UNIVERSIDADE FEDERAL RURAL DO RIO DE JANEIRO
INSTITUTO MULTIDISCIPLINAR

VICTOR DINIZ AUGUSTO ANDRADE

Detecção Automática de Discurso de Ódio em Textos do Twitter

Prof. Filipe Braida do Carmo, D.Sc.
Orientador

Nova Iguaçu, Maio de 2021

Detecção Automática de Discurso de Ódio em Textos do Twitter

Victor Diniz Augusto Andrade

Projeto Final de Curso submetido ao Departamento de Ciência da Computação do Instituto de Multidisciplinar da Universidade Federal Rural do Rio de Janeiro como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Apresentado por:

_____

Victor Diniz Augusto Andrade

Aprovado por:

_____

Prof. Filipe Braida do Carmo, D.Sc.

_____

Prof. Leandro Guimaraes Marques Alvim, D.Sc.

_____

Prof. Bruno José Dembogurski, D.Sc.

NOVA IGUAÇU, RJ - BRASIL

Maio de 2021

Agradecimentos

Primeiramente, agradeço a Deus por sempre estar comigo em tudo o que faço.

Agradeço também aos meus pais, Mônica Maria e José de Alencar, que sempre fizeram de tudo por mim e, principalmente, pela minha educação.

Agradeço pela presença do Instituto Multidisciplinar da Universidade Federal Rural do Rio de Janeiro no município de Nova Iguaçu.

Meus agradecimentos ao Departamento de Ciência da Computação e, em especial, ao professor Filipe Braida por ter me motivado do início ao fim e ter me oferecido todo o suporte necessário para a conclusão deste trabalho.

À toda minha família por todo apoio e, em especial, à minha avó Severina por se preocupar tanto comigo, à minha tia Rosemary por ter me inspirado a cursar o ensino superior e à minha prima Thainá que sempre esteve ao meu lado em todos os momentos;

Aos meus irmãos, Matheus Ramos e Thaís Leotério, por tudo o que representam na minha vida;

Aos meus amigos Bruno Martins, Gabriela Gomes, Júlia Rodrigues, Matheus Abreu e, em especial, à Natália Francisco por me acompanharem nessa jornada;

Ao CEFET/RJ *campus* Nova Iguaçu por tudo o que me deu e, em especial, aos queridos Ana Carolina, Cesar Renato e Luane Fragoso;

Aos integrantes do grupo Waddles por me ajudarem a manter o equilíbrio;

Meus sinceros agradecimentos.

RESUMO

Detecção Automática de Discurso de Ódio em Textos do Twitter

Victor Diniz Augusto Andrade

Maio/2021

Orientador: Filipe Braida do Carmo, D.Sc.

Atualmente, as mídias sociais contam com um número muito expressivo de usuários ativos em todo o mundo, os quais exploram os diferentes recursos de interação e participação existentes. Esses recursos interativos foram introduzidos a partir da versão 2.0 da tecnologia *World Wide Web* que possibilitou a criação e o compartilhamento de mídia espontânea na Internet. Entretanto, apesar do propósito muito positivo dessa tecnologia, um fenômeno conhecido como discurso de ódio ou, em inglês, *hate speech* passou a ser facilmente disseminado nas redes devido à velocidade e ao alcance global alcançados por meio das interações virtuais. Nesse contexto, o objetivo deste trabalho é atuar na análise de uma estratégia de identificação desse tipo de discurso baseada em aprendizado de máquina. Isso porque o discurso de ódio se torna mais prejudicial na medida em que permanece online. Sendo assim, conseguir identificá-lo é um passo importante para lidar com o problema. Propõe-se, então, o desenvolvimento de uma ferramenta de detecção do discurso de ódio online através da análise experimental de diferentes modelos de classificação textual baseados em aprendizado de máquina. O processo experimental inclui a combinação de dois conjuntos de dados diferentes e o refinamento dos hiperparâmetros de um dos algoritmo de classificação. A média dos melhores resultados alcançados na tarefa de identificar o discurso de ódio, de acordo com a métrica F1, foi de $0,58$.

ABSTRACT

Detecção Automática de Discurso de Ódio em Textos do Twitter

Victor Diniz Augusto Andrade

Maio/2021

Advisor: Filipe Braida do Carmo, D.Sc.

Currently, social media has a very expressive number of active users worldwide, who exploit the different resources of interaction and participation that exist. These interactive features were introduced starting with version 2.0 of the World Wide Web technology, which enabled the creation and sharing of spontaneous media on the Internet. However, despite the very positive purpose of this technology, a phenomenon known as hate speech started to be easily disseminated on networks due to the speed and global reach achieved through virtual interactions. In this context, the objective of this work is to analyze an identification strategy for this type of discourse based on machine learning. That's because hate speech becomes more damaging as it stays online. Therefore, being able to identify it is an important step in dealing with the problem. Then is proposed the development of an online hate speech detection tool through the experimental analysis of different textual classification models based on machine learning. The experimental process includes combining two different data sets and refining the hyperparameters of one of the classification algorithms. The average of the best results achieved in the task of identifying hate speech, according to the F1 metric, was 0.58.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

TF	Term Frequency
BoW	Bag of Words
TF-IDF	Term Frequency - Inverse Document Frequency
CBOW	Continuous Bag of Words
PLN	Processamento de Linguagem Natural
SVM	Support Vector Machines
LSTM	Long Short-Term Memory
CNN	Redes Neurais Convolucionais

Sumário

Capítulo 1

Introdução

É notório o papel e o impacto que as mídias sociais possuem hoje na sociedade. A cada 60 segundos, 510.000 comentários são gerados no *Facebook*[1] e cerca de 350.000 *tweets* são publicados no *Twitter*[2] segundo Gaydhani et al. (2018). Essa grande quantidade de conteúdo gerada em um curto período de tempo somada ao anonimato e mobilidade intrínsecos ao meio de comunicação em questão criou um ambiente muito favorável para que o discurso de ódio ganhasse uma projeção significativa na Internet (ZHANG; LUO, 2018).

Os ataques odientos são direcionados a pessoas ou grupos de pessoas com base em estereótipos de suas características e atributos. Esse fenômeno, apesar de ter obtido grande repercussão através da Internet, era disseminado anteriormente em meios de comunicação mais tradicionais como em transmissões de televisão e rádio ou jornais (CHETTY; ALATHUR, 2018). Esses ataques podem ser definidos de duas outras perspectivas: eles estigmatizam o alvo, de forma explícita ou não, atribuindo-lhe qualidades consideradas amplamente indesejáveis e, além disso, eles fazem o alvo ser visto como um objeto legítimo de hostilidade (GELBER; MCNAMARA, 2016).

Nos espaços de interação virtual, as palavras utilizadas e o estilo de escrita denotam preferências, pensamentos, emoções e comportamentos dos usuários. Através desses recursos, o discurso odioso pode ser identificado e tratado pelas plataformas

[1] <https://www.facebook.com/>
[2] <https://www.twitter.com/>

que gerenciam tais espaços de interação (MARTINS et al., 2018). Essa necessidade existe por conta da pressão exercida pelos governos e sociedades civis sobre essas empresas em relação ao controle do escalonamento, da duração e da difusão do discurso de ódio online. Ao permanecer online, esse tipo de discurso tem o poder de assustar, silenciar ou intimidar as pessoas atingidas por ele (SILVA et al., 2019).

Muitas companhias responsáveis pelas plataformas de mídias sociais, como o *Facebook*, o *Twitter* e o *YouTube*[3], investiram, ao longo dos anos, centenas de milhões de euros por ano na identificação e no monitoramento do discurso de ódio e, ainda assim, sofreram e sofrem com críticas sobre não estarem agindo à altura do problema. Isso ocorre devido ao fato de que essas empresas focaram por muito tempo em soluções manuais para moderar o conteúdo. Esse tipo de moderação é considerado trabalhoso, demorado, não escalonável e não sustentável (ZHANG; LUO, 2018). Dessa forma, é notória a necessidade de soluções que realizem essa moderação de maneira automatizada.

Nos últimos anos, muito se pesquisou a respeito das soluções automatizadas para identificação do discurso de ódio no âmbito das mídias sociais (ZHANG; LUO, 2018). Muitas soluções aplicáveis foram desenvolvidas como as ontologias baseadas em palavras-chave de ódio comumente utilizadas nos ataques, as soluções baseadas em metadados extraídos a respeito dos perfis disseminadores de ódio e os modelos de classificação de texto baseados em aprendizado de máquina (MACAVANEY et al., 2019).

No caso específico dos modelos baseados em aprendizado de máquina, estes são capazes de performar a tarefa de classificação textual, a qual se relaciona com os domínios da mineração de texto e do processamento de linguagem natural. Por meio de conjuntos de dados textuais rotulados extraídos das plataformas de mídias sociais, os modelos extraem padrões de conhecimento e os aplicam para identificar o discurso odioso com resultados considerados promissores (ZHANG; LUO, 2018). Desse modo, neste trabalho, propõe-se uma ferramenta de identificação do discurso de ódio online em português a partir da experimentação de alguns modelos de aprendizado de

[3]<https://www.youtube.com/>

máquina.

Como contribuição, o trabalho em questão expõe os resultados experimentais da combinação de diferentes entradas com diferentes algoritmos de classificação baseados em aprendizado de máquina. Essas entradas foram obtidas a partir da estruturação de conjuntos de dados em português disponibilizados para a comunidade científica. Além disso, o trabalho também contribui relatando e analisando alguns dos principais desafios associados à tarefa de detecção do discurso de ódio por meio da abordagem proposta.

1.1 Objetivos

O objetivo deste trabalho é desenvolver um detector de discurso de ódio para textos online. Assim, para que esse objetivo seja alcançado, este trabalho se dividiu em:

- Construir e analisar os resultados de modelos de identificação do discurso de ódio online em português, buscando o modelo com o melhor desempenho.
- Comparar o desempenho dos algoritmos dos modelos com diferentes tipos de entrada, analisando o impacto da aplicação da técnica de refinamento de hiperparâmetros.
- Tornar o código-fonte do experimento aberto para futuras análises e construções científicas.

1.2 Organização do Trabalho

Este trabalho tem seu enfoque no desenvolvimento e na análise experimental para a construção de uma ferramenta de detecção automática de discurso de ódio online e sua organização é descrita a seguir:

- Capítulo 2: todos os fundamentos teóricos que embasam o trabalho serão

descritos nesse capítulo, contextualizando o processo de mineração de texto voltado para a classificação com o aprendizado de máquina e as técnicas utilizadas em cada etapa do processo.

- Capítulo 3: serão relatados a motivação do desenvolvimento deste trabalho, os trabalhos relacionados que também focaram na construção e na análise de modelos de classificação do discurso de ódio e uma perspectiva macro do modelo proposto.

- Capítulo 4: nesse capítulo, serão fornecidos todos os detalhes da construção e da avaliação dos modelos de identificação do discurso de ódio baseado em aprendizado de máquina. Serão descritas as bases de dados, as fases de preparação de dados, de extração de características, de extração de padrões de conhecimento e, por fim, a organização e os resultados dos experimentos realizados.

- Capítulo 5: aqui estarão as considerações finais e possíveis trabalhos futuros voltados para a ferramenta de detecção.

Capítulo 2

Embasamento Teórico

Nesse capítulo, serão descritos os detalhes a respeito do referencial teórico que serve de embasamento para as técnicas e conceitos aplicados durante a construção deste trabalho.

2.1 O Discurso de Ódio

De acordo com a organização internacional *United Nations*[1] fundada em 1945, não existe uma definição universal legal para o discurso de ódio. No entanto, segundo um plano estratégico de ação em relação ao discurso de ódio elaborado pela organização, esse fenômeno pode ser entendido como qualquer tipo de comunicação, discurso ou comportamento que ataca ou discrimina, de forma pejorativa, um indivíduo ou um grupo. Essas ações seriam baseadas em características que definem o indivíduo ou o grupo como religião, sexualidade, etnia, nacionalidade, raça, cor, descendência, gênero ou outros fatores de identidade.

As mídias sociais, as quais incluem redes sociais, sites de notícias, blogs e outros ambientes virtuais interativos, são espaços onde é possível encontrar o discurso de ódio sendo amplamente disseminado. Através desse tipo de tecnologia, foram adicionadas novas dimensões para o que já se definia como um problema complexo

[1] https://www.un.org/

na sociedade civil (ALKIVIADOU, 2019).

Em janeiro de 2020, 3.80 bilhões era o número que representava a quantidade total de usuários ativos nas mídias sociais em escala global. Em comparação com janeiro de 2019, constatou-se um acréscimo de 321 milhões de novos usuários em um ano (KEMP, 2020). Essas informações mostram o quão expressivo é o número de pessoas que interagem na Internet todos os dias gerando uma quantidade considerável de diferentes tipos de dados.

2.2 Mineração de Texto

Ao focar nos dados do tipo textual continuamente gerados em mídias sociais, avalia-se que a maior parte deles está armazenada de maneira não estruturada ou em estado bruto. Além disso, o formato em texto é considerado a forma mais natural de armazenamento de informações (TAN et al., 2000). Nesse cenário, a mineração de texto consiste em um conjunto de algoritmos utilizados para estruturar esses dados de texto bruto e consiste também nos métodos quantitativos usados na análise das informações textuais pós-estruturadas (DANG, 2015).

Como afirma Dang (2015), a mineração de texto é uma área considerada multidisciplinar por incorporar conceitos e técnicas de diversos domínios como o de mineração de dados, processamento de linguagem natural (PLN), recuperação e extração de informação, classificação, agrupamento e sumarização de texto. Além disso, tem como objetivo principal, através da combinação das técnicas supracitadas, a obtenção de informação útil e significativa por meio das fontes de dados textuais disponíveis.

2.2.1 Etapas da Mineração de Texto

De acordo com os autores Sinoara, Antunes e Rezende (2017), o processo de mineração de texto se divide em cinco etapas de maneira geral: especificação do objetivo, preparação de dados, extração de padrões, avaliação de conhecimento e uso de conhecimento. É necessário que, ao alcançar a fase de avaliação de conhecimento,

seja possível retornar para fase de preparação de dados ou de extração de padrões no caso da avaliação realizada sugerir possíveis ajustes ou melhorias a serem implementados. Essa possibilidade cria ciclos que são benéficos ao processo. Através da figura 2.1, é possível visualizar e entender melhor o fluxo entre as etapas mencionadas.

