Maíra Baptista de Almeida

Detecção Automática de Discurso de Ódio em Redes Sociais

São José dos Campos, SP

Maíra Baptista de Almeida

Detecção Automática de Discurso de Ódio em Redes Sociais

Trabalho de conclusão de curso apresentado ao Instituto de Ciência e Tecnologia – UNIFESP, como parte das atividades para obtenção do título de Bacharel em Engenharia de Computação.

Universidade Federal de São Paulo – UNIFESP

Instituto de Ciência de Tecnologia

Bacharelado em Engenharia de Computação

Orientador: Profª. Dra. Lilian Berton

São José dos Campos, SP

Elaborado por sistema de geração automática com os dados fornecidos pelo(a) autor(a).

Baptista de Almeida, Maíra
Detecção Automática de Discurso de Ódio em Redes Sociais/ Maíra Baptista de Almeida
Orientador(a) Lilian Berton-São José dos Campos, 2020.
66 p.

Trabalho de Conclusão de Curso-Engenharia da Computação-Universidade Federal de São Paulo-Instituto de Ciência e Tecnologia, 2020.

1. classificação automática. 2. aprendizado de máquina. 3. processamento de texto. 4. discurso de ódio. I. Berton, Lilian , orientador(a). II. Título.

“As AI becomes ever more entwined into everyone’s daily life,
it is crucial that we as trailblazers take our responsibility seriously —
to build out AI that empowers people and that fosters equality, not inequality.“
(Rana El Kaliouby)

Resumo

As redes sociais trouxeram uma mudança de paradigma em relação à maneira que as pessoas se comunicam. Elas permitem que usuários expressem suas opiniões "livremente", sem nenhum tipo de contato humano direto. Isso abre brechas para o surgimento de discurso de ódio na internet. Discurso de ódio se refere a qualquer comentário que ataque um indivíduo/grupo com relação a sua raça, gênero, etnia, nacionalidade, religião, orientação sexual ou outro aspecto passível de discriminação. Em redes sociais que permitem um certo grau de anonimidade, como o Twitter, este problema pode ser exacerbado. Notou-se que poucas contribuições científicas têm sido feitas para contra-atacar este problema em línguas diferentes do inglês. Este trabalho tem como objetivo empregar técnicas de processamento de texto e aprendizado de máquina para fazer a classificação de dados de discurso de ódio. Objetivou-se explorar diferentes algoritmos de classificação em um conjunto de dados formado por *tweets* em português. Os resultados foram avaliados por meio de métricas estatísticas, e através delas, foram feitas comparações entre os resultados obtidos e as principais abordagens consideradas como estado-da-arte. O método obtido foi uma combinação do classificador *Support Vector Machines*, com vetorização através da técnica de *TF-IDF*. Além das técnicas de pré-processamento e vetorização, quatro novas características foram geradas para cada exemplo do conjunto de dados, tendo como base, a contagem de palavras. Este método obteve um *F1-score* de 0.94.

Palavras-chaves: classificação automática. aprendizado de máquina. processamento de texto. discurso de ódio.

Abstract

Social networks brought a paradigm shift when it comes to how humans interact with each other. They allow users to freely express their opinions, without any contact with each other. This may directly influence the spread of hate speech. Hate speech refers to any commentary that is considered an attack to an individual or group, targeting race, gender, ethinicity, nationality, religion, sexual orientation or any other discriminatory aspect. On social networks that allow for a certain degree of anonymity, like Twitter, this problem is exacerbated. Even more, few scientific contributions have been done to explore the hate detection problem in languages other than English. This work aims to explore different text processing and machine learning techniques to perform classification on a dataset made up of portuguese tweets. The results were evaluated according to statistic metrics and compared to current state-of-the-art approaches. The final method was a combination of TF-IDF vectorization paired with a Support vector machine classifier. Along with the pre-processing and vectorization techniques, four new features were engineered for each dataset example, based on word count. The method had a F1-score of 0.94.

Key-words: automatic classification. machine learning. text processing. hate speech.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

Redes sociais são comunidades virtuais onde interações entre pessoas acontecem, podendo abrigar tópicos de discussão específicos ou serem generalistas (MURRAY; WALLER, 2007). Redes sociais vêm sendo utilizadas desde o início da Internet, com uma grande explosão de uso nos últimos anos. A quota de mercado das 20 maiores plataformas sociais aumentou em 11.5% de Janeiro de 2007 para Fevereiro de 2007. Além disso, neste mesmo mês, o tráfego proveniente de redes sociais compunha 6.5% de todo o tráfego da Internet (HITWISE, 2007). Em 2020, o número de usuários ativos de redes sociais é de aproximadamente 3,96 bilhões de pessoas, tendo um crescimento de 10% quando comparado com os números do ano passado (SMARTINSIGHTS, 2020).

Com o advento das redes sociais, e muitas vezes, com a anonimidade que elas trazem, a quantidade de publicações ofensivas vem se tornando um problema crescente (ZHANG; LUO, 2019). O termo discurso de ódio vem sendo utilizado para englobar postagens que atacam ou são de alguma forma ofensivas a uma pessoa ou grupo de pessoas. NOCKLEBY define discurso de ódio como "qualquer tipo de comunicação, que ofende uma pessoa ou grupo com base em alguma característica, como raça, cor, etnia, gênero, sexualidade, nacionalidade, e religião, ou outras características".

Certos grupos que promovem discurso de ódio online estão inclusive ligados a ataques terroristas, sugerindo que redes sociais podem contribuir para a radicalização (ROBERTSON; MELE; TAVERNISE, 2018),(MACAVANEY et al., 2019). Um dos exemplos mais recentes de como uma rede social pode contribuir para um ataque terrorista guiado pelo ódio foi o incidente em Christchurch, na Nova Zelândia, onde um tiroteio contra uma mesquita foi transmitido ao vivo pelo Facebook (TIMES, 2019).

As principais plataformas de redes sociais, como Twitter (WIRED, 2019), Facebook (VERGE, 2019) e YouTube (FAIR, 2019) têm feito esforços para tentar combater este problema, seja através de moderadores humanos ou através de algum tipo de detecção automática de postagens ofensivas. No entanto, a sub-classificação, ou mesmo, classificações errôneas ainda são muito comuns nestes casos.

A maior parte dos dados postados em redes sociais se dão por meio de imagens, vídeos e textos. Nesse trabalho, o objeto de estudo será apenas dados textuais, em especial da plataforma Twitter. O Twitter foi fundado em 2006 por Jack Dorsey, Noah Glass, Biz Stone e Evan Williams, sendo uma rede social de micro-postagens denominadas *tweets*. Permite a postagem de imagens, vídeos e textos com 280 caracteres. Possui cerca de 321 milhões de usuários ativos, com uma receita de US$3.46 bilhões de dólares em 2019 [1].

[1] https://about.twitter.com/pt.html

O processamento de textos é uma área da computação que lida com a manipulação automática de documentos, tendo aplicações em várias áreas como tradução automática, análise de sentimentos, extração de informações, entre várias outras (AGGARWAL; ZHAI, 2012). Dentro do ramo de Processamento de Linguagem Natural (PLN), o processamento de texto tem papel importante, já que prepara entradas para serem interpretadas posteriormente. Dentro deste contexto, *word embeddings* são usadas para mapear características de textos em vetores, tornando possível uma interpretação mais aprofundada (MIKOLOV et al., 2013b). Várias técnicas de mineração de texto podem ser utilizadas em tarefas de PLN. Desta forma, textos vindos de fontes não estruturadas, como *blogs*, *sites*, redes sociais, etc. podem ser utilizados para suprir as necessidades das tarefas de PLN (AGGARWAL; ZHAI, 2012).

De acordo com MITCHELL et al., Aprendizado de Máquina (AM) se refere ao "estudo de algoritmos que permitem que programas de computador melhorem de forma automática através de experiências". O AM possui um papel importante na análise de textos, pois permite o processamento de grandes quantidades de dados e reconhecimento automático de padrões. Dentro da área de AM existem outras sub-áreas, como o aprendizado supervisionado, onde o agente aprende por meio de exemplos rotulados, o aprendizado semissupervisionado, onde o agente aprende por meio de exemplos não rotulados e poucos dados rotulados, e o aprendizado não supervisionado, onde o agente reconhece padrões sem o auxílio de rótulos (LORENA; GAMA; FACELI, 2000).

1.1 Motivação

Ferramentas de detecção e prevenção de discurso de ódio ainda são pouco exploradas, e sua melhoria pode contribuir positivamente, tanto no âmbito econômico como no âmbito social. Tais ferramentas são cruciais para empresas de redes sociais, onde manter um ambiente agradável e justo significa manter o engajamento de clientes, melhorando a saúde econômica dessas empresas. E de forma geral, detectar falas ofensivas na *internet* é benéfico para o empoderamento de minorias e até mesmo a diminuição da radicalização de grupos extremistas (MACAVANEY et al., 2019). Além disso, é crítica para algumas aplicações que utilizam PLN, como *chatbots*, recomendação de conteúdo, análise de sentimentos, entre outras (BADJATIYA et al., 2017).

Apesar do português ser a língua oficial de 9 países e falado por 273 milhões de pessoas, foram encontrados poucos trabalhos que exploram a detecção de discurso de ódio para a língua portuguesa (FORTUNA et al., 2019) (NASCIMENTO et al., 2019) (PELLE; MOREIRA, 2017). Alguns autores tem focado no desenvolvimento/apresentação de técnicas para português (CARVALHO; SIMOES, 2018). Porém, ainda há espaço para melhoria das abordagens aplicadas nessa língua.

Neste trabalho, serão abordadas diferentes técnicas de aprendizado supervisionado para

a detecção de discurso de ódio em língua portuguesa. Os dados explorados são da rede social Twitter. Além das técnicas de classificação, será analisado como diferentes características do texto podem influenciar no resultado final da classificação. Uma comparação dessas abordagens será realizada, a fim de buscar um resultado que represente o estado da arte na detecção de discurso de ódio.

1.2 Objetivos

1.2.1 Objetivo Geral

Esta pesquisa tem como objetivo comparar diferentes técnicas de extração de características textuais e diferentes algoritmos de aprendizado de máquina para a classificação de textos em português relacionados a discurso de ódio.

1.2.2 Objetivos Específicos

Os objetivos específicos são apresentados a seguir:

1. Pesquisar e empregar conjuntos de dados de discurso de ódio na língua portuguesa, com classificação binária.
2. Analisar se a utilização de técnicas de *embeddings* na classificação de textos de discurso de ódio permite melhores resultados que os algoritmos tradicionais baseados em frequência de palavras e bola de palavras.
3. Comparar o desempenho de diferentes algoritmos de classificação supervisionada nos vetores de atributos gerados.
4. Aplicar testes estatísticos para validar se houve precisão nas classificações de textos realizadas.
5. Comparar com resultados reportados na literatura.

1.3 Organização do Documento

O restante deste documento se organiza da seguinte forma. No capítulo 2 são apresentadas as bases teóricas para a compreensão do trabalho. São introduzidos conceitos de processamento e representação de texto, além de uma breve apresentação sobre a teoria de Aprendizado de Máquina, juntamente com alguns algoritmos clássicos. No capítulo 3 são analisados trabalhos relacionados ao tema, além de revisões sistemáticas. Finalmente, no capítulo 4, os métodos para a confecção do trabalho são mostrados. Nele é discutido o conjunto de dados a ser utilizado, as técnicas de pré-processamento e classificação que serão empregadas e também as bibliotecas

utilizadas para auxiliar na implementação. No capitulo 5 são mostrados em detalhes os passos para a montagem de um modelo base. Este modelo é usado para comparar os testes subsequentes, com experimentos utilizando diferentes combinações de vetorização e algoritmos de classificação. Por fim, no capítulo 6, são apresentadas as considerações finais sobre o trabalho.

2 Fundamentação Teórica

Neste capítulo são descritos os principais conceitos relacionados com o desenvolvimento deste trabalho. Na Seção 2.1 apresenta-se a temática discurso de ódio. Em seguida, na Seção 2.2, é feita uma breve introdução acerca de análise de sentimentos e alguns casos de uso. Na Seção 2.3, as etapas que englobam pré-processamento e representação de texto são discutidas com mais detalhes. A Seção 2.4 é dedicada a introduzir conceitos básicos de aprendizado de máquina, bem como apresentar os algoritmos usados no Capítulo 5 deste trabalho. Por fim, nas Seções 2.5 e 2.6 são introduzidos conceitos de amostragem de conjunto de dados e avaliação estatística dos algoritmos, respectivamente.

2.1 Discurso de Ódio

A temática acerca de discurso de ódio vem se tornando mais popular com o passar dos anos, sendo tratada de forma jurídica em diversos países. De acordo com estudos feitos pelo Centro de Direitos Humanos da Universidade de Essex, desde 2006 há relatos sobre o tratamento deste tópico na Austrália, Estados Unidos, Argentina, Canadá, Rússia, Dinamarca, França, Uruguai, Israel, Alemanha, Índia, Holanda, Sri Lanka, África do Sul e Inglaterra (SARMENTO, 2006).

A crescente urgência sobre o tema pode advir tanto de uma maior cobertura por parte da mídia, quanto por uma maior atenção política. Por exemplo, a Comissão Europeia (*European Union Commission*) tem conduzido iniciativas com o objetivo de diminuir a presença de discurso de ódio. Uma dessas iniciativas, a "*No Hate Speech Movement*" patrocinada pelo Conselho Europeu (*Council of Europe*), vem desde 2013 buscando melhorias e mobilizando pessoas para reduzir a presença de discurso de ódio tanto *online* quanto *offline* (EUROPE, 2017). A própria Comissão Europeia tem pressionado rede sociais para tentar garantir um ambiente seguro *online*. Esta pressão veio na forma de uma tentativa de fazer com que as redes sociais Facebook, Twitter e Youtube, além de companhias de tecnologia, como a Microsoft, assinassem um 'código de conduta' para o discurso de ódio (HERN, 2016). Um dos requisitos deste código é a remoção de discurso de ódio das plataformas em até 24 horas. Não obstante, alguns órgãos reguladores afirmam que o Twitter, em especial, não é bom o bastante em retirar mensagens contendo discurso de ódio de sua plataforma (KOTTASOVA, 2017).

No entanto, apesar de todas as tratativas atuais acerca de discurso de ódio, ainda é relativamente difícil decidir se um fragmento de texto faz parte ou não deste contexto, mesmo sendo analisado por humanos. É um fenômeno complexo, e pode mudar radicalmente quando contexto, relacionamento entre grupos e nuances da língua são levados em consideração (FORTUNA; NUNES, 2018).

Mesmo com a dificuldade presente em decidir se um trecho de texto é ou não discurso de ódio, é importante tentar definir no que consiste discurso de ódio. Diversas empresas e organizações, sejam públicas ou privadas, possuem definições próprias sobre o que é discurso de ódio. Na Tabela 1 são apresentadas algumas destas definições.

A escolha de tais organizações se deu da seguinte forma:

- Código de conduta da Comissão Europeia: este órgão regula outras instituições e empresas, contendo uma definição mais generalista.
- Associações Internacionais de Minorias (ILGA): buscam proteger os grupos de pessoas que normalmente são alvos de discurso de ódio.
- Twitter: como é o ponto focal deste trabalho, é necessário trazer a definição de discurso de ódio da plataforma.

Tabela 1 – Definições acerca de discurso de ódio

Organização	Definição
Código de conduta entre a União Europeia e empresas	"Qualquer conduta que incita violência ou ódio de forma pública, direcionada contra um grupo de pessoas ou contra algum membro deste grupo, sendo definida por raça, religião, cor, ascendência, nacionalidade ou etnia (tradução livre)" (WIGAND; VOIN, 2017).
Associações Internacionais de Minorias (ILGA)	"Discurso de ódio é uma expressão pública que espalha, incita, promove ou justifica ódio, discriminação ou hostilidade contra algum grupo especifico. Contribui para um clima generalizado de intolerância, o que torna novos ataques a estes grupos mais prováveis de acontecerem (tradução livre)" (ILGA, 2016).
Twitter	"Conduta de propagação de ódio: não é permitido promover violência, atacar diretamente ou ameaçar outras pessoas com base em raça, etnia, nacionalidade, orientação sexual, sexo, identidade de gênero, religião, idade, deficiência ou doença grave. Também não permitimos contas cuja finalidade principal seja incitar lesões a outros com base nessas categorias" (TWITTER, 2017).

Apesar de ser o berço de nascimento de todas as plataformas sociais mais utilizadas atualmente, não há uma definição legal sobre discurso de ódio nos Estados Unidos (ASSOCIATION et al., 2017). Além disso, discurso de ódio é protegido pela Primeira Emenda da constituição estadunidense, que garante liberdade de expressão por completo. Sob a jurisprudência da Primeira Emenda, discurso de ódio só pode ser criminalizado quando incita atividades criminais diretamente ou então quando traz ameaças de violência contra uma pessoa ou um grupo.

É importante acrescentar que na legislação brasileira não há artigos direcionados à temática de discurso de ódio. No entanto, o inciso IV, do artigo de número 5 da Constituição

Brasileira de 1988 (UNIAO, 1988) mostra que é livre a manifestação do pensamento. No entanto, esta liberdade é limitada, "já que os limites à liberdade de expressão são impostos pelos direitos da personalidade" (GOMES, 2010).

2.2 Análise de Sentimentos

Historicamente, o termo análise de sentimentos se refere ao ato de analisar de forma automática algum fragmento textual, tendo em vista os julgamentos preditivos que os seguem (PANG; LEE, 2009). Assim, o contexto de um determinado fragmento de texto é minerado, tendo como objetivo extrair informações intrínsecas, e por vezes, subjetivas contidas nele.

