna-se inviável rotular mais de 100,000 entradas. O código utilizado para análise de sentimentos usando os dicionários pode ser visualizado no Apêndice A.2.

3.4 Extração da Localização e Data

Nessa seção será detalhado como foi o processo de extração de data e localização que foram usados para a construção das análises espaço-temporal desse trabalho, na Figura

3.4 é possível visualizar um exemplo de *tweet* onde aparece a data de publicação e o seu conteúdo.

O quarto bloco da Figura 3.1 corresponde à extração da localização, onde são obtidas informações sobre as coordenadas geográficas do tweet. Como o banco de dados apresenta o nome do autor da publicação, a latitude e a longitude registradas em seu perfil pessoal, quando disponíveis, podem ser obtidas. Permite determinar a localização geográfica dos usuários que apresentam boas ou más opiniões sobre um determinado candidato e, consequentemente, intensificar as ações de marketing a serem tomadas naquela região. Portanto, o objetivo do trabalho é unir a geolocalização com a análise sentimental. Foi utilizada a biblioteca denominada *tweepy* [47], o código na seção A.3 detalha como é feita a requisição para obter os dados referentes a localização do usuário.

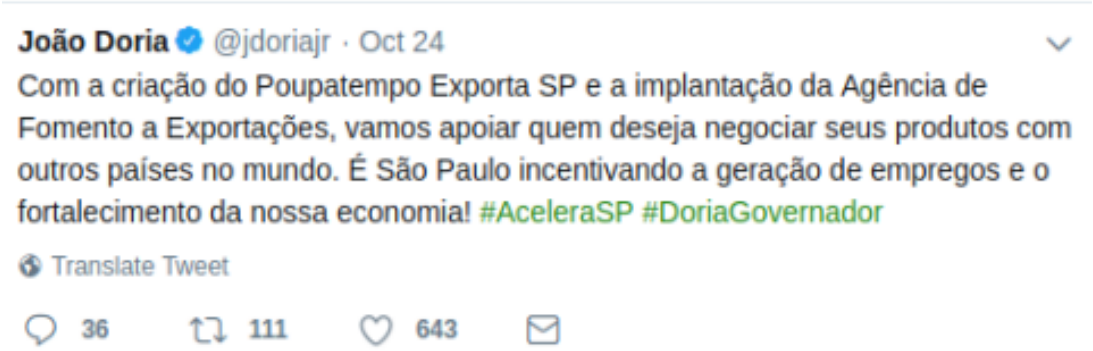

Figura 3.4: Exemplo de um tweet extraído da rede social analisada.

Na Figura 3.5 é apresentada as informações pessoais desse perfil, a localização será extraída desse campo. Essa data será agrupada a cada dia, para que seja possível analisar a série temporal de forma precisa.

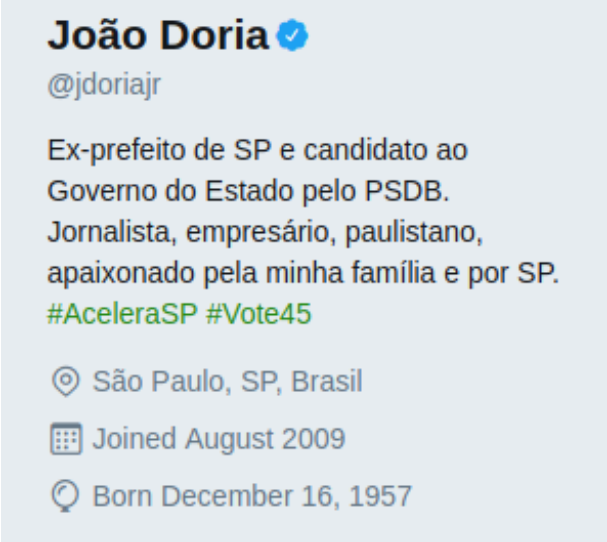

Figura 3.5: Exemplo de um perfil na rede social analisada.

3.5 Aprendizado de Máquina Supervisionado

Após realizar o tratamento dos dados, o texto está pronto para ser utilizado para treinamento. Esse *dataset* será utilizado como treinamento inicial dos classificadores que serão utilizados para analisar os textos futuros.

Os algoritmos utilizados para criar o modelo de AM supervisionado foram detalhados nas seções 2.1.2.1.1 a 2.1.2.1.4, esses algoritmos foram utilizados para classificar novos dados que irão ser extraídos.

