
Detecção de comentários ofensivos na rede social YouTube

Kelly Cristina Alves



UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

Monte Carmelo - MG
2024

Kelly Cristina Alves

Detecção de comentários ofensivos na rede social YouTube

Trabalho de Conclusão de Curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como requisito exigido parcial à obtenção do grau de Bacharel em Sistemas de Informação.

Área de concentração: Bacharelado em Sistemas de Informação

Orientador: Fabíola Souza Fernandes Pereira

Monte Carmelo - MG

2024

Este trabalho é dedicado a Deus e a todas as pessoas que estiveram ao meu lado durante esta jornada.

Agradecimentos

Agradeço primeiramente a Deus pela oportunidade de entender a importância da educação e por me permitir vivenciá-la. Agradeço a Ele por ter sido minha força durante os anos de graduação e no desenvolvimento deste trabalho.

Agradeço aos meus pais que, mesmo sem a oportunidade de vivenciar a educação, me ensinaram o valor dela.

Agradeço a minha avó Gasparina por todas as orações e cuidado desde sempre.

Ao meu irmão Rafael, que se tornou minha maior inspiração ao escolher se graduar.

Agradeço ao meu marido Raul, que esteve ao meu lado desde o primeiro dia de graduação, literalmente, e por ter sido meu maior apoio durante o desenvolvimento deste trabalho.

Agradeço também à professora e orientadora Fabiola Souza, por toda a dedicação e atenção, que foram essenciais para a conclusão deste trabalho.

E, por fim, agradeço a todos os professores que passaram pela minha vida, em especial aos da graduação, que contribuíram para minha tão sonhada formação acadêmica.

*“Não a nós, Senhor, nenhuma glória para nós, mas sim ao Teu nome, por Teu amor e
por Tua fidelidade.”
(Salmos 115:1)*

Resumo

Comentários ofensivos e o discurso de ódio têm se tornado um desafio para a moderação de conteúdo nas redes sociais virtuais, e pesquisa sobre técnicas de moderação automatizada para a língua portuguesa brasileira ainda é limitada. Neste contexto, este estudo visa contribuir para o desenvolvimento de um sistema eficiente para a detecção e classificação de comentários ofensivos em português brasileiro, utilizando técnicas de processamento de linguagem natural e aprendizado de máquina.

A abordagem adotada explora um conjunto de dados composto por 4.139 comentários em português brasileiro extraídos do *YouTube* e propõe detectar e classificar automaticamente comentários ofensivos em português brasileiro. Foram comparados quatro algoritmos clássicos de classificação de texto — *Naive Bayes*, *Support Vector Machine* (SVM), *Random Forest* e *Gradient Boosting Machines* (GBM) — aplicados a vetorizadores *CountVectorizer* e *Term Frequency Inverse Document Frequency* (TF-IDF). O modelo *Random Forest*, combinado com *CountVectorizer*, apresentou o melhor desempenho, alcançando 86% de acurácia e medida F1. Esse resultado evidencia a viabilidade do uso de métodos clássicos de aprendizado de máquina na moderação de conteúdo em português brasileiro. Este estudo contribui com a construção e disponibilização de uma base de dados especializada, promovendo avanços no campo da moderação automatizada e fornecendo um recurso útil para o desenvolvimento de modelos voltados ao português. Com isso, o trabalho reforça o potencial do aprendizado de máquina em promover ambientes online mais seguros e inclusivos.

Palavras-chave: Processamento de Linguagem Natural, Aprendizado de Máquina, Comentário ofensivo, Classificação de texto, Moderação de conteúdo.

Lista de tabelas

Tabela 1	–	Comentários retirados da base de dados normalizada.	42
Tabela 2	–	Acurácia média do algoritmo Naive Bayes com diferentes configurações	46
Tabela 3	–	Acurácia média do algoritmo SVM com diferentes configurações	47
Tabela 4	–	Acurácia média do algoritmo Random Forest com diferentes configurações	49
Tabela 5	–	Acurácia média do algoritmo GBM com diferentes configurações	51
Tabela 6	–	Melhores resultados dos algoritmos em termos de acurácia e medida F1	52

Lista de ilustrações

Figura 1 – Gráfico de frequência de palavras.	42
Figura 2 – Nuvem de palavras mais raras.	42
Figura 3 – Gráfico dos 3-grams mais comuns.	43