É uma técnica utilizada por empresas dos mais diversos ramos, com o intuito de avaliar a satisfação de usuários e alavancar novas estratégias de negócios. Por exemplo, a rede de comida rápida KFC vem passando por um *rebranding* alimentado pelo uso desta técnica. Através de campanhas de *marketing* inovadoras, um engajamento social é gerado. Este engajamento é usado para monitoramento de performance com análise de sentimentos baseada em aspectos, de forma a descobrir quais pontos da campanha obtiveram maior atenção. Além disso, a mineração de tópicos em redes sociais também é feita, extraindo novas ideias e variações para novas campanhas. Este processo pode ser feito de forma contínua, gerando ganhos positivos (BILYK, 2020).

Além do KFC, a Google também emprega análise de sentimentos. Uma das vertentes onde a técnica é empregada é a melhoria de produtos. O time de desenvolvimento da empresa monitora constantemente o a opinião e comportamento dos usuários do navegador *Chrome*, seja de forma direta ou indireta. Ou seja, é analisado o sentimento em si ao redor do produto (positivo ou negativo), menções de aspectos específicos (como extensões, segurança ou usabilidade), e também recomendações do produto em si. Estes comentários são monitorados através da ferramenta do *Chrome* e também através de mineração de opiniões em *sites* ou *blogs*. Esses dados mostram com clareza os pontos fracos do produto, o que torna o processo de melhoria do mesmo mais direto (BILYK, 2020).

A respeito do tema deste trabalho, a análise de sentimentos possui grande valor para plataformas de redes sociais, já que a moderação realizada por humanos é inviável, devido ao imenso número de postagens. No entanto, esta técnica sozinha, muitas vezes, não é capaz de gerar resultados relevantes, sendo necessária a ação conjunta com outras técnicas. Tais técnicas, relacionadas à aplicação de algoritmos de inteligência artificial e aprendizado de máquina, serão descritas posteriormente.

2.3 Processamento de Textos

O pré-processamento dos textos é crucial não somente para convertê-los em estruturas interpretáveis pelos algoritmos de aprendizado de máquina, mas para melhorar a performance dos mesmos, removendo informações irrelevantes para a tarefa de classificação textual. É comum aplicar diversas técnicas de pré-processamento em conjunto. Algumas das mais utilizadas são descritas a seguir.

2.3.1 Normalização de texto

Uma das etapas cruciais que ocorre durante o processamento de texto é a normalização textual. Ela deve ocorrer antes de qualquer manipulação do texto (JURASKY; MARTIN, 2000). A normalização é um processo composto por várias etapas, sendo as mais comuns discutidas a seguir.

2.3.1.1 Conversão para caixa baixa e remoção de pontuações

A conversão para caixa baixa é uma etapa recorrente em várias tarefas de PLN, como recuperação de informação ou reconhecimento de voz (JURASKY; MARTIN, 2000). Com ela, a palavra “Pássaro” e “pássaro” passam a ser representadas de forma idêntica. De maneira similar, a remoção de pontuações também é comumente realizada, para que o processamento se torne mais simples.

Certas tarefas de Processamento de Linguagem Natural, no entanto, podem se beneficiar em pular essa etapa de pré-processamento, como tradução automática, análise de sentimentos e extração de informações (JURASKY; MARTIN, 2000).

2.3.1.2 Segmentação

Também conhecida por seu termo em inglês, *tokenization*, a segmentação é o ato de segmentar um texto em palavras, ditos *token* (JURASKY; MARTIN, 2000) (MANNING; SCHÜTZE; RAGHAVAN, 2008). Este processo pode ou não manter pontuações e números. A decisão para retirá-los ou mantê-los deve depender da aplicação. Um *token* pode ser definido como “uma instância de uma sequência de caracteres em um documento, agrupados de tal forma que produzem um unidade semântica útil para processamento” (MANNING; SCHÜTZE; RAGHAVAN, 2008).

2.3.1.3 Remoção de *stopwords*

Em qualquer tipo de texto, certas palavras ocorrem com uma frequência muito maior do que outras, oferecendo pouco valor semântico. Essas palavras são conhecidas como *stopwords* e devem ser retiradas do vocabulário para permitir a redução de ruído em tarefas de processamento (MANNING; SCHÜTZE; RAGHAVAN, 2008).

2.3.1.4 Lematização

Lematização é o ato de determinar a raiz de uma palavra, através de análises morfológicas e de vocabulário (JURASKY; MARTIN, 2000). O retorno de processo de lematização é o chamado lema, que é a palavra resultante sem inflexões (MANNING; SCHÜTZE; RAGHAVAN, 2008).

Um exemplo de lematização na língua inglesa pode ser dado através das várias formas de conjugação do verbo *to be*. Palavras como *am*, *are*, ou *is*, após o processo de lematização são transformadas para *be*.

2.3.1.5 *Stemming*

Como algoritmos de lematização tendem a ser complexos por necessitarem de análises morfológicas, uma heurística mais simples, como o *stemming*, pode ser aplicada em seu lugar. No *stemming*, os sufixos de uma palavra são removidos. O algoritmo mais comum para a realização do *stemming* é o *Porter Stemming*, onde cinco rodadas de reduções de palavras são feitas de forma sequencial. Em cada rodada, diferentes regras são aplicadas para selecionar a palavra final (MANNING; SCHÜTZE; RAGHAVAN, 2008).

Em um exemplo dado por (JURASKY; MARTIN, 2000) em seu livro *Speech and Language Processing*, quando aplicado ao seguinte parágrafo:

> *"This was not the map we found in Billy Bones's chest, but an accurate copy, complete in all things-names and heights and soundings-with the single exception of the red crosses and the written notes."*

O *Porter Stemmer* retornará:

> *"Thi wa not the map we found in Billi Bone s chest but an accur copi complet in all thing name and height and soundwith the singl except of the red cross and the written note"*

2.3.2 Representação de texto

Muitas aplicações que utilizam Processamento de Linguagem Natural, como alguns algoritmos de aprendizado de máquina, não conseguem aceitar entradas de texto puras, necessitando que sejam convertidos para vetores de números.

Desta forma os vetores são derivados das informações contidas no texto, refletindo suas propriedades linguísticas (GOLDBERG; HIRST,). Este processo é chamado de extração de características.

A seguir serão elencadas as técnicas mais comuns de representação textual.

2.3.2.1 *Bag-of-words*

O *Bag-of-words* (BoW) ou bolsa de palavras, é uma maneira de representar um texto que descreve a ocorrência de palavras em um documento. Para isso, leva em consideração um vocabulário das palavras conhecidas e também uma medida da presença dessas palavras.

Como é feito um histograma das palavras em um texto, a contagem das palavras é considerada uma característica. Desta forma, a estrutura e a ordem das palavras são descartadas (GOLDBERG; HIRST,).

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

Um dos problemas do uso de BoW é que a simples contagem de palavras pode não ser discriminativa o bastante, não sendo portanto, uma boa maneira de medir associações entre palavras. Além disso, muitas vezes palavras que ocorrem muito em um texto podem não oferecer alto significado semântico, enquanto palavras que ocorrem com menos frequência são mais importantes. O algoritmo TF-IDF reescala a frequência de palavras através do texto, penalizando palavras que surgem mais vezes (MANNING; SCHÜTZE; RAGHAVAN, 2008).

O TF-IDF considera dois termos, um chamado de *term frequency* tf (ou frequência do termo) e o outro chamado de *document frequency* df (ou frequência no documento). O $tf_{t,d}$ representa a contagem de cada termo t por um documento d, enquanto df_t representa a quantidade de documentos em que certo termo apareceu. Para enfatizar palavras que aparecem com menos frequência, o inverso do df é usado, o chamado de *inverse document frequency*, idf. O idf é dado por:

$$idf_t = \frac{N}{df_t}$$

onde N é o número total de documentos na coleção (JURASKY; MARTIN, 2000).

Assim, o $tf - idf$ é o produto desses dois termos:

$$tf - idf = tf_{t,d} \times idf_t$$

2.3.2.3 *Word-embedding* via redes neurais

As vetorizações geradas por técnicas como o TF-IDF geram longos vetores esparsos, porém, vetores densos podem ser melhores para representar textos em tarefas de PLN (JURASKY; MARTIN, 2000). Para gerar esse vetores densos, *word embeddings* são utilizados.

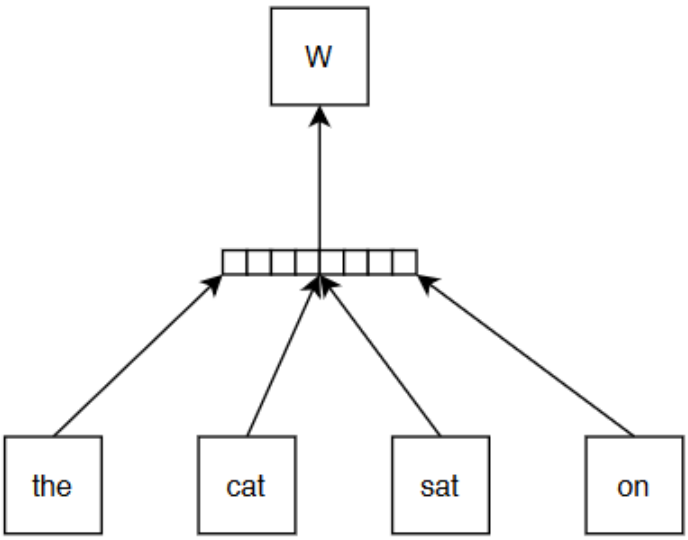
Word embedding é uma técnica que aprende a representação dos significados das palavras, fazendo com que seja possível capturar contexto em um texto (JURASKY; MARTIN, 2000). Uma das técnicas para gerar *embeddings* é a word2vec, criada por (MIKOLOV et al., 2013a). Ao invés de contar o quão frequente uma certa palavra p aparece próxima de uma outra palavra w, um classificador é treinado para predizer se é provável que a palavra p apareça perto da palavra w (JURASKY; MARTIN, 2000). Desta forma, com o word2vec, é possível agrupar

palavras similares em um espaço vetorial. O word2vec pode ser implementado através de duas arquiteturas diferentes, ambas desenvolvidas por MIKOLOV et al.: a *Continuous-Bag-of-Words* (CBOW) e a *Skipgram*.

As duas arquiteturas possuem camadas de Entrada, Projeção e Saída, porém os processos de derivação acontecem de forma diferente (JANG; KIM; KIM, 2019). A camada de entrada recebe, como argumento, n palavras W_n, onde $W_n = W_{(t-2)}, W_{(t-1)}, ..., W_{(t+1)}, W_{(t+2)}$. A camada de projeção corresponde a um vetor de vetores multi-dimensionais, e armazena a soma desses vetores. Finalmente, a camada de saída diz respeito à camada que mostra os resultados dos vetores da camada de projeção (JANG; KIM; KIM, 2019).

- **CBOW:** a arquitetura CBOW é similar a uma rede neural, predizendo a palavra de saída a partir de vetores próximos, ou seja, baseada em seu contexto. A camada de projeção da CBOW projeta todos os vetores na mesma posição, e assim os vetores de todas as palavras mantém uma média, compartilhando as posições de todas as palavras (JANG; KIM; KIM, 2019). A vantagem desta arquitetura é organizar de forma uniforme a informação que está distribuída no conjunto de dado. Uma representação gráfica da arquitetura CBOW pode ser vista na Figura 1.

Figura 1 – Diagrama exemplificando a arquitetura CBOW.

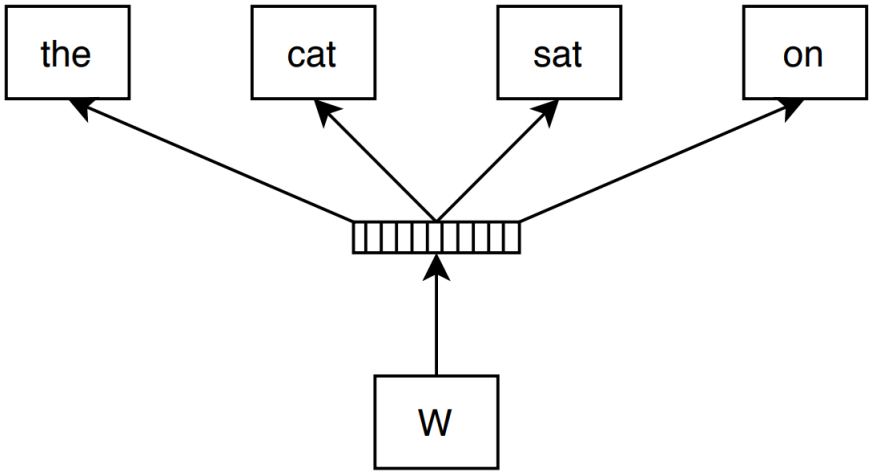


Fonte: elaborada pela autora.

- ***Skipgram*:** já a arquitetura *skipgram* tenta predizer vetores a partir de outras palavras, ou até mesmo a partir de uma palavra. Ou seja, o contexto é predito a partir de uma palavra. A camada de projeção da *skipgram* usa as palavras vindas da camada de entrada para predizer palavras próximas. Desta forma, tem como vantagem a vetorização de palavras conforme elas aparecem, possuindo uma performance melhor do que a CBOW no

aprendizado de novas palavras (JANG; KIM; KIM, 2019). No entanto, é mais custosa em conjuntos de dados maiores. Uma representação gráfica da arquitetura *skipgram* pode ser vista na Figura 2.

Figura 2 – Diagrama exemplificando a arquitetura *Skipgram*.



Fonte: elaborada pela autora.

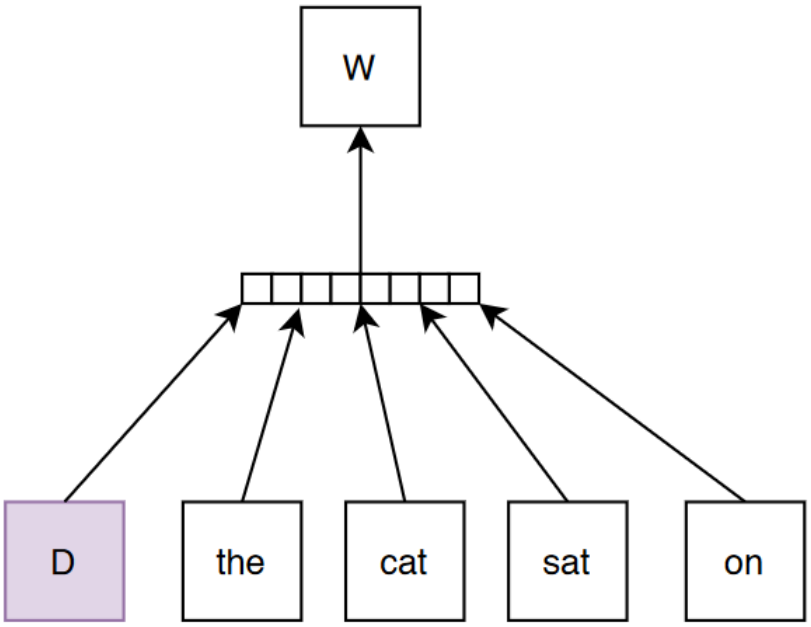
Enquanto o word2vec considera palavras, há uma outra técnica de *embedding* que leva em consideração grupos de palavras, podendo estes serem frases, parágrafos ou até mesmo documentos inteiros (LE; MIKOLOV, 2014). Esta técnica é chamada de Doc2Vec, e cria vetores para toda uma porção de texto. Diferentemente de palavras, documentos não possuem um estrutura lógica. Portanto, um novo vetor é adicionado ao modelo de word2vec, o *Paragraph ID*, funcionando como uma forma de identificar parágrafos.

Assim como o word2vec, o Doc2Vec pode ser implementado de acordo com duas arquiteturas diferentes, baseadas nas arquiteturas CBOW e Skipgram.

- **PV-DM:** a arquitetura baseada em CBOW é chamada de *Distributed Memory Model of Parapgraph Vectors*, ou PV-DM. Nesta arquitetura, exemplificada pela Figura 3, cada parágrafo é mapeado em um vetor único, representado por uma coluna em um matriz D. O vetor de parágrafos e os vetores de palavras são concatenados para predizer a palavra seguinte, dado o contexto. De certa forma, a única diferença entre o PV-DM e o CBOW é a introdução de um vetor de parágrafos, e este vetor pode ser considerado como uma palavra extra (LE; MIKOLOV, 2014).

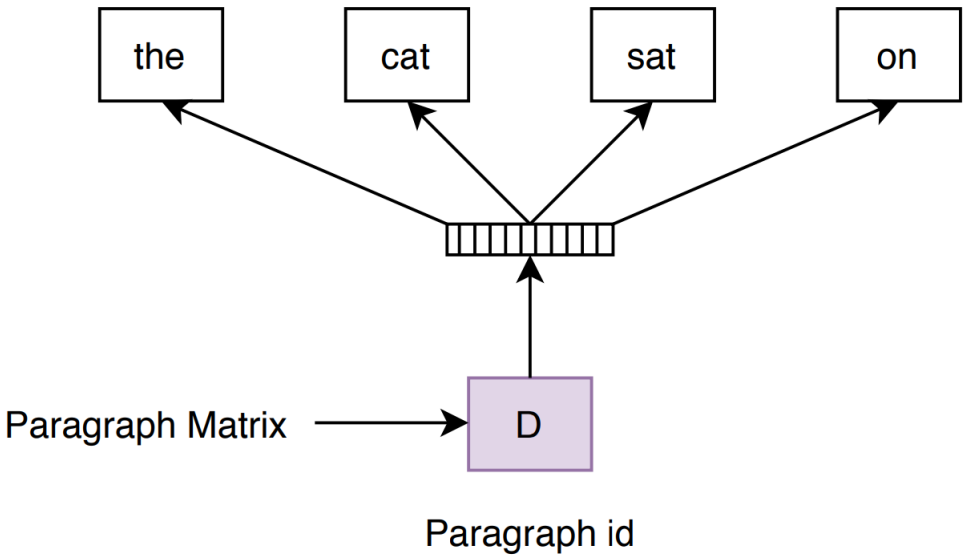
- **DBOW:** A outra arquitetura presente em implementações Doc2Vec é a *Distributed Bag-of-Words* (DBOW). A arquitetura DBOW é similar à arquitetura Skipgram, com o diferencial de usar o *Paragraph ID* ao invés de uma palavra para predizer o contexto de um documento. A Figura 4 mostra um exemplo gráfico da arquitetura DBOW (LE; MIKOLOV, 2014).