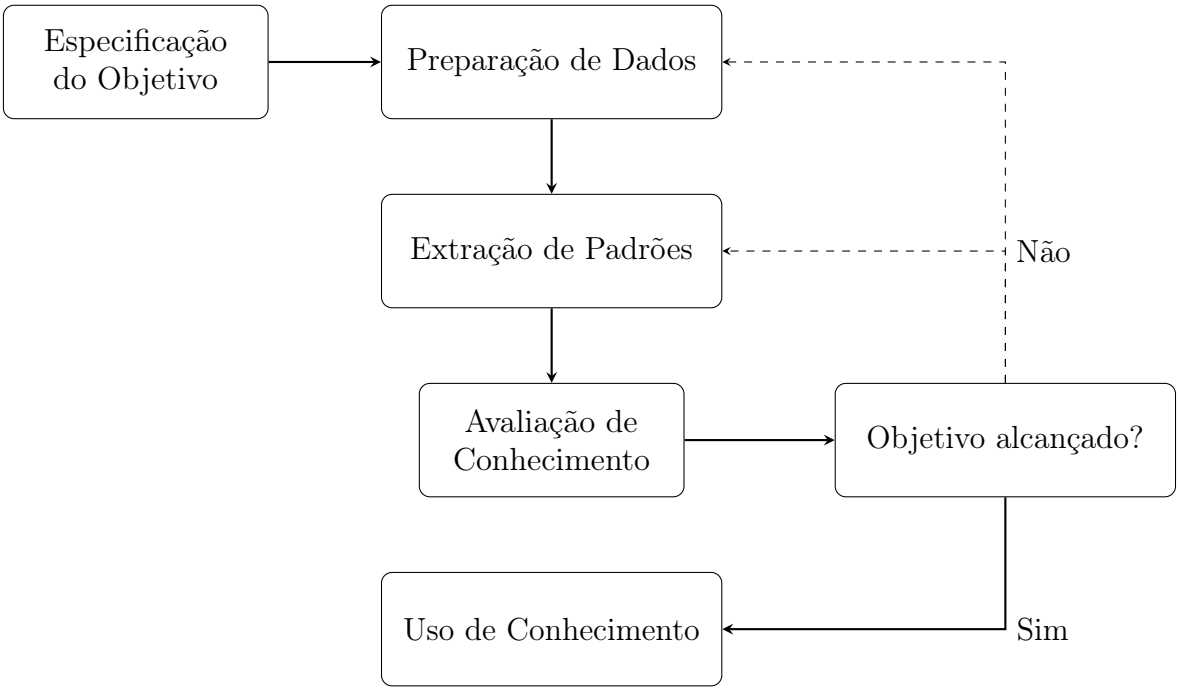

Figura 2.1: Processo de Mineração de Texto

2.2.1.1 Especificação do Objetivo

Considerada a fase de definição de diretrizes de todo o processo, a etapa de especificação do objetivo é onde se identifica a finalidade a ser alcançada com a mineração de texto. O escopo é delimitado e o conjunto de documentos de texto bruto, também conhecido como corpus, é especificado. Tais definições irão nortear cada etapa subsequente no processo de mineração.

2.2.1.2 Preparação de Dados

É na etapa de preparação de dados, também conhecida como etapa de pré-processamento, que ocorre a conversão dos dados que se encontram em formato de texto bruto, os quais compõem o corpus endereçado na fase de especificação do objetivo. O conjunto de documentos é, então, convertido para um formato que

permite a obtenção de padrões de conhecimento através dos algoritmos da etapa de extração de padrões.

A nova representação estruturada desses dados precisa conservar os padrões implícitos que estavam presentes no formato não estruturado, pois isso permite que eles possam ser identificados na próxima etapa. Sendo assim, é possível notar que a qualidade do resultado produzido na fase em questão impactará todo o processo de mineração de texto.

2.2.1.3 Extração de Padrões

A etapa de extração de padrões foca na obtenção dos padrões implícitos no conjunto de dados textuais através de seu formato estruturado gerado na etapa de preparação de dados. A extração é feita por meio de um algoritmo específico escolhido com base no tipo de padrão de conhecimento que se deseja obter, no conjunto de documentos de texto disponível e no objetivo declarado no escopo da primeira etapa do processo.

2.2.1.4 Avaliação de Conhecimento

Após a obtenção dos padrões de conhecimento, inicia-se a fase de avaliação também conhecida como etapa de pós-processamento. Nesse estágio, é avaliado se o objetivo do escopo foi alcançado através da análise dos padrões obtidos até então por meio de métricas específicas. Caso a avaliação produza um resultado não satisfatório, os ciclos presentes no fluxo de trabalho do processo, que partem da etapa de avaliação do conhecimento, permitem a realização das modificações ou melhorias necessárias. Sendo assim, é possível remodelar a etapa de conversão dos dados brutos em estruturados e/ou reformular a extração de padrões de conhecimento até que os resultados contemplem o objetivo adequadamente.

2.2.1.5 Uso de Conhecimento

Ao concluir, na etapa anterior, que o modelo construído cumpre satisfatoriamente as especificações da primeira fase do processo de mineração de texto, o resultado produzido até então pode ser disponibilizado para utilização dos usuários. Isso marca o início da última etapa do processo de mineração de texto denominada uso de conhecimento. Por fim, se for necessário realizar quaisquer mudanças no escopo ou no corpus de texto, será necessário refazer todo o processo desde sua primeira etapa.

2.2.2 Processamento de Linguagem Natural

Segundo Reshamwala, Mishra e Pawar (2013), o termo linguagem natural refere-se a qualquer linguagem desenvolvida e aprendida por seres humanos no ambiente em que estão inseridos, onde a utilizam para comunicar-se entre si. Nesse ambiente, através da linguagem natural, é possível expressar sentimentos e conhecimentos de maneira não dependente do formato de comunicação que se adota. O Processamento de Linguagem Natural (PLN) é uma ramificação oriunda dos domínios da inteligência artificial e da linguística que se combinam para investigar e compreender as interações entre linguagem natural humana e computadores.

O PLN foca nas estruturas semântica e gramatical presentes na linguagem. Essa característica o permite lidar com as diversas ramificações da comunicação humana como texto, discurso, imagens, entre outras. Através disso, é possível realizar muitas tarefas como traduções de muita qualidade entre diferentes línguas, análise dos sentimentos descritos na comunicação ou, ainda, simplificar e tornar favorável a interação com os computadores para os seres humanos (TITENOK, 2020).

2.2.3 O PLN e o Processo de Mineração de Texto

Ao dar ênfase às técnicas advindas do processamento de linguagem natural, é preciso entender qual a relação delas com a mineração de texto, isto é, quais são as diferenças e os pontos de convergência dos dois domínios citados até então. Segundo Kao e Poteet (2006), o PLN pode ser incorporado ao processo de mineração de

texto através do uso de conceitos linguísticos do texto como as classes e estruturas gramaticais e as propriedades semânticas. Tais elementos ganham enfoque por meio das técnicas presentes no domínio do processamento de linguagem natural e podem impulsionar o alcance do objetivo definido no escopo do modelo de mineração de texto.

O processo de mineração consiste na extração de objetos de interesse úteis e significativos do conjunto de dados textuais, os quais compõem os padrões de conhecimento. No cumprimento desse objetivo, o processo conta com uma série de metodologias de processamento textual. Nesse contexto, o processamento de linguagem natural é uma das mais importantes metodologias disponíveis para processar os dados textuais presentes nas etapas de mineração de texto (LINGUAMATICS, 2020).

Enquanto a mineração de texto possui seu foco em técnicas advindas principalmente dos campos de conhecimento da extração de informação, estatística e aprendizado de máquina, o PLN traz técnicas inspiradas na linguística com o uso de analisadores sintáticos, onde o resultado dessas análises é interpretado, muitas vezes, com o objetivo de realizar uma extração de informação semântica dos dados (KAO; POTEET, 2005).

2.2.4 Técnicas de PLN na Extração de Características

O PLN fornece uma série de conceitos cruciais para muitas implementações do processo de classificação textual que se relaciona diretamente com o domínio da mineração de texto. Dentre as técnicas que permeiam esses domínios, estão as utilizadas na etapa de extração de características como as técnicas de palavras ponderadas combinadas com a estrutura *Bag of Words*, os *N-Grams* e a técnica *word embeddings*. Essas técnicas são comumente utilizadas na criação de representações estruturadas de conjuntos de dados textuais em sua forma bruta ou não processada (KOWSARI et al., 2019).

2.2.4.1 Técnicas de Palavras Ponderadas

A técnica de palavras ponderadas é uma das maneiras de preparar um corpus de texto bruto para ser submetido aos algoritmos de aprendizado de máquina. Essa técnica foca em associar um peso numérico a cada termo único presente no conjunto de dados textual. Cada documento ou sentença torna-se um vetor de números totalmente processável pelos algoritmos em questão. A forma mais básica de se aplicar o método é através da abordagem denominada *Term Frequency* (TF). Ademais, existe também uma outra abordagem que funciona em conjunto com a TF definida como *Term Frequency - Inverse Document Frequency* (TF-IDF) (KOWSARI et al., 2019).

As abordagens presentes nessa técnica variam em relação à forma como constroem os vetores que representam os documentos ou sentenças. Nesse contexto, o *Term Frequency* realiza o mapeamento de cada uma das palavras para um número que significa a quantidade de vezes que elas aparecem nos documentos sem manter nenhuma informação sintática ou de similaridade entre elas. Cada documento se torna um vetor com uma posição para cada uma das palavras únicas presentes no próprio documento e, então, as frequências são contadas e armazenadas nas posições das palavras correspondentes.

Uma estrutura muito utilizada em diversos domínios do aprendizado de máquina é definida como *Bag of Words* (BoW) (Tabela 2.1). Ela é capaz de representar um conjunto inteiro de documentos usando um critério específico para isso, como a frequência das palavras por exemplo. Além disso, ela contém todos os termos únicos presentes no conjunto de dados e cada sentença se torna um vetor com tamanho igual ao número total desses termos, onde os valores das posições são preenchidos seguindo o critério adotado. Por fim, a estrutura se torna um mapeamento das sentenças com o conjunto de palavras únicas, preenchendo as posições dos vetores com o número de ocorrências das palavras nas sentenças caso o critério seja a frequência delas.

Em 1972, Jones (1972) propôs o método *Inverse Document Frequency* (IDF), o qual passou a ser usado em conjunto com o *Term Frequency*. O objetivo dessa combinação foi lidar com os termos que podem não ser tão frequentes dentro de um

	meu	*primo*	*pegou*	*celular*	*ele*	*parece*	*com*
$S1$	2	1	1	1	0	0	0
$S2$	1	1	0	0	1	1	1

Tabela 2.1: Exemplo da estrutura BoW para as sentenças "meu primo pegou meu celular" (S1) e "ele parece com meu primo" (S2).

documento específico, mas aparecem com muita frequência dentre os documentos do conjunto. Por conta disso, esses termos são classificados como implicitamente comuns e a combinação *Term Frequency - Inverse Document Frequency* (TF-IDF) tenta diminuir o efeito desses termos no conjunto de dados textual. Através da equação 2.1 é possível entender como os pesos são atribuídos aos termos nessa combinação.

$$W(d,t) = TF(d,t) * \log\left(\frac{N}{df(t)}\right) \tag{2.1}$$

Na equação 2.1, $W(d,t)$ significa um peso W sendo atribuído ao termo t localizado no documento ou sentença d. Esse cálculo envolve a frequência do termo t no documento d ($TF(d,t)$) multiplicada pelo logaritmo do número total de documentos do conjunto (N) dividido pela quantidade de documentos que contêm o termo t ($df(t)$). Dessa forma, a abordagem TF-IDF lida o problema das palavras muito comuns no documento aplicando essa equação para cada termo presente no corpus com um contexto específico para cada sentença.

2.2.4.2 N-Gram

A técnica *N-Gram* consiste em uma maneira de conceder alguma representação sintática aos modelos de extração de características que não mantêm essa informação. Os *N-Grams* são conjuntos de sequências de palavras, onde N representa o número de *tokens* dentro de cada sequência. Sendo assim. a abordagem *Term Frequency*, também conhecida *Bag of Words*, ao realizar o mapeamento tradicional das palavras, utiliza o *1-Gram* ou unigramas, pois extrai as palavras individualmente no processo. A extração dos conjuntos *2-Gram* e *3-Gram* faz mais sentido ao se utilizar o *N-Gram*, pois, com mais palavras presentes nas sequências, é possível obter mais informação

sintática (KOWSARI et al., 2019).

Exemplo de *1-Gram*

> *"Ele está triste e sem rumo na vida."*
> No caso da sentença acima, o conjunto *1-Gram* fica da seguinte forma:
> { *"Ele", "está", "triste", "e", "sem", "rumo", "na", "vida"* }.

Exemplo de *2-Gram*

> *"Ele está triste e sem rumo na vida."*
> No caso da sentença acima, o conjunto *2-Gram* fica da seguinte forma:
> { *"Ele está", "está triste", "triste e", "e sem", "sem rumo", "rumo na", "na vida"*}.

Exemplo de *3-Gram*

> *"Ele está triste e sem rumo na vida."*
> No caso da sentença acima, o conjunto *3-Gram* fica da seguinte forma:
> { *"Ele está triste", "está triste e", "triste e sem", "e sem rumo", "sem rumo na", "rumo na vida"*}.

2.2.4.3 Word Embeddings

Apesar da informação sobre a sintaxe fornecida pelo *N-Gram*, as representações estruturadas do conjunto de dados apresentadas, até então, não são capazes de respeitar totalmente a ordem das palavras e de lidar com as similaridades semânticas atribuídas ao corpus textual. Nesse contexto, surgem os modelos de *word embeddings Skip-gram* e *Continuous Bag of Words* (CBOW) que propõem uma nova representação dos dados para lidar com os problemas apresentados (KOWSARI et al., 2019).

A abordagem de *word embeddings* consiste em transformar as palavras presentes no conjunto de dados em vetores reais N-dimensionais. Dessa forma, apresentada em Mikolov et al. (2013), a técnica denominada *Word2Vec* usa *Shallow Neural Networks* para criar esses vetores e implementa os modelos *Skip-gram* e CBOW. O primeiro modelo foca em inferir uma palavra através do contexto dela, enquanto o segundo

objetiva encontrar as palavras que compõem o contexto por meio de um termo específico. Ambos os modelos atuam no espaço vetorial, no qual as similaridades entre as palavras foram calculadas com a análise de um grande volume de documentos de texto. A figura 2.2 apresenta uma ideia visual de como essas similaridades ocorrem entre os vetores das palavras.

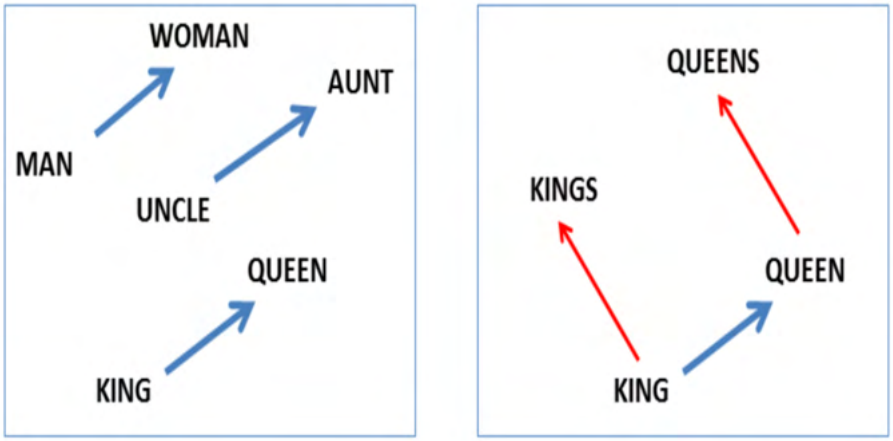

Figura 2.2: Representação visual das similaridades presentes no espaço vetorial das *word embeddings*.
Fonte: (MIKOLOV et al., 2013)

2.3 Classificação de Texto

A classificação de texto foi mencionada anteriormente como um dos domínios que compõem a mineração de texto, assim como o processamento de linguagem natural. Além disso, de acordo com os autores Kamruzzaman, Haider e Hasan (2010), ela constitui-se como o processo de classificar os documentos de texto presentes no conjunto de dados em categorias definidas previamente de acordo com o conteúdo presente em cada documento. Os sistemas de recuperação e sumarização de texto possuem como requisito primário a técnica de classificação textual.