Lista de siglas

AM Aprendizagem de Máquina

BERT *Bidirectional Encoder Representations from Transformers*

CNN Redes Neurais Convolucionais

GBM *Gradient Boosting Machines*

GPT *Generative Pre-trained Transformer*

GRU *Gated Recurrent Units*

IA Inteligência Artificial

KNN *K-Nearest Neighbors*

LSTM *Long Short-Term Memory*

NMT *Neural Machine Translation*

PLN Processamento de Linguagem Natural

PCA *Análise de Componentes Principais*

RSV *Redes Sociais Virtuais*

RNA *Redes Neurais Artificiais*

RNN *Redes Neurais Recorrente*

SVM *Support Vector Machine*

SMOTE *Synthetic Minority Over-sampling Technique*

TF-IDF *Term Frequency Inverse Document Frequency*

XGB *Gradient Boosting*

Sumário

Lista de ilustrações		7
1	INTRODUÇÃO	12
1.1	Objetivos e desafios da pesquisa	13
1.2	Organização monográfica	14
2	FUNDAMENTAÇÃO TEÓRICA	16
2.1	Linguagem ofensiva e discurso de ódio	16
2.2	Aprendizado de Máquina	18
2.2.1	Aprendizado supervisionado	19
2.2.2	Aprendizado não-supervisionado	19
2.2.3	Divisão de Dados e Avaliação de Desempenho	20
2.2.4	Ajuste de Hiperparâmetros	21
2.2.5	Subamostragem e sobreamostragem	22
2.2.6	Classificação	23
2.3	Processamento de Linguagem Natural	28
2.3.1	Principais Abordagens de Processamento de Linguagem Natural	29
2.3.2	Ferramentas e Técnicas de Pré-Processamento	33
2.4	Ferramentas Utilizadas	36
2.4.1	API YouTube Data	36
2.4.2	Google Colab	37
2.4.3	Biblioteca Scikit-learn	37
2.5	Trabalhos Correlatos	37
3	MATERIAIS E MÉTODOS	40
3.1	Conjunto de Dados	40
3.2	Análise Exploratória dos Dados	41
3.3	Design Experimental	43

3.4	Etapas de Treinamento dos Classificadores	44
4	EXPERIMENTOS E ANÁLISE DOS RESULTADOS	45
4.1	Resultados por algoritmo	45
4.1.1	Naive Bayes	46
4.1.2	SVM	47
4.1.3	Random Forest	48
4.1.4	GBM	50
4.1.5	Sumário dos resultados	52
5	CONCLUSÃO	55
	Referências	57

Introdução

A Internet é frequentemente descrita como "terra de ninguém", um ditado popular que ganhou força com o advento dessa tecnologia. Esse conceito se baseia na percepção de liberdade de expressão desenfreada, que muitos usuários sentem ao explorar suas possibilidades. As Redes Sociais Virtuais (RSV), por exemplo, podem proporcionar um ambiente confortável para aqueles que se aproveitam do anonimato e da distância física para interagir de forma insensível, cruel e desrespeitosa, através de comentários depreciativos, humor insensível ou sarcasmo (Phillips, 2011). Essa forma de comunicação é não apenas ofensiva, mas também pode ter consequências significativas para o bem-estar psicológico, incluindo estresse emocional, depressão, ansiedade e até mesmo pensamentos suicidas.

Diversos autores, tais como Phillips (2011), Waldron (2012) e Hardaker (2013) definem como comentário ofensivo qualquer expressão verbal ou escrita que visa prejudicar, desumanizar, humilhar, marginalizar ou causar desconforto emocional a indivíduos ou grupos. Como já mencionado, um comentário deixado em uma plataforma de mídia social pode ter um impacto significativo na vida das pessoas. Ademais, o discurso de ódio nas redes sociais não é apenas prejudicial para as vítimas diretas, mas também afeta a sociedade como um todo, pois pode se espalhar rapidamente e influenciar muitos outros usuários. A disseminação desenfreada de comentários ofensivos também pode contribuir para a polarização social, o aumento da intolerância e a desintegração do tecido social (Waldron, 2012).

A preocupação com o tema torna-se ainda mais aflitiva diante do seu crescimento alarmante. A matéria veiculada pela revista BBC (Brasil, 2022) traz um estudo realizado no Reino Unido e nos Estados Unidos, expondo um aumento de 20% no discurso de ódio desde o início da pandemia de COVID-19. Este estudo analisou dados de redes sociais, abrangendo plataformas como *Twitter*, *Facebook* e *Instagram*, totalizando 263 milhões de conversas nos dois países entre 2019 e meados de 2021. De acordo com Zhang and Luo (2019), a prevalência de comentários ofensivos nas plataformas de mídia social é preocupante. Dados sugerem que uma porcentagem considerável de usuários já foi exposta a algum tipo de discurso de ódio ou *cyberbullying*, tornando-se um problema sério que

exige intervenções eficazes e uma abordagem multidisciplinar (Zhang and Luo, 2019).

Nesse sentido, a detecção de comentários ofensivos em mídias sociais pode desempenhar um papel crucial na manutenção de um ambiente virtual seguro e respeitoso. De acordo com o artigo "*Detecting Offensive Language in Social Media to Protect Adolescent Online Safety*" (Chen et al., 2012), a utilização de técnicas de Processamento de Linguagem Natural (PLN) e Aprendizagem de Máquina (AM) possibilitam a análise automática e rápida de grandes volumes de texto, permitindo a identificação de padrões de linguagem ofensiva e a classificação precisa dos comentários. Ao empregar PLN e AM, as plataformas de mídia social podem agir na remoção ou moderação desses conteúdos, fomentando um ambiente *online* mais saudável. Como dito em Fabriz and Mendonça (2022), é crucial que as plataformas assumam a responsabilidade de promover a segurança e o bem-estar de seus usuários, mantendo a autorregulação e transparência sobre suas políticas e processos de moderação. Afinal, a Internet não deve ser tratada como uma 'terra de ninguém'.