A fase treinamento funcionou da seguinte forma, foi utilizado o príncipio de Pareto para realização do treinamento, onde 20% dos dados foram separados para validação do modelo e os outros 80% são usados na fase de treinamento [48].

Foi necessário transformar o *dataframe* obtido ao realizar a extração de dados da rede social em uma matriz utilizando a técnica *Bag-of-Words*, juntamente com a técnica que foi apresentada na seção 2.2.4, onde foi contabilizada a quantidade de informações e normalizado os valores, para que o resultado da análise de sentimentos fosse otimizado.

Após transformar todos de dados disponíveis para o formato vetorial e divisão dos dados entre conjunto de testes e treinamento, utilizou-se os algoritmos biblioteca *Scikit-Learn* que foram detalhados nas Seções 2.1.2.1.1 a 2.1.2.1.4 [23].

O primeiro algoritmo a ser utilizado na fase de treinamento foi o SVM, que segundo a literatura é o algoritmo que melhor tem resultados para classificação de textos. Nessa etapa utilizou-se o kernel linear, e utilizou-se de funções de *gridsearch* para encontrar o valor da constante C que é uma variável que de normalização para minimizar os erros no conjunto de treinamento e obter as melhores métricas de análise [25]. O valor encontrado

foi para $C = 100$ e para otimizar a fase de treinamentos, foi utilizada uma função de paralelismo para que o treinamento não fosse longo e consumisse muitos recursos computacionais.

O algoritmo de Naive Bayes (NB) foi o segundo a ser utilizado para treinamento. A implementação desse algoritmo acontece de forma simples, pois é computada apenas a probabilidade condicional para cada valor de saída a partir do texto de entrada.

As árvores de decisão foi o classificador que mais consumiu recursos e tempo na fase de treinamento, pois o dataset apresenta muitas linhas para análise e a variável de entrada pode ter até 140 caracteres, mesmo otimizando os dados através dos processos apresentados na Seção 3.2, a árvore construída teve uma alta profundidade tornando-se extremamente custoso ao computador que realiza o treinamento.

Por último utilizou-se a regressão logística que o tempo de treinamento é considerável ao ser comparado com a árvore de decisão e esse classificador, também já tem uma função implementada na biblioteca utilizada nessa etapa.

Capítulo 4

Resultados

Esta seção apresenta a validação do framework proposto. Comparamos as previsões do framework com os resultados das eleições presidenciais de 2014 extraídos do banco de dados do Tribunal Superior Eleitoral Brasileiro. Analisamos apenas os resultados referentes aos dois candidatos com maior número de votos obtidos no primeiro turno, Dilma Rousseff e Aécio Neves, que consequentemente se classificaram para o segundo turno

4.1 Avaliação da Performance dos Algoritmos

O desempenho dos algoritmos Naive Bayes (NB), Máquina de Vetores de Suporte (SVM), Regressão Logística (LR) e Árvores de Decisão (DT) foram avaliados através das seguintes métricas: precisão, F1-Score, recall e precisão. Essas informações podem ser visualizadas na Tabela 4.1. O algoritmo de árvore de decisão apresentou os piores resultados em todas as métricas avaliadas e também apresentou maior custo computacional. Os algoritmos com os melhores resultados em termos de precisão e custo computacional foi o SVM, pois foi utilizada a função de *gridsearch* para diminuir os erros. Os dados de texto são ideais para classificação de SVM devido à natureza esparsa do texto, em que poucos recursos são irrelevantes, mas tendem a ser correlacionados entre si e geralmente organizados em categorias linearmente separáveis [49].

4.2 Validação Cruzada N-Fold

Na validação cruzada de $N - fold$, o conjunto de dados é particionado em subconjuntos mutuamente exclusivos de N. Um subconjunto é usado como dados de validação para testar o modelo, e os subconjuntos $N1$ restantes são usados como dados de treinamento. Como o conjunto de dados tem mais de 100.000 *tweets*, a validação cruzada

Tabela 4.1: Performance dos classificadores utilizados

Métricas	Classificador	TextBlob	OpLexicon/Sentilex
Acurácia	NB	0.82	0.93
	SVM	0.94	0.98
	LR	0.70	0.65
	DT	0.64	0.85
Precisão	NB	0.83	0.92
	SVM	0.94	0.98
	LR	0.79	0.83
	DT	0.69	0.89
Recall	NB	0.79	0.92
	SVM	0.93	0.97
	LR	0.60	0.58
	DT	0.56	0.81
F1-score	NB	0.80	0.92
	SVM	0.94	0.98
	LR	0.55	0.60
	DT	0.55	0.84

de $N - fold$ apresentaria um alto custo computacional se um grande valor de N fosse escolhido. Consequentemente, usamos $N = 5$. Os dados do teste podem ser visualizados na Tabela 4.2. Usando validação cruzada de 5 vezes, obtivemos resultados semelhantes na primeira análise e o SVM permaneceu como o melhor algoritmo para mineração de texto em português.