Formalmente, seja d_i um documento de texto presente no conjunto de documentos (corpus) D e $\{c_1, c_2, c_3, ..., c_n\}$ o conjunto de todas as categorias ou classes predefinidas, então o processo de classificação de texto significa a atribuição de uma categoria c_j ao documento d_i. Se somente uma classe ou categoria é atribuída ao documento, logo ele é definido como *single-label* e, caso mais de uma classe seja associada ao

documento, ele é definido como *multi-label* (ABIKOYE; OMOKANYE; ARO, 2018).

Ainda de acordo com Abikoye, Omokanye e Aro (2018), um problema de classificação de texto definido como *single-label* pode assumir dois tipos diferentes: *binary class* e *multi-class*. No primeiro tipo, existem duas classes e somente uma delas é atribuída a cada documento de texto presente no corpus. O segundo tipo é caracterizado por documentos, aos quais são atribuídas, ao mesmo tempo, N classes mutuamente exclusivas. Por fim, o principal objetivo presente em todas as definições de classificação textual citadas seria a criação de um modelo que delimita as classes predefinidas com o intuito de predizer qual delas será atribuída a instâncias ainda não conhecidas.

2.3.1 Algoritmos

Os algoritmos de classificação encaixam-se no domínio do aprendizado de máquina supervisionado, isto é, nesse cenário, se torna necessário fornecer, para a fase de aprendizado, um conjunto de dados previamente anotado ou rotulado. Como foi dito anteriormente, o objetivo principal consiste em classificar as instâncias desconhecidas através dos padrões de conhecimento extraídos dos dados textuais já anotados (PELLE, 2019).

É possível categorizar a mineração de texto como uma ciência experimental por conta dos testes que podem ser feitos com vários algoritmos de classificação diferentes e com a alteração dos hiperparâmetros envolvidos em cada um deles. Isso permite que o melhor classificador para um cenário específico possa ser encontrado. Além dos testes, a natureza e o tipo do conjunto de dados disponível também devem ser levados em consideração na escolha dos algoritmos (ABIKOYE; OMOKANYE; ARO, 2018).

2.3.1.1 Logistic Regression

Um dos primeiros algoritmos utilizados no domínio de classificação é conhecido como *Logistic Regression*. Desenvolvido por David Cox em 1958, esse algoritmo é um

modelo linear cujo objetivo principal é realizar o treinamento a partir da probabilidade de variáveis serem de uma classe representada por 0 ou de outra representada por 1 dado um conjunto de exemplos específico (KOWSARI et al., 2019). Ainda que carregue o nome *regression*, o qual se trata de outra tarefa realizada no domínio do aprendizado de máquina supervisionado, esse é um modelo de classificação baseado em probabilidades contínuas no conjunto dos números reais e limitadas entre 0 e 1 (BONACCORSO, 2017).

2.3.1.2 Naïve Bayes

Baseada no Teorema de Bayes formulado por Thomas Bayes entre 1701 e 1761, a família de algoritmos *Naïve Bayes* é um conjunto de ferramentas poderosas e práticas que classificam por meio da determinação da probabilidade dado um conjunto de condições específicas. Desde 1950, o modelo é muito utilizado na tarefa de categorização de documentos e responde muito bem em muitos outros contextos também. Essa técnica é considerada um modelo generativo, a qual é a forma mais tradicional de categorização textual (KOWSARI et al., 2019).

2.3.1.3 Support Vector Machines

A primeira versão do algoritmo *Support Vector Machines* foi desenvolvida em 1963 e tratava-se de uma versão linear focada em resolver tarefas de classificação binária. No entanto, essa técnica foi explorada também para tarefas de multi-classificação, a qual envolve a presença de mais de 2 classes. Na década de 90, essa versão foi modificada para uma variação que lidava também com cenários não-lineares. Quando combinado com redes neurais, esse algoritmo se torna uma das melhores técnicas para se aplicar nos casos em que não é fácil achar um hiperplano de separação (BONACCORSO, 2017).

2.3.1.4 Random Forest

Desenvolvido em 1999, o algoritmo *Random Forest* é uma combinação de um determinado número de árvores de decisão, as quais foram um dos primeiros algorit-

mos de classificação para o domínio da mineração de dados e de texto. Na árvore de decisão, os dados são decompostos hierarquicamente e a maior questão que o método precisa resolver é a de quais parâmetros e características irão ocupar os nós mais altos e mais baixos na hierarquia (KOWSARI et al., 2019). No caso do *Random Forest*, o conjunto de árvores de decisão é construído em amostras aleatórias que variam o modo em que dividem seus nós. O objetivo do algoritmo é separar os dados da melhor forma possível entre as amostras, distribuindo subconjuntos aleatórios do conjunto de dados para isso.

Capítulo 3

Detecção Automática de Discurso de Ódio em Textos do Twitter

Nesse capítulo, serão apresentados as motivações para o desenvolvimento da ferramenta de detecção, os trabalhos relacionados a ele e os detalhes da proposta de solução planejada que será construída ao longo deste trabalho.

3.1 Motivação

Os recentes avanços tecnológicos trouxeram a possibilidade dos usuários serem não só consumidores mas também autores das informações que circulam na Internet. Esses avanços criaram a *Word Wide Web* (WEB) 2.0, a qual introduziu o conceito de sites dinâmicos que, ao contrário das páginas estáticas, possibilitaram o surgimento da interação através das plataformas de mídias sociais (LEINER et al., 2009). Sendo assim, a dinamicidade da WEB 2.0 permitiu que os usuários fizessem, por exemplo, postagens de opiniões em blogs, publicações em redes sociais, comentários em fóruns ou sites de notícias, entre outras modalidades de interação online (PELLE, 2019).

De acordo com Kane et al. (2014), os pesquisadores utilizam a área de análise de redes sociais para entender e estudar o fenômeno das mídias sociais e seus impactos. Dessa forma, é notório o papel de destaque das redes sociais dentro das inovações

ocasionadas pela versão 2.0 da tecnologia WEB. Hoje, as redes sociais possuem muitos papéis na comunicação online, como o de ampliar o alcance das interações e publicação de conteúdos e ideias ou permitir a criação de comunidades voltadas para postagens de interesse comum (PELLE, 2019).

No entanto, em meio aos avanços relacionados às mídias sociais, um comportamento praticado por muitos usuários, que se manifesta através de uma linguagem ofensiva, começou a ganhar destaque nas plataformas recentemente. Conhecido como discurso de ódio, esse comportamento é definido por Fortuna e Nunes (2018) como uma linguagem usada para atacar, diminuir e incitar violência ou ódio contra grupos, baseando-se em características associadas a eles.

Aparência física, religião, descendência, nacionalidade ou origem étnica, orientação sexual e identidade de gênero são alguns exemplos dos aspectos intrínsecos aos grupos utilizados no discurso de ódio, o qual pode ser sutil, conter humor ou ainda aparecer em outros formatos linguísticos (FORTUNA; NUNES, 2018). Redes sociais como o *Facebook* e o *Twitter* possuem, em suas condições e termos de uso, as próprias definições de discurso de ódio, visto que é nesse tipo de plataforma que esse tipo de discurso tem acontecido de forma regular.

Mesmo não permitindo o discurso de ódio em suas políticas, as empresas responsáveis por gerenciar as plataformas de mídias sociais continuam possuindo um número significativo de usuários que o reproduzem em suas interações, gerando conflitos e processos judiciais. Isso acarretou em um aumento do empenho dessas empresas no combate a esse tipo de discurso através da aceleração de sua remoção nas aplicações (PELLE, 2019). No entanto, nem sempre há sucesso com relação às medidas de combate, como, por exemplo, quando o *Facebook* falhou em conter a incitação de ódio contra um grupo minoritário mulçumano em Mianmar no ano de 2013[1].

No Brasil, com 140 milhões de usuários ativos nas mídias sociais registrados em janeiro de 2020 (KEMP, 2020), o discurso de ódio também é muito frequente nas plataformas. A Nova/SB[2], uma das maiores agências de publicidade do país,

[1] <https://reut.rs/2vKN0ov>
[2] <http://www.novasb.com.br/>

publicou, em 2016, um dossiê sobre a intolerância nas redes disponível em NOVA/SB (2016). No documento, foi realizada uma análise de sentimentos com cerca de 542 mil postagens de brasileiros retiradas de redes sociais, blogs e sites da Internet, contendo termos que possivelmente caracterizam o discurso intolerante. Segundo os resultados, em todas as categorias de intolerância analisadas, o percentual classificado como negativo ou depreciativo foi acima de 84%.

A legislação no Brasil, em relação aos crimes de ódio, é considerada defasada quando comparada ao cenário internacional de acordo com Silva (2019). Segundo ele, em muitas situações, a expressão "discurso de ódio" foi utilizada em denúncias sem ter sido devidamente definida em termos jurídicos específicos. Isso abriu espaço para que essas situações tivessem múltiplas interpretações, principalmente, quando elas ocorriam no meio virtual. Atualmente, existe um projeto de lei em tramitação no Congresso Nacional identificado pelo número 7582/2014[3], o qual prevê a definição e a punição dos crimes relacionados ao discurso de ódio. O artigo 5º do projeto menciona que os meios de comunicação, inclusive a Internet, estão incluídos nos espaços, nos quais a lei poderá ser aplicada.

Em Gagliardone Danit Gal (2015), afirma-se que o discurso de ódio pode permanecer online por um longo período de tempo em diferentes plataformas e formatos. Ademais, enquanto esse tipo de conteúdo se mantém acessível através da Internet, mais destrutivo ele se torna para as vítimas e mais força ganham os emissores do discurso. Em contrapartida, afirma-se também que a remoção do conteúdo em estado precoce pode limitar a exposição sofrida pelas vítimas. Dessa forma, é notória a necessidade de identificação e contenção do discurso de ódio presente na Internet como um todo.

Com um número significativo de plataformas de mídias sociais proibindo o discurso de ódio em suas regras de uso, foi criada a possibilidade dos usuários reportarem publicações, as quais, possivelmente, poderiam ser identificadas como uma manifestação do discurso em questão. Sendo assim, para lidar com a análise dos relatórios, é necessário um grande esforço manual das equipes responsáveis pela moderação de

[3]<https://www.camara.leg.br/proposicoesWeb/fichadetramitacao?idProposicao=616270>

conteúdo. Nesse contexto, as ferramentas e abordagens automáticas podem auxiliar na aceleração do processo de revisão e permitir que os recursos humanos sejam alocados onde realmente é necessária uma análise mais crítica (MACAVANEY et al., 2019).

Dentre as ferramentas e abordagens automáticas existentes, estão os classificadores baseados em aprendizado de máquina, os quais, através de amostras de texto previamente anotadas, são capazes de identificar o discurso de ódio. Nesse processo, é necessária a extração de características do texto que apontam o ódio, como palavras ou sequências de palavras. Na categoria de abordagens baseadas em aprendizado de máquina, existem quatro modelos muito utilizados nessa tarefa: *Logistic Regression*, *Naïve Bayes*, *Support Vector Machine* e *Random Forest*. Esses modelos possuem implementações de código aberto disponíveis no pacote de aprendizado de máquina da linguagem de programação *Python3*[4] denominado *sci-kit learn*[5] (MACAVANEY et al., 2019).

3.2 Trabalhos Relacionados

Nesse tópico, serão destacados alguns trabalhos que desenvolveram abordagens de classificadores baseados em aprendizado de máquina com o objetivo de identificar o discurso de ódio online.

Em Pelle (2019), a abordagem proposta é denominada *Hate2Vec* e objetiva a detecção de comentários ofensivos em português por meio de um meta-classificador, o qual combina o resultado de três classificadores empilhados. Internamente, o modelo possui um classificador baseado na proximidade semântica da representação vetorial dos vocábulos da língua portuguesa; outro que utiliza um algoritmo de regressão logística como método de classificação dos vetores que representam os comentários; e um outro classificador definido pelo autor como *bag of words* baseado em uni-gramas do texto. Além disso, o autor propõe também duas variações de um conjunto de

[4] <https://www.python.org/>
[5] <https://scikit-learn.org/>

dados[6] anotados com 1.250 comentários retirados do portal de notícias brasileiro G1[7].

Fortuna et al. (2019) propuseram a construção de um conjunto de dados[8] anotados com 5.668 *tweets* em português. Dois tipos de anotação foram feitas: uma binária, na qual os anotadores rotularam os dados entre os termos "*hate*" e "*no-hate*" e outra, onde foi desenvolvido um método de classificação multi-rótulo hierárquica com 81 categorias do discurso de ódio relacionadas entre si. Sendo assim, para validar o conjunto de dados gerado, realizou-se um experimento de *baseline* com os dados anotados de forma binária, utilizando representações vetoriais de palavras baseadas na proximidade semântica entre elas e uma rede neural do tipo recorrente denominada *Long Short-Term Memory* (LSTM).

Outro modelo de meta-classificação para identificação do discurso de ódio foi proposto em MacAvaney et al. (2019). Em sua proposta, o autor combina duas aplicações diferentes do algoritmo de *Support Vector Machines* (SVM): uma *multi-view* com múltiplas camadas de visualização empilhadas, uma para cada característica (*feature*) extraída dos documentos de texto; e uma abordagem linear do mesmo algoritmo. Na aplicação *multi-view*, cada camada empilhada do modelo possui seu próprio classificador SVM linear e, além disso, esse modelo de múltiplas camadas funciona em conjunto com um classificador SVM linear separado, formando o meta-classificador. A proposta em questão foi testada em vários conjuntos de dados de discurso de ódio nos idiomas inglês, espanhol e hindi presentes na literatura e obteve resultados promissores.

Em sua pesquisa, Aljarah et al. (2020) coletaram 3.696 *tweets* em árabe, os quais foram rotulados com os termos "*hate*" e "*non-hate*" e também foram submetidos a um processo de análise de sentimentos. Além dos *tweets*, dados dos perfis dos usuários que os publicaram também foram coletados. Essas informações tornaram possível a extração de 15 combinações de características (*features*) diferentes através de técnicas de processamento de linguagem natural (PLN). As características extraídas foram

[6] <https://github.com/rogersdepelle/OffComBR>
[7] <https://g1.globo.com>
[8] <https://github.com/paulafortuna/Portuguese-Hate-Speech-Dataset>

experimentadas com os algoritmos SVM, Naïve Bayes, Decision Tree e Random Forest. O melhor resultado avaliado foi do experimento que utilizou *Term Frequency - Inverse Document Frequency* para extrair características dos *tweets*, características retiradas dos dados dos perfis e o algoritmo Random Forest.