A interação entre PLN e AM oferece oportunidades empolgantes para a detecção e classificação de comentários ofensivos em redes sociais. Como descrito em Jurafsky and Martin (2019), o PLN é uma disciplina que explora como os computadores podem compreender e interagir com a linguagem humana. Ela abrange uma ampla gama de técnicas para processar, analisar e gerar texto e fala de maneira significativa. Por sua vez, a AM, uma área da Inteligência Artificial (IA) que ensina os computadores a reconhecer padrões nos dados e a fazer previsões ou tomar decisões, é impulsionada pelo PLN, que fornece as técnicas e ferramentas necessárias para compreender e processar eficientemente a linguagem humana. Essa cooperação PLN-AM é crucial para melhorar a capacidade dos sistemas de identificar e lidar com conteúdo ofensivo nas mídias sociais.

A análise de dados textuais por meio dessas técnicas tem gerado resultados significativos, e essa prática tem ganhado cada vez mais atenção. No entanto, na literatura, há uma escassez de estudos sobre a automação da detecção de comentários ofensivos na língua portuguesa brasileira (de Pelle and Moreira, 2017). Diante dessa lacuna, este estudo visa abordar essa questão, investigando e desenvolvendo métodos de detecção de comentários ofensivos na língua portuguesa brasileira. Utilizando um conjunto de dados coletados a partir de comentários em vídeos da rede social *YouTube*, a hipótese deste trabalho sugere que é viável construir um classificador de comentários em português brasileiro que apresente uma boa performance em termos de acurácia e medida F1.

1.1 Objetivos e desafios da pesquisa

O objetivo geral deste estudo é a identificação e classificação de mensagens ofensivas em português brasileiro, empregando técnicas de PLN e AM. Este esforço visa contribuir significativamente para o desenvolvimento de filtros eficazes em português brasileiro para plataformas online. Como objetivos específicos este trabalho busca:

- ❑ Desenvolver uma base de dados robusta e representativa composta por comentários ofensivos extraídos de canais do *YouTube* em português brasileiro, garantindo uma amostra diversificada que capture a variedade de expressões linguísticas presentes no contexto de redes sociais virtuais;
- ❑ Investigar e analisar padrões linguísticos ofensivos por meio de técnicas de mineração de texto e PLN, identificando características relevantes associadas à linguagem ofensiva em português brasileiro;
- ❑ Propor um classificador eficaz para a detecção automática de comentários ofensivos em português brasileiro, visando melhorar a eficiência dos sistemas de moderação automática em redes sociais virtuais nesse idioma.
- ❑ Implementar e avaliar modelos de classificação de comentários ofensivos em português brasileiro, explorando a eficácia de técnicas de AM para automatizar a moderação de conteúdo, contribuindo para o aprimoramento de sistemas de filtragem e detecção em redes sociais.

A hipótese do trabalho é que é possível desenvolver um modelo de AM clássico capaz de identificar e classificar automaticamente comentários ofensivos em português brasileiro, com desempenho satisfatório em termos de acurácia e medida F1. Esse modelo pode então ser usado para melhorar a eficácia dos sistemas de moderação de conteúdo em redes sociais, contribuindo para um ambiente virtual mais seguro e inclusivo. Essa hipótese é baseada na premissa de que técnicas de PLN e AM são adequadas para capturar os padrões linguísticos associados a comentários ofensivos em português, e que é viável adaptar esses métodos a uma base de dados composta por comentários de usuários em redes sociais, como o *YouTube*.

1.2 Organização monográfica

Os demais capítulos da monografia estão organizados da seguinte maneira:

- ❑ O capítulo 2 fornece a fundamentação teórica sobre comentários ofensivos e discurso de ódio em plataformas de mídias sociais, aprendizado de máquina, processamento de linguagem natural, ferramentas e ambientes utilizados e também aborda os trabalhos relacionados ao tema discutido neste estudo.
- ❑ O capítulo 3 traz os materiais e métodos, abordando o conjunto de dados, a análise exploratória dos mesmos e o *design* experimental seguido durante o treinamento dos algoritmos.

- ❑ O capítulo 4 apresenta a realização dos experimentos e os resultados obtidos por cada algoritmo. Também apresenta o sumário dos melhores resultados e a análise do desempenho de cada modelo.
- ❑ O capítulo 5 apresenta as principais conclusões do trabalho e sugestões de trabalhos futuros.