Tabela 4.2: Validação cruzada $N - Fold$ com $N = 5$

Métrica	Algorithm	TextBlob	OpLexicon/Sentilex
Acurácia	NB	0.81	0.92
	SVM	0.93	0.98
	LR	0.86	0.66
	DT	0.64	0.84
Precisão	NB	0.82	0.92
	SVM	0.92	0.98
	LR	0.78	0.83
	DT	0.70	0.88
Recall	NB	0.86	0.94
	SVM	0.93	0.98
	LR	0.90	0.80
	DT	0.54	0.81
F1-score	NB	0.79	0.91
	SVM	0.92	0.98
	LR	0.54	0.58
	DT	0.56	0.84

Tabela 4.3: Distribuição dos resultados da eleição presidencial de 2014

Dilma Roussef	51,64%
Aécio Neves	48,36%

4.3 Avaliação do erro utilizando o modelo

4.4 Resultado das Eleições

Finalmente, nessa seção ocorre a comparação entre os resultados obtidos pelo classificador e os dados reais das eleições anteriores fornecidas pelo Tribunal Superior Eleitoral brasileiro [50] para obter a previsão eleitoral. Como estamos observando apenas as eleições presidenciais, os candidatos em análise são aqueles que obtiveram, no primeiro turno, votos suficientes para participar do segundo turno.

A Tabela 4.3 mostra um exemplo da distribuição dos votos dos candidatos às eleições presidenciais brasileiras de 2014, Aécio Neves e Dilma Rousseff, obtidos do banco de dados do Tribunal Superior Eleitoral.

4.5 Análise Espaço-Temporal

Usando a análise temporal, é possível observar a intensidade de cada candidato no dia da eleição, e serviu para demonstrar como as reações dos brasileiros estavam acompanhando os resultados do Tribunal Superior Eleitoral em tempo real. E na Figura 4.1

apresenta a quantidade de tweets classificados em três categorias para os candidatos Dilma Roussef e Aécio Neves.

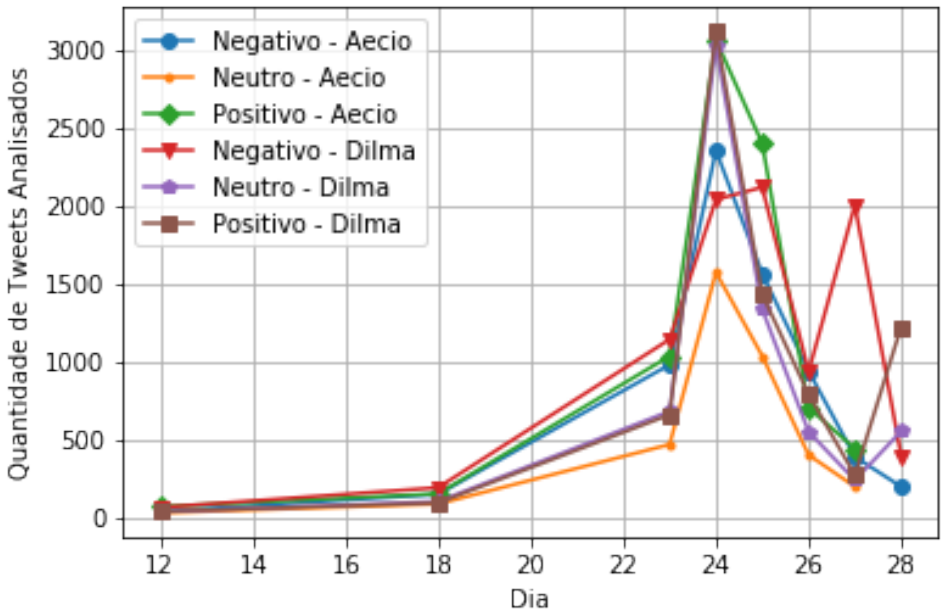

Figura 4.1: Série temporal utilizando os tweets classificados durante o segundo turno com suas polaridades.