Gibert et al. (2018) realizaram a construção de um conjunto de dados[9] em inglês sobre discurso de ódio com o conteúdo retirado de um fórum criado em 1996 denominado *Stormfront*, o qual tentava promover a supremacia branca. Foram coletadas 10.568 sentenças do fórum que passaram por um processo de rotulação, onde 9.656 sentenças foram selecionadas e rotuladas com os termos "*hate*" e "*noHate*"Ós experimentos de classificação utilizando o conjunto de dados em questão ocorreram com os algoritmos de SVM, de redes neurais convolucionais (CNN) e o de redes neurais do tipo recorrente (LSTM). De acordo com a fase de avaliação dos modelos experimentados, o classificador baseado em LSTM obteve os melhores resultados.

3.3 Proposta

A partir dos cenários apresentados, nos quais a identificação automatizada do discurso de ódio na Internet se torna uma necessidade, é notório que a pesquisa e a construção de soluções nesse sentido se tornam uma prioridade. Uma prova disso são os esforços reunidos no *The Alan Turing Institute*[10] voltados para monitorar, entender e combater os ataques de ódio online. Nesse instituto, dentre as inúmeras iniciativas, muito se pesquisa também sobre as estratégias de classificação textual e de aprendizado de máquina no intuito de produzir as soluções tão necessárias[11].

Propõe-se, então, neste trabalho, uma análise experimental que envolve a avaliação de uma série de modelos de classificação textual baseados em aprendizado de máquina voltados para a detecção do discurso de ódio em português na Internet. Serão analisados os resultados produzidos na execução de quatro algoritmos de classificação diferentes que utilizarão representações estruturadas de um conjunto de dados

[9] <https://github.com/Vicomtech/hate-speech-dataset>
[10] <https://www.turing.ac.uk/>
[11] <https://www.turing.ac.uk/research>

anotado como entrada. Essa análise envolverá a possibilidade de um reforço na fase de treinamento dos algoritmos por meio de uma injeção de comentários ofensivos provenientes de uma segunda base de dados e o refinamento dos hiperparâmetros de um dos algoritmos utilizados também.

Mais detalhadamente, os dados de cada uma das bases serão submetidos ao fluxo de preprocessamento que envolve a preparação dos dados, onde ocorrem os procedimentos de padronização, *tokenização*, limpeza e extração de características dos conjuntos de dados textuais. Os produtos obtidos através do fluxo de preprocessamento serão as entradas para os algoritmos presentes na etapa de extração de padrões de conhecimento. A partir de então, cada entrada será dividida em um grupo que segue para o treinamento dos algoritmos e outro que segue para a avaliação dos padrões de conhecimento que foram extraídos.

Seguidamente, a etapa de avaliação dos padrões de conhecimento calculará as medidas de desempenho do modelo, fornecendo os resultados experimentais para a análise. Os experimentos serão separados com base nas entradas a serem submetidas aos algoritmos e também pela utilização ou não da injeção de comentários ofensivos do conjunto de dados de reforço. Em cada experimento, serão executados e avaliados os quatro algoritmos de uma só vez.

Capítulo 4

Avaliação Experimental

Nesse capítulo, serão detalhadas a composição da base de dados adotada, as etapas de construção dos modelos de identificação automática do discurso de ódio online e a avaliação dos resultados obtidos através da execução dos experimentos realizados com os modelos em questão.

4.1 Bases de Dados

O conceituado *The Alan Turing Institute* fornece uma ferramenta online de mapeamento de conjuntos de dados anotados com enfoque em discurso de ódio, linguagem ofensiva e abuso na Internet denominada *Hate speech data*[1]. Essa ferramenta cataloga os conjuntos de dados desenvolvidos e disponibilizados em diversos idiomas juntamente com as referências dos estudos que os originaram. Através dela, o critério de busca de uma base de dados em português foi cumprido de forma facilitada, já que ela mapeia também os conjuntos de dados para o idioma.

O *Hate speech data* possuía em seu catálogo, em janeiro de 2021, apenas dois conjuntos de dados para a língua portuguesa: um desenvolvido por Fortuna et al. (2019) e um outro desenvolvido por Pelle (2019). Os dois conjuntos de dados são compostos de sentenças extraídas de mídias sociais na Internet. Além disso, eles já

[1] <https://hatespeechdata.com/>

foram citados na seção 3.2 de trabalhos relacionados anteriormente. Desse modo, neste trabalho, optou-se pela utilização de uma metodologia que envolveu as duas bases de dados supracitadas.

Por meio das figuras 4.1 e 4.2, é possível visualizar as palavras e suas frequências nos conjuntos de dados. As figuras trazem um conceito de representação visual denominado nuvem de palavras. Através desse tipo de visualização, os termos presentes em um ou vários documentos são sumarizados de forma que sua frequência no texto seja descrita de forma proporcional pelo tamanho da fonte que é atribuída a cada palavra (CIDELL, 2010). Nas imagens, como exemplo, estão as sentenças rotuladas em positivo para o discurso ofensivo.

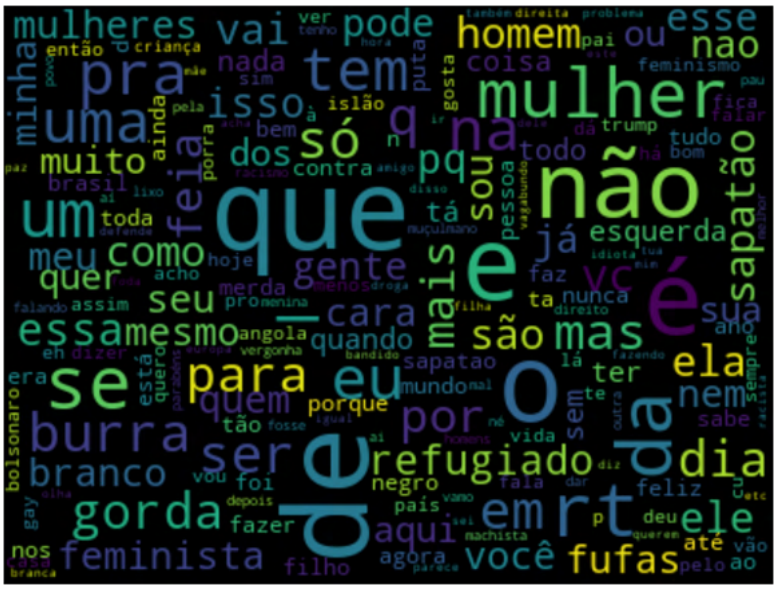

Figura 4.1: Nuvem de palavras das sentenças rotuladas como discurso de ódio do conjunto de dados desenvolvido em Fortuna et al. (2019).

4.1.1 Fortuna et al. (2019)

Apesar de possuir dois processos de anotação diferentes, um binário e outro que envolve uma estrutura de dados hierárquica, somente a abordagem de anotação binária presente em Fortuna et al. (2019) foi utilizada neste trabalho. Nessa abordagem, as sentenças são discriminadas em positivo ou negativo para discurso de ódio. Ademais, estiveram envolvidos três anotadores voluntários e o Kappa de Fleiss (FLEISS, 1971)

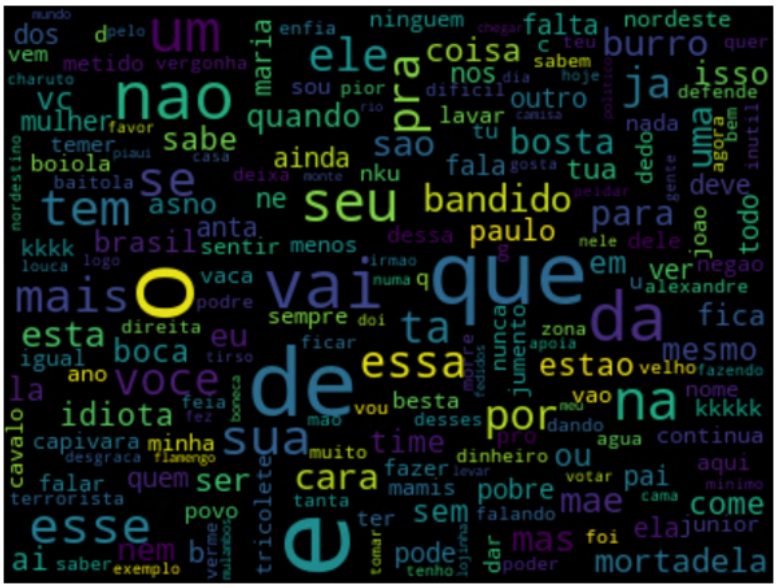

Figura 4.2: Nuvem de palavras das sentenças rotuladas como ofensivas do conjunto de dados desenvolvido em Pelle (2019).

foi calculado como medida de concordância entre eles com o valor de $0,17$. O corpus resultante desse processo possui no total 5.668 *tweets* coletados e anotados com voto majoritário no período de janeiro a março de 2017. Além disso, a quantidade de sentenças apontadas como discurso de ódio é de $31,5\%$.

4.1.2 Pelle (2019)

O site de notícias G1[2], um dos portais de notícias mais acessados no Brasil segundo o ranqueamento da plataforma Alexa[3] em janeiro de 2021, foi o foco da extração de dados realizada em Pelle (2019). O portal permite a publicação de comentários nas páginas das notícias e, através da técnica de *web scraping*, o autor extraiu 10.336 comentários de 115 notícias diferentes. Desses comentários, foi selecionada aleatoriamente uma amostra com 1.250, os quais foram anotados nas categorias ofensivo e não ofensivo por três julgadores. Em suma, duas bases de dados foram geradas, levando em consideração o nível de concordância entre os anotadores.

Na primeira, denominada *OFFCOMBR2*, os comentários são considerados ofensivos quando pelo menos 2 julgadores concordam nesse sentido. Já na segunda, definida

[2] <https://g1.globo.com>
[3] <https://www.alexa.com/topsites/countries/BR>

como *OFFCOMBR3*, os comentários só são rotulados como ofensivos se os 3 anotadores concordarem entre si. Nesse caso, o Kappa de Fleiss (FLEISS, 1971) resultou em $0,71$ e foi calculado como medida de concordância apenas para o conjunto de dados *OFFCOMBR2*, já que no *OFFCOMBR3* existe uma unanimidade associada à escolha dos rótulos. Por fim, somente a base *OFFCOMBR2* com $33,5\%$ de comentários ofensivos foi escolhida para ser um dos corpus utilizados neste trabalho.

4.2 Metodologia e Organização dos Experimentos

Nessa seção, a metodologia pensada através do embasamento teórico fornecido por Sinoara, Antunes e Rezende (2017) a respeito do processo de mineração de texto voltado para a classificação textual foi descrita. Logo em seguida, a organização do processo experimental, ao qual os modelos de classificação construídos foram submetidos também é explicada.

4.2.1 Preparação dos Dados

A etapa de preparação dos dados é responsável por transformar os dados textuais brutos em representações estruturadas do conjunto de dados, as quais serão submetidas como entrada para os algoritmos presentes na etapa seguinte. Essa etapa é dividida em três subetapas que serão contextualizadas nessa seção e podem ser visualizadas através da figura 4.3.

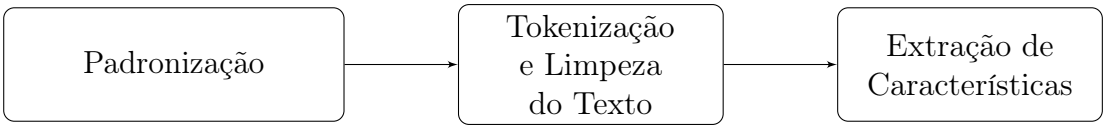

Figura 4.3: Subetapas da etapa de preparação dos dados pertencente ao processo de mineração de texto.

4.2.1.1 Padronização

Os dois conjuntos de dados textuais adotados encontram-se em diferentes formatos de armazenamento no início do processo. Isso ocorre, pois as fontes que os disponi-

bilizaram utilizaram meios distintos para extrair e construir tais conjuntos. Sendo assim, inicia-se a preparação desses dados através da padronização do formato de armazenamento dos corpus de texto utilizando uma estrutura de dados denominada *DataFrame*[4] presente no pacote *Pandas*[5] para a linguagem *Python3*. A estrutura do *DataFrame* é tabular, assim como a dos arquivos que contêm os dados dos corpus anotados, o que permite guardá-los e manipulá-los de maneira prática.

Cada um dos corpus foi alocado em um *DataFrame* exclusivo, no qual existem duas colunas: uma para guardar o texto da sentença convertido para minúsculo e outra para armazenar o rótulo que lhe foi atribuído durante a anotação. Os rótulos foram padronizados em "1" para positivo e "0" para negativo em relação à presença do discurso de ódio na sentença correspondente. Além disso, o conjunto de dados, disponibilizado em Fortuna et al. (2019), é composto por *tweets* e, portanto, foi necessária a remoção das menções aos usuários do *Twitter*, quebras de linha e links presentes no corpus em questão.

4.2.1.2 Tokenização e Limpeza do Texto

Com a padronização do formato dos dois corpus anotados inseridos nos *DataFrames*, é possível avançar no processo de preparação dos dados. Na próxima etapa, uma técnica denominada *tokenização*, a qual transforma cada sentença em um vetor de elementos significativos ou *tokens*, foi aplicada. Os tokens podem ser palavras, frases, símbolos ou outros elementos textuais com significado (KOWSARI et al., 2019). A *tokenização*, nesse caso, teve enfoque na discriminação de palavras e cada uma delas preencheu uma posição do vetor na ordem exata de ocorrência na sentença analisada.

A partir da obtenção dos *tokens*, é possível aplicar uma série de métodos, a nível de palavra, nos *DataFrames* visando a limpeza dos textos. Isso porque, segundo os autores Kowsari et al. (2019), muitos conjuntos de dados possuem ruídos e características desnecessárias que podem prejudicar a performance dos algoritmos de classificação de texto. Dessa forma, os procedimentos de remoção de sinais de pontuação, de

[4] <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html>
[5] <https://pandas.pydata.org/pandas-docs/stable/index.html>

símbolos numerais, de símbolos de acentuação gráfica e de palavras consideradas irrelevantes para a classificação definidas como *stop words* foram executados nos *tokens* das sentenças.

Após a execução da *tokenização* e das técnicas de limpeza de texto, uma coluna denominada "*preprocessing*" e outra intitulada "*pp_tokens*" foram adicionadas aos *DataFrames*. As duas novas colunas surgem com o objetivo de registrar um histórico da preparação dos dados textuais realizada até então. A primeira coluna armazena as sentenças após as fases da preparação e a segunda guarda os *tokens* dessas mesmas sentenças para possivelmente serem utilizados em etapas posteriores do processo de classificação textual.

Nas figuras 4.4 e 4.5, são apresentadas novamente, como forma de exemplificação, as nuvens de palavras associadas às sentenças contendo o discurso de ódio nos conjuntos de dados. No entanto, em comparação com as nuvens exibidas anteriormente em 4.1 e 4.2, essas nuvens representam o conteúdo das sentenças que passou pelas subetapas de padronização, tokenização e limpeza supracitadas. Portanto, é possível notar uma diferença significativa ao remover o que se considera como ruído no texto.