Fundamentação Teórica

Este capítulo apresenta uma fundamentação teórica sobre os principais assuntos envolvidos neste trabalho e também descreve os trabalhos relacionados. Na seção 2.1 discute-se linguagem ofensiva e discurso de ódio. Na seção 2.2 apresenta-se conceitos importantes de aprendizado de máquina, aprendizado supervisionado e não supervisionado, divisão de dados e avaliação de desempenho, ajustes de hiperparâmetros, subamostragem e sobreamostragem e algoritmos de classificação. A seção 2.3 traz os fundamentos do processamento de linguagem natural, as principais abordagens e as ferramentas e técnicas de pré-processamento e vetorização. A sessão 2.4 traz as ferramentas utilizadas neste trabalho, sendo elas *API Youtube Data*, *Google Colab* e a biblioteca *Scikit-learn* e, por fim, a sessão 2.5 apresenta os principais trabalhos relacionados na literatura.

2.1 Linguagem ofensiva e discurso de ódio

As RSV são plataformas *online* que possibilitam aos usuários interagirem, compartilharem conteúdo e se comunicarem com outros membros. Uma das formas mais comuns de interação é por meio de comentários em publicações de terceiros, permitindo a expressão de opiniões, o esclarecimento de dúvidas e a participação em discussões. No entanto, é nesse ambiente que surgem fenômenos preocupantes, como a linguagem ofensiva e o discurso de ódio.

A relação entre discurso de ódio e linguagem ofensiva é próxima, embora se trate de conceitos distintos. Conforme evidenciado por Paasch-Colberg et al. (2021), a linguagem ofensiva é frequentemente encontrada em comentários *online* e pode assumir várias formas, incluindo insultos diretos, linguagem depreciativa, estereótipos negativos, ameaças e até mesmo discurso de ódio. Essa ampla gama de expressões negativas demonstra a interseção clara entre essas duas categorias. Segundo os autores, enquanto alguns comentários podem ser considerados ofensivos pela sua carga depreciativa ou insultuosa, outros ultrapassam essa linha e caracterizam-se como discurso de ódio, promovendo ativamente intolerância, preconceito ou violência contra grupos específicos.

Diversas perspectivas e abordagens surgem quando se trata de definir o discurso de ódio. Sellars (2016) aponta a complexidade e as nuances desse debate. Sob a ótica da incitação ao ódio e à violência, o discurso de ódio é definido como qualquer forma de expressão que incite discriminação, violência ou ódio contra um grupo específico, baseado em características como raça, religião, etnia, orientação sexual, gênero, entre outras. Já a abordagem focada no dano causado se preocupa com os efeitos prejudiciais do discurso de ódio sobre os grupos-alvo, independentemente de incitação direta à violência, considerando seu impacto psicológico, emocional e social nas vítimas e comunidades afetadas.

Um dos primeiros estudos a abordar a questão da agressividade *online*, o livro de Herring (1993), oferece uma análise crítica das oportunidades e desafios que a comunicação mediada por computador impõe à democracia. Desde então, os autores já alertavam que, apesar de a internet democratizar o acesso à comunicação, ela também acentua desigualdades de gênero e cria um espaço propício para o assédio. Por exemplo, Kwon and Gruzd (2017) analisaram a contagiosidade dos comentários ofensivos e concluíram que a exposição a esse tipo de linguagem aumenta a probabilidade de os usuários adotarem comportamentos semelhantes. Isso demonstra a importância de entender os padrões de comportamento na *web* e desenvolver estratégias eficazes para criar um ambiente mais saudável nas mídias sociais.

Suler (2004) traz contribuições valiosas ao discutir estes comportamentos, especialmente o fenômeno da "desinibição *online*", em que as pessoas tendem a agir de forma mais impulsiva e sem filtros em suas interações na internet. Ele distingue entre desinibição benigna e tóxica, sendo esta última marcada por comportamentos agressivos e antiéticos. O anonimato, a invisibilidade, a ausência de consequências imediatas, a distância física e a natureza despersonalizada da comunicação são, segundo o autor, os principais fatores que facilitam esse comportamento. Embora a desinibição possa ajudar na expressão criativa e na formação de laços sociais, ela também pode levar a comportamentos negativos, como *bullying*, *cyberbullying* e *trolling*.

Para além do comportamento, Lenhart et al. (2016) apresentam um estudo abrangente sobre os impactos do assédio online, do abuso digital e da perseguição cibernética, destacando como esses problemas afetam significativamente os usuários da internet. O artigo descreve as várias formas de assédio, incluindo mensagens ameaçadoras e comentários ofensivos. Uma pesquisa nacional conduzida nos Estados Unidos revelou que uma parcela considerável de adultos já vivenciou algum tipo de abuso online, com certos grupos, como mulheres, jovens adultos e minorias étnicas, sendo desproporcionalmente afetados. Além disso, muitas vítimas relataram experiências de ansiedade, medo, depressão e estresse como consequência desses abusos.

Esses estudos evidenciam os desafios representados pelo abuso digital e pela perseguição cibernética para a segurança e o bem-estar emocional dos indivíduos. Portanto, a compreensão e a gestão do discurso *online*, especialmente em relação aos comentários

ofensivos e ao discurso de ódio e , são essenciais para promover interações mais saudáveis nas plataformas digitais contemporâneas. Uma abordagem promissora envolve o uso de sistemas de moderação aprimorados por PLN , que oferecem ferramentas eficazes para analisar grandes volumes de textos de redes sociais, auxiliando na detecção automática de comentários ofensivos.