O segundo turno das eleições presidenciais de 2014, ocorreu no dia 26 de outubro. É possível visualizar na Figura 4.1 que nesse dia a quantidade de informações rotuladas como positiva para a candidata Dilma Roussef é superior com uma pequena distância em relação ao oponente Aécio Neves. A queda na popularidade da candidata que ganhou as eleições está ligado ao fato da operação da Polícia Federal, denominada Lava Jato, onde vários político foram presos.

É possível visualizar o aumento de mensagens de teor negativo a partir do dia 24 de outubro de 2014, isso se dá ao fato do doleiro responsável pelo esquema de corrupção desse governo ter realizado declarações de que a presidente Dilma Roussef e o ex-presidente Lula tinham conhecimento de todo o esquema de corrupção na empresa estatal Petrobras e no dia 28 o executivo de uma empresa envolvida em esquemas de proprina ter fechado um acordo de delação premiada com os procuradores da força tarefa da operação [51].

A localização dos votos de cada seção eleitoral são disponibilizada pelo TSE combinadas com o quantitativo de votos na Figura 6. As cores vermelha e azul representam, respectivamente, as regiões onde os candidatos Dilma Rousseff e Aécio Neves tiveram maior número de votos nas eleições presidenciais de 2014.

A idéia principal é comparar com um local extraído do *twitter* conjuntamente com a análise de sentimentos. Utilizando a geolocalização extraída é possível gerar o mapa com

a localização e o envio dos *tweets* daquela região. Na Figura 7 é apresentada a distribuição da polaridade do tweet para a candidata Dilma Roussef para cada Estado, a cor verde significa um sentimento positivo, o laranja é neutro e o vermelho representa os *tweets* negativos.

4.6 Plataforma de Análise de Sentimentos *Online*

Capítulo 5

Conclusão

Neste trabalho foi desenvolvido um *framework* capaz de analisar dados extraídos de uma rede social utilizando técnicas de processamento de linguagem natural. Nosso objetivo foi explorar a análise de sentimento em textos em língua portuguesa e validar os dados, cruzando os dados disponíveis pelo Tribunal responsável pela condução das eleições no Brasil.

Foi possível verificar nesse trabalho que quanto melhor for o pré-processamento dos dados melhor será o resultado final. Durante o trabalho foram feitos vários ajustes nessa etapa para que o dado chegasse ao classificador da melhor maneira possível.

Para as eleições de 2014, os modelos propostos conseguiram prever de forma eficaz o resultado daquela eleição, foi uma eleição onde grande parte da sociedade esteve crítica a reeleição daquele governo, portanto é possível entender o elevado número de *tweets* negativos naquele cenário. E dois anos após a reeleição, a então presidente eleita sofreu um *impeachment*, assim então validando todo o nosso estudo no que diz respeito a identificação de texto com a polaridade negativa.

Ao comparar os resultados obtidos pela análise de sentimento com os resultados que os dicionários proporcionaram e comparar os dados do Tribunal Superior Eleitoral, foi possível constatar que o *framework* teve um bom resultado.

Ademais o trabalho propõe estudar outras áreas além da política, pois tem o intuito de aquecer as pesquisas na área de análise de sentimentos no idioma português.

Após a utilização de quatro classificadores distintos para a criação de modelos de aprendizado de máquina, podemos dizer que o algoritmo que teve melhores resultados foi o SVM que obteve mais de 90% de acurácia na fase de validação do modelo e árvore de decisões foi o que se mostrou ineficiente para essa tarefa de análise de sentimento de textos provenientes de redes sociais.

De contribuições ao meio científico, entende-se que o desenvolvimento desse *framework* e do modelo treinado facilita pesquisas nessa áreas e um artigo foi publicado para que

outras pessoas possam replicar as etapas que foram aplicadas nesse trabalho.