Figura 4.4: Nuvem de palavras das sentenças rotuladas com "1" do conjunto de dados desenvolvido em Fortuna et al. (2019) após as duas primeiras subetapas da fase de preparação dos dados.

Figura 4.5: Nuvem de palavras das sentenças rotuladas com "1" do conjunto de dados desenvolvido em Pelle (2019) após as duas primeiras subetapas da fase de preparação dos dados.

4.2.1.3 Extração de Características

Como o texto das sentenças já foi refinado anteriormente, é possível obter as representações estruturadas que seguirão para a etapa de extração de padrões de conhecimento. Sendo assim, neste trabalho, quatro diferentes técnicas de extração de características foram utilizadas para gerar, no total, sete entradas ou representações estruturadas diferentes do conjunto de dados: *Term Frequency*, *Term Frequency-Inverse Document Frequency*, *N-Gram* e *Word2Vec*. As duas primeiras são definidas como técnicas de palavras ponderadas, a terceira é uma técnica de linguística computacional adaptável a vários cenários e a última trata-se de uma técnica de *word embeddings* (KOWSARI et al., 2019).

Dessa forma, através do pacote *scikit-learn* desenvolvido para a linguagem *Python3*, as duas primeiras representações estruturadas supracitadas do conjunto de dados textual foram geradas. Para isso, os módulos *CountVectorizer*[6] e *TfidfVectorizer*[7] presentes no pacote foram empregados no processo. Esses módulos implementam

[6]<https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html>

[7]<https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html>

a técnica de palavras ponderadas, a qual, segundo Kowsari et al. (2019), implica na tradução de cada sentença em um vetor com número de posições igual ao total de palavras únicas da base de dados. O conjunto de sentenças traduzidas forma, então, uma matriz.

A matriz mencionada anteriormente é uma representação simplificada e estruturada para documentos de texto definida também como *Bag of Words*. O módulo *CountVectorizer* converte cada *DataFrame* em uma BoW, mapeando os *tokens* através do seu número de ocorrências ou TF, nas sentenças, sem manter a informação da ordem das palavras nos documentos. Ao passo que, o módulo *TfidfVectorizer* utiliza o cálculo da frequência inversa das sentenças combinado com a frequência dos termos em seu método de mapeamento. Essa combinação é conhecida como a técnica TF-IDF e objetiva diminuir o efeito de palavras implicitamente comuns no corpus.

Existe uma outra técnica que também atua na construção das representações estruturadas do conjunto de dados denominada *N-Gram*. Neste trabalho, ela foi utilizada em conjunto com as variações TF e TF-IDF da técnica de palavras ponderadas mencionada anteriormente com o objetivo de amenizar a perda das relações sintáticas entre os termos que ocorre na estrutura da BoW. O *N-Gram* consiste na geração de conjuntos de sequências de palavras, onde o N corresponde ao número de palavras consideradas nas sequências na ordem exata em que ocorrem no texto. Dessa forma, os *tokens*, a partir de N igual a 2, passam a ser formados por sequências de termos, ao invés de só por palavras.

Os módulos *CountVectorizer* e *TfidfVectorizer*, que são apresentados nos trechos de código 4.1 e 4.2, implementam um intervalo de números inteiros para o N da técnica *N-Gram*, onde, num intervalo de 1 a 2, por exemplo, são consideradas como elementos do conjunto as palavras individualmente e também as sequências de duas palavras na construção das representações estruturadas do conjunto de dados. Quando o intervalo vai de 1 até 1, isso implica em considerar somente uma palavra em cada elemento do conjunto de *N-Grams*. Para este trabalho, foram considerados os intervalos inteiros $[1, 1]$, $[1, 2]$ e $[1, 3]$ nas aplicações do *N-Gram*, atribuindo esses valores ao parâmetro *ngram_range* dos módulos.

```
from sklearn.feature_extraction.text import CountVectorizer
bow_vectorizer = CountVectorizer(ngram_range=ngram_range)
```

Código 4.1: Módulo *CountVectorizer* utilizado para gerar as entradas baseadas em TF a partir do conjunto de dados.

```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf_vectorizer = TfidfVectorizer(ngram_range=ngram_range)
```

Código 4.2: Módulo *TfidfVectorizer* utilizado para gerar as entradas baseadas em TF-IDF a partir do conjunto de dados.

A quarta e última técnica de extração de características utilizada é definida como *Word2Vec*. Essa técnica consiste em uma arquitetura aperfeiçoada de *word embeddings*. Nela, cada palavra ou sentença é mapeada em um vetor n-dimensional de números reais. Através de uma rede neural, a abordagem é capaz de descobrir similaridades semânticas entre os vetores que representam as palavras (*1-Gram*) analisando uma quantidade significativa de corpus de texto (KOWSARI et al., 2019).

Em Hartmann et al. (2017), foi desenvolvido e disponibilizado um conjunto de *word embeddings* pré-treinadas para a língua portuguesa. Ao todo, foram incluídos 17 corpus de texto no treinamento desses vetores, o qual foi realizado através da abordagem *Word2Vec* e de outras técnicas de *word embeddings* também. Além disso, para cada técnica, existem vetores disponíveis em dimensões e modelos variados. Optou-se por utilizar, neste trabalho, os vetores 100-dimensionais pré-treinados com a abordagem *Word2Vec* do modelo CBOW acessíveis por meio do repositório do Núcleo Interinstitucional de Linguística Computacional da Universidade de São Paulo[8].

Por conseguinte, as palavras das sentenças presentes nos *DataFrames* foram mapeadas nos vetores 100-dimensionais pré-treinados através do módulo *KeyedVectors*[9] do pacote *Gensim*[10] cuja implementação é apresentada no código 4.3. Com as palavras convertidas nesses vetores e as similaridades entre eles identificadas, é possível combiná-las de forma a manter as relações de conformidade. Isso permite a utilização de estratégias para a representação das sentenças a partir do cálculo da

[8] <http://nilc.icmc.usp.br/nilc/index.php/repositorio-de-word-embeddings-do-nilc>
[9] <https://radimrehurek.com/gensim/models/keyedvectors.html>
[10] <https://radimrehurek.com/gensim/>

média dos vetores que retratam as palavras. Esse método, proposto em Lin (2019), foi adotado na construção de mais uma representação estruturada dos corpus de texto para os algoritmos de classificação.

```
from gensim.models import KeyedVectors
w2v_model = KeyedVectors.load_word2vec_format(
            "datasets/cbow_s100_USP.txt"
            )
```

Código 4.3: Módulo *KeyedVectors* utilizado para a contextualização de vetores de *word embeddings* pré-treinados.

4.2.2 Extração dos Padrões de Conhecimento

Na fase de extração dos padrões de conhecimento ocorrem as subetapas de divisão do conjunto de dados, de reforço de sentenças ofensivas, de execução dos algoritmos e de refinamento dos hiperparâmetros do algoritmo *Random Forest*. É possível visualizar como essas subetapas se relacionam através da figura 4.6 e cada uma delas será detalhada a seguir.

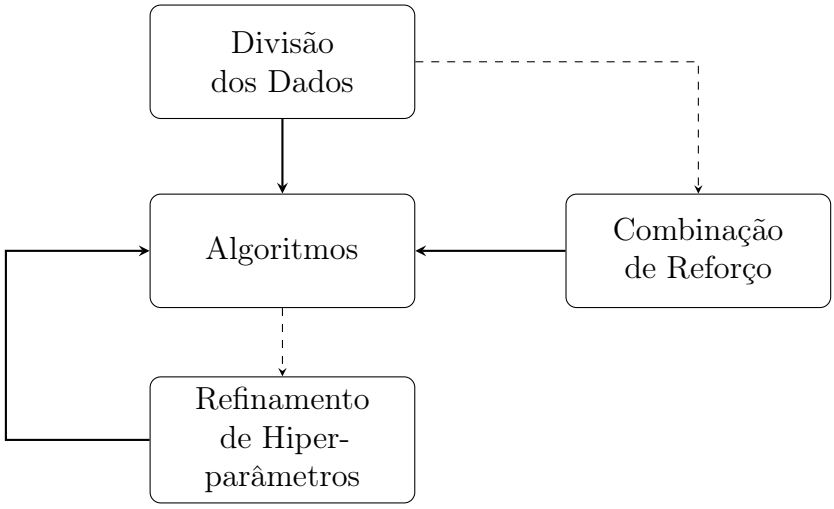

Figura 4.6: Subetapas da etapa de extração dos padrões de conhecimento pertencente ao processo de mineração de texto.

Com o conjunto de dados devidamente preparado e estruturado, o mesmo encontra-se em um formato que pode ser submetido como entrada para os algoritmos de aprendizado de máquina da fase de extração dos padrões de conhecimento presentes

no texto. No entanto, antes da execução desses algoritmos, é necessário que a base da dados seja dividida em duas partes: treino e teste. Somente a porção de dados reservada para treino será utilizada na extração de padrões, enquanto a seção de teste serve para que as predições e, posteriormente, a avaliação dos modelos de classificação sejam realizadas (KOWSARI et al., 2019).

4.2.2.1 Divisão do Conjunto de Dados

A divisão ocorreu através do módulo *StratifiedShuffleSplit*[11] também presente no pacote *scikit-learn*. Esse módulo, além de permitir o controle do número de vezes que o particionamento ocorrerá e o tamanho para cada partição, ainda utiliza uma técnica estratificada e randomizada na realização da tarefa. No processo, cada vez que uma divisão dos dados em treino e teste ocorre, a escolha das amostras para cada subconjunto é aleatória. No entanto, por ser estratificada, a amostragem ainda preserva a porcentagem de amostras presente em cada classe no conjunto de dados para cada subconjunto gerado. O código 4.4 apresenta a implementação do módulo.

```
from sklearn.model_selection import StratifiedShuffleSplit
splitter = StratifiedShuffleSplit(n_splits=10,
                                  test_size=0.2,
                                  random_state=42)
```

Código 4.4: Módulo *StratifiedShuffleSplit* utilizado para a divisão do conjunto de dados em 10 subconjuntos de treino e de teste reproduzíveis por meio do parâmetro *random_state* com 20% dos dados setados para o subconjunto de treino.

Anteriormente, constatou-se, na seção 4.1, que a taxa de sentenças anotadas na classe de discurso de ódio do corpus produzido em Fortuna et al. (2019) era de $31,5\%$. Sendo assim, a construção dos subconjuntos de treino e de teste, na abordagem estratificada, manterá essa mesma proporção ao escolher as sentenças para ambos os subconjuntos. Como a abordagem em questão foi aplicada através do módulo *StratifiedShuffleSplit*, foi necessário setar os parâmetros que regem a quantidade

[11] <https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedShuffleSplit.html>

total de execuções do método de divisão randômica e estratificada e o tamanho dos subconjuntos. Em suma, foram 10 divisões ao todo e o tamanho atribuído ao bloco de teste foi de 20% do tamanho total, o que, consequentemente, seta o bloco de treino para 80% desse valor.

4.2.2.2 Combinação de Reforço

Como já foi mencionado, existem dois conjuntos de dados que são utilizados: um construído em Fortuna et al. (2019) e outro criado em Pelle (2019). O primeiro trata-se da principal fonte de dados de todo o processo, ao passo que o segundo foi aproveitado como reforço para os subconjuntos de treino gerados através da fonte principal. Desse modo, somente as sentenças ofensivas do segundo conjunto foram cogitadas para o reforço atribuído à fase de extração de padrões. Um total de 202 novas sentenças ofensivas são injetadas apenas nos grupos de treinamento, não afetando os subconjuntos de teste nessa combinação.

4.2.2.3 Algoritmos

As principais ferramentas presentes na etapa em questão são os algoritmos de classificação. Eles são os responsáveis por extrair os padrões de conhecimento do conjunto de dados de treinamento e, posteriormente, classificar, a partir dos padrões extraídos, as sentenças de texto do conjunto de dados reservado para teste. Após isso, a performance da classificação pode, então, ser calculada e analisada. Para fins de experimentação, optou-se por analisar o processo de classificação de quatro algoritmos diferentes obtidos através do pacote *scikit-learn*: *Logistic Regression*, *Support Vector Machines*, *Naïve Bayes* e *Random Forest*.

Os módulos dos algoritmos de classificação possuem valores configuráveis, denominados hiperparâmetros, que são responsáveis por conduzir a operação de aprendizado. Cada um desses módulos pode ser executado mantendo os valores já predefinidos neles ou alterando-os de acordo com o contexto e o objetivo da execução. Neste trabalho, a maioria dos algoritmos foi executada com hiperparâmetros fixados e quase todos eles já predefinidos como mostrado no código 4.5. Somente *Logistic Regression* e *Support*

Vector Machines, os quais tiveram um hiperparâmetro que altera o quanto cada classe pesa no processo de classificação modificado, não utilizaram a configuração padrão.

```
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn import svm
clf_lr = LogisticRegression(solver="lbfgs",
                            class_weight="balanced")
clf_svm = svm.SVC(kernel="linear",class_weight="balanced")
clf_nb = GaussianNB()
```

Código 4.5: Módulos *LogisticRegression*, *svm* e *GaussianNB* utilizados na aplicação dos algoritmos *Logistic Regression*, *Support Vector Machines* e *Naïve Bayes* respectivamente.

Tal modificação ocorreu por conta da diferença significativa existente entre a quantidade de sentenças anotadas como positivas e negativas para discurso de ódio no conjunto de dados principal. Sendo assim, optou-se por definir o peso das classes como balanceado, pois essa configuração concede um peso maior para a classe que é minoria e menor para a que é maioria. Isso, segundo Yanminsun, Wong e Kamel (2011), traz melhores resultados ao processo de classificação, uma vez que ameniza a disparidade entre a quantidade de elementos anotados em cada classe do conjunto de dados.

4.2.2.4 Refinamento de Hiperparâmetros

O algoritmo *Random Forest* foi selecionado para ser submetido a um procedimento de refinamento de hiperparâmetros. De acordo com os autores Probst, Wright e Boulesteix (2019), esse procedimento consiste em tentar encontrar os melhores valores de hiperparâmetros de um algoritmo em relação a um conjunto de dados em específico. Esses valores são considerados melhores ou piores em relação a um determinado tipo de métrica de avaliação ou em relação ao tempo de execução do algoritmo. Nesse caso, os hiperparâmetros do *Random Forest* serão refinados considerando uma métrica de avaliação previamente definida.

A estratégia escolhida para refinar os hiperparâmetros do algoritmo em questão é baseada em técnicas de busca. Nesse conjunto de estratégias de refinamento, as opções variam na forma como escolhem os candidatos a valores para os hiperparâmetros. Desse modo, a estratégia adotada é definida como *Random Search* e, nesse caso, ela foi aplicada através do módulo *RandomizedSearchCV*[12], presente no pacote *scikit-learn*, e combina aleatoriamente valores retirados de intervalos predefinidos para cumprir a tarefa de obter os valores refinados (PROBST; WRIGHT; BOULESTEIX, 2019).