Figura 3 – PV-DM é a versão para documentos da arquitetura CBOW.



Fonte: elaborada pela autora.

Figura 4 – Diagrama exemplificando a arquitetura DBOW



Fonte: elaborada pela autora.

2.3.3 Modelagem de Linguagem

Modelos de linguagem são modelos que atribuem probabilidades a sequencias de palavras (JURASKY; MARTIN, 2000). Esse tipo de probabilidades são úteis para identificar palavras relevantes no meio de entradas ambíguas. É possível citar o N-grama como o modelo de linguaguem mais simples (JURASKY; MARTIN, 2000), sendo este o modelo usado neste

trabalho.

O N-grama é uma sequência de n-palavras. Utilizando o exemplo de JURASKY; MARTIN, na frase "*The quick brown fox jumps over the lazy dog*", podemos derivar os seguintes bigramas (n-gramas de duas palavras):

- <start> The
- The quick
- quick brown
- brown fox
- fox jumps
- jumps over
- over the
- the lazy
- lazy dog
- dog <end>

O mesmo princípio pode ser aplicado para trigramas (n-gramas de 3 palavras), quadrigramas (n-gramas de quatro palavras) e assim por diante.

A probabilidade do modelo de bigrama predizer a palavra seguinte é dada por JURASKY; MARTIN:

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})} \tag{2.1}$$

onde $P(w_n|w_{n-1})$ é a probabilidade da palavra atual w_n dada pela palavra anterior w_{n-1}. O numerador da equação ($C(w_{n-1}w_n)$) pode ser calculado considerando todo o vocabulário ou *corpus*, contando a frequência de quantas vezes a palavra atual apareceu ao lado da palavra anterior. O denominador da equação ($C(w_{n-1})$) é a frequência de todas as instâncias da palavra anterior.

2.4 Aprendizado de Máquina

Aprendizado de Máquina (AM) é uma área de pesquisa da computação que busca encontrar maneiras de induzir hipóteses de forma automática, com base em experiências passadas

(LORENA; GAMA; FACELI, 2000). MITCHELL et al. também define aprendizado de máquina como " a capacidade de melhorar o desempenho na realização de alguma tarefa por meio da experiência".

Existem diversos paradigmas dentro da área de Aprendizado de Máquina:

- ***Aprendizado supervisionado:*** usa um conjunto de dados rotulado para treinar um modelo a aprender uma função que prevê corretamente a saída de uma entrada desconhecida (KOTSIANTIS; ZAHARAKIS; PINTELAS, 2007). Dentre os algoritmos de classificação supervisionada é possível citar o *Naive-Bayes*, *Support Vector Machines* (SVMs), Árvores de Decisão, Redes Neurais Artificiais, etc.

- ***Aprendizado não supervisionado:*** usa dados não rotulados como entrada para fazer uma predição . Desta forma, dados similares são identificados e agrupados entre si. A classificação não supervisionada acontece através de algoritmos de clusterização, como *K-means* (BISHOP, 2006) (JAIN; DUBES, 1988).

- ***Aprendizado semissupervisionado:*** é uma técnica híbrida, onde uma pequena parte dos dados é rotulada e a grande parte não é (OLIVIER; BERNHARD; ALEXANDER, 2006).

- ***Aprendizado por reforço:*** tem por objetivo maximizar o resultado numérico de uma função para obter o melhor resultado possível em um objetivo a longo prazo. *Q-Learning* e Teoria de Jogos Bayesiana são algumas das técnicas de classificação apresentadas em (WATKINS; DAYAN, 1992).

- ***Aprendizado ativo***: é uma técnica de amostragem seletiva onde o protocolo de aprendizado possui o controle sobre os dados que serão usados para treinamento (TOMANEK; HAHN, 2009). Com isso, a quantidade de amostras que precisam ser de fato rotuladas diminui, sendo uma ótima abordagem para conjuntos de dados incompletos.

Este trabalho tem como foco a utilização de algoritmos de aprendizado supervisionado. Desta forma, são mostradas a seguir algumas definições consideradas importantes acerca deste paradigma, para a total compreensão do trabalho.

- **Indutor**: o indutor é qualquer algoritmo de indução, e sua principal tarefa consiste em extrair um classificador a partir de um conjunto de dados rotulados (REZENDE, 2003).

- **Exemplo**: um exemplo, no contexto de aprendizado de máquina, se refere a um vetor de valores de atributos (REZENDE, 2003). Também pode ser referido por registro ou dado, sendo a origem da expressão conjunto de dados, ou conjuntos de dados. Um dado descreve o objeto de interesse, podendo ser características de uma casa à venda, ou então características de flores, por exemplo.

Um conjunto de dados T é então, um conjunto de n exemplos com m atributos (REZENDE, 2003), como pode ser visto na Tabela 2. Neste caso, a linha i se refere ao i-ésimo exemplo, com $i = 1, 2, ..., n$; as entradas do conjuntos de dados são os atributos x_{ij}, onde j se refere ao j-ésimo atributo X, sendo $j = 1, 2, ..., m$. Por fim, as saídas y_i se referem às i-ésimas saídas pertinentes a cada entrada.

Tabela 2 – Conjunto de exemplos

	X_1	X_2	...	X_m	Y
T_1	x_{11}	x_{12}	...	x_{1m}	y_1
T_2	x_{21}	x_{22}	...	x_{2m}	y_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
T_n	x_{n1}	x_{n2}	...	x_{nm}	y_n

Um conjunto de exemplos é usualmente dividido em conjuntos de treinamento e conjuntos de teste para tarefas de aprendizado supervisionado. Um conjunto de treinamento é usado para aprender o conceito e o conjunto de teste é usado para testar o grau de efetividade do conceito aprendido.

As divisões de treino e teste são feitas de forma aleatória para assegurar que o resultado final seja estatisticamente válido.

- **Atributo**: de acordo com REZENDE, "um atributo descreve alguma característica ou aspecto de um exemplo", podem ser nominais ou contínuos. Atributos nominais são as características que são exemplificadas por um nome, ou um adjetivo, como cores (azul, vermelho, amarelo), tamanho (grande, pequeno), entre outros. Já um atributo contínuo descreve um valor real, como peso, altura, metragem de um ambiente, etc.

  Indutores geralmente assumem que os atributos usados como entrada são relevantes; no entanto é necessário observar o domínio dos dados além de qual pergunta específica procura ser respondida com a predição. Por exemplo, para determinar se é possível dar uma volta no parque, usar atributos com alto poder preditivo (como temperatura, clima, chance de chuva, etc) é muito mais eficiente do que usar atributos de baixo poder preditivo (como modelo do carro, cor de roupa, etc).

- **Classe**: classes são também conhecidas como rótulos, e descrevem o fenômeno de interesse (REZENDE, 2003). São presentes em tarefas de classificação através de um conjunto discreto $C_1, C_2, ...C_k$. Em tarefas de regressão são representadas por valores reais.

- **Ruído**: é chamado de ruído a interferência presente em um conjunto de dados. Essa interferência pode advir do próprio processo de geração ou aquisição do conjunto, da etapa de processamento, ou até mesmo por classes rotuladas de maneira incorreta (REZENDE, 2003).

- ***Bias***: um *bias* ou viés se refere à qualquer preferência de uma hipótese sobre outra (REZENDE, 2003). Ou seja, um classificador pode produzir várias hipóteses consistentes com o conjunto de dados.

- ***Underfitting* e *Overfitting***: um problema muito comum em tarefas de aprendizado supervisionado é a existência de *underfitting* ou *overfitting*. *Overfitting* está relacionado ao classificador se ajustar de forma muito específica com os dados utilizados no conjunto de treinamento. Isso pode levar erroneamente a um bom resultado de treino, enquanto são observados resultados piores quando testados com outros conjuntos de dados similares (REZENDE, 2003).

  Por outro lado, também é possível que o contrário ocorra. Quando há poucos dados relevantes para a tarefa de predição, o classificador tem dificuldades de traçar uma hipótese robusta, fazendo com que o resultado sobre o conjunto de treino seja ruim. Este fenômeno é chamado de *underfitting* (REZENDE, 2003).

- ***Overtuning***: o fenômeno de *overtuning* é similar ao de *overfitting*, mas ocorre quando o programador ajusta o algoritmo de indução em excesso. Isso pode levar a um desempenho extremamente otimizado, no entanto a performance sobre outros conjuntos de dados similares pode ser pior (REZENDE, 2003). Tanto *overfitting* quanto *overtuning* podem ser amenizados fazendo a separação do conjunto de dados em treino e teste.

Nesse trabalho serão utilizados diferentes algoritmos supervisionados de classificação como: *k-NN*, *Naive Bayes*, *Random Forest*, Regressão Logística, *CatBoost*, *Support Vector Machines* e MLP sumarizados a seguir.

2.4.1 *k-Nearest Neighbors* (k-NN)

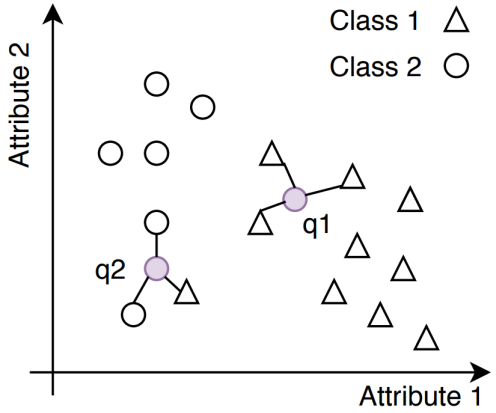
A técnica de classificação baseada no vizinho mais próximo permite que as amostras sejam classificadas baseadas na classe de seus vizinhos mais próximos (CUNNINGHAM; DELANY, 2020). Quando há mais de um vizinho, é chamada então de classificação *k-Nearest Neighbors* (k-NN), onde k é um número inteiro que representa os vizinhos de um exemplo, sendo usado para determinar a classe de uma determinada amostra.

A Figura 5 mostra um exemplo de um classificador k-NN utilizando três vizinhos.

Os vizinhos da amostra são selecionados a partir da distância d:

$$d(q, x_i) = \sum_{f \in F} w_f \sigma(q_f, x_{if}) \tag{2.2}$$

Figura 5 – Exemplo de um classificador k-NN com três vizinhos.



Fonte elaborada pela autora.

onde F é o conjunto de características de um conjunto de dados D com X amostras de treinamento. Para classificar uma amostra q, a distância entre q e cada x_i é calculada, usando um determinado peso w. A forma mais básica da função σ, que faz parte do cálculo da distância, é definida por (CUNNINGHAM; DELANY, 2020):

$$\sigma(q_f, x_{if}) = \begin{cases} 0, \text{com } f \text{ discreto e } q_f = x_{if}; \\ 1, \text{com } f \text{ discreto e } q_f \neq x_{if}; \\ |q_f - x_{if}| \text{com } f \text{ continuo} \end{cases} \tag{2.3}$$

Desta forma, fica claro que $q1$ pertence a classe triângulo. A amostra $q2$ possui tanto vizinhos do tipo círculo quanto vizinhos do tipo triângulo, mas a regra do voto majoritário faz com que seja classificada como círculo.

2.4.2 *Naive Bayes*

O classificador *Naive Bayes* é um método de aprendizado Bayesiano com suposições de independência de características. Assim, um classificador Bayesiano deve assumir que a presença, ou ausência, de uma certa caraterística não depende da presença ou ausência de outras características. A classificação acontece de forma probabilística e independente de outros atributos (KANTARCIOGLU; VAIDYA; CLIFTON, 2003).

A seguinte equação representa o Teorema de Bayes:

$$P(c \mid x) = \frac{P(x \mid c)P(c)}{P(x)} \tag{2.4}$$

onde $P(c \mid x)$ representa a probabilidade de uma classe c acontecer dado um atributo preditor x; $P(x \mid c)$ representa a probabilidade de um preditor x dada uma certa classe c; por fim, $P(c)$ e $P(x)$, que representam a probabilidade da classe c e do preditor x, respectivamente.

2.4.3 Árvore de Decisão

Uma árvore de decisão é composta por nós folhas e nós de decisão. Cada nó de decisão realiza um teste sobre um único atributo da amostra, e possui um certo número de filhos, onde cada um lida com o resultado do teste feito pelo nó de decisão pai. Cada nó folha representa a classe resultado da decisão (STEIN et al., 2005).

A construção da árvore em si depende de diferentes medidas de pureza, que determinam como uma árvore é dividida. Para um problema de classificação, dado um nó m, que representa uma região R_m do conjunto de dados, com N_m observações, a equação 2.5 mede a proporção de k classes observadas no nó m (HASTIE; TIBSHIRANI; FRIEDMAN, 2009):

$$p_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k) \tag{2.5}$$

As observações no nó m são classificadas na classe $k(m) = \arg max_k p_{mk}$, que é a classe majoritária no nó m. Desta forma temos três diferentes medidas de pureza que guiam a divisão de árvores de maneira ótima. Será adotada a notação $i(m)$ para representar impureza.

- **Erro de classificação errônea:** esta medida quantifica a probabilidade mínima de uma padrão ser classificado de forma errada em m (DUDA; HART; STORK, 1973).

$$i(m) = \frac{1}{N_m} \sum_{i \in R_m} I(y_i \neq k) = 1 - p_{mk(m)} \tag{2.6}$$

- **Índice de Gini:** o índice de Gini é taxa de erro esperada no nó m, se o rótulo é selecionado de forma aleatória a partir da distribuição de classes K presente em m (DUDA; HART; STORK, 1973).

$$i(m) = \sum_{k \neq' k} p_{mk}\, p_{mk'} = \sum_{k=1}^{K} p_{mk}\,(1 - p_{mk}) \tag{2.7}$$

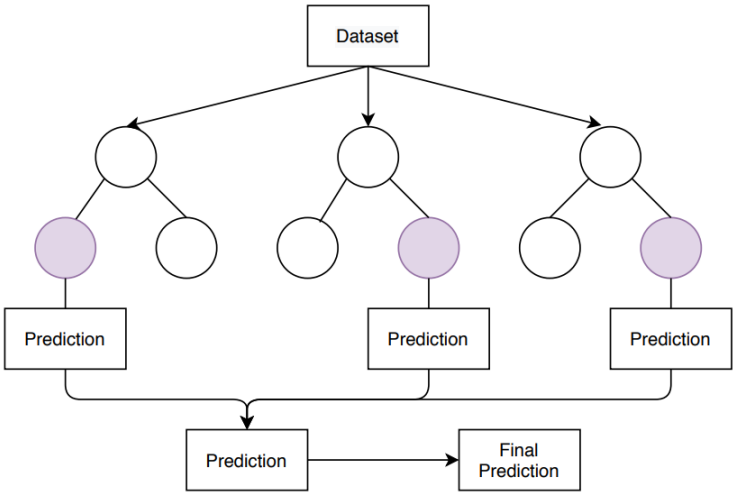
- **Entropia:** se todos os padrões foram da mesma categoria, a impureza é igual à 0. Caso contrário, é positiva, com o maior valor ocorrendo quando as classes são igualmente prováveis (DUDA; HART; STORK, 1973).

$$i(m) = -\sum_{k=1}^{K} p_{mk}\, log(p_{mk}) \tag{2.8}$$

2.4.3.1 *Random Forest*

O classificador *Random Forest* usa uma combinação de árvores de decisão, onde cada classificador é gerado usando um vetor aleatório amostrado de forma independente do vetor de entrada. A floresta então faz uma decisão baseada no voto majoritário dado por cada árvore (PAL, 2005). Uma representação gráfica deste processo pode ser visto na Figura 6.

Figura 6 – Uma representação de *Random Forests*, onde cada árvore individual faz uma predição, e então, o voto majoritário é aplicado.



Fonte: elaborada pela autora.

2.4.4 *Logistic Regression*

Logistic Regression ou Regressão Logística é uma extensão do modelo clássico de regressão linear, sendo mais apropriada para tarefas de regressão.

Em sua forma mais simples, a Regressão Logística modela probabilidades de classificação com dois possíveis resultados (MOLNAR, 2019). Apesar de ser classicamente empregada em tarefas de classificação binária, com alguns ajustes, é possível usá-la para problemas com várias classes.

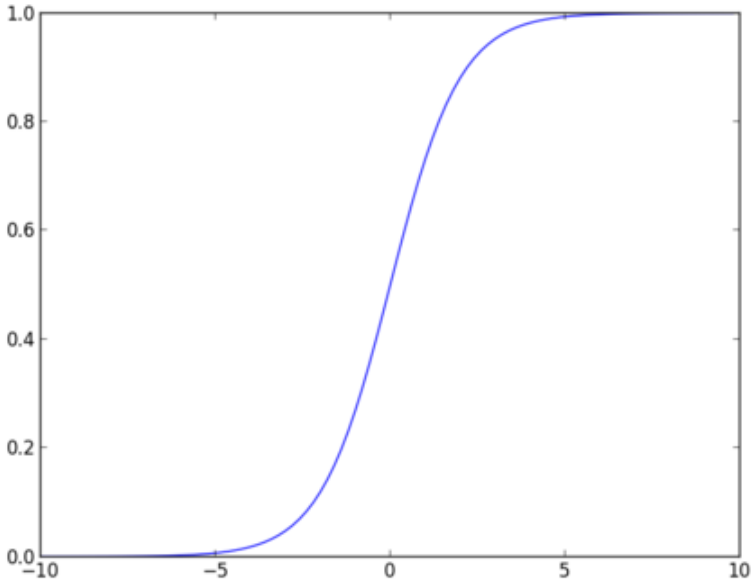
O objetivo da Regressão Linear é de ajustar uma linha ou um hiperplano a um conjunto de dados, de modo a conseguir a melhor aproximação possível entre os pontos do conjunto de dados e o próprio hiperplano. De maneira similar, a Regressão Logística tenta separar os pontos de um conjunto de dados usando a função logística:

$$logistic(\nu) = \frac{1}{1 + exp(-\nu)} \tag{2.9}$$

onde ν é uma equação linear. Assim, ao invés de somente modelar a relação entre o resultado e as características usando uma equação linear, como a Regressão Linear faz, a Regressão Logística 'empacota' uma equação linear dentro de uma função logística. Isso faz com que apenas valores entre 0 e 1 sejam permitidos como resultado, gerando probabilidades.