5.1 Trabalhos Futuros

Referências

[1] W3Techs: *Historical trends in the usage of content languages for websites.* `https://w3techs.com/technologies/history_overview/content_language`. ix, 2

[2] Andreoni, Martin: *An intrusion detection and prevention architecture for software defined networking*, abril 2014. ix, 15

[3] Song, M., M. C. Kim e Y. K. Jeong: *Analyzing the political landscape of 2012 Korean presidential election in Twitter.* IEEE Intelligent Systems, 29(2):18–26, 2014. 1, 4

[4] Araniti, G., I. Bisio e M. De Sanctis: *Towards the reliable and efficient interplanetary internet: A survey of possible advanced networking and communications solutions.* Em *2009 First International Conference on Advances in Satellite and Space Communications*, páginas 30–34, July 2009. 1

[5] Stats, Internet Live: *Elaboration of data by international telecommunication union (itu), united nations population division, internet mobile association of india (iamai), world bank.* `http://www.internetlivestats.com/internet-users-by-country/`. 1

[6] Miller, G. A.: *Wordnet: a lexical database for English.* Communications of the ACM, 38(11):39–41, 1995. 2, 24

[7] Twitter: *Twitter developer platform.* `https://developer.twitter.com`. 2

[8] Comunicação, Empresa Brasil de: *Facebook chega a 127 milhões de usuários no brasil.* `http://agenciabrasil.ebc.com.br/economia/noticia/2018-07/facebook-chega-127-milhoes-de-usuarios-no-brasil`. 2

[9] País, El: *Cambridge analytica, empresa pivô no escândalo do facebook, é fechada.* `https://brasil.elpais.com/brasil/2018/05/02/internacional/1525285885_691249.html`. 2

[10] Souza, Marlo e Renata Vieira: *Sentiment analysis on twitter data for portuguese language.* Em *Proceedings of the 10th International Conference on Computational Processing of the Portuguese Language*, PROPOR'12, páginas 241–247, Berlin, Heidelberg, 2012. Springer-Verlag, ISBN 978-3-642-28884-5. `http://dx.doi.org/10.1007/978-3-642-28885-2_28`. 3, 24

[11] Neuenschwander, Bruna, Adriano C.M. Pereira, Wagner Meira, Jr. e Denilson Barbosa: *Sentiment analysis for streams of web data: A case study of brazilian financial markets.* Em *Proceedings of the 20th Brazilian Symposium on Multimedia*

and the Web, WebMedia '14, páginas 167–170, New York, NY, USA, 2014. ACM, ISBN 978-1-4503-3230-9. http://doi.acm.org/10.1145/2664551.2664579. 3

[12] Loria, S.: *TextBlob: Simplified text processing*. http://textblob.readthedocs.io/en/dev/index.html. 3, 16, 24

[13] Wagh, Rasika e Payal Punde: *Survey on sentiment analysis using twitter dataset*. Em *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, páginas 208–211. IEEE, 2018. 3

[14] Wang, Yuanhong, Yang Liu e Xiaohui Yu: *Collaborative filtering with aspect-based opinion mining: A tensor factorization approach*. Em *Data Mining (ICDM), 2012 IEEE 12th International Conference on*, páginas 1152–1157. IEEE, 2012. 4

[15] Wegrzyn-Wolska, K. e L. Bougueroua: *Tweets mining for French presidential election*. Em *2012 Fourth International Conference on Computational Aspects of Social Networks (CASoN)*, páginas 138–143, Nov 2012. 4, 16

[16] Cerón-Guzmán, J. A. e E. León-Guzmán: *A sentiment analysis system of Spanish tweets and its application in Colombia 2014 presidential election*. Em *2016 IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom) (BDCloud-SocialCom-SustainCom)*, páginas 250–257, Oct 2016. 4

[17] Joyce, B. e J. Deng: *Sentiment analysis of tweets for the 2016 US presidential election*. Em *2017 IEEE MIT Undergraduate Research Technology Conference (URTC)*, páginas 1–4, Nov 2017. 4

[18] Nasrabadi, Nasser M: *Pattern recognition and machine learning*. Journal of electronic imaging, 16(4):049901, 2007. 6, 12

[19] Bonaccorso, Giuseppe: *Machine Learning Algorithms*. Packt Publishing Ltd, 2017. 6, 13

[20] Marks, II, Robert J., Jacek M. Zurada e Charles J. Robinson (editores): *Computational Intelligence: Imitating Life*. IEEE Press, Piscataway, NJ, USA, 1994, ISBN 0780311043. 6

[21] Mitchell, Thomas M.: *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1ª edição, 1997, ISBN 0070428077, 9780070428072. 7

[22] Chang, Chih Chung e Chih Jen Lin: *Libsvm: a library for support vector machines*. ACM transactions on intelligent systems and technology (TIST), 2(3):27, 2011. 7