Para cada hiperparâmetro envolvido no refinamento, existirá um intervalo de possíveis valores previamente determinados. Ao escolher um valor de cada intervalo, a estratégia em questão cria uma combinação com eles e, posteriormente, avalia, de acordo com a métrica especificada, a performance do algoritmo setado com tal combinação de valores em relação ao conjunto de dados reservado para treino. O número de vezes que o processo ocorrerá é controlado através da quantidade de combinações que se deseja explorar.

Por meio da tabela 4.1, é possível visualizar os intervalos setados para cada hiperparâmetro do *Random Forest* submetido ao refinamento. Os conjuntos de valores, dependendo do parâmetro, podem ser textuais, numéricos ou envolver os dois tipos. No caso dos valores para o parâmetro "Peso das Classes", o conjunto inclui o tipo textual e uma estrutura de dados nativa da linguagem *Python3* denominada *dictionary* que associa valores com chaves específicas. Por exemplo, no caso do elemento "{1: 2}" presente no intervalo em questão, associa-se o valor 2 ao peso da classe representada pela chave 1. Quando o peso de uma classe não é definido no módulo, ele é automaticamente setado para 1.

O código 4.6 apresenta como ocorreu a aplicação do refinamento de parâmetros no trabalho em questão. Dessa forma, nos módulos, o parâmetro *random_state* foi usado para tornar os experimentos reproduzíveis, ao passo que o parâmetro *param_distributions* recebe, na estrutura de dicionário do *Python3*, os intervalos mostrados na tabela 4.1. Ademais, o parâmetro *scoring* recebe a métrica a ser utilizada como referência no processo de refinamento. Por fim, o classificador que

[12] <https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html>

Tabela 4.1: Lista dos hiperparâmetros e seus respectivos valores para o algoritmo *Random Search* que foi refinado através do módulo *RandomizedSearchCV*.

Hiperparâmetro	Valores
Nº de Árvores	$\{x \mid x = 200 + 200 \cdot i, 0 \leq i \leq 9\}$
Máximo de *Features*	$\{"auto", "sqrt"\}$
Profundidade Máxima	$\{x \mid x = 10 + 10 \cdot i, 0 \leq i \leq 10\} \cup \{"None"\}$
Mínimo de Amostras de Corte	$\{2, 5, 10\}$
Mínimo de Amostras por Folha	$\{1, 2, 4\}$
Bootstrap	$\{"true", "false"\}$
Peso das Classes	$\{"balanced", "\{1:1\}", "\{1:2\}", "\{1:4\}", "\{1:6\}", "\{1:8\}"\}$

obteve a melhor performance ao processar o subconjunto de treino, é selecionado.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import RandomizedSearchCV
rf_model = RandomForestClassifier(random_state=42)
rf_random = RandomizedSearchCV(estimator=rf_model,
            param_distributions=random_grid,
            n_iter=200, cv=2, verbose=2,
            random_state=42, scoring='f1')
rf_random.fit(X_train, y_train)
clf_rf = rf_random.best_estimator_
```

Código 4.6: Aplicação dos módulos *RandomForestClassifier* e *RandomizedSearchCV* no refinamento de parâmetros do algoritmo *Random Forest* por meio da técnica *Random Search*.

4.2.3 Avaliação do Conhecimento

Para avaliar a performance dos modelos na execução dos experimentos foram utilizados os módulos das métricas acurácia, precisão, *recall* e medida F1 (ou *F-measure*) contextualizados pelo trecho de código 4.7 e também presentes no pacote *scikit-learn*. Cada vez que um algoritmo de extração de padrões é executado, esses módulos analisam os resultados obtidos através dos classificadores e as métricas geradas são armazenadas em um novo *DataFrame*. Dessa forma, os resultados dos experimentos podem ser registrados para posteriormente serem analisados.

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
ac = accuracy_score(y_test, pred)
f1 = f1_score(y_test, pred)
rec = recall_score(y_test, pred)
pr = precision_score(y_test, pred)
```

Código 4.7: Módulos das métricas extraídas ao longo da execução dos experimentos.

Em relação ao fundamento das métricas, tem-se que acurácia, precisão, *recall* e F1 são baseadas em um conceito denominado matriz de confusão. Essa matriz armazena os valores verdadeiros positivos (VP), verdadeiros negativos (VN), falsos positivos (FP) e falsos negativos (FN) que são obtidos após a execução de um algoritmo de classificação sobre um conjunto de dados de teste anotado (KOWSARI et al., 2019). A tabela 4.2 representa uma matriz de confusão, onde o rótulo verdadeiro faz referência à classe previamente anotada nos dados de teste e a predição diz respeito ao rótulo inferido pelo algoritmo de classificação analisado.

		Predição	
		Negativo	Positivo
Rótulo Verdadeiro	Negativo	VN	FP
	Positivo	FN	VP

Tabela 4.2: Matriz de Confusão

As métricas baseadas na matriz de confusão surgem a partir da manipulação dos valores armazenados nela. A acurácia (AC), por exemplo, é obtida por meio da divisão das predições corretas pelo total de predições realizadas como pode ser visto na equação 4.1. Ao passo que a precisão (PR) é a taxa de verdadeiros positivos sobre todas as predições positivas (Equação (4.2)). Em relação ao *recall*, ele representa a taxa de verdadeiros positivos sobre o total de verdadeiros positivos e falsos negativos somados (Equação (4.3)). Por fim, a métrica F1 corresponde à média harmônica entre o *recall* e a precisão (Equação (4.4)) e combina as duas métricas para obter um só número que indique a qualidade geral do classificador (KOWSARI et al., 2019).

$$AC = \frac{VP + FP}{VP + FP + VN + FN} \tag{4.1}$$

$$PR = \frac{VP}{VP + FP} \tag{4.2}$$

$$RC = \frac{VP}{VP + FN} \tag{4.3}$$

$$F1 = \frac{2 * PR * RC}{PR + RC} \tag{4.4}$$

4.2.4 Organização dos Experimentos

A partir de toda metodologia apresentada até então, os experimentos foram pensados de maneira que fosse possível comparar os resultados com e sem a utilização das técnicas de reforço de sentenças ofensivas na fase de treinamento. Além disso, definiu-se também que todos os algoritmos de classificação serão executados para cada entrada gerada. Dessa forma, os experimentos serão separados por entrada e também pela utilização ou não do reforço de sentenças ofensivas proveniente do conjunto de dados secundário para a etapa de treinamento.

O conjunto de entradas para os algoritmos nos experimentos inclui as abordagens de palavras ponderadas TF e TF-IDF, variando, para os dois casos, o intervalo de números inteiros do *N-Gram* em $[1, 1]$, $[1, 2]$ e $[1, 3]$. Além disso, ainda há a representação estruturada do conjunto de dados criada com o *Word2Vec* através dos *word embeddings* pré-treinados. Com isso, ao todo, existem sete possibilidades de entradas que serão submetidas à fase de extração de padrões de conhecimento.

O objetivo, nessa fase, é poder comparar a performance dos algoritmos de classificação tendo como entrada cada uma das possibilidades supracitadas. No entanto, é importante analisar também o impacto que causa a técnica de reforço mencionada na seção 4.2.2.2. Sendo assim, são, no total, quatorze experimentos, nos

quais todas as entradas são submetidas aos quatro algoritmos com e sem a presença do reforço no treinamento.

4.3 Experimentos

Nessa seção, serão apresentados os detalhes das execuções dos experimentos mencionados em 4.2.4 e os resultados documentados de cada um deles. Os resultados são obtidos a partir das métricas de performance extraídas ao longo do processo de classificação e serão os principais objetos de análise a seguir.

4.3.1 Execução

Em um computador com sistema operacional *Microsoft Windows* 10, memória RAM de 16 *gigabytes* e um processador *Intel Core i7* de sétima geração com frequência de $2,70$ GHz e dois núcleos de processamento, os experimentos levaram cerca de 18 horas contínuas para serem completados utilizando os recursos presentes na configuração em questão. Ademais, os arquivos referentes à ferramenta e aos experimentos estão disponibilizados em um repositório online na plataforma GitHub[13].

Nesse tempo, cada um deles foi executado seguindo os critérios definidos no módulo *StratifiedShuffleSplit*, o qual realizou 10 divisões aleatórias diferentes do conjunto de dados para cada experimento em grupos de treino e de teste, mantendo as proporções de cada classe em cada subconjunto. Isso implicou em 10 combinações de treino e teste diferentes que foram submetidas aos algoritmos em cada uma das 14 possíveis composições experimentais. Como os quatro algoritmos são executados por experimento e os experimentos estão organizados por entrada, ocorreu um total de 140 execuções.

Além disso, o refinamento dos hiperparâmetros do algoritmo *Random Forest* realizou 200 combinações dos valores presentes nos intervalos apresentados na tabela 4.1 uma vez por experimento. Cada uma das sete representações estruturadas do conjunto de dados teve o primeiro subconjunto separado para treino submetido

[13] <https://github.com/dinizvictor/hatedetectiontool>

ao processo de refinamento. Como o reforço de sentenças ofensivas injetadas nos subconjuntos de treino também ocorre, as entradas são reexecutadas com a utilização dele. Logo, o refinamento acontece 14 vezes, somando-se os experimentos com e sem a aplicação do reforço.

4.3.2 Resultados

Os resultados obtidos ao longo das predições realizadas sobre o conjunto de dados principal foram armazenados também em *DataFrames*. Como são realizadas 10 divisões em grupos de treino e de teste, os resultados de cada submissão dos subconjuntos aos algoritmos foi registrado nessas estruturas. A utilização da estrutura do *DataFrame* facilitou o cálculo das médias das métricas geradas nessas submissões e o processo de plotagem dos gráficos também.

Em seguida, serão apresentadas as visualizações dos resultados e suas descrições. Nesse caso, a métrica de avaliação adotada para os resultados foi a definida como F1 já apresentada através da equação 4.4 anteriormente. Essa métrica é calculada através da média harmônica entre o *precision* e o *recall*, trazendo um só valor que representa as duas métricas combinadas.

As visualizações se deram por meio de gráficos em barra, onde, no eixo x, constam as variações do *N-Gram* das representações estruturadas do conjunto de dados baseadas na técnica de palavras ponderadas. Ao passo que, no eixo y, estão os possíveis valores para a média da métrica F1 obtida em cada experimento. Como o *Word2Vec* possui somente uma estrutura de entrada, o eixo x de seus gráficos só contém um único valor. Sendo assim, as barras de cada entrada demonstram o valor da média dos valores de F1 alcançados para cada algoritmo utilizando a entrada especificada em x.

Na figura 4.7, a representação estruturada do conjunto de dados retratada é a *Bag of Words*, também conhecida como *Term Frequency*. É demonstrado que, para as variações de intervalo $[1, 1]$, $[1, 2]$ e $[1, 3]$ do *N-Gram* para esse tipo de entrada, o algoritmo *Random Forest* alcançou os melhores resultados nos três casos. Nesse caso, o algoritmo obteve $0,58$ de média para o intervalo $[1, 1]$ e $0,57$ para os outros

dois experimentos.

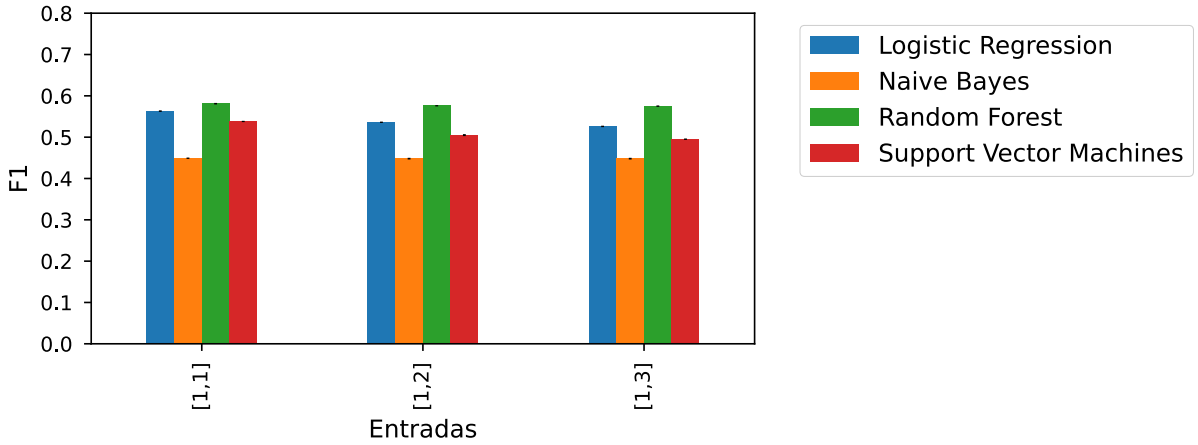

Figura 4.7: Experimentos com *Bag of Words*.

O *Random Forest* seguiu sendo o algoritmo que obteve o melhor resultado no caso representado pela figura 4.8. A melhor média obtida foi de $0,58$ para a variação do *N-Gram* com intervalo $[1, 2]$. Para os outros dois experimentos, a média foi de $0,57$. Nesse caso, o algoritmo *Logistic Regression* alcançou resultados muito próximos ao *Random Forest*, chegando a ter $0,57$ de média para os intervalos $[1, 1]$ e $[1, 2]$.

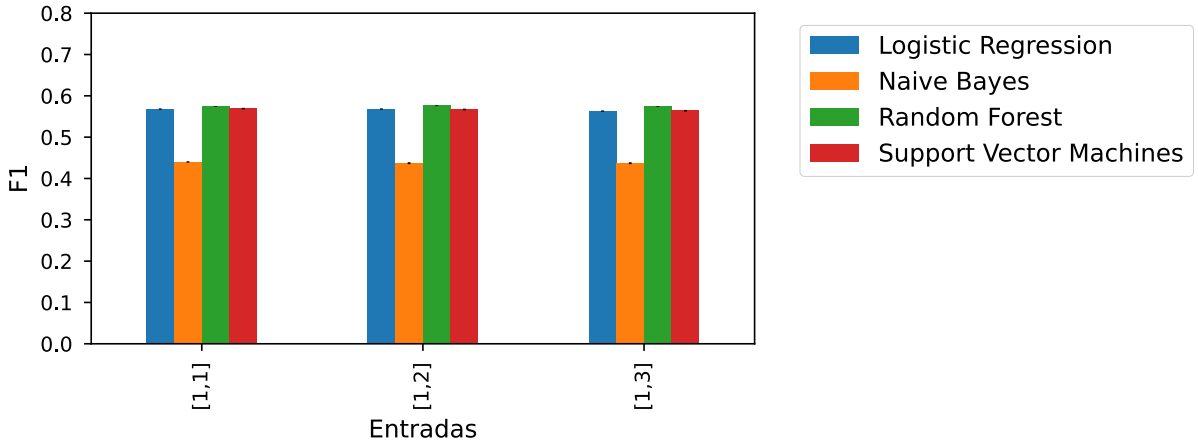

Figura 4.8: Experimentos com TF-IDF.