2.2 Aprendizado de Máquina

O AM é um campo dentro da IA essencial para sistemas que precisam analisar grandes volumes de dados e tomar decisões precisas. Conforme define Mohri (2018), o AM desenvolve algoritmos que extraem e reconhecem padrões nos dados para construir sistemas capazes de realizar previsões ou classificar novas informações com base nesses padrões. Este processo é caracterizado pela capacidade dos modelos adaptarem-se continuamente a novos dados, desencadeando uma adaptação contínua sem que seja necessário programar regras explícitas.

Bishop and Nasrabadi (2006) definem AM como "o estudo de algoritmos que se adaptam com o uso de dados", e defende uma abordagem estatística para entender as relações entre as variáveis e fazer previsões sobre novos dados. Para Bishop, a capacidade adaptativa dos modelos de AM é fundamental, pois os modelos são expostos a uma grande variedade de dados e, com isso, podem fazer generalizações, reduzindo o erro ao serem aplicados em dados inéditos.

Já Hastie et al. (2009) define que o AM descreve o comportamento dos dados através de modelos matemáticos, sendo ele uma extensão da análise estatística tradicional, mas com a capacidade de processar grandes volumes de dados e lidar com uma complexidade elevada. Para eles, o AM é uma forma de "descoberta de conhecimento", que permite aos sistemas reconhecerem padrões ocultos nos dados e aplicarem esse conhecimento em situações práticas.

Goodfellow (2016) apresenta o processo de aprendizado de um modelo como uma série de técnicas matemáticas e computacionais que buscam aproximar uma função desconhecida f que mapeia as entradas (ou características) X para uma saída Y . O objetivo é estimular uma função $\hat{f}$ que seja a mais próxima possível de f , tal que a diferença entre as saídas previstas e as reais seja mínima. Durante o treinamento, o modelo ajusta seus parâmetros para reduzir a função de erro, que mensura o quão distante as previsões estão dos valores reais. Goodfellow destaca que, além de adaptar-se aos dados, o AM pode aprender representações de alto nível, capturando características complexas dos dados.

Dentro das abordagens de AM, duas das categorias principais são o aprendizado supervisionado e o aprendizado não supervisionado.

2.2.1 Aprendizado supervisionado

O aprendizado supervisionado é uma técnica que se baseia no uso de dados rotulados, ou seja, cada entrada possui uma saída conhecida para, "ensinar" ao modelo a identificar uma relação entre as entradas e saídas. O processo de aprendizado supervisionado envolve dividir a base de dados em conjuntos de treinamento e teste. No conjunto de treinamento, o algoritmo utiliza as entradas e saídas conhecidas para aprender a relação entre elas e, ao passo em que o conjunto de treinamento vai sendo consumido, o algoritmo vai ajustando seus parâmetros para minimizar a diferença entre suas previsões e os rótulos reais (Hastie et al., 2009).

Como explica Hastie et al. (2009), o aprendizado supervisionado é especialmente útil em tarefas de classificação e regressão. Na classificação, o objetivo é categorizar dados em classes discretas. Sebastiani (2002) aponta a utilidade de um sistema de classificação de texto na tarefa de distinguir mensagens “*spam*” e “*não spam*”, com base em características como palavras-chave, frequência de termos e outras variáveis extraídas do texto. A partir de uma grande quantidade de textos já rotulados, o modelo aprende as características específicas de cada rótulo e consegue identificar padrões específicos em novos textos, classificando-os corretamente.

O processo de treinamento supervisionado geralmente envolve algoritmos como regressão linear, árvores de decisão e redes neurais, cada um com suas particularidades. Redes neurais, por exemplo, ajustam pesos nas conexões entre neurônios artificiais para minimizar a função de erro, permitindo que a rede aprenda representações complexas dos dados (Goodfellow, 2016).

2.2.2 Aprendizado não-supervisionado

O aprendizado não supervisionado, diferentemente do supervisionado, trabalha com dados não rotulados. O objetivo é explorar a estrutura oculta dos dados e encontrar padrões, sem qualquer orientação explícita ou rótulos de saída pré-definidos. Segundo Murphy (2012), uma das abordagens comuns no aprendizado não supervisionado é o agrupamento (*clustering*), que organiza os dados em grupos com características semelhantes. Por exemplo, em uma análise de textos, o algoritmo pode agrupar documentos de forma temática, mesmo sem saber previamente quais temas existem nos dados. Essa abordagem é ideal para examinar dados de maneira exploratória, revelando informações ocultas ou *insights* nos dados.

O aprendizado não supervisionado se baseia em algoritmos como o *k-means* e a Análise de Componentes Principais (PCA). No *k-means*, o modelo inicializa uma série de "centros" para cada *cluster* e atribui cada ponto de dados ao centro mais próximo, recalculando os centros até atingir a estabilidade. Na PCA, o objetivo é reduzir a dimensionalidade dos dados, extraindo os componentes principais que mais contribuem para a variação dos

dados (Murphy, 2012). Esses métodos permitem que o algoritmo identifique estruturas e agrupamentos que não seriam evidentes em um contexto supervisionado.