A função logística também é conhecida como função sigmóide, e pode ser representada graficamente de acordo com a Figura 7.

Figura 7 – Representação gráfica da função logística ou sigmóide



Fonte: elaborada pela autora.

2.4.5 *CatBoost*

O *CatBoost* é uma biblioteca *open source* desenvolvida pela empresa russa Yandex [1]. Seu nome vem da combinação das palavras *Category* e *Boosting*.

Boosting é um método de aprendizado por *ensemble*. O aprendizado por *ensemble* tem por objetivo treinar vários classificadores base, combinando suas predições em uma única saída, fazendo com que a performance final seja maior.

Além do método de *boosting*, também existe o método de *bagging*.

- ***Bagging*:** é o método de *ensemble* mais simples. O processo de *bagging* gera vários membros do *ensemble* construídos com réplicas do conjunto de treinamento. Todos os membros são então combinados, formando um classificador final que é um agregado de todos os membros. Para a saída final, é aplicado o voto majoritário. Para melhorar a performance desse método, é possível simplesmente construir membros do *ensemble* com amostras do conjunto de treinamento, aumentando a diversidade entre cada membro. Desta forma, é possível reduzir o viés e a variância nos resultados de classificação (YANG, 2016).

  O algoritmo de *RandomForest* é um exemplo de aprendizado de *ensemble* que usa o método de *bagging*.

- ***Boosting*:** o método de *boosting* foi criado a partir de um outro algoritmo, o *Adaboost*. Este método constrói de forma sequencial uma combinação linear de classificadores base,

[1] https://catboost.ai/

se concentrando em exemplos difíceis. Ou seja, os pesos dos classificadores são atualizados quando um erro é cometido por parte do classificador anterior (YANG, 2016).

Existem duas formas de combinar os membros individuais do *ensemble*: de forma linear e através do voto majoritário.

- **Combinação linear:** os combinadores lineares são mais utilizados em tarefas de classificação ou regressão supervisionadas, já que as saídas de cada membro são números reais (incluindo probabilidades) de cada rótulo vindo da entrada. Assim, a combinação de todos os membros tem a seguinte estimativa (YANG, 2016):

$$p(x|y) = \sum_{t=1}^{T} w_t p_t(y|x) \tag{2.10}$$

  onde $p_t(y|x)$ é a probabilidade de estimar a classe y, dada a entrada x, feita por um membro t. O peso w_t determina a performance baseada no conjunto de treinamento.

- **Voto Majoritário:** se os membros classificadores base estimam os rótulos diretamente, a opção de voto majoritário pode ser feita. Desta forma os votos de cada membro são contados, o que pode ser visto na equação a seguir (YANG, 2016):

$$H^i = \arg\, max_j \sum_{t=1}^{T} w_t H_t^{i,j}, \text{ onde } H_t^{i,j} = \begin{cases} 1 \text{ se } i \text{ é classificado como } j \\ 0 \text{ caso contrário} \end{cases} \tag{2.11}$$

O algoritmo *Catboost* é construído a partir do algoritmo de *boosting* ou aperfeiçoamento de gradiente. No *Adaboost*, os pesos eram atualizados quando havia erro, e todo o pacote de informação era passado para o próximo membro do *ensemble*. No *boosting* de gradiente, apenas o erro residual é enviado para o próximo membro. O erro residual é a chamada função de perda, e pode ser considerada, por exemplo, como a diferença entre o valor real e o valor predito, nos casos mais simples. Assim, usando este método, ao passar para o próximo membro do *ensemble*, o método de gradiente descendente é feito para minimizar esse erro residual.

Normalmente, as implementações de *boosting* de gradiente utilizam árvores de decisão, assim como outros métodos de *ensemble*. Árvores de decisão são muito úteis quando utilizadas em aplicações numéricas, e para usá-las em conjuntos de dados categóricos, é necessário convertê-los para números antes do treinamento. Já o *Catboost* é uma implementação de *boosting* de gradiente que permite o uso de dados categóricos de forma direta, fazendo a conversão dos rótulos em tempo de execução (DOROGUSH; ERSHOV; GULIN, 2018). Além disso, diferentes permutações do conjunto de dados são utilizadas, e cada permutação é usada para treinar um modelo diferente. Segundo os autores, isso diminui as chances de *overfitting*. Assim, o *Catboost* é capaz de selecionar o melhor modelo y, baseado em um conjunto de características x_i, que melhor resolva o problema para qualquer tipo de entrada (DOROGUSH; ERSHOV; GULIN, 2018).

Esta biblioteca foi escolhida em particular, por questões de performance e robustez. Além disso, é notória por reduzir a necessidade de afinação de hiper parâmetros, e também reduzir as chances de *overfitting*, como exemplificado anteriormente.

2.4.6 *Support Vector Machine* (SVM)

O termo *Support Vector Machine* (SVM) ou máquina de vetores de suporte é tipicamente usado para descrever classificação usando métodos de vetores de suporte. Os princípios básicos de SVM estão embasados na teoria de aprendizado estatístico.

2.4.6.1 Teoria de Aprendizado Estatístico

Segundo LORENA; CARVALHO, a Teoria de Aprendizado Estatístico "estabelece condições matemáticas que auxiliam na escolha de um classificador a partir de um conjunto de dados de treinamento". Ao se modelar um problema, o objetivo é escolher um modelo, a partir do espaço de hipóteses, que é o mais próximo possível da função subjacente no espaço alvo. Estar 'o mais próximo possível' depende da medida de erro utilizada podendo partir de dois casos (GUNN et al., 1998):

- **Erro de Aproximação:** acontece quando o espaço de hipóteses é menor do que o espaço alvo, fazendo com que a função subjacente fique fora do espaço de hipóteses.
- **Erro de Estimação:** um erro durante o processo de aprendizado, levando à escolha de um modelo não ótimo.

Estes dois tipos de erros, quando combinados, formam o erro de generalização. O objetivo então é encontrar uma função f (o classificador) que minimiza o erro (que também é chamado de risco):

$$R[f] = \int_{X*Y} L(y, f(x))P(x, y)dxdy \tag{2.12}$$

onde $P(x, y)$ descreve a relação entre os dados de um conjunto de dados e seus rótulos. $L(y, f(x))$ é uma função de custo relacionada a previsão $f(x)$ quando a saída desejada é y. Como $P(x, y)$ é desconhecido, é possível encontrar uma aproximação, através do princípio de minimização de risco empírico:

$$R_{emp}[f] = \frac{1}{l}\sum_{i=1}^{l} L(y^i, f(x^i)) \tag{2.13}$$

Assim, ao minimizar o risco empírico, é possível obter a função f:

$$f = argmin R_{emp}[f] \tag{2.14}$$

2.4.7 SVM Lineares

Considerando a Teoria de Aprendizado Estatístico vista na subseção anterior, é possível dizer que o objetivo do algoritmo de máquina de vetores de suporte, para o problema de classificação, é encontrar um função que minimize o erro de separação. Ou seja, encontrar um hiperplano, em um espaço n-dimensional, sendo n o número de características, que separe os pontos de um conjunto de dados de forma ótima (GUNN et al., 1998).

É possível caracterizar hiperplanos como margens que ajudam a classificar pontos em um conjunto de dados. A dimensão de um hiperplano depende do número de características presentes em um conjuntos de dados. Já os chamados vetores de suporte são pontos do conjunto de dados que estão mais próximos do hiperplano, e influenciam a posição e orientação do mesmo.

Para obter o hiperplano ótimo, é necessário maximizar a margem entre os pontos do conjunto de dados e o hiperplano em si. A função que minimiza esta margem é a função de perda de articulação:

$$c(x, y, f(x)) = \begin{cases} 0, & \text{se } y * (fx) \geq 1. \\ 1 - y * f(x), & \text{caso contrário.} \end{cases} \tag{2.15}$$

$f(x)$ corresponde à $f(x) = w \cdot x + b = 0$, que é a equação de um hiperplano e w é o vetor normal ao hiperplano (LORENA; CARVALHO, 2007).

Nesta função, o custo é igual a zero se o valor predito e o valor verdadeiro obtiverem a mesma polaridade. Caso contrário a perda é calculada.

Para calcular a função de custo de forma correta, é necessário adicionar um parâmetro α para regularização. Este parâmetro tem como objetivo balancear a margem de maximização e a perda. A maximização de margem pode ser obtida através da minimização de $\|w\|$ (LORENA; CARVALHO, 2007). A função de custo, terá então o formato da Equação 2.16:

$$c(x, y, f(x)) = min_w \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} max\ \alpha_i(y_i\langle x_i, w\rangle - 1) \tag{2.16}$$

Assim, resolvendo-se a equação de forma a encontrar valores que minimizem w e b (w^* e b^*) e maximizem α (a^*), é possível obter o classificador final $g(x)$ (LORENA; CARVALHO, 2007):

$$g(x) = sgn(f(x)) = sgn(\sum_{w_i \in SV} y_i\alpha_i^* x_i \cdot x + b*) \tag{2.17}$$

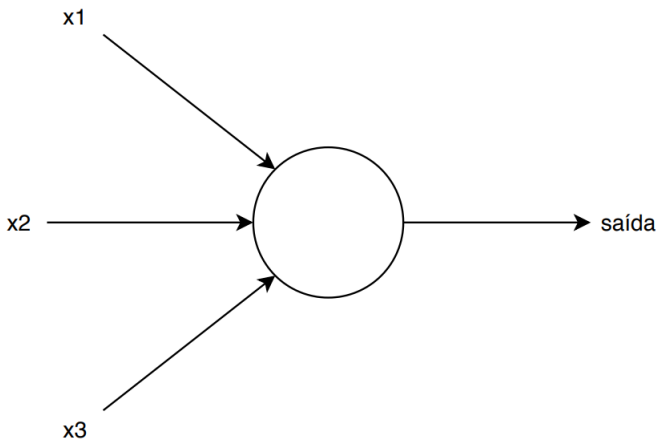
Os cálculos para a solução da Equação 2.16 fogem do escopo do trabalho, e portanto não são representados.

2.4.8 *Multi-Layer Perceptron* (MLP)

Também conhecido como rede de perceptrons multicamadas, é uma evolução do modelo de perceptron. Um perceptron pode ser entendido como um algoritmo de classificação binária, proposto por Frank Rosenblatt entre os anos 1950 e 1960, a partir de trabalhos anteriores por Warren McCulloch e Walter Pitts (NIELSEN, 2015). O perceptron é um algoritmo bio-inspirado, sendo modelado a partir do neurônio humano.

Um perceptron, como pode ser visto na Figura 8, pode possuir várias entradas $x_1, x_2, x_3, ...$, e produz uma única saída binária (NIELSEN, 2015).

Figura 8 – Representação de um neurônio (perceptron), com suas entradas e saída binária.



Fonte: elaborada pela autora.

De acordo com o modelo de Rosenblatt, a saída de um perceptron é calculada através de pesos. Esses pesos são atribuídos às entradas, e denotam a importância de cada uma delas. Dessa forma, a saída do perceptron é determinada pela soma ponderada das entradas associadas com seus respectivos pesos, como na Equação 2.18

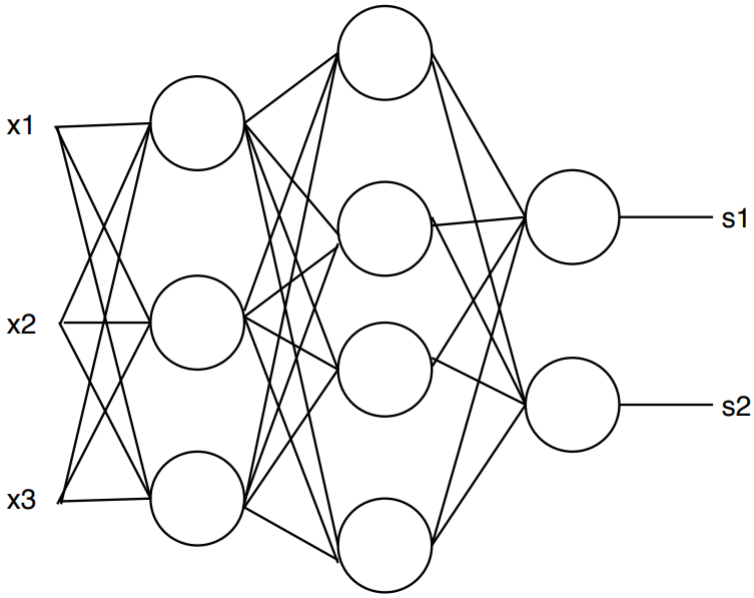
$$saida = \begin{cases} 0 \text{ se } w \cdot x + b \leq 0 \\ 1 \text{ se } w \cdot x + b > 0 \end{cases} \tag{2.18}$$

onde b é o viés para o perceptron. Assim, se a soma ponderada for menor ou igual à 0, a saída do perceptron será 0, caso contrário, a saída será 1 (NIELSEN, 2015).

Já uma rede de perceptrons multicamadas possui vários perceptrons, e é muito mais eficiente que um modelo de perceptron comum para resolver problemas mais complexos.

Diferentemente do perceptron, que é capaz de aproximar funções lineares, uma rede MLP é capaz de aproximar funções não-lineares com a mesma facilidade (GARDNER; DORLING, 1998). A Figura 9 representa uma rede MLP, com três entradas (x_1, x_2 e x_3) e duas saídas (s_1 e s_2), com duas camadas escondidas.

Figura 9 – Representação de uma rede MLP composta por vários perceptrons.



Fonte: elaborada pela autora.

A soma dos nós a cada camada é modificada por uma função de ativação (também chamada de função transferência) não-linear. É esta superposição de várias funções de ativação não-lineares que permite que uma rede MLP aproxime funções não-lineares. Uma função de ativação bastante comum é a função sigmóide, devido à sua derivada ser facilmente calculável (GARDNER; DORLING, 1998).

A arquitetura de uma rede MLP é customizável, podendo ter um número variável de nós de entrada, camada escondida e saída. A própria quantidade de camadas escondidas também é variável. A informação passa de uma camada à outra, implicando uma direção para a rede (GARDNER; DORLING, 1998).

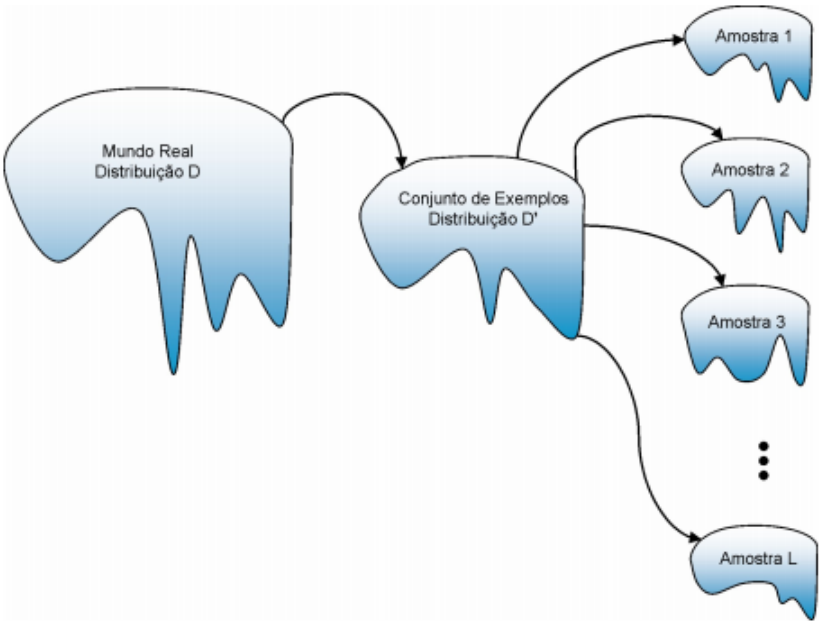
Uma rede MLP aprende através de treinamento, que necessita de exemplos rotulados. Durante o treinamento, os exemplos de um conjuntos de dados são mostrados repetidamente à rede, e com isso, os pesos associados a cada nó são atualizados, até que a combinação correta de entrada e saída ocorra. Os pesos são atualizados levando em consideração um erro, que é a diferença entre o resultado esperado e o resultado obtido (GARDNER; DORLING, 1998).

2.5 Amostragem de Algoritmos

A partição entre conjuntos de treino e conjuntos de teste é importante para que o resultado da classificação possa ser avaliado de forma justa. Diversas técnicas de partição podem ser utilizadas para este fim, todas baseadas na ideia de amostragem (REZENDE, 2003).

Como pode ser visto na Figura 10, de um conjunto D de dados do mundo real podem ser extraídos exemplos D'. Assume-se que a distribuição D' seja similar à distribuição D. Para comparar o desempenho de algoritmos de indução, são retiradas amostras dessa distribuição D', e cada indutor é treinado com uma amostra diferente, e sua performance é medida. Este processo busca simular o processo de amostragem que ocorre no mundo real. As métricas de performance de algoritmos serão tratadas com mais detalhes na seção seguinte.

Figura 10 – Estimativas baseadas em amostragem.



Fonte: (REZENDE, 2003)

A seguir, são listadas as diferentes técnicas de amostragem não paramétricas:

- *Holdout*: nesta técnica de estimação, o *holdout* divide exemplos em uma porcentagem fixa de exemplos p para treinamento e $1 - p$ para teste (REZENDE, 2003). Apesar de ser uma técnica bastante comum, não há fundamentos teóricos sobre os valores exatos de p e $1 - p$, ficando à cargo da aplicação.