Mais uma vez, como demonstra a figura 4.9, o *Random Forest* obteve a melhor performance de acordo com a métrica F1. O algoritmo alcançou $0,53$ de média ao executar a representação estruturada do conjunto de dados produzida através da técnica *Word2Vec*. Além disso, novamente o *Logistic Regression* chegou bem perto

obtendo $0,52$ de média.

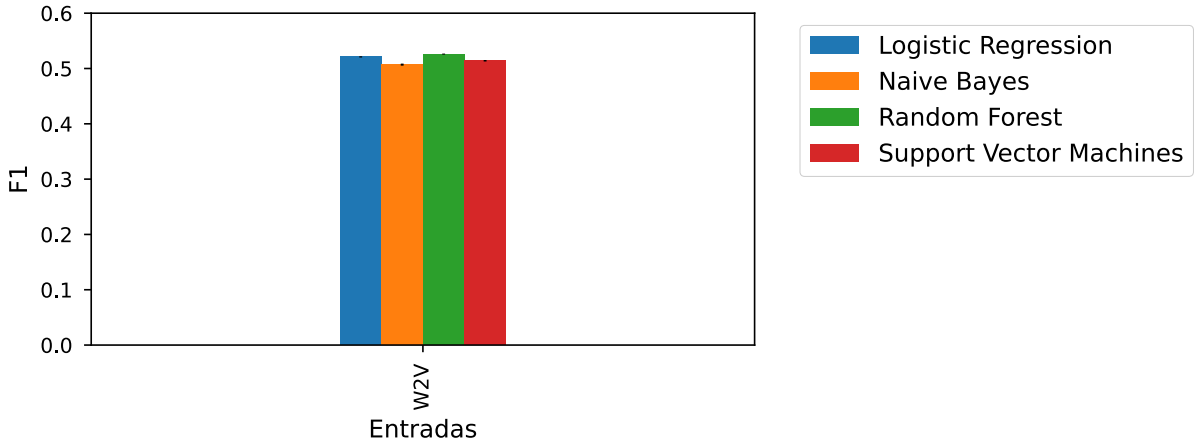

Figura 4.9: Experimentos com *Word2Vec*.

Ao aplicar a técnica de reforço de sentenças ofensivas na fase de treinamento dos algoritmos através do conjunto de dados secundário, os resultados mudam significativamente. Como pode ser visto na figura 4.10, o algoritmo *Logistic Regression*, obteve a melhor performance dentre os demais com uma média de $0,56$ em suas execuções com a variação $[1,1]$ do *N-Gram*.

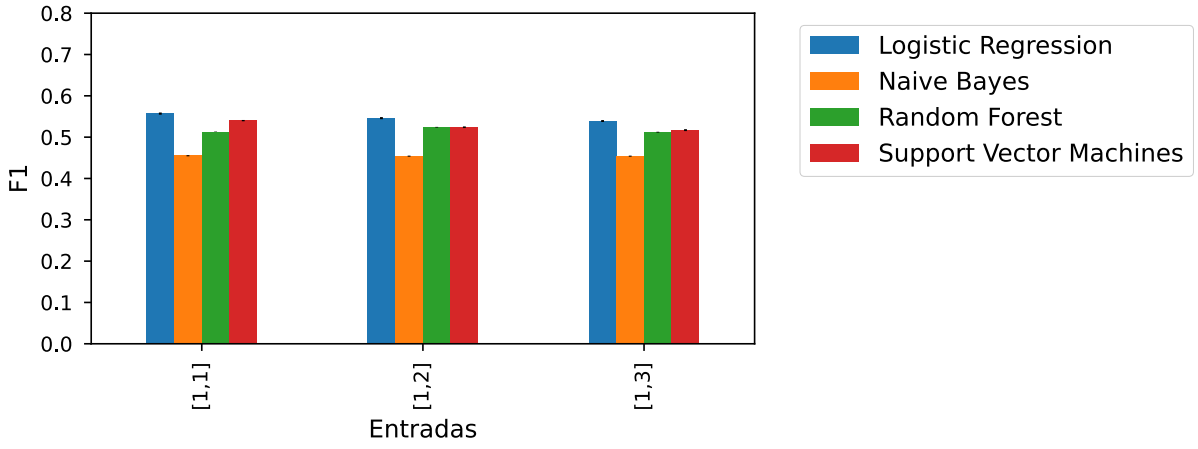

Figura 4.10: Experimentos com *Bag of Words* e reforço no treinamento por meio do conjunto de dados secundário disponibilizado em Pelle (2019).

No caso do TF-IDF combinado com o reforço realizado em seus subconjuntos de treino, o *Logistic Regression* passou a alcançar, mais uma vez, o melhor desempenho em relação aos demais algoritmos como mostra a figura 4.11. O algoritmo obteve

uma média de $0,57$ para os três experimentos que variam os intervalos de *N-Gram* da entrada em questão. Logo em seguida, está o *Support Vector Machines* com $0,56$ de média para todos os experimentos.

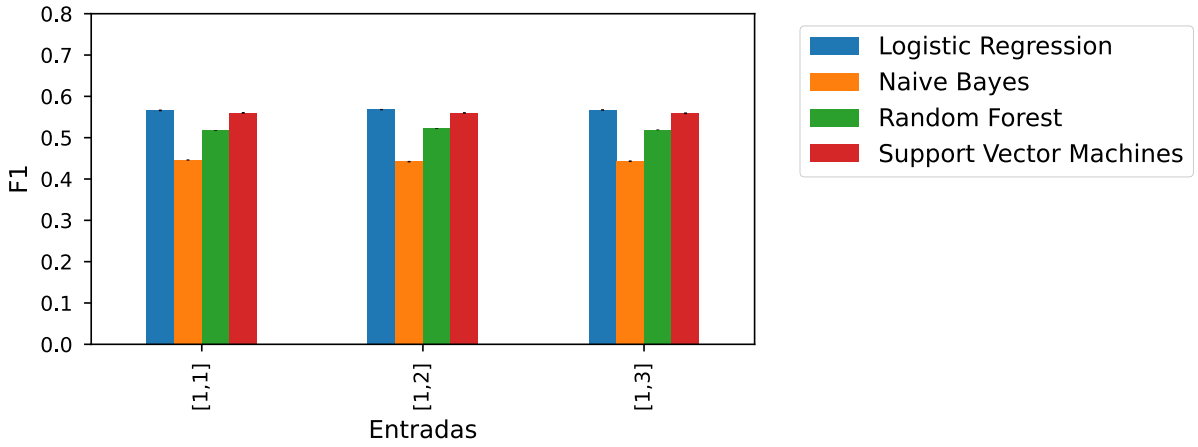

Figura 4.11: Experimentos com TF-IDF e reforço no treinamento por meio do conjunto de dados secundário disponibilizado em Pelle (2019).

Em relação à entrada produzida através do *Word2Vec* utilizando o reforço do conjunto de dados secundário, nota-se que, com uma média igual a $0,52$, os melhores resultados foram obtidos através das predições realizadas com o algoritmo *Logistic Regression*. O *Random Forest* e o *Support Vector Machines*, nesse caso, foram os que mais se aproximaram obtendo um valor $0,51$ de média.

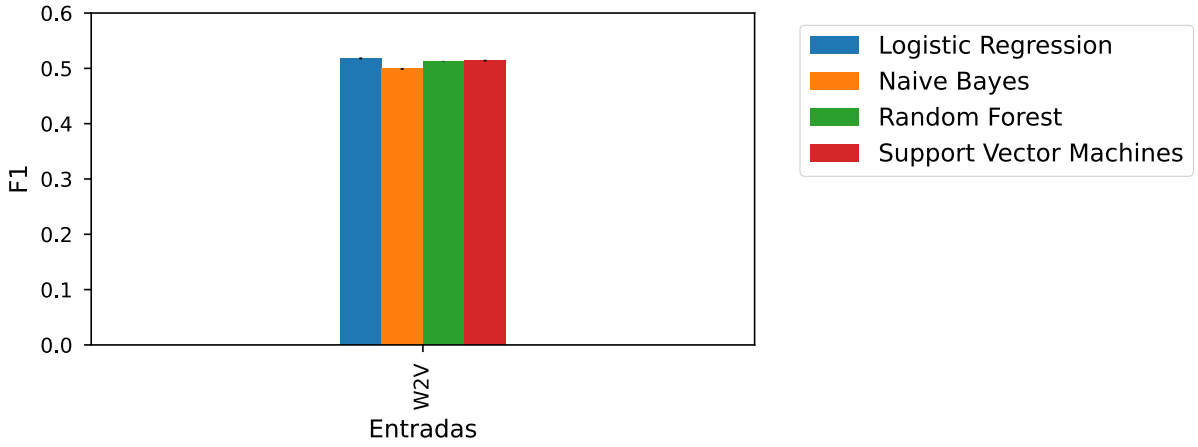

Figura 4.12: Experimentos com *Word2Vec* e reforço no treinamento por meio do conjunto de dados secundário disponibilizado em Pelle (2019).

Dentre todos os resultados apresentados acima, a melhor performance avaliada

foi alcançada através do cenário, onde a representação estruturada do conjunto de dados *Bag of Words* foi utilizada com o intervalo de *N-Gram* igual $[1, 1]$. O algoritmo presente nesse cenário foi o *Random Forest*, o qual alcançou o valor de $0,58$ ao calcular a média das métricas F1 obtidas nas predições realizadas com as subdivisões da entrada em questão.

É possível notar que o algoritmo *Random Forest* obteve seus melhores resultados quando a técnica de reforço não foi aplicada nos subconjuntos, os quais utilizou em sua etapa de treinamento. Em contrapartida, o algoritmo *Logistic Regression* melhorou ligeiramente a maioria de seus resultados quando as representações estruturadas utilizadas por ele adotaram o reforço de sentenças ofensivas presentes no conjunto de dados desenvolvido em Pelle (2019). Isso mostra que, em muitos casos, o reforço não foi benéfico para o processo de classificação dos algoritmos.

O refinamento dos hiperparâmetros do *Random Forest*, o qual, uma vez em cada experimento, seleciona a melhor combinação de hiperparâmetros dentre 200, resultou no alcance das melhores performances para o algoritmo nos experimentos em que não houve aplicação do reforço de sentenças ofensivas. O algoritmo, inclusive, conseguiu alcançar o melhor resultado em comparação a todos os outros experimentos realizados neste trabalho.

Através das matrizes de confusão nas figuras 4.13 e 4.14, é possível comparar um cenário em que não se aplica o reforço na entrada *Bag of Words* com intervalo de *N-Gram* igual a $[1, 1]$ com o cenário em que o reforço é aplicado. O algoritmo que realiza as predições mostradas é o *Random Forest* com hiperparâmetros refinados. No primeiro cenário, 82% das sentenças que não contêm discurso de ódio foram corretamente classificadas e, das sentenças que contêm o discurso, 59% foram preditas de maneira correta.

No segundo cenário apresentado pela figura 4.14 ocorrem mudanças significativas. Nesse cenário, o reforço foi aplicado em conjunto com o algoritmo *Random Forest* e com a estrutura BoW de intervalo $[1, 1]$ no *N-Gram*. Em relação às sentenças que acusam falso para a presença do discurso de ódio, apenas 33% foram corretamente preditas pelo algoritmo. Ao passo que, das sentenças que contêm o ódio, 86% foram

classificadas da forma correta.

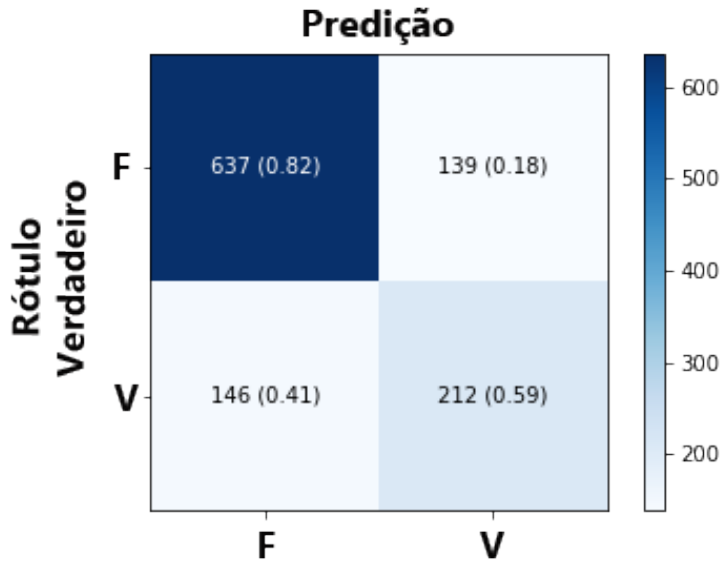

Figura 4.13: Matriz de confusão das predições realizadas pelo algoritmo *Random Forest* com os primeiros subconjuntos de treino e teste gerados a partir de uma entrada BoW com intervalo de *N-Gram* igual a $[1, 1]$.

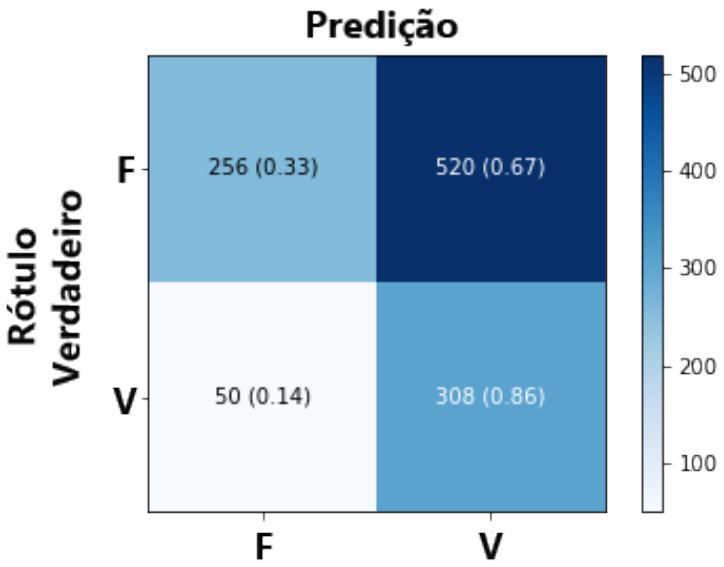

Figura 4.14: Matriz de confusão das predições realizadas pelo algoritmo *Random Forest* com os primeiros subconjuntos de treino e teste gerados a partir de uma entrada BoW com intervalo de *N-Gram* igual a $[1, 1]$ e subconjunto de treino reforçado com as sentenças ofensivas presentes no conjunto de dados secundário.

Sendo assim, é notório que a aplicação do reforço piorou a performance do algoritmo *Random Forest* nos cenários analisados. No entanto, apesar de ter aumentado

a quantidade de predições erradas para as sentenças negativas para discurso de ódio em 49%, o reforço na etapa de treinamento aumentou também a quantidade de acertos em relação às sentenças positivas em 27% para o caso em análise. Ainda assim, a não utilização do reforço concedeu ao *Random Forest*, no caso mostrado pela figura 4.13, o valor de $0,6$ de F1.