2.2.3 Divisão de Dados e Avaliação de Desempenho

A divisão da base de dados em subconjuntos de treinamento e teste é uma etapa primordial e indispensável no processo de criação de modelos de AM. Através desse processo o modelo pode ser avaliado de maneira confiável, além de tornar-se capaz de generalizar bem para novos dados. Como discutido por Hastie et al. (2009), essa separação permite que o modelo aprenda a partir de um conjunto de dados (treinamento), enquanto o conjunto de teste serve como referência isenta para avaliar a eficácia do modelo em prever novos exemplos.

Ao adotar essa abordagem evita-se o sobreajuste (*overfitting*), que acontece quando o modelo adapta-se excessivamente aos dados de treinamento e torna-se ineficiente em prever novos resultados. Goodfellow (2016) explica que um modelo sobreajustado captura o “ruído” ou as peculiaridades dos dados de treinamento em vez dos padrões subjacentes que são realmente relevantes. Para evitar o sobreajuste, algumas técnicas comuns incluem regularização, *cross-validation*, *early stopping* ou aumentar o volume de dados.

Uma vez que o modelo tenha sido treinado e testado, é necessário avaliar seu desempenho de forma quantitativa. Dentre as métricas disponíveis, a acurácia e o *F1-score* são amplamente utilizadas, cada uma com aplicabilidades específicas.

2.2.3.1 Acurácia

A acurácia mede a proporção de previsões corretas em relação ao total de previsões. Segundo Mohri (2018), a acurácia é calculada como a razão entre o número de previsões corretas e o número total de previsões, representando a porcentagem de acertos do modelo. Em termos matemáticos, ela pode ser expressa pela fórmula:

$$\text{Acurácia} = \frac{\text{Número de previsões corretas}}{\text{Número total de previsões}}$$

Essa métrica é amplamente utilizada quando as classes estão balanceadas, ou seja, possuem uma quantidade relativamente semelhante de amostras. Para bases desbalanceadas a acurácia pode ser ultrajante, uma vez que um modelo que prediz sempre a classe majoritária terá uma alta acurácia sem de fato capturar a classe minoritária. Murphy (2012) destaca que, nesses casos, a acurácia isoladamente pode não ser suficiente para avaliar o desempenho de modelos de classificação.

2.2.3.2 F1-Score

A medida *F1-score*, todavia, é uma métrica que combina precisão (*precision*) e revocação (*recall*) em uma única medida, sendo especialmente útil em situações onde a base

de dados está desbalanceada. Como explica Powers (2020) essa métrica traz a média harmônica entre precisão e *recall*. A precisão é a quantidade de previsões corretas entre todas as previsões que o modelo fez para a classe positiva e o *recall* é a quantidade de acertos entre todas as instâncias que realmente pertencem à classe positiva. A fórmula do *F1-score* é dada por:

$$\mathbf{F1-Score} = \frac{2 \cdot \text{Precisão} \cdot \text{Revocação}}{\text{Precisão} + \text{Revocação}}$$

onde:

□ **Precisão** é dada por:

$$\text{Precisão} = \frac{\text{VP}}{\text{VP} + \text{FP}}$$

Verdadeiros Positivos (VP): Casos em que o modelo previu positivo e o rótulo real também é positivo.

Falsos Positivos (FP): Casos em que o modelo previu positivo, mas o rótulo real é negativo.

□ **Recall** é dado por:

$$\text{Revocação} = \frac{\text{VP}}{\text{VP} + \text{FN}}$$

Verdadeiros Positivos (VP): Casos em que o modelo previu positivo e o rótulo real também é positivo.

Falsos Negativos (FN): Casos em que o modelo previu negativo, mas o rótulo real é positivo.

Sebastiani (2002) realça que o *F1-score* é essencial para tarefas em que a classificação correta de uma classe minoritária é crítica, pois uma alta acurácia geral não é suficiente para garantir que o modelo esteja, de fato, identificando as classes minoritárias de maneira adequada. A precisão contribui para garantir que as previsões feitas pelo modelo para a classe positiva estejam corretas, enquanto o *recall* assegura que a maioria dos exemplos positivos seja identificada. Isso torna a medida F1 uma métrica mais informativa para problemas onde há uma desigualdade significativa entre as classes.

2.2.4 Ajuste de Hiperparâmetros

Para melhorar a performance de modelos de AM, é crucial que os hiperparâmetros estejam ajustados corretamente, pois eles influenciam diretamente como o modelo se adapta aos dados. Hiperparâmetros são parâmetros definidos antes do treinamento, ou seja, valores que o próprio algoritmo de aprendizado não aprende e que devem ser configurados para que o modelo possa generalizar de maneira adequada (Raschka and Mirjalili, 2019).

Eles incluem aspectos como a taxa de aprendizado em redes neurais, a profundidade máxima das árvores em algoritmos de classificação e o número de vizinhos em métodos de classificação por proximidade, entre outros.