- Amostragem Aleatória: em uma amostragem aleatória, são induzidas L hipóteses sobre os conjuntos de treinamento, sendo L muito menor que a quantidade de exemplos n. O erro final é a média de erros de todas as hipóteses induzidas. Os conjuntos de testes são independentes e extraídos de forma aleatória (REZENDE, 2003). Esta técnica produz estimativas de erros melhores do que a técnica de *holdout*.

- *Cross-validation*: o *cross-validation* ou validação cruzada é uma evolução da técnica de *holdout*. O termo *k-fold cross-validation* ou CV, se refere ao ato de se dividir o conjunto de exemplos em k partições ou *folds* mutualmente exclusivas, com tamanho aproximadamente igual a n/k, onde n é o número de exemplos. O treinamento é feito nos exemplos das $k - 1$ partições, e testado na partição remanescente. Este processo é repetido por k

vezes, onde cada vez é considerado um *fold* diferente para teste (REZENDE, 2003). O erro desta técnica de amostragem é a média de todos os erros.

Uma variação do estimador *cross-validation* é o *stratified cross-validation* ou validação cruzada estratificada. A diferença está na abordagem de geração de partições mutualmente exclusivas. Na validação cruzada estratificada, a distribuição das classes é observada (REZENDE, 2003). Ou seja, se um *fold* possui uma distribuição de classes de 70% e 30%, todos os outros *folds* seguirão a mesma proporção.

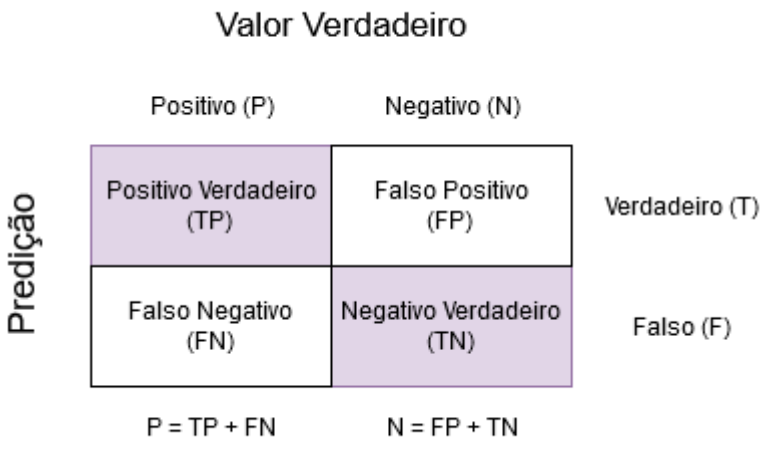
- *Leave-one-out*: este estimador é um caso especial de *cross-validation*. Em um conjunto de dados de n exemplos, o treinamento acontece em $n - 1$ exemplos e é testado no exemplo remanescente. Este processo se repete n vezes, onde cada vez o treinamento deixa de considerar um exemplo (REZENDE, 2003). O erro final é calculado pela soma dos erros divididos por n.

2.6 Métricas de Avaliação de Algoritmos

A métrica de avaliação de um algoritmo é um fator decisivo para avaliar a performance de um classificador, após a etapa de treinamento.

Assumindo uma tarefa de classificação binária, como é o caso deste trabalho, é possível elencar duas classes diferentes: P para a classe positiva e N para a classe negativa. Sendo completada a etapa de treinamento, o classificador induzido é idealmente testado contra dados ainda não vistos. As possíveis comparações entre o resultado predito pelo algoritmo e o resultado verdadeiro podem ser resumidas em uma matriz de confusão (THARWAT, 2018), como na Figura 11.

Figura 11 – Um exemplo de uma matriz de confusão para um problema de classificação binária



Fonte: elaborada pela autora, baseado em (THARWAT, 2018)

A diagonal de cor lilás representa predições corretas, enquanto a diagonal de cor branca representa predições incorretas.

Desta forma, é possível listar quatro possíveis avaliações (THARWAT, 2018):

- **Positivo Verdadeiro:** ou *True Postive* (TP), mostra as classificações corretamente classificadas como positivas.
- **Negativo Verdadeiro:** ou *True Negative* (TN), mostra as classificações corretamente classificadas como negativas.
- **Falso Negativo:** ou *False Negative* (FN), mostra as classificações erroneamente classificadas como negativas, sendo na verdade positivas.
- **Falso Positivo:** ou *False Positive* (FP), mostra as classificações erroneamente classificadas como positivas, sendo na verdade negativas.

A matriz de confusão que contabiliza falsos positivo e negativos, além de positivos e negativos verdadeiros é a base para o cálculo das métricas de performance que serão usadas neste trabalho, detalhadas a seguir (THARWAT, 2018).

- **Erro e Acurácia**: a acurácia (Acc) é a razão entre amostras classificadas de forma correta pelo número total de amostras, como é mostrado na Equação 2.19.

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.19}$$

  O complemento natural da métrica de acurácia é a métrica de erro (ERR), representando o número de amostras classificadas de forma errada, considerando tanto amostras de classes positivas e negativas. A equação é mostrada a seguir:

$$ERR = 1 - Acc = \frac{FP + FN}{TP + TN + FP + FN} \tag{2.20}$$

  É importante frisar que ambas as métricas são sensíveis a dados desbalanceados.

- **Precisão**: ou *Positive Prediction Value* (PPV) é a métrica que representa a proporção entre as amostras positivas que foram corretamente classificadas como positivas, e o número total de amostras classificadas como positivas, como é mostrado na Equação 2.21.

$$PPV = \frac{TP}{FP + TP} \tag{2.21}$$

  A precisão pode ser medida com relação à classe negativa, tendo o seguinte formato:

$$NPV = \frac{TN}{FN + TN} \tag{2.22}$$

  onde NPV significa *Negative Prediction Value*.

- ***Recall***: o *recall* é uma métrica que mede a taxa de positivos verdadeiros. Ou seja, representa a razão entre amostras positivas corretamente classificadas como positivas e número total de amostras positivas, como pode ser visto na Equação 2.23.

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{P} \tag{2.23}$$

  O *recall* também pode ser medido de acordo com as classes negativas, sendo chamado de *recall* invertido (*True Negative Rate, ou TNR*).

$$TNR = \frac{TN}{FP + TN} = \frac{TN}{N} \tag{2.24}$$

- ***F1-score***: também conhecido como medida-F, representa a média harmônica entre a precisão e o *recall*, como pode ser visto na Equação 2.25.

$$F1 = \frac{2PPV * TPR}{PPV + TPR} = \frac{2TP}{2TP + FP + FN} \tag{2.25}$$

  Os valores da medida-F variam entre 0 e 1, e quanto mais próxima de 1, melhor é a performance de classificação.

3 Trabalhos Relacionados

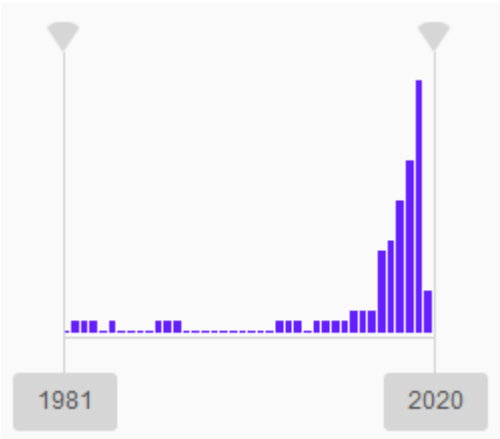
Trabalhos relacionados a discurso de ódio são prevalentes nas ciências sociais (WALKER, 1994) (WALDRON, 2014), porém, ainda é um tópico recente no campo da computação, e a quantidade de pesquisas ainda é baixa. Contudo, nota-se que por ser um assunto de extrema importância, tanto social, como econômica, esforços estão sendo feitos para o aperfeiçoamento das técnicas de detecção automática de discurso de ódio.

Inicialmente, utilizamos uma *string* de pesquisa generalista na biblioteca digital ACM[1] apresentada a seguir:

(("Hate Speech") OR ("Sexism") OR ("Racism")) AND (automatic OR detection)

É possível perceber o aumento exponencial no número de pesquisas feitas nos últimos anos (Figura 12).

Figura 12 – Distribuição da quantidade de artigos publicados através dos anos na biblioteca digital ACM



Fonte: ACM [2]

3.1 Principais trabalhos encontrados

Apenas duas revisões sistemáticas foram encontradas, sendo usadas como ponto de partida para guiar os desafios e necessidades do tema. A primeira delas, intitulada "*Survey on Hate Speech Detection Using Natural Language Processing*" (SCHMIDT; WIEGAND, 2017), apresenta uma pesquisa estruturada, e separa os artigos reunidos em tópicos como análise de características (*features*) e métodos de classificação.

A segunda revisão sistemática, intitulada "*A Survey on Automatic Detection of Hate Speech in Text*" (FORTUNA; NUNES, 2018), além de analisar uma gama maior de trabalhos,

[1] https://dl.acm.org/

foca não somente em pesquisas que apresentam algoritmos, mas também naquelas que focam na estatística descritiva na detecção de discurso de ódio. Além disso, esta revisão lista diversos conjuntos de dados disponíveis para o problema.

Além das revisões sistemáticas, alguns trabalhos foram analisados com maior rigor. Os autores MACAVANEY et al. utilizam uma técnica de máquina de vetores de suporte combinadas (*Multi-view* SVM) e TF-IDF para realizar a tarefa de classificação. Vários conjuntos de dados foram usados e uma comparação com resultados de outros trabalhos também foi feita.

Em "*Identification of Hate Speech*" in Social Media, (RUWANDIKA; WEERASINGHE, 2018), um conjunto de dados foi construído a partir de comentários em um fórum de notícias. A partir dele, os autores utilizaram diversas técnicas diferentes de extração de características e representação de texto, assim como algumas técnicas de classificação automática. O objetivo deste trabalho era comparar a performance de um algoritmo de classificação não supervisionado (*K-means*) com outros algoritmos de classificação supervisionada.

Em (SAHA; DUBEY; BHATTACHARYYA, 2019), foram utilizadas SVMs e redes neurais convolucionais, a fim de comparar a performance de cada um dos algoritmos. Diferentes características também foram usadas para cada teste, como uso de *hashtags*, *emoticons* e *n-grams*.

As revisões sistemáticas e trabalhos descritos acima tinham como foco a detecção de ódio usando textos em inglês. Também foi possível encontrar trabalhos em outras línguas, apesar do número reduzido. No entanto, o uso de textos na língua portuguesa foi encontrado em apenas três trabalhos.

Em (FORTUNA et al., 2019), foi apresentado um novo conjunto de dados para discurso de ódio composto por *tweets* em português. Este também é o conjunto de dados que será utilizado neste trabalho. Ele será visto com mais detalhes no Capítulo 4. O conjunto de dados apresentado possui rótulos para classificação binária e multi classe. Além de introduzir um novo conjunto de dados, os autores também fizeram um breve experimento de classificação. Usando *GloVe embeddings* e redes neurais *Long Short Term Memory*, combinadas com a técnica de *boosting xgBoost* atingiram um *F1-score* de 0,78.

NASCIMENTO et al. também propôs uma solução para detecção de discurso de ódio na língua portuguesa, usando os chamados *imageboards*. *Imageboards* são fóruns de discussão anônimos, baseados na postagem de imagens e texto. São também conhecidos como *chans* (FONTANELLA, 2010). Um dos *chans* mais conhecidos atualmente é o *4chan*, no entanto, existem fóruns brasileiros deste tipo. O fórum usado como fonte dos dados para o trabalho de NASCIMENTO et al. é o *55chan*. A anonimidade dos *imageboards* faz com que sejam conhecidos por serem um ambiente tóxico (FONTANELLA, 2010). Os autores do trabalho utilizaram *TF-IDF* para a preparação de características, e compararam a performance de classificação entre os algoritmos *Naive-Bayes* Multinomial, *Random Forest* e SVM. Com o classificador SVM,

o trabalho atingiu um *F1-score* de 0,955.

Os autores de (UNSVÅG; GAMBÄCK, 2018) usaram três conjuntos de dados diferentes, um para língua inglesa, um para a língua portuguesa e um para língua alemã. O conjunto de dados de língua portuguesa é o mesmo criado por FORTUNA et al.. O objetivo do trabalho foi extrair características, como gênero e customização de perfil, do conjunto de dados, para impulsionar a tarefa de classificação. Além da extração de características, foram utilizados a técnica de *TF-IDF* e o algoritmo de Regressão Logística. Com essa combinação, foi possível atingir um *F1-score* de 0,79 para o conjunto de dados em Português. Para esse conjunto de dados, o processo de extração de características não surtiu impacto sobre o resultado final.

3.2 Discussão

Os trabalhos reunidos até aqui mostram uma certa escassez de fontes que têm como tema a detecção de discurso de ódio. A grande maioria das pesquisas realizadas até o momento foram feitas na língua inglesa. No entanto, um classificador treinado para uma língua dificilmente servirá com a precisão adequada para outras línguas.

Além disso, a maior parte das pesquisas mostram resultados de classificação binária, dividindo textos entre portadores e não portadores de discurso de ódio. Porém, os ataques às minorias possuem muitas facetas, e detalhar essa classificação pode ser de grande valor.

A maior parte dos pesquisadores se concentrou em iniciativas que usavam técnicas supervisionadas de aprendizado de máquina. Algoritmos de aprendizado não-supervisionado, semissupervisionado e de *deep learning* foram pouco explorados.

4 Materiais e Métodos

Neste capítulo são apresentados o conjunto de dados textual de discurso de ódio a ser explorado na Seção 4.1 e as técnicas e bibliotecas referentes a Processamento de Linguagem Natural e Aprendizado de Máquina utilizadas na Seções 4.2 e 4.3, respectivamente.

4.1 *Conjunto de dados*

Os trabalhos relacionados no Capítulo 3 indicam a clara predominância de pesquisas na língua inglesa no que tange a detecção automática de discurso de ódio. No entanto, é de interesse social e econômico que ferramentas que possam detectar ataques automaticamente em todas as línguas sejam criadas e aperfeiçoadas. Outro fator importante é a simplicidade na detecção: a maioria dos trabalhos apresentam resultados de classificação binária ou com poucas classes, apesar da existência de muitos subtipos de discurso de ódio diferentes.

A partir desta necessidade, um conjunto de dados na língua portuguesa foi selecionado. O conjunto de dados utilizado neste trabalho, compilado por (FORTUNA et al., 2019), possui características que o tornam competitivo com conjuntos semelhantes e já consolidados, porém, em língua inglesa.

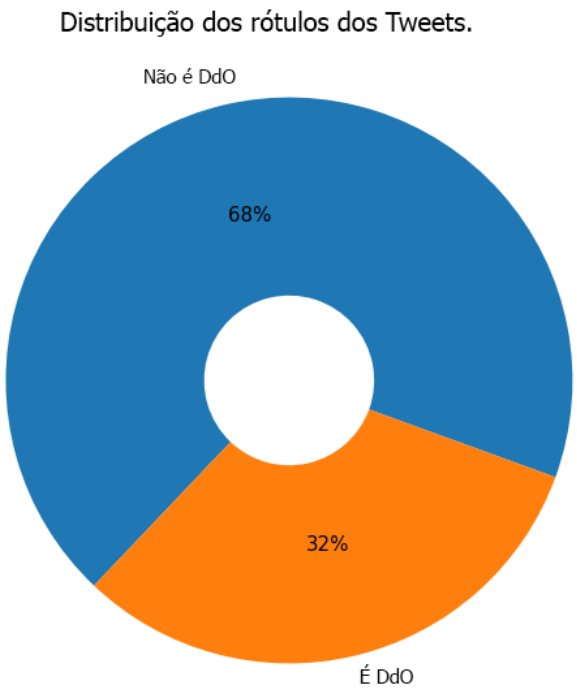
Os autores utilizaram a rede social Twitter como fonte de dados, e dois métodos de agregação de dados foram empregados. Primeiramente, com o auxílio da *API* do Twitter, foram realizadas buscas por palavras-chave e *hashtags* consideradas ofensivas, como *"sapatão"* e *#LugarDeMulherENaCozinha*. Além disso, também foi feita uma busca por perfis que costumam espalhar discurso de ódio e possuem comportamento ofensivo. O objetivo dos autores com esse processo de coleta foi o de reunir diferentes tipos de discriminação, seja ela baseada em gênero, religião, orientação sexual, etnia e status migratório.

É importante ressaltar que até o momento não existe um léxico de palavras de ódio disponível para a língua portuguesa.

O conjunto de dados final, após a filtragem e amostragem de *tweets*, é composto por 5668 mensagens, de 1156 usuários diferentes. O processo para a rotulação das mensagens foi feito em duas etapas:

1. As mensagens foram rotuladas de forma binária (“discurso de ódio” e “não contém discurso de ódio”), utilizando 18 rotuladores humanos sem experiência de área, e validando a concordância entre eles usando o método de Fleiss Kappa (FLEISS, 1971). Neste *datastet* binário, 32% das mensagens foram consideradas como discurso de ódio (Figura 13).

Figura 13 – Distribuição dos *tweets* classificados como discurso de ódio *hate speech* (IsHS) e como não sendo discurso de ódio (not HS).



Fonte : elaborada pela autora.

2. As mensagens foram rotuladas de forma hierárquica, contando com a experiência de dois rotuladores: um pesquisador na área de detecção automática de discurso de ódio e outro com treinamento em psicologia social. Os mesmos critérios de concordância entre rotuladores foram utilizados. Ao fim deste processo, um conjunto de dados com 81 classes diferentes de rótulos foi obtido.

As 10 classes mais predominantes do conjunto de dados são apresentadas na Tabela 3.

Tabela 3 – Distribuição de mensagens de discurso de ódio para as 10 classes mais frequentes presentes no conjunto de dados (FORTUNA et al., 2019).

Classe	Frequência
Discurso de ódio	1228
Sexismo	672
Mulheres	544
Homofobia	322
Homossexuais	288
Lésbicas	248
Corpo	164
Pessoas obesas	160
Mulheres obesas	153
Pessoas feias	131

Para este trabalho, apenas o conjunto de dados para classificação binária será utilizado, com o objetivo de comparar resultados com os atuais trabalhos de estado da arte.