[23] Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg *et al.*: *Scikit-learn: Machine learning in python*. Journal of machine learning research, 12(Oct):2825–2830, 2011. 7, 8, 26

[24] Vieira, Lucas Maciel, Clicia Grativol, Flavia Thiebaut, Thais G Carvalho, Pablo R Hardoim, Adriana Hemerly, Sergio Lifschitz, Paulo Cavalcanti Gomes Ferreira e Maria Emilia MT Walter: *Plantrna_sniffer: a svm-based workflow to predict long intergenic non-coding rnas in plants*. Non-coding RNA, 3(1):11, 2017. 9

[25] Lorena, Ana Carolina e André CPLF de Carvalho: *Uma introdução às support vector machines*. Revista de Informática Teórica e Aplicada, 14(2):43–67. 9, 26

[26] McCallum, Andrew e Kamal Nigam: *A comparison of event models for naive bayes text classification*, 1998. 10

[27] Zhang, Harry: *The optimality of naive bayes*. AA, 1(2):3, 2004. 11

[28] Coelho, Vinícius Coutinho Guimarães: *Análise de logs de interação em ambiente educacional corporativo via mineração de dados educacionais*. 11

[29] Breiman, Leo: *Classification and regression trees*. Routledge, 2017. 11

[30] MacQueen, James *et al.*: *Some methods for classification and analysis of multivariate observations*. Em *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1 de *1*, páginas 281–297. Oakland, CA, USA, 1967. 13

[31] Zurada, Jacek M.: *Introduction to Artificial Neural Systems*. PWS Publishing Co., Boston, MA, USA, 1st edição, 1999, ISBN 053495460X. 14

[32] Mazurowski, Maciej A, Piotr A Habas, Jacek M Zurada, Joseph Y Lo, Jay A Baker e Georgia D Tourassi: *Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance*. Neural networks, 21(2-3):427–436, 2008. 14

[33] Aalst, Wil MP Van der, Vladimir Rubin, HMW Verbeek, Boudewijn F van Dongen, Ekkart Kindler e Christian W Günther: *Process mining: a two-step approach to balance between underfitting and overfitting*. Software & Systems Modeling, 9(1):87, 2010. 14

[34] Kohavi, Ron *et al.*: *A study of cross-validation and bootstrap for accuracy estimation and model selection*. Em *Ijcai*, volume 14, páginas 1137–1145. Montreal, Canada, 1995. 15

[35] Liddy, Elizabeth D: *Natural language processing*. 2001. 16

[36] Golbeck, Jennifer, Justin M Grimes e Anthony Rogers: *Twitter use by the us congress*. Journal of the American Society for Information Science and Technology, 61(8):1612–1621, 2010. 17

[37] Ferreira, Lohann Paterno Coutinho, Mariza Miola Dosciatti e Emerson Cabrera: *Um método para o pré-processamento de textos na identificação automática de emoções para o português do brasil*. 17

[38] Mcnamee, Paul e James Mayfield: *Character n-gram tokenization for european language text retrieval*. Information retrieval, 7(1-2):73–97, 2004. 17

[39] Bird, Steven e Edward Loper: *Nltk: the natural language toolkit*. Em *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, página 31. Association for Computational Linguistics, 2004. 17, 18

[40] Wallach, Hanna M: *Topic modeling: beyond bag-of-words*. Em *Proceedings of the 23rd international conference on Machine learning*, páginas 977–984. ACM, 2006. 18

[41] Salton, Gerard e Christopher Buckley: *Term-weighting approaches in automatic text retrieval*. Information processing & management, 24(5):513–523, 1988. 19

[42] Sparck Jones, Karen: *A statistical interpretation of term specificity and its application in retrieval*. Journal of documentation, 28(1):11–21, 1972. 19

[43] McKinney, Wes: *pandas: a foundational python library for data analysis and statistics*. Python for High Performance and Scientific Computing, páginas 1–9, 2011. 21

[44] Oliveira, Derick M de, Roberto CSNP Souza, Denise EF de Brito, Wagner Meira Jr, Gisele L Pappa e Belo Horizonte-MG-Brasil: *Uma estratégia não supervisionada para previsão de eventos usando redes sociais.* Em *SBBD*, páginas 137–148, 2015. 21, 23

[45] Sharma, Anuj e Shubhamoy Dey: *A comparative study of feature selection and machine learning techniques for sentiment analysis*. Em *Proceedings of the 2012 ACM Research in Applied Computation Symposium*, RACS '12, páginas 1–7, New York, NY, USA, 2012. ACM, ISBN 978-1-4503-1492-3. `http://doi.acm.org/10.1145/2401603.2401605`. 23