O ajuste de parâmetros é uma etapa importante no processo de aprendizado e, quando executado inadequadamente, prejudica a performance do modelo, levando a um sobreajuste (*overfitting*), já discutido na sessão anterior, ou um subajuste (*underfitting*), quando o modelo é muito simples para capturar os padrões dos dados de treinamento (Raschka and Mirjalili, 2019). Encontrar o equilíbrio certo é crucial para criar um modelo que generalize bem para novos dados.

2.2.4.1 Busca em Grade

A busca em grade, ou *Grid Search*, é uma técnica de otimização de hiperparâmetros, onde uma busca exaustiva é executada para testar todas as combinações possíveis de um conjunto pré-determinado de valores para cada hiperparâmetro. Esse método, segundo Raschka and Mirjalili (2019), "é vantajoso por garantir que todas as combinações possíveis serão testadas, assegurando que nenhuma opção será negligenciada".

A técnica envolve definir um conjunto de valores possíveis para cada hiperparâmetro e, em seguida, realizar experimentos com cada combinação desses valores. Neste caso, um modelo é treinado com cada conjunto de hiperparâmetros e avaliado com uma métrica de desempenho, como a acurácia. A combinação que resulta na melhor acurácia é então selecionada como a configuração ideal (Raschka and Mirjalili, 2019). Essa técnica demanda um alto custo computacional, dado que o número de combinações cresce exponencialmente de acordo com o número de hiperparâmetros e seus respectivos valores.

2.2.5 Subamostragem e sobreamostragem

Em tarefas de aprendizado supervisionado, como apontado por Japkowicz and Stephen (2002), o desbalanceamento do conjunto de dados faz com que os algoritmos tendam a favorecer a classe majoritária, o que resulta em vieses preditivos e baixa sensibilidade para identificar a classe minoritária. Isso ocorre porque muitos algoritmos de AM são projetados para maximizar a precisão global, frequentemente resultando em uma tendência a prever a classe mais comum. Esse panorama afeta especialmente métricas como a precisão e medida F1, onde a classificação correta da classe minoritária é importante. Assim, é essencial aplicar técnicas de balanceamento de classes para assegurar que o modelo aprenda com paridade a representação das diferentes classes e, conseqüentemente, ofereça previsões mais confiáveis.

Subamostragem

Uma das técnicas mais comuns para lidar com o desbalanceamento de um conjunto de dados é a subamostragem, ou *undersampling*. Ela reduz a quantidade de exemplos da classe majoritária a fim de equilibrar a distribuição entre as classes. Segundo He and Garcia (2009), "essa técnica reduz a quantidade de amostras na classe mais frequente, permitindo que o modelo tenha um treinamento mais equilibrado". Esse método pode ser útil quando o objetivo é garantir que o modelo não seja excessivamente influenciado pelos exemplos da classe majoritária. Porém, a subamostragem pode acarretar a perda de informações relevantes, já que exemplos valiosos da classe majoritária são descartados e, dependendo do contexto, pode afetar negativamente o desempenho do modelo (He and Garcia, 2009).

A subamostragem é indicada em situações onde o conjunto de dados original é muito extenso, e a perda de dados da classe majoritária não compromete a aprendizagem do modelo. Contudo, quando o volume de dados é limitado, a aplicação dessa técnica requer maior cautela para não prejudicar a performance geral do modelo.

Sobreamostragem

Outra técnica largamente empregada para o balanceamento de dados é a sobreamostragem, ou *oversampling*, que demanda o aumento da quantidade de exemplos na classe minoritária. Esse processo pode ser feito replicando amostras existentes ou gerando novos exemplos sintéticos, como no método *Synthetic Minority Over-sampling Technique* (SMOTE), desenvolvido por Chawla et al. (2002). De acordo com o autor, "o SMOTE cria novos exemplos interpolando dados reais da classe minoritária, o que ajuda a melhorar a capacidade do modelo de reconhecer a classe menos frequente".

A sobreamostragem é especialmente útil em casos onde a quantidade de dados é limitada e a perda de dados pode comprometer a generalização do modelo. Além disso, a criação de exemplos sintéticos permite uma maior variabilidade nos dados da classe minoritária, contribuindo significativamente para com a capacidade do modelo em detectar padrões mais complexos e representativos desta classe. Contudo, como apontado por He and Garcia (2009), o aumento excessivo da classe minoritária pode levar ao sobreajuste (*overfitting*), caso o modelo se torne excessivamente adaptado às amostras sintéticas geradas.

2.2.6 Classificação

A classificação é uma tarefa primordial em AM. O objetivo central de um modelo de classificação é aprender a partir de um conjunto de dados rotulados, reconhecendo padrões e relacionando características dos dados (também chamados de *features* ou atributos) com

classes específicas. Esse aprendizado permite ao modelo fazer previsões sobre a classe de novas amostras desconhecidas por ele, mas que compartilham características relevantes com o conjunto de dados de treinamento (Bishop and Nasrabadi, 2006).