4.2 Técnicas Empregadas

4.2.1 Pré-processamento

A etapa de pré-processamento é extremamente importante para a presente tarefa de classificação, para que a busca por padrões relevantes seja simplificada. As etapas de pré-processamento explicitadas no Capítulo 2 foram aplicadas ao conjunto de dados. Primeiramente, foi feita a normalização do texto, onde a conversão dos *tweets* para caixa baixa foi realizada, além da remoção de pontuações.

Nesta etapa, também é necessário levar em consideração o domínio que está sendo trabalhado. Para o caso de um conjunto de dados provenientes da rede social Twitter, também foram removidos *links* para outros *websites*. Além disso, no Twitter, o uso de *hashtags* é prevalente, e as palavras usadas junto ao símbolo (#) podem ser relevantes para a tarefa de detecção de discurso de ódio. Portanto, para estes casos, o símbolo foi retirado. Menções a usuários são feitas através da simbologia @. Porém, manter o nome de usuário não traz benefícios à tarefa de classificação. Logo, para estes casos, tanto o símbolo como o nome de usuário que o segue, foram removidos do conjunto de dados.

A tokenização também foi realizada para todo o conjunto de dados. A remoção de *stopwords* também foi feita de forma exploratória. Desta forma, testes preliminares foram realizados com três diferentes estratégias de utilização de *stopwords*:

- Sem remoção de *stopwords*.
- Remoção de *stopwords* com o auxílio do conjunto padrão disponibilizado pela biblioteca *Sci-kit Learn*[1].
- Remoção de um conjunto de *stopwords* customizado, reunindo as 10 palavras mais comuns entre os dados de entrada.

Esta análise exploratória da remoção de *stopwords* foi realizada para determinar qual alternativa resulta na melhor precisão de classificação.

Além da análise exploratória com relação a *stopwords* também foi feita uma comparação utilizando os diferentes tipos de modelagem de linguagem descritos no Capítulo 2, como BoW e TF-IDF, além de representações com *embeddings*, como Doc2Vec.

[1] https://scikit-learn.org/

4.2.2 Extração de Características

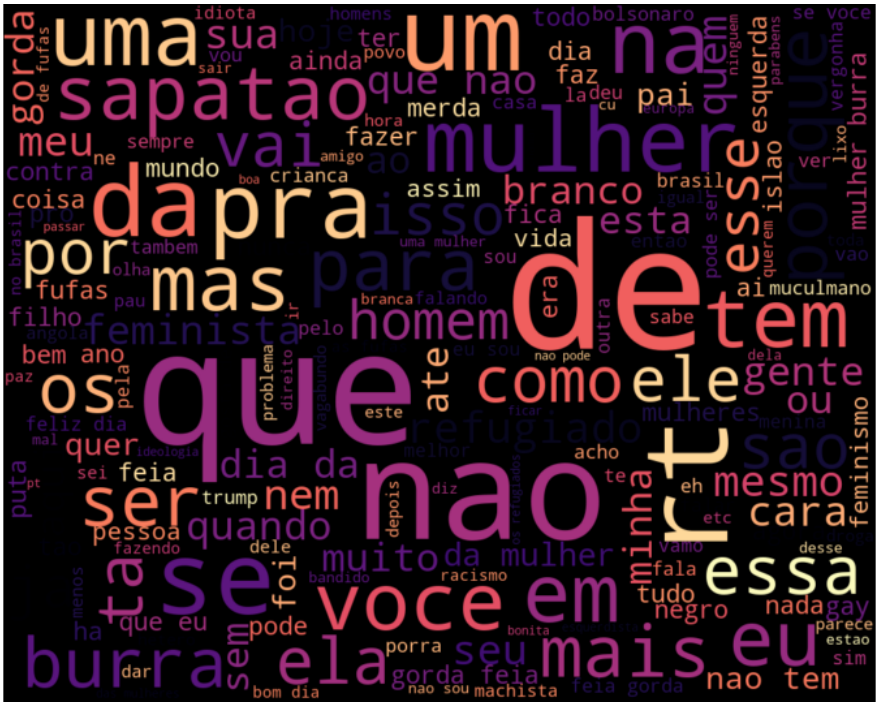
Ainda durante a etapa de pré-processamento, foi realizada uma etapa extra, chamada de *fetaure engineering* ou engenharia de características, já que as técnicas de pré-processamento discutidas no Capítulo 2 podem não ser suficientes para gerar os melhores resultados possíveis.

Alguns dos trabalhos relacionados, vistos no Capítulo 3, apresentaram o uso de características extras para melhorarem o resultado de classificação. As características escolhidas foram particulares de cada trabalho, porém, incluíam meta-informação de *tweets*, como geolocalização ou gênero, ou também características textuais, como tamanho das frases ou uso de *emojis*.

Dada a tarefa de classificação, é trivial perceber que palavras ofensivas têm mais chances de aparecer em *tweets* classificados como discurso de ódio. No entanto, como discutido anteriormente, criar um dicionário de palavras categorizadas como ruins ou ofensivas e usá-las como base para a classificação pode levar a erros.

Sendo assim, a interpretação do conjunto de dados através da visualização de suas características pode ajudar na criação de características extras, melhorando a precisão final. As Figuras 14 e 15 mostram a nuvem de palavras mais utilizada para os *tweets* classificados como sendo discurso de ódio e como não sendo discurso de ódio, respectivamente.

Figura 14 – A nuvem de palavras resultantes para *tweets* classificados como sendo discurso de ódio



Fonte: elaborada pela autora.

Observando as palavras geradas em cada uma das nuvens, é possível notar que há pouca sobreposição de palavras entre as duas classificações. Sendo assim, foram criadas quatro novas características:

Figura 15 – A nuvem de palavras resultantes para *tweets* classificados como não sendo discurso de ódio

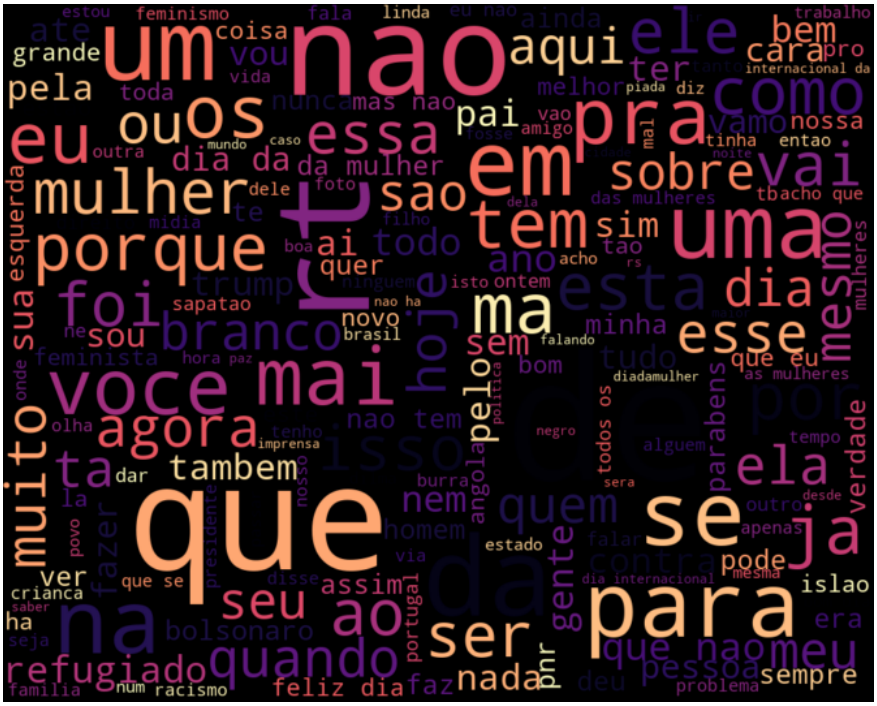


Figura elaborada pela autora.

- *sum_idf_0*: contagem de palavras para a classe 0.
- *sum_idf_1*: contagem de palavras para a classe 1.
- *count_0*: IDF para a classe 0.
- *count_1*: IDF para a classe 1.

Com estas características, foi possível diferenciar um *tweet* de acordo com as classes de forma mais eficaz.

Para fins de exemplo, é mostrado na Tabela 4, os valores resultantes das novas características a partir de dois *tweets* quaisquer, um classificado como discurso de ódio e outro não.

Tabela 4 – Exemplo de valores atribuídos à novas características baseado em dois *tweets* quaisquer

É discurso de ódio (DO)?	**sum_idf_0**	**sum_idf_1**	**count_0**	**count_1**
É DO	125.61	238.39	22	38
Não é DO	188.52	64.02	28	13

Para computar os valores resultantes contidos na coluna *count_0*, foram armazenadas em um dicionário todas as palavras que apareceram em *tweets* classificados como não sendo discurso de ódio, levando em consideração todo o *corpus*. Em seguida, foi feita a contagem das palavras em cada um dos dois *tweets* usados como exemplo. A cada vez que uma palavra do primeiro *tweet* era encontrada no dicionário referente à *count_0*, um contador era incrementado.

De maneira semelhante, o mesmo processo foi feito para a coluna *count_1*. Para as colunas *sum_idf_0* e *sum_idf_1* o procedimento também foi similar, no entanto, uma pontuação maior era atribuída a palavras que aparecem mais frequentemente em *classificados* como discurso de ódio.

Sendo assim é possível notar uma certa diferença no pontuação obtido entres as diferentes classes. *Tweets* que são considerados discurso de ódio possuem mais palavras ofensivas, além de obterem uma maior pontuação no que tange o IDF.

Assim, é possível inferir que o uso de engenharia de características pode vir a melhorar os resultados da tarefa de classificação. Uma análise exploratória foi feita, comparando resultados tanto com, como sem o uso de novas características.

4.2.3 Algoritmos Utilizados

Os algoritmos de aprendizado de máquina eleitos para aplicação sobre o conjunto de dados foram o k-NN, *Naive Bayes*, *Random Forest*, Regressão Logística e *CatBoost*, SVM e Rede de Perceptrons Multicamadas, todos pormenorizados no Capítulo 2.

Para oferecer robustez ao resultado e torná-lo relevante estatisticamente, foi usada a ferramenta de seleção de modelo *GridSearch* da biblioteca Sci-kit Learn. Esta ferramenta permite que os melhores parâmetros de um determinado algoritmo sejam encontrados, além de realizar a validação cruzada com o método *K-fold*, neste caso, com k=5 dobras. O método *K-fold* é equivalente ao método de *Leave-One-Out*. O número de dobras foi reduzido para que fosse possível realizar uma maior quantidade de testes com diferentes tipo de algoritmos de aprendizado de máquina e técnicas de pré-processamento, a fim de obter uma análise horizontal frente a outros trabalhos similares.

As métricas de avaliação impostas sobre os resultados são precisão, *Recall*, *F1-Score* e acurácia, descritos com mais detalhes no Capítulo 2.

4.3 Bibliotecas

Para o desenvolvimento deste trabalho, Python foi a linguagem de programação utilizada para a confecção dos classificadores. Ela foi escolhida pela existência de bibliotecas usadas para computação científica já desenvolvidas para esta linguagem.

Desta forma, as bibliotecas a serem utilizadas neste trabalho são:

- **Scikit-Learn** [2]**:** é uma biblioteca de aprendizado de máquina, que contém diversos algoritmos de classificação, regressão e agrupamento. Possui alta sinergia com outras bibliotecas que serão utilizadas neste trabalho, como a Numpy.

[2] https://scikit-learn.org/

- **Natural Language Toolkit (NLTK)** [3]**:** é um conjunto de bibliotecas capazes de processar, de maneira simbólica e estatística, linguagem natural. A NLTK é capaz de executar funcionalidades de classificação, segmentação, *stemming*, análise e raciocínio semântico.

- **NumPy** [4]**:** é uma biblioteca que permite a construção e manipulação de vetores e matrizes multidimensionais de grandes tamanhos, além de reunir funções matemáticas de alta complexidade para operar nesse vetores.

- **Pandas** [5]**:** é uma biblioteca usada para manipulação e análise de dados. Permite que informações sejam manipuladas usando tabelas e séries temporais.

- **Gensim** [6]**:** é uma biblioteca de código aberto para a modelagem de tópicos, sendo bastante popular em aplicações de *Word2Vec* e *Doc2Vec*.

[3] https://www.nltk.org/
[4] https://numpy.org/
[5] https://pandas.pydata.org/
[6] https://radimrehurek.com/gensim/

5 Resultados

Os resultados obtidos com base nos experimentos, cujos métodos foram descritos no Capítulo 4 serão divididos em duas fases. A primeira fase, descrita na Seção 5.1 elenca os passos tomados para obter um modelo *baseline* ou modelo base. Este modelo base é utilizado posteriormente na segunda fase, descrita na Seção 5.2, a fim de comparar com diferentes técnicas consideradas estado da arte.

A máquina onde os experimentos foram realizados consiste num processador Intel i7-8565U com 4 núcleos e 8 *threads*, com 32GB de memória *RAM*, operando numa arquitetura de 64 *bits*. Neste trabalho não foi usado aceleração gráfica.

5.1 Método Base

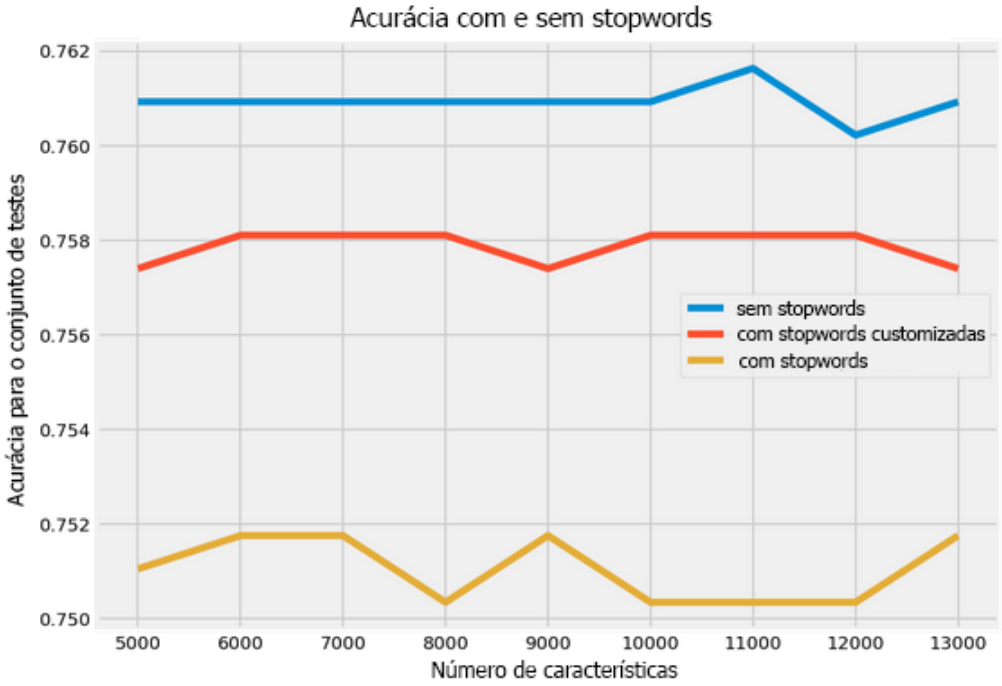
O maior objetivo desta pesquisa foi examinar como a aplicação de diferentes algoritmos, assim como diferentes técnicas de processamento de texto influenciavam o resultado final da tarefa de classificação. Desta forma, diversos conjuntos de classificadores e vetorização foram testados e comparados. A comparação entre eles foi dada através das medidas de precisão, *recall*, *F1-Score* e acurácia, descritos com detalhes no Capítulo 2.

A criação de um método base simplificado foi escolhida para que a posterior comparação com algoritmos mais sofisticados fosse feita. Desta forma, foi possível avaliar os prós e os contras sobre o tempo de processamento *versus* a precisão da classificação final para detecção de discurso de ódio.

Para a criação do método base foi necessário levar em consideração algumas características do conjunto de dados. O conjunto de dados é relativamente pequeno, com pouco menos de 6000 entradas. Além disso, as entradas individuais não contêm grandes quantidades de informações, já que o comprimento médio de cada *tweet* é de 28 caracteres. Para prevenir *overfitting*, o algoritmo de Regressão Logística foi selecionado, por oferecer certa robustez contra esse tipo de problema.

Para o método base, foram criados dois cenários diferentes. No primeiro cenário foi explorado como o uso de *stopwords* afetava a precisão de classificação. O segundo cenário teve como objetivo explorar o efeito que diferentes contagens de N-gramas tinham sobre a precisão da classificação final. Em ambos os cenários a modelagem de BoW foi utilizada, com diferentes quantidades de palavras.

Figura 16 – Acurácias de acordo com os diferentes usos de *stopwords*, dada uma variação na quantidade máxima de palavras



Fonte: elaborada pela autora.

5.1.1 Primeiro Cenário

Neste cenário, o uso de *stopwords* foi explorado. As acurácias foram medidas para os seguintes casos:

- Remoção de *stopwords* com base no conjunto de palavras fornecidos pela biblioteca Sci-kit Learn.
- Criação de um conjunto de *stopwords* customizado para o conjunto de dados em questão, com as 10 palavras mais utilizadas.
- *Stopwords* não foram removidas.

Os resultados estão resumidos na Figura 16. Na linha em amarelo é possível ver os resultados para a classificação sem o uso de *stopwords*. Nas linhas de cor vermelha e azul são mostrados os resultados para o conjunto de *stopwords* customizadas e o conjunto de palavras fornecido pela biblioteca Sci-kit Learn, respectivamente.

Mesmo com um modelo simples e sem utilizar métodos de otimização, como *GridSearch*, é possível notar que não retirar *stopwords* pode causar grande confusão para o algoritmo de classificação, o que está de acordo com a literatura. O uso de um conjunto de palavras customizado obteve melhores resultados, no entanto, o uso das *stopwords* fixas da biblioteca se mostrou superior.