Seguidamente, são apresentados exemplos de frases presentes no conjunto de dados principal, as quais aparecem em seu estado original, isto é, sem a utilização de nenhuma técnica de preprocessamento. Essas frases, conhecidas como *tweets*, foram erroneamente classificadas pelo modelo mencionado acima, através do qual obteve-se um F1 de $0,6$. Portanto, as sentenças a seguir compõem os grupos de falsos positivos e falsos negativos do subconjunto de teste predito pelo algoritmo *Random Forest*.

Exemplos de falsos positivos:

1. *"ela eh sapatao eh larissa joga ideia vai q ela se encanta"*
2. *"tem nada de gorda e feia, smp foi linda nenem"*
3. *"só reparei nas tetona da ana paula mesmo, a merda que vcs escrevem eu ja nem me surpreendo mais"*
4. *"a sorosmídia tenta de todas as maneiras fazer o ocidente engolir os refugiados. aqui não neném"*

Exemplos de falsos negativos:

1. *"a crimeia foi ocupada pela rússia durante o governo obama. será que obama foi muito bonzinho com a rússia?"*
2. *"nem as próprias candidatas votam nelas mesmas"*
3. *"uns tem saco, nós temos culhões !!!!!!!!!!!!!!!!!!"*
4. *"era obvio que o estadio do palmeiras tambem ia cair"*

Essas sentenças auxiliam no processo de análise do resultado das predições, uma vez que elas permitem que se faça um comparativo entre o que foi rotulado como discurso de ódio ou não e o que foi classificado pelo modelo. No caso dos exemplos de falsos positivos, é possível notar que existem palavras, as quais aparecem com grande frequência na nuvem de palavras da classe ofensiva apresentada na figura 4.4. Por mais que a sentença, não seja enquadrada como discurso de ódio, a presença dessas palavras tornam difícil a correta classificação dessas sentenças aparentemente.

Em relação aos falsos negativos, os quais são sentenças que contêm o discurso de ódio segundo o processo de anotação do conjunto de dados principal, ao contrário dos falsos positivos, as palavras que aparecem com frequência na classe ofensiva quase não ocorrem nas sentenças. Além disso, o contexto das frases é de difícil interpretação e a ausência de palavras intrínsecas a esse tipo de discurso torna ainda mais complexa a classificação dessas sentenças.

Capítulo 5

Conclusões

Nesse capítulo, serão descritas as considerações finais a respeito deste trabalho, as limitações associadas a ele e as perspectivas de possíveis trabalhos futuros.

5.1 Considerações finais

A identificação do discurso de ódio na Internet permanece sendo, não só um óbice enfrentado pelas corporações que gerenciam as plataformas de mídias sociais, mas também uma necessidade daqueles que são os maiores alvos desse tipo de discurso.

O processo de automatização da identificação do discurso de ódio através do domínio de classificação textual baseada em aprendizado de máquina é uma solução muito promissora para o problema. No entanto, existem muitos desafios presentes em cada etapa da implementação desse tipo de solução.

Nesse contexto, o trabalho em questão propôs o desenvolvimento de uma ferramenta de detecção do discurso de ódio online em português. Para isso, foram realizados experimentos combinando diferentes entradas e algoritmos de classificação na busca pela combinação com a melhor performance segundo a métrica F1. Além disso, diversas análises relacionadas aos resultados foram apresentadas.

Nos experimentos associados às combinações das entradas com os algoritmos de aprendizado de máquina, ocorreu a aplicação do refinamento de hiperparâmetros no

algoritmo *Random Forest* e a análise da adição de sentenças ofensivas retiradas de um segundo conjunto de dados utilizado como reforço na fase de treinamento dos algoritmos.

5.2 Limitações e trabalhos futuros

Na seção em questão, serão explicitados possíveis pontos de melhoria, os quais poderão ser desenvolvidos futuramente a partir deste trabalho. Além disso, serão apresentados os aspectos limitantes identificados ao longo de seu desenvolvimento.

Em Zhang e Luo (2018), os autores esclarecem que é muito mais difícil identificar o discurso de ódio do que apontar quando ele não está presente. Os resultados apresentados através da execução dos experimentos reforçam que a identificação do discurso de ódio é uma tarefa complexa. No experimento que resultou na melhor performance alcançada neste trabalho, segundo a métrica F1, é possível verificar esse ponto, pois 82% das sentenças não odiosas foram preditas corretamente contra 59% dos acertos relacionados às sentenças odientas.

Através do experimento que alcançou a melhor performance, ao analisar as sentenças presentes nos falsos positivos e falsos negativos, notou-se que o processo de anotação do conjunto de dados textual com enfoque no apontamento do discurso de ódio é uma das etapas mais importantes da construção da base de dados. Isso porque, segundo Abro et al. (2020), um dos fatores que torna a identificação automática do discurso odiento algo desafiador são as divergências em relação às diferentes definições do discurso. Dessa forma, o processo de anotação precisa levar em consideração a questão de qual definição utilizar como norte.

Além disso, ainda há a complexidade associada com a integração de dois conjuntos de dados que, apesar de se encaixarem no mesmo domínio, possuem processos de construção distintos. A estratégia de integração pensada neste trabalho consistiu em reforçar a base de dados principal com os dados anotados para discurso ofensivo presentes na base de dados secundária na fase de treinamento dos algoritmos. Essa integração foi realizada com o objetivo de lidar com o desbalanceamento entre as

classes que ocorre nos dois conjuntos de dados.

No entanto, a integração diminuiu significativamente o desempenho do algoritmo *Random Forest* que passou pelo refinamento de hiperparâmetros uma vez em cada experimento. Ao atuar em conjunto com o reforço na fase de treinamento, o algoritmo conseguiu aumentar sua capacidade de identificar o discurso de ódio, porém diminuiu muito seus acertos ao predizer as sentenças que não continham o discurso.

Como, até o momento, só existem duas bases de dados na língua portuguesa mapeadas no *Hate speech data*, um possível trabalho futuro seria a confecção de um conjunto de dados anotado para o discurso de ódio em português grande o suficiente para evitar o desbalanceamento das classes existentes. O processo de anotação poderia permitir, não só a realização de modelos de classificação binária, mas também a possibilidade de realizar tarefas de multiclassificação, já que o discurso de ódio possui também suas ramificações como afirmam Fortuna e Nunes (2018).

A representação estruturada do conjunto de dados gerada com o *Word2Vec* utilizou uma técnica básica, onde os vetores das palavras nas sentenças são combinados em um só por meio da média aritmética realizada entre eles. Como perspectiva de melhoria, essa combinação dos vetores poderia ocorrer através de técnicas mais complexas como a proposta em Djaballah, Boukhalfa e Boussaid (2019) que se trata de uma média ponderada capaz de performar melhor que as técnicas tradicionais.

Outros pontos de melhoria são os relacionados aos experimentos, onde o refinamento de parâmetros poderia ser ampliado para todos os algoritmos utilizados neste trabalho. Além disso, como foi feito em Gaydhani et al. (2018), poderiam ser comparadas as duas formas de implementação da técnica de *Term Frequency - Inverse Document Frequency*, já que, no trabalho em questão, somente foi utilizada a implementação já pré-definida no módulo *TfidfVectorizer*.

Referências

ABIKOYE, O.; OMOKANYE, S.; ARO, T. Text classification using data mining techniques: A review. *Information Systems Education Journal*, v. 22, p. 1–8, 05 2018.

ABRO, S. et al. Automatic hate speech detection using machine learning: A comparative study. *International Journal of Advanced Computer Science and Applications*, The Science and Information Organization, v. 11, n. 8, 2020. Disponível em: <http://dx.doi.org/10.14569/IJACSA.2020.0110861>.

ALJARAH, I. et al. Intelligent detection of hate speech in arabic social network: A machine learning approach. *Journal of Information Science*, p. 016555152091765, 05 2020.

ALKIVIADOU, N. Hate speech on social media networks: towards a regulatory framework? *Information & Communications Technology Law*, Routledge, v. 28, n. 1, p. 19–35, 2019. Disponível em: <https://doi.org/10.1080/13600834.2018.1494417>.

BONACCORSO, G. *Machine Learning Algorithms: A Reference Guide to Popular Algorithms for Data Science and Machine Learning*. [S.l.]: Packt Publishing, 2017. ISBN 1785889621.

CHETTY, N.; ALATHUR, S. Hate speech review in the context of online social networks. *Aggression and Violent Behavior*, v. 40, p. 108–118, 2018. ISSN 1359-1789. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1359178917301064>.

CIDELL, J. Content clouds as exploratory qualitative data analysis. *Area*, v. 42, p. 514 – 523, 11 2010.

DANG, S. A review of text mining techniques associated with various application areas. *International Journal of Science and Research (IJSR)*, v. 4, p. 2461–2466, 02 2015.

DJABALLAH, K. A.; BOUKHALFA, K.; BOUSSAID, O. Sentiment analysis of twitter messages using word2vec by weighted average. In: *2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS)*. [S.l.: s.n.], 2019. p. 223–228.

FLEISS, J. L. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, v. 76, n. 5, p. 378–382, 1971. ISSN 1939-1455(Electronic),0033-2909(Print). Place: US Publisher: American Psychological Association.

FORTUNA, P.; NUNES, S. A survey on automatic detection of hate speech in text. *ACM Comput. Surv.*, Association for Computing Machinery, New York, NY, USA, v. 51, n. 4, jul. 2018. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/3232676>.

FORTUNA, P. et al. A hierarchically-labeled Portuguese hate speech dataset. In: *Proceedings of the Third Workshop on Abusive Language Online*. Florence, Italy: Association for Computational Linguistics, 2019. p. 94–104. Disponível em: <https://www.aclweb.org/anthology/W19-3510>.

GAGLIARDONE DANIT GAL, T. A. G. M. U. I. *Countering online hate speech*. [S.l.]: Paris : United Nations Educational, Scientific and Cultural Organization, 2015. ISBN 9231001051 9789231001055.

GAYDHANI, A. et al. *Detecting Hate Speech and Offensive Language on Twitter using Machine Learning: An N-gram and TFIDF based Approach*. 2018.

GELBER, K.; MCNAMARA, L. Evidencing the harms of hate speech. *Social Identities*, Routledge, v. 22, n. 3, p. 324–341, 2016. Disponível em: <https://doi.org/10.1080/13504630.2015.1128810>.

GIBERT, O. de et al. Hate speech dataset from a white supremacy forum. In: *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*. Brussels, Belgium: Association for Computational Linguistics, 2018. p. 11–20. Disponível em: <https://www.aclweb.org/anthology/W18-5102>.

HARTMANN, N. et al. *Portuguese Word Embeddings: Evaluating on Word Analogies and Natural Language Tasks*. 2017.

JONES, K. S. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, v. 28, p. 11–21, 1972.

KAMRUZZAMAN, S. M.; HAIDER, F.; HASAN, A. R. Text classification using data mining. *CoRR*, abs/1009.4987, 2010. Disponível em: <http://arxiv.org/abs/1009.4987>.

KANE, G. C. et al. What's different about social media networks? a framework and research agenda. *MIS Quarterly*, Management Information Systems Research Center, University of Minnesota, v. 38, n. 1, p. 275–304, 2014. ISSN 02767783, 21629730. Disponível em: <https://www.jstor.org/stable/26554878>.

KAO, A.; POTEET, S. Text mining and natural language processing-introduction for the special issue. *SIGKDD Explorations*, v. 7, p. 1–2, 01 2005.

KAO, A.; POTEET, S. R. *Natural Language Processing and Text Mining*. [S.l.]: Springer Publishing Company, Incorporated, 2006. ISBN 184628175X.

KEMP, H. . W. A. S. S. *DIGITAL 2020: BRAZIL*. 2020. <https://datareportal.com/reports/digital-2020-brazil>. Acessado em 28/06/2020.

KOWSARI, K. et al. Text classification algorithms: A survey. *Information (Switzerland)*, v. 10, 04 2019.

LEINER, B. M. et al. A brief history of the internet. *SIGCOMM Comput. Commun. Rev.*, Association for Computing Machinery, New York, NY, USA, v. 39, n. 5, p. 22–31, out. 2009. ISSN 0146-4833. Disponível em: <https://doi.org/10.1145/1629607.1629613>.

LIN, T. *Performance of Different Word Embeddings on Text Classification*. 2019. <https://towardsdatascience.com/nlp-performance-of-different-word-embeddings-on-text-classification-de648c6262b>. Acessado em 05/04/2021.

LINGUAMATICS. *What is Text Mining, Text Analytics and Natural Language Processing?* 2020. <https://www.linguamatics.com/what-text-mining-text-analytics-and-natural-language-processing>. Acessado em 12/08/2020.

MACAVANEY, S. et al. Hate speech detection: Challenges and solutions. *PLOS ONE*, Public Library of Science, v. 14, n. 8, p. 1–16, 08 2019. Disponível em: <https://doi.org/10.1371/journal.pone.0221152>.

MARTINS, R. et al. Hate speech classification in social media using emotional analysis. In: . [S.l.: s.n.], 2018. p. 61–66.

MIKOLOV, T. et al. *Efficient Estimation of Word Representations in Vector Space*. 2013.

NOVA/SB. *The Webcertain Global Search Social Report*. 2016. <http://www.comunicaquemuda.com.br/dossie/quando-intolerancia-chega-as-redes/>. Acessado em 20/10/2020.

PELLE, R. P. de. *Identificação de Comentários Ofensivos na WEB*. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, 2019.

PROBST, P.; WRIGHT, M. N.; BOULESTEIX, A.-L. Hyperparameters and tuning strategies for random forest. *WIREs Data Mining and Knowledge Discovery*, v. 9, n. 3, p. e1301, 2019. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/widm.1301>.

RESHAMWALA, A.; MISHRA, D.; PAWAR, P. Review on natural language processing. *IRACST – Engineering Science and Technology: An International Journal (ESTIJ)*, v. 3, p. 113–116, 02 2013.

SILVA, L. L. et al. A gestão do discurso de ódio nas plataformas de redes sociais digitais: um comparativo entre facebook, twitter e youtube. *Revista Ibero-Americana de Ciência da Informação*, v. 12, 04 2019.

SILVA, L. L. da. *Crimes de discurso de ódio na internet*. 2019. <https://jus.com.br/artigos/78119/crimes-de-discurso-de-odio-na-internet>. Acessado em 30/10/2020.

SINOARA, R. A.; ANTUNES, J.; REZENDE, S. O. Text mining and semantics: a systematic mapping study. *Journal of the Brazilian Computer Society*, v. 23, n. 1, p. 9, jun. 2017. ISSN 1678-4804. Disponível em: <https://doi.org/10.1186/s13173-017-0058-7>.

TAN, A.-H. et al. Text mining: The state of the art and the challenges. 11 2000.

TITENOK, Y. *Natural Language Processing vs Text Mining*. 2020. <https://sloboda-studio.com/blog/natural-language-processing-vs-text-mining>. Acessado em 06/08/2020.

YANMINSUN; WONG, A.; KAMEL, M. S. Classification of imbalanced data: a review. *International Journal of Pattern Recognition and Artificial Intelligence*, v. 23, 11 2011.

ZHANG, Z.; LUO, L. *Hate Speech Detection: A Solved Problem? The Challenging Case of Long Tail on Twitter*. 2018.