[46] Souza, M., R. Vieira, D. Busetti, R. Chishman e I. M. Alves: *Construction of a portuguese opinion lexicon from multiple resources*. Em *Proceedings of the 8th Brazilian Symposium in Information and Human Language Technology*, 2011. 24

[47] Roesslein, Joshua: *tweepy documentation*. Online] http://tweepy. readthedocs. io/en/v3, 5, 2009. 25

[48] Jin, Yaochu e Bernhard Sendhoff: *Pareto-based multiobjective machine learning: An overview and case studies*. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 38(3):397–415, 2008. 26

[49] Medhat, Walaa, Ahmed Hassan e Hoda Korashy: *Sentiment analysis algorithms and applications: A survey*. Ain Shams Engineering Journal, 5(4):1093 – 1113, 2014, ISSN 2090-4479. `http://www.sciencedirect.com/science/article/pii/S2090447914000550`. 28

[50] Eleitoral Tribunal Superior: *Estatísticas Eleitorais: eleições anteriores*. `http://www.tse.jus.br/eleitor-e-eleicoes/estatisticas/eleicoes/eleicoes-anteriores/estatisticas-eleitorais-anos-anteriores`. 30

[51] São Paulo, Folha de: *Entenda a operação lava jato, da polícia federal*. https://www1.folha.uol.com.br/poder/2014/11/1548049-entenda-a-operacao-lava-jato-da-policia-federal.shtmls, Acessado em 20.10.2018. 31

Apêndice A

A.1 CÃşdigo para limpeza dos dados

```
import pandas as pd
import numpy as np
from nltk.tokenize import WordPunctTokenizer
tok = WordPunctTokenizer()

def removestopwords(texto):
    frases = []

    for palavras in texto:

        palavras = palavras.lower()
        palavras = palavras.replace('?',' ')
        palavras = palavras.replace('#','')
        palavras = palavras.replace('!',' ')
        palavras = palavras.replace('%',' ')
        palavras = palavras.replace('. ',' ')
        palavras = palavras.replace(')',' ')
        palavras = palavras.replace('(',' ')
        palavras = palavras.replace('-',' ')
        palavras = palavras.replace(',',' ')
        palavras = palavras.replace('/ ',' ')
        palavras = palavras.replace('*',' ')
        palavras = palavras.replace('=',' ')
        palavras = palavras.replace(':','')

        palavras = palavras.replace('r\$',' ')

        palavras = palavras.replace('Ãą','a')
        palavras = palavras.replace('Ãã','a')
        palavras = palavras.replace('Ãć','a')
        palavras = palavras.replace('Ãč','a')
```

```
        palavras = palavras.replace('Ãľ','e')
        palavras = palavras.replace('ÃÍ','e')
        palavras = palavras.replace('Ãł','e')


        palavras = palavras.replace('Ãŋ','i')
        palavras = palavras.replace('Ãň','i')
        palavras = palavras.replace('Ãő','i')

        palavras = palavras.replace('Ãş','o')
        palavras = palavras.replace('Ãš','o')
        palavras = palavras.replace('Ãť','o')
        palavras = palavras.replace('Ãţ','o')

        palavras = palavras.replace('Ãž','u')
        palavras = palavras.replace('Ãź','u')
        palavras = palavras.replace('Ãż','u')

        palavras = palavras.replace('Ãğ','c')

        frases.append(palavras)


    return frases

dataset['text'] = removestopwords(dataset['text'])
```

A.2 CÃşdigo para anÃąlise de sentimentos usando dicionÃąrio

```
dicionario = open("dicionario.txt", 'r')

# Criamos um dicionÃąrio para receber as informaÃğÃţes
dic_palavra_polaridade = {}

# Neste trecho atribuÃŋmos a cada palavra uma Polaridade
for i in dicionario.readlines():
    pos_ponto = i.find('.')
    palavra = (i[:pos_ponto])
    pol_pos = i.find('POL')
```

```
    polaridade = (i[pol_pos+4:pol_pos+6]).replace(';','')
    dic_palavra_polaridade[palavra] = polaridade


def PontuaÃğÃčo_sentimento(frase):
    frase = frase.lower()
    l_sentimento = []

    for p in frase.split():
        l_sentimento.append(int(dic_palavra_polaridade.get(p, 0)))

    if sum(l_sentimento) > 0:
        return 'Positivo, PontuaÃğÃčo: {}'.format(sum(l_sentimento))
    elif sum(l_sentimento) == 0:
        return 'Neutro, PontuaÃğÃčo:{}'.format(sum(l_sentimento))
    elif sum(l_sentimento) < 0:
        return 'Negativo, PontuaÃğÃčo:{}'.format(sum(l_sentimento))


frase = PontuaÃğÃčo_sentimento('Sua mÃče Ãľ feliz')
print(frase)
```