Isso provavelmente se deve ao fato que as 10 palavras mais comuns no conjunto de dados podem, na verdade, ser importantes para o entendimento do algoritmo. Este problema pode ser contornado utilizando um conjunto de dados maior. Apesar disso, é necessário levar em consideração o domínio: muitas vezes *tweets* são informais em demasia, o que acarreta em um vocabulário empobrecido, de maneira geral.

O algoritmo de BoW selecionado foi o *CountVectorizer*, da biblioteca Sci-kit Learn. O *CountVectorizer* forma vetores de palavras a partir da contagem das mesmas. O conjunto de dados escolhido possui em torno de 15000 palavras, portanto, para a construção do método base, optou-se por investigar o efeito de diferentes contagens de palavras. Como pode ser visto na Figura 16, os resultados mostram que o uso de *stopwords* possui um impacto maior na acurácia final do que a quantidade de palavras vetorizadas, para este caso de uso.

Removendo as *stopwords* fornecidas pela biblioteca Sci-kit Learn, foi possível encontrar o modelo de maior acurácia para o primeiro cenário, com 76,02%.

5.1.2 Segundo Cenário

O melhor modelo do primeiro cenário, foi então, levado ao segundo cenário, onde foi explorado o efeito que diferentes contagens de N-gramas tinham sobre o resultado final de classificação. Os resultados dos testes para o segundo cenário estão resumidos na Figura 17.

Novamente, os testes foram realizados utilizando a modelagem de BoW, implementado através do algoritmo *CountVectorizer*. Os testes foram feitos realizando a mesma variação na contagem de palavras.

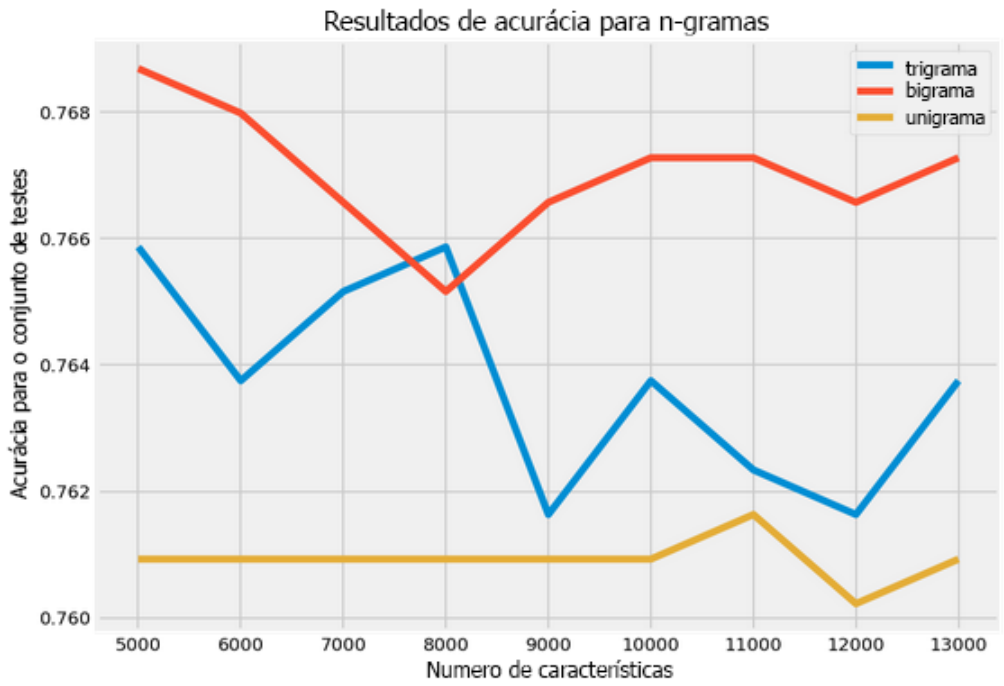
Neste cenário, três quantidades de N-gramas foram exploradas: unigrama, bigrama e trigrama. Como mostrado na Figura 17, as linhas de cores amarelo, vermelho e azul representam a variação de acurácia para modelagem via unigrama, bigrama e trigrama, respectivamente.

A Figura mostra a performance pior da modelagem via unigrama em relação às outras contagens. Isso pode se dar ao fato que o unigrama é uma modelagem muito simples, oferecendo um menor grau de compreensão textual.

A acurácia para as contagens de bigrama e trigrama variaram de forma relevante com relação à contagem de palavras fornecida pelo *CountVectorizer*. De maneira geral, obteve-se melhores resultados com uma menor quantidade máxima de palavras, em ambos os casos. Desta forma foi possível obter um modelo vencedor para o segundo cenário, com uma combinação de bigrama e modelo de BoW de 5000 palavras.

Para que os resultados do modelo base sejam estatisticamente relevantes, o método de *GridSearch* com validação cruzada foi realizado. Com isso, obtemos os melhores parâmetros para o método, que podem ser vistos na Tabela 5. A acurácia final para o modelo foi de 76,87%, marginalmente superior ao modelo vencedor do primeiro cenário.

Figura 17 – Acurácias de acordo com as diferentes contagens de N-gramas, dada uma variação na quantidade máxima de palavras



Fonte: elaborada pela autora.

Tabela 5 – Melhores parâmetros para o modelo base

Nome do parâmetro	**Valor obtido**
C	1.0
intercept_scaling	1
penalty	l2
solver	lbfgs
dual	False
multi_class	auto
fit_intercept	True
tol	0.0001

5.2 Classificação

Os resultados obtidos no método base obtiveram acurácia superior à 70%, podendo ser considerados satisfatórios. No entanto, trabalhos similares realizados com conjunto de dados composto por *tweets* em outra línguas se mostram superiores, como visto no Capítulo 3.

Alguns destes trabalhos, além de utilizarem diferentes técnicas de engenharia de características, também usam uma combinação de algoritmos sofisticados, como redes neurais e redes de aprendizado profundo. Apesar de produzirem resultados muito bons, tanto no âmbito de tarefas relacionadas ao Processamento de Linguagem Natural, quanto em outros ramos de Inteligência Artificial, necessitam de longos períodos de treinamento, ou então grande capacidade

de processamento. Este problema é agravado com o tamanho do conjunto de dados.

Desta forma, a proposta desta seção é utilizar algoritmos de classificação mais ágeis, que não necessitem de aceleração gráfica para atingir bons resultados. Espera-se então, com o auxílio da técnica de *feaure engineering*, descrita no Capítulo 4, atingir resultados similares à métodos mais complexos.

O modelo *baseline* obteve uma acurácia final de 76,87%. Nele foi utilizada apenas uma técnica de vetorização, o BoW em conjunto com o algoritmo de Regressão Logística. Para os testes a seguir, foram utilizados os algoritmos *Naive-Bayes*, k-NN, *Random Forest*, *CatBoost*, Regressão Logística, SVM e MLP aliados a diferentes métodos de vetorização.

Todos os testes foram realizados com a adição de quatro características extras, descritas anteriormente, baseadas na contagem e frequência das palavras. Assim, os classificadores receberam além do texto pré-processado, a soma das palavras para cada classe e também o IDF das palavras para cada classe como entrada.

5.2.1 *Bag-of-Words*

A Tabela 6 sumariza os resultados obtidos através da modelagem via BoW. Os testes feitos com todos os algoritmos partem do modelo base, onde foram usados uma contagem de palavras igual a 5000 e modelo de bigrama.

É possível perceber, que atentando-se à precisão prevista com o algoritmo de Regressão Logística, o uso de extração de características traz uma diferença considerável no resultado final. O algoritmo MLP trouxe o melhor resultado para este caso. É interessante apontar que os resultados apresentam baixa taxa de falsos negativos e positivos, devido aos altos *F1-score* e *recall*.

Tabela 6 – Métricas para o modelo de vetorização *Bag-ofWords*

Algoritmo	**Precisão**	***Recall***	***F1-Score***	**Acurácia**
Naive Bayes	0,920706	0,917989	0,918778	0,917989
Regressão Logística	0,928985	0,929453	0,928808	0,929453
k-NN	0,925573	0,925925	0,924980	0,925925
RandomForest	0,925774	0,925044	0,923465	0,925044
CatBoost	0,934639	0,934744	0,933911	0,934744
SVM	0,935684	0,932098	0,932924	0,932098
MLP	**0,936137**	**0,936507**	**0,936201**	**0,936507**

5.2.2 *TF-IDF*

De modo a enriquecer os resultados obtidos com a representação via BoW, também foi feita a representação com a técnica de TF-IDF, vista com detalhes no Capítulo 2. Para este tipo

de vetorização, foram testadas duas classes de N-gramas, unigrama e bigrama. Modelos de n-gramas mais complexos não foram incluídos nos testes, devido ao desempenho pouco estável, como mostrado na Figura 17

Os resultados para o caso de TF-IDF utilizando unigramas com as métricas apropriadas podem ser sumarizados na Tabela 7. Novamente, os mesmos classificadores foram utilizados, juntamente com as características extras criadas na etapa de pré-processamento. De maneira similar, a Tabela 8 mostra os resultados para vetorização usando TF-IDF e modelagem de bigrama, sob as mesmas condições.

Tabela 7 – Métricas para o modelo de vetorização TF-IDF utilizando unigramas

Algoritmo	**Precisão**	***Recall***	***F1-Score***	**Acurácia**
Naive Bayes	0,916458	0,914462	0,915115	0,914462
Regressão Logística	0,931104	0,931217	0,930275	0,931217
k-NN	0,928191	0,928571	0,927758	0,928571
RandomForest	0,923510	0,922399	0,920564	0,922399
CatBoost	0,932826	0,932981	0,932125	0,932981
SVM	**0,945638**	**0,945326**	**0,944577**	**0.945326**
MLP	0,935448	0,934744	0,933534	0,934744

Tabela 8 – Métricas para o modelo de vetorização TF-IDF utilizando bigramas

Algoritmo	**Precisão**	***Recall***	***F1-Score***	**Acurácia**
Naive Bayes	0,922020	0,922399	0,921335	0,922399
Regressão Logística	0,932064	0,932099	0,931136	0,932099
k-NN	0,928191	0,928571	0,927758	0,928571
RandomForest	0,901195	0,895944	0,891177	0,895944
CatBoost	0,931104	0,931217	0,930275	0,931217
SVM	**0,948434**	**0,947971**	**0,947234**	**0.947971**
MLP	0,944266	0,943562	0,942627	0,943562

É possível perceber que não houve diferenças significativas, tanto com relação à quantidade de N-grama quanto com relação à vetorização via BoW. Em ambos os casos, o algoritmo SVM trouxe os melhores resultados, possuindo métricas muito similares tanto para utilização de unigramas quanto para a utilização de bigramas. Já o algoritmo MLP obteve uma melhora de aproximadamente 1% com a mudança de unigrama para bigrama. Isso pode indicar que a rede de Perceptrons consegue compreender melhor o contexto quando uma estrutura mais complexa de vetorização é usada. As métricas de F1-score e *recall*, para ambos os casos, permanecem elevadas, indicando baixa taxa de falso negativos.

5.2.3 *Doc2Vec*

A Tabela 9 mostra uma sumarização dos resultados para os testes utilizando uma vetorização com a técnica *Doc2Vec* e a arquitetura DBOW, vista no Capítulo 2. Os resultados apresentados foram vetorizados seguindo a arquitetura DBOW, e foram similares aos resultados obtidos com as técnicas de vetorização BoW e TF-IDF.

Tabela 9 – Métricas para o modelo de vetorização *Doc2Vec*

Algoritmo	**Precisão**	***Recall***	***F1-Score***	**Acurácia**
Naive Bayes	0.915508	0.912698	0.913538	0.912698
Regressão Logística	0.931013	0.931217	0.930339	0.931217
k-NN	0.930935	0.925925	0.930402	0.925925
RandomForest	0.925376	0.925926	0.925314	0.925926
CatBoost	0.931012	0.931217	0.930339	0.931217
SVM	**0.938171**	**0.938271**	**0.937540**	**0.938271**
MLP	0.937110	0.936507	0.935393	0.936507

Novamente, é possível notar altas pontuações de *recall* e *F1-score*, indicando baixa taxa de falsos negativos e falsos positivos. Além disso, todos algoritmos obtiveram resultados iguais de *recall* e acurácia. Isso pode indicar que tais algoritmos atingiram a mesma taxa de seletividade, ou seja, *True Negative Rate* (TNR). Com ambos *recall* e TNR iguais, então deriva-se que a acurácia também será a mesma.

Para os testes feitos com *Doc2Vec*, o parâmetro de dimensionalidade não foi alterado, permanecendo com o valor constante de 100 em todas as iterações. O parâmetro de janela não foi fornecido em nenhum teste, já que é opcional para a biblioteca utilizada (Gensim [1]).

O algoritmo vencedor foi novamente o SVM, porém, ainda com valores inferiores à combinação SVM + TF-IDF com bigrama.

5.3 Discussão

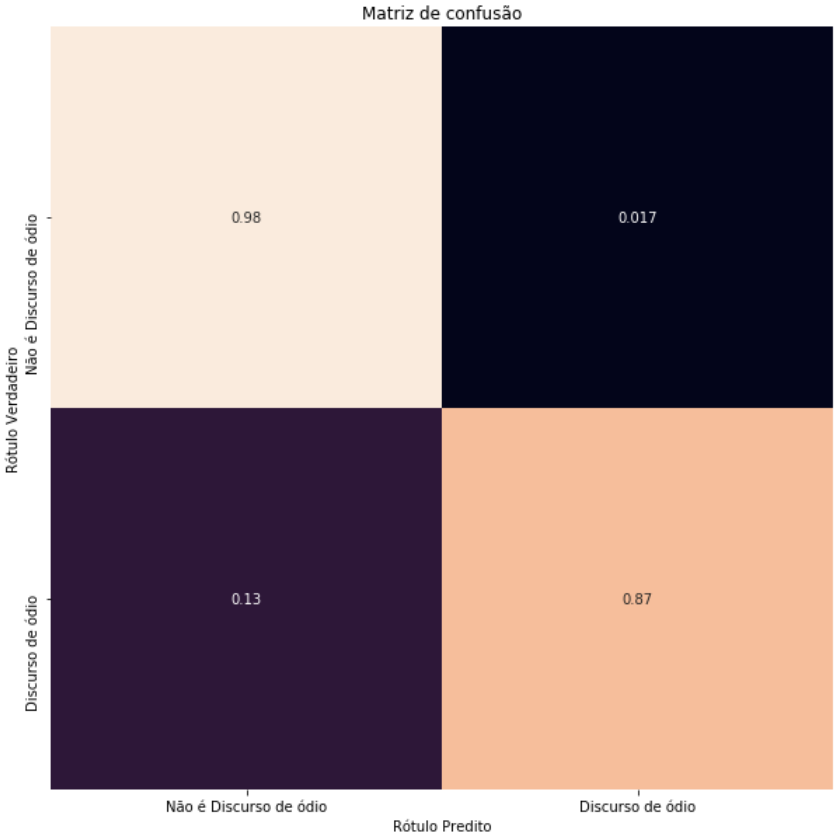
A Figura 18 mostra a matriz de confusão para o melhor modelo obtido. Ele é composto pelo classificador SVM e uma representação em *TF-IDF* e bigramas.

Todos os testes foram executados através do método *GridSearch* e validação cruzada com $k = 5$. Para que o trabalho contenha apenas informações relevantes, foi incluído na Tabela 10 os melhores parâmetros apenas para a combinação vencedora.

Os resultados apresentados neste trabalho, quando comparados a outros trabalhos que usam o mesmo conjunto de dados, se mostram melhores, conforme apresentado na Tabela 11.

[1] https://radimrehurek.com/gensim/

Figura 18 – A matriz de confusão para o modelo usando *SVM + TF-IDF* com bigrama



Fonte: elaborada pela autora.

Tabela 10 – Melhores parâmetros para o modelo de *SVM*

Nome do parâmetro	**Valor obtido**
C	1.0
loss	squared_hinge
penalty	l2
tol	$1e^4$
dual	True
multi_class	ovr
fit_intercept	True
intercept_scaling	1

É interessante mencionar que os trabalhos que utilizaram o conjunto de dados de FORTUNA et al. apenas aplicaram algoritmos de classificação, sem o uso de características extras.

Além disso, o tempo de processamento e a complexidade dos algoritmos e técnicas utilizados também é um fator a ser levado em consideração. A Tabela 12 apresenta o compilado para os tempos de processamento para todos os testes.

Alguns testes diferem consideravelmente no tempo gasto por critérios técnicos: alguns algoritmos possuem mais opções de customização através da biblioteca Sci-kit Learn do que

Tabela 11 – Comparação entre os diferentes trabalhos de estado da arte.