A.3 CÃşdigo para extraÃğÃčo da localizaÃğÃčo do usuÃąrio

```
    # coding: utf-8
    import pandas as pd
    import tweepy

    consumer_key = ""

    consumer_secret = ""

    access_token_secret = ""

    access_token = ""

    auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_token, access_token_secret)

    api = tweepy.API(auth)
```

```
    location = []
    for tweet in entrada['username']:
        try:
            user = api.get_user(tweet)
            print(user.location)
            location.append(user.location)
        except tweepy.TweepError as e:
            location.append('Error')

    entrada['local'] = location
    entrada.to_csv(saida, sep=';', mode='a',index=False)
```

A.4 CÃşdigo para geraÃğÃčo da sÃľries temporais

```
import pandas as pd      # To handle data
import numpy as np       # For number computing

# For plotting and visualization:
from IPython.display import display
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

dataset=pd.read_csv("/home/bruno/Documents/artigo/Dados Brutos/
                                    Dilma_final_prog1.csv",delimiter=","
                                    ,encoding='latin-1').set_index('Date
                                    ')
dataset1=pd.read_csv("/home/bruno/Documents/artigo/Dados Brutos/
                                    Aecio_final_prog1.csv",delimiter=","
                                    ,encoding='latin-1').set_index('Date
                                    ')

positivo = dataset.loc[dataset['Sentiment'] == 'Positive']
negativo = dataset.loc[dataset['Sentiment'] == 'Negative']
neutro  = dataset.loc[dataset['Sentiment'] == 'Neutral']

positivo1 = dataset1.loc[dataset1['Sentiment'] == 'Positive']
negativo1 = dataset1.loc[dataset1['Sentiment'] == 'Negative']
neutro1  = dataset1.loc[dataset1['Sentiment'] == 'Neutral']

positivo.index = pd.to_datetime(positivo.index)
negativo.index = pd.to_datetime(negativo.index)
neutro.index = pd.to_datetime(neutro.index)
```

```
positivo1.index = pd.to_datetime(positivo1.index)
negativo1.index = pd.to_datetime(negativo1.index)
neutro1.index = pd.to_datetime(neutro1.index)

positivo = positivo.rename(columns={"Sentiment":"Positive - Dilma"})
positivo1 = positivo1.rename(columns={"Sentiment":"Positive - Aecio"})
positivo_plot = positivo['Positive - Dilma'].groupby( [positivo.index.
                                    day] ).count()
positivo_plot1 = positivo1['Positive - Aecio'].groupby( [positivo1.index
                                    .day] ).count()

negativo = negativo.rename(columns={"Sentiment":"Negative - Dilma"})
negativo1 = negativo1.rename(columns={"Sentiment":"Negative - Aecio"})
negative_plot = negativo['Negative - Dilma'].groupby( [negativo.index.
                                    day] ).count()
negative_plot1 = negativo1['Negative - Aecio'].groupby( [negativo1.index
                                    .day] ).count()

neutro = neutro.rename(columns={"Sentiment":"Neutral - Dilma"})
neutral_plot = neutro['Neutral - Dilma'].groupby( [neutro.index.day] ).
                                    count()
neutro1 = neutro1.rename(columns={"Sentiment":"Neutral - Aecio"})
neutral_plot1 = neutro1['Neutral - Aecio'].groupby( [neutro1.index.day]
                                    ).count()

#CriaÃğÃčo do GrÃąfico Temporal
import matplotlib.dates as mdates
negative_plot1.plot(marker='o')
neutral_plot1.plot(marker='.')
positivo_plot1.plot(marker='D')


negative_plot.plot(marker='v')
neutral_plot.plot(marker='p')
positivo_plot.plot(marker='s')

plt.ylabel("Sentiment Count")
plt.xlabel("Day")
plt.legend()
plt.grid(True)
```