Autor	*Dataset*	Model	F1
Este trabalho	FORTUNA et al.	SVM + TF-IDF + *features*	0.944
MACAVANEY et al.	TRAC (Facebook)	Multi-view SVM	0.5368
MACAVANEY et al.	Stormfornt	Multi-view SVM	0.8031
RUWANDIKA; WEERASINGHE	Próprio	Naive-Bayes	0.709
SAHA; DUBEY; BHATTACHARYYA	HASOC-2019	SVM	0.64
FORTUNA et al.	FORTUNA et al.	LSTM + xgBoost	0.78
NASCIMENTO et al.	Próprio	LinearSVC (SVM)	0.955
UNSVÅG; GAMBÄCK	FORTUNA et al.	LogReg + metadata	0.79

Tabela 12 – Medidas de tempo de processamento para todos os testes

Pré-processamento	Modelo	Tempo
Count Vectorize	Naive Bayes	00:00.019347000
Count Vectorize	Regressão Logística	00:02.343595000
Count Vectorize	KNN	00:18.715416000
Count Vectorize	CatBoost	00:28.607344000
Count Vectorize	RandomForest	10:07.596466000
Count Vectorize	SVM	00:01.652220000
Count Vectorize	MLP	00:58.596666000
TF-IDF 1-grams	Naive Bayes	00:00.022793000
TF-IDF 1-grams	Regressão Logística	00:03.029153000
TF-IDF 1-grams	KNN	00:19.786714000
TF-IDF 1-grams	CatBoost	00:42.151486000
TF-IDF 1-grams	RandomForest	10:22.701351000
TF-IDF 1-grams	SVM	00:02.357018000
TF-IDF 1-grams	MLP	01:14.272502000
TF-IDF 2-grams	Naive Bayes	00:00.041663000
TF-IDF 2-grams	Regressão Logística	00:07.093292000
TF-IDF 2-grams	KNN	00:26.267021000
TF-IDF 2-grams	CatBoost	02:21.806615000
TF-IDF 2-grams	RandomForest	22:14.096713000
TF-IDF 2-grams	SVM	00:02.705719000
TF-IDF 2-grams	MLP	10:03.351424000
Doc2Vec	Naive Bayes	00:00.050586000
Doc2Vec	Regressão Logística	00:29.928659000
Doc2Vec	KNN	00:36.510388000
Doc2Vec	CatBoost	00:04.135514000
Doc2Vec	RandomForest	02:46.075434000
Doc2Vec	SVM	00:01.254356000
Doc2Vec	MLP	00:15.236144000

outros; logo, o tempo de execução para estes será maior. Ainda assim, é possível observar características esperadas. O algoritmo *Naive Bayes* utilizado (MultinomialNB), além de ser inerentemente rápido, não possui parâmetros para otimização, fazendo com que sua execução seja ágil. Os algoritmos *RandomForest* e k-NN possuem muitas alternativas de otimização, fazendo com que seus testes sejam mais demorados. De forma geral, o SVM foi ágil em todos os casos, apresentando um bom balanceamento entre tempo e precisão.

Estes fatos, associados aos resultados apresentados no Capitulo 5, levam a fortes indícios que o uso de engenharia de características possui um efeito extremamente positivo na tarefa de detecção de discurso de ódio, ao menos para conjuntos de dados de tamanho similar.

6 Considerações Finais

Discurso de ódio é um problema crescente no âmbito das mídias sociais. Para proteger a saúde mental da população e também prevenir possíveis incidentes violentos, a supervisão de atividades *online* pode se tornar realidade. Construir métodos de detecção automática que possam detectar e classificar este tipo de discurso de forma precisa é um importante passo para garantir que a *Internet* seja um espaço seguro para livre expressão.

Neste trabalho foi usado um conjunto de dados composto por textos da plataforma social *Twitter*, chamados de *tweets*. Este conjunto de dados foi previamente classificado por humanos, com o intuito de criar um conjunto de dados padrão para a língua portuguesa.

Este trabalho se propôs a comparar diferentes métodos de Aprendizado de Máquina para a classificação de discurso de ódio na *Internet*, além de investigar o efeito que diferentes técnicas de pré-processamento surtiam na precisão da classificação final. O uso de técnicas mais simples de Aprendizado de Máquina foi proposital, para tentar mostrar que dependendo da aplicação, não é necessário utilizar um algoritmo 'caixa-preta', ainda mais quando estamos lidando com assuntos sensíveis, com possíveis bloqueios de contas em redes sociais. Outra vantagem é o custo de processamento, já que algoritmos mais simples não precisam de elevado poder computacional para terminarem a tarefa com sucesso.

Os resultados foram satisfatórios, e foi percebido que o uso de engenharia de características melhora drasticamente o resultado final da classificação, sem grande ônus ao tempo de processamento. Este trabalho atingiu uma precisão de 94.8% utilizando a técnica de vetorização TF-IDF com modelagem de bigrama, combinada com uso de características extras, e com o auxílio da classificador *SVM*, é um resultado que rivaliza o atual estado da arte para o tema.

As limitações atuais do classificador podem ser derivadas dos recursos escassos para a língua portuguesa, como a falta de um dicionário léxico de palavras ofensivas e também de outras ferramentas de extração de características. Para trabalhos futuros, é recomendada a inclusão de rotulação gramatical como caraterística adicional para classificação. Os resultados implicam que o uso de diferentes características textuais possuem mais impacto que o classificador em si. Para confirmar esta hipótese, é recomendado que outros classificadores sejam testados e avaliados.

Referências

AGGARWAL, C. C.; ZHAI, C. *Mining text data*. [S.l.]: Springer Science & Business Media, 2012. Citado na página 14.

ASSOCIATION, A. L. et al. *Hate speech and hate crime*. 2017. Disponível em: <http://www-.ala.org/advocacy/intfreedom/hate>. Citado na página 18.

BADJATIYA, P. et al. Deep learning for hate speech detection in tweets. p. 759–760, 2017. Citado na página 14.

BILYK, V. *Why Business Applies Sentiment Analysis? 5 Successful Examples*. 2020. Disponível em: <https://theappsolutions.com/blog/development/sentiment-analysis-for-business/>. Citado na página 19.

BISHOP, C. M. *Pattern recognition and machine learning*. [S.l.]: springer, 2006. Citado na página 27.

CARVALHO, N. R.; SIMOES, A. Pln. pt: Processamento de linguagem natural para português como um serviço. *Linguamática*, v. 10, n. 1, p. 29–33, 2018. Citado na página 14.

CUNNINGHAM, P.; DELANY, S. J. k-nearest neighbour classifiers–. *arXiv preprint arXiv:2004.04523*, 2020. Citado 2 vezes nas páginas 29 e 30.

DOROGUSH, A. V.; ERSHOV, V.; GULIN, A. Catboost: gradient boosting with categorical features support. *arXiv preprint arXiv:1810.11363*, 2018. Citado na página 34.

DUDA, R. O.; HART, P. E.; STORK, D. G. *Pattern classification and scene analysis*. [S.l.]: Wiley New York, 1973. Citado na página 31.

EUROPE, C. of. *No Hate Speech Movement*. 2017. Disponível em: <https://www.coe.int/en-/web/no-hate-campaign/no-hate-speech-movement>. Citado na página 17.

FAIR, V. *YouTube Is Finally Taking a Harder Line on Hate Speech. Is It Too Little, Too Late?* 2019. Disponível em: <https://www.vanityfair.com/news/2019/06/youtube-hate-speech-too-little-too-late>. Citado na página 13.

FLEISS, J. L. Measuring nominal scale agreement among many raters. *Psychological bulletin*, American Psychological Association, v. 76, n. 5, p. 378, 1971. Citado na página 47.

FONTANELLA, F. Nós somos anonymous: anonimato, trolls e a subcultura dos imageboards. *Intercom – Sociedade Brasileira de Estudos Interdisciplinares da Comunicação.*, 2010. Http://www.intercom.org.br/papers/nacionais/2010/resumos/R5-1964-1.pdf. Citado na página 44.

FORTUNA, P.; NUNES, S. A survey on automatic detection of hate speech in text. *ACM Computing Surveys (CSUR)*, ACM New York, NY, USA, v. 51, n. 4, p. 1–30, 2018. Citado 2 vezes nas páginas 17 e 43.

FORTUNA, P. et al. A hierarchically-labeled portuguese hate speech dataset. In: *Proceedings of the Third Workshop on Abusive Language Online*. [S.l.: s.n.], 2019. p. 94–104. Citado 8 vezes nas páginas 9, 14, 44, 45, 47, 48, 62 e 63.

GARDNER, M. W.; DORLING, S. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric environment*, Elsevier, v. 32, n. 14-15, p. 2627–2636, 1998. Citado 2 vezes nas páginas 37 e 38.

GOLDBERG, Y.; HIRST, G. *Neural Network Methods in Natural Language Processing. Morgan & Claypool Publishers(2017)*. [S.l.: s.n.]. Citado 2 vezes nas páginas 21 e 22.

GOMES, R. de E. L. F. *O que se entende por Hate Speech? Daniella Parra Pedroso Yoshikawa*. 2010. Disponível em: <https://lfg.jusbrasil.com.br/noticias/2489788/o-que-se-entende-por-hate-speech-daniella-parra-pedroso-yoshikawa>. Citado na página 19.

GUNN, S. R. et al. *Support vector machines for classification and regression*. [S.l.], 1998. v. 14, n. 1, 5–16 p. Citado 2 vezes nas páginas 35 e 36.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. *The elements of statistical learning: data mining, inference, and prediction*. [S.l.]: Springer Science & Business Media, 2009. Citado na página 31.

HERN, A. *Facebook, YouTube, Twitter and Microsoft sign EU hate speech code*. 2016. Disponível em: <https://www.theguardian.com/technology/2016/may/31/facebook-youtube-twitter-microsoft-eu-hate-speech-code>. Citado na página 17.

HITWISE. *Social networking visits increase 11.5 percent from January to February*. 2007. Citado na página 13.

ILGA. *Hate crime and hate speech*. 2016. Disponível em: <http://www.ilga-europe.org/what-we-do/our-advocacy-work/hate-crime-hate-speech>. Citado na página 18.

JAIN, A. K.; DUBES, R. C. *Algorithms for clustering data*. [S.l.]: Prentice-Hall, Inc., 1988. Citado na página 27.

JANG, B.; KIM, I.; KIM, J. W. Word2vec convolutional neural networks for classification of news articles and tweets. *PloS one*, Public Library of Science San Francisco, CA USA, v. 14, n. 8, p. e0220976, 2019. Citado 2 vezes nas páginas 23 e 24.

JURASKY, D.; MARTIN, J. H. *Speech and Language Processing: An introduction to natural language Processing*. [S.l.: s.n.], 2000. Citado 5 vezes nas páginas 20, 21, 22, 25 e 26.

KANTARCIOGLU, M.; VAIDYA, J.; CLIFTON, C. Privacy preserving naive bayes classifier for horizontally partitioned data. In: *IEEE ICDM workshop on privacy preserving data mining*. [S.l.: s.n.], 2003. p. 3–9. Citado na página 30.

KOTSIANTIS, S. B.; ZAHARAKIS, I.; PINTELAS, P. Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, Amsterdam, v. 160, n. 1, p. 3–24, 2007. Citado na página 27.

KOTTASOVA, I. *Europe says Twitter is failing to remove hate speech*. 2017. Disponível em: <https://money.cnn.com/2017/06/01/technology/twitter-facebook-hate-speech-europe/index-.html>. Citado na página 17.

LE, Q.; MIKOLOV, T. Distributed representations of sentences and documents. In: *International conference on machine learning*. [S.l.: s.n.], 2014. p. 1188–1196. Citado na página 24.

LORENA, A. C.; CARVALHO, A. C. de. Uma introdução às support vector machines. *Revista de Informática Teórica e Aplicada*, v. 14, n. 2, p. 43–67, 2007. Citado 2 vezes nas páginas 35 e 36.

LORENA, A. C.; GAMA, J.; FACELI, K. *Inteligência Artificial: Uma abordagem de aprendizado de máquina*. [S.l.]: Grupo Gen-LTC, 2000. Citado 2 vezes nas páginas 14 e 27.

MACAVANEY, S. et al. Hate speech detection: Challenges and solutions. *PloS one*, Public Library of Science San Francisco, CA USA, v. 14, n. 8, p. e0221152, 2019. Citado 4 vezes nas páginas 13, 14, 44 e 63.

MANNING, C. D.; SCHÜTZE, H.; RAGHAVAN, P. *Introduction to information retrieval*. [S.l.]: Cambridge university press, 2008. Citado 3 vezes nas páginas 20, 21 e 22.

MIKOLOV, T. et al. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013. Citado 2 vezes nas páginas 22 e 23.

MIKOLOV, T. et al. Distributed representations of words and phrases and their compositionality. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2013. p. 3111–3119. Citado na página 14.

MITCHELL, T. M. et al. Machine learning. 1997. *Burr Ridge, IL: McGraw Hill*, v. 45, n. 37, p. 870–877, 1997. Citado 2 vezes nas páginas 14 e 27.

MOLNAR, C. *Interpretable machine learning. A Guide for Making Black Box Models Explainable*. [S.l.]: https://christophm.github.io/interpretable-ml-book/, 2019. Citado na página 32.

MURRAY, K. E.; WALLER, R. Social networking goes abroad. *International Educator*, NAFSA: ASSOCIATION OF INTERNATIONAL EDUCATORS, v. 16, n. 3, p. 56, 2007. Citado na página 13.

NASCIMENTO, G. et al. Hate speech detection using brazilian imageboards. p. 325–328, 2019. Citado 3 vezes nas páginas 14, 44 e 63.

NIELSEN, M. *Neural Networks and Deep Learning*. [S.l.]: Determination Press, 2015. Citado na página 37.

NOCKLEBY, J. T. *Hate speech*. [S.l.]: Macmillan Reference US Detroit, 2000. 1277–1279 p. Citado na página 13.

OLIVIER, C.; BERNHARD, S.; ALEXANDER, Z. *Semi-supervised learning*. [S.l.: s.n.], 2006. 542–542 p. Citado na página 27.

PAL, M. Random forest classifier for remote sensing classification. *International journal of remote sensing*, Taylor & Francis, v. 26, n. 1, p. 217–222, 2005. Citado na página 31.

PANG, B.; LEE, L. Opinion mining and sentiment analysis. *Comput. Linguist*, v. 35, n. 2, p. 311–312, 2009. Citado na página 19.

PELLE, R. P. de; MOREIRA, V. P. Offensive comments in the brazilian web: a dataset and baseline results. In: SBC. *Anais do VI Brazilian Workshop on Social Network Analysis and Mining*. [S.l.], 2017. Citado na página 14.

REZENDE, S. O. Sistemas inteligentes: fundamentos e aplicações. In: ____. [S.l.]: Editora Manole Ltda, 2003. Citado 6 vezes nas páginas 27, 28, 29, 38, 39 e 40.

ROBERTSON, C.; MELE, C.; TAVERNISE, S. *11 Killed in Synagogue Massacre; Suspect Charged With 29 Counts*. 2018. Disponível em: <https://www.nytimes.com-/2018/10/27/us/active-shooter-pittsburgh-synagogue-shooting.html>. Citado na página 13.

RUWANDIKA, N.; WEERASINGHE, A. Identification of hate speech in social media. In: IEEE. *2018 18th International Conference on Advances in ICT for Emerging Regions (ICTer)*. [S.l.], 2018. p. 273–278. Citado 2 vezes nas páginas 44 e 63.

SAHA, U.; DUBEY, A.; BHATTACHARYYA, P. Iit bombay at hasoc 2019: Supervised hate speech and offensive content detection in indo-european languages. In: *FIRE (Working Notes)*. [S.l.: s.n.], 2019. p. 352–358. Citado 2 vezes nas páginas 44 e 63.

SARMENTO, D. *A LIBERDADE DE EXPRESSÃO E O PROBLEMA DO "HATE SPEECH"*. Dissertação (Mestrado) — Pontifícia Universidade Católica de Goiás, 2006. Citado na página 17.

SCHMIDT, A.; WIEGAND, M. A survey on hate speech detection using natural language processing. In: *Proceedings of the Fifth International workshop on natural language processing for social media*. [S.l.: s.n.], 2017. p. 1–10. Citado na página 43.

SMARTINSIGHTS. *Global social media research summary July 2020*. 2020. Disponível em: <https://www.smartinsights.com/social-media-marketing/social-media-strategy/new-global-social-media-research/>. Citado na página 13.

STEIN, G. et al. Decision tree classifier for network intrusion detection with ga-based feature selection. In: *Proceedings of the 43rd annual Southeast regional conference-Volume 2*. [S.l.: s.n.], 2005. p. 136–141. Citado na página 31.

THARWAT, A. Classification assessment methods. *Applied Computing and Informatics*, 2018. Https://www.sciencedirect.com/science/article/pii/S2210832718301546. Citado 2 vezes nas páginas 40 e 41.

TIMES, T. N. Y. *New Zealand Shooting Live Updates: 49 Are Dead After 2 Mosques Are Hit*. 2019. Citado na página 13.

TOMANEK, K.; HAHN, U. Semi-supervised active learning for sequence labeling. In: *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. [S.l.: s.n.], 2009. p. 1039–1047. Citado na página 27.

TWITTER. *Política contra propagação de ódio*. 2017. Disponível em: <https://help.twitter-.com/pt/rules-and-policies/hateful-conduct-policy>. Citado na página 18.

UNIAO, D. O. de. *Constituição Federal Brasileira de 1988*. [S.l.]: D.O.U. Online, 1988. Citado na página 19.

UNSVÅG, E. F.; GAMBÄCK, B. The effects of user features on twitter hate speech detection. p. 75–85, 2018. Citado 2 vezes nas páginas 45 e 63.

VERGE, T. *Facebook says it's taking new steps to stop hate speech in Sri Lanka and Myanmar*. 2019. Disponível em: <https://www.theverge.com/2019/6/21/18701073/facebook-myanmar-sri-lanka-messenger-hate-speech>. Citado na página 13.

WALDRON, J. *The Harm in Hate Speech*. [S.l.]: Harvard University Press, 2014. Citado na página 43.

WALKER, S. *Hate Speech: The History of an American Controversy*. [S.l.]: University of Nebraska Press, 1994. Citado na página 43.

WATKINS, C. J.; DAYAN, P. *Q-learning*. [S.l.]: Springer, 1992. 279–292 p. Citado na página 27.

WIGAND, C.; VOIN, M. *Speech by Commissioner Jourová—10 years of the EU Fundamental RightsAgency: A call to action in defence of fundamental rights, democracy and the rule of law*. 2017. Disponível em: <https://ec.europa.eu/commission/presscorner/detail/en-/SPEECH_17_403>. Citado na página 18.

WIRED. *Twitter and Instagram Unveil New Ways to Combat Hate—Again*. 2019. Disponível em: <https://www.wired.com/story/twitter-instagram-unveil-new-ways-combat-hate-again/>. Citado na página 13.

YANG, Y. Temporal data mining via unsupervised ensemble learning. In: ____. [S.l.]: Elsevier, 2016. cap. Ensemble Learning. Citado 2 vezes nas páginas 33 e 34.

ZHANG, Z.; LUO, L. Hate speech detection: A solved problem? the challenging case of long tail on twitter. *Semantic Web*, IOS Press, v. 10, n. 5, p. 925–945, 2019. Citado na página 13.