Segundo Hastie et al. (2009), a classificação é definida como o processo de atribuir uma etiqueta de classe Y a uma observação x com base em exemplos anteriores, onde a relação entre x e y é aprendida por um modelo. Para aprender relações e atribuir etiquetas a classe y , o modelo analisa os atributos verificando a similaridade, distância ou distribuição dos mesmos. Esses atributos podem ser qualquer tipo de dado, porém o rótulo da classe precisa ser discreta ou categórica.

O processo de classificação se dá a partir de duas etapas. Na primeira fase, chamada de aprendizado ou fase de treinamento, constrói-se um modelo para analisar um conjunto de dados rotulados montando assim o classificador. Na segunda fase, chamada de classificação, o classificador é executado para prever a classe do objeto desconhecido. Neste contexto, a tarefa de classificação é caracterizada como aprendizado supervisionado, dado que a classe do conjunto de treinamento é conhecida (Mining, 2006).

As próximas sessões descrevem o funcionamento dos 4 algoritmos aplicados neste estudo: *Naive Bayes*, *SVM*, *Random Forest* e *GBM*.

2.2.6.1 Naive Bayes

O *Naive Bayes* é um algoritmo de classificação probabilístico baseado no Teorema de Bayes. A hipótese deste algoritmo é que todas as características são independentes entre si, sendo chamada de independência condicional. Esse algoritmo, quando aplicado a grandes bases de dados, possui resultados de alta precisão e são rápidos em suas respostas (McCallum et al., 1998). A fórmula é dada por:

$$P(C|X) = \frac{P(C) \prod_{i=1}^n P(x_i|C)}{P(X)}$$

onde:

- $P(C|X)$ é a probabilidade da classe C dado o vetor de características X
- $P(C)$ é a probabilidade a priori da classe C
- $P(x_i|C)$ é a probabilidade condicional de x_i dado que a classe é C (assumida independente, o que caracteriza o "naive").
- $P(X)$ é a probabilidade de observar o vetor X , que é um fator de normalização.

A probabilidade dos atributos é calculada para cada uma das classes individualmente e, no fim, realiza-se o produto de todas as probabilidades. A fórmula é dada por:

$$P(C|X) = \frac{P(C) \prod_{i=1}^n P(x_i|C)}{\sum_{C'} P(C') \prod_{i=1}^n P(x_i|C')}$$

Posteriormente executa-se a normalização destes valores, e a classe que possuir a maior probabilidade é atribuída ao objeto.

2.2.6.2 Support Vector Machine (SVM)

O SVM é um algoritmo cujo principal hipótese é que, dadas as características de um conjunto de dados, existe um hiperplano que pode separar as diferentes classes de forma ótima. Portanto, o objetivo do algoritmo é encontrar esse hiperplano, de modo a garantir que a margem entre as classes seja maximizada. Margem é a distância entre o hiperplano e os pontos de dados mais próximos de cada classe, chamados de vetores de suporte. Quanto maior essa margem, mais robusta é a classificação, pois a probabilidade de erro em dados novos tende a ser menor (Cortes, 1995). A fórmula para o modelo SVM é dada por:

$$f(X) = w^T X + b$$

Onde:

- $f(X)$ é a função de decisão que determina de que lado do hiperplano o ponto X se encontra.
- w é o vetor de pesos, que define a direção do hiperplano.
- b é o viés, que controla a posição do hiperplano no espaço.
- X é o vetor de características do ponto de dados.

A otimização do SVM busca minimizar a norma do vetor de pesos w , ou seja, maximizar a margem. A função objetivo do SVM é:

$$\text{minimize} \quad \frac{1}{2} \|w\|^2$$

Com as restrições:

$$y_i(w^T x_i + b) \geq 1, \quad \forall i$$

Onde:

- y_i são os rótulos das classes (+1 ou -1).
- x_i são os pontos de dados de treinamento.

2.2.6.3 Random Forest

Random Forest é um algoritmo baseado em uma coleção de árvores de decisão. Cada árvore é treinada em um subconjunto aleatório dos dados, e as previsões finais são feitas por votação (no caso de classificação) ou média (no caso de regressão) dos resultados de todas as árvores. A principal vantagem do *Random Forest* é sua robustez e capacidade de generalização, o que ajuda a reduzir o risco de *overfitting* em comparação com uma única árvore de decisão (Breiman, 2001).

O algoritmo segue as seguintes fases:

- **Construção das árvores:** Para cada árvore na floresta, um subconjunto aleatório dos dados de treinamento é escolhido (com reposição, o que é chamado de *bootstrap*). Além disso, para cada divisão em cada nó, um subconjunto aleatório de características é selecionado, o que aumenta a diversidade entre as árvores.
- **Predição:** A predição de cada árvore é feita com base nas regras de decisão de cada nó, e a previsão final do modelo é feita por votação (para classificação) ou média (para regressão) das previsões de todas as árvores. A fórmula de predição de uma árvore de decisão é:

$$f(X) = \text{class}(X)$$

Onde:

- $f(X)$ é a classe ou o valor predito para a entrada X .
- $\text{class}(X)$ é a classe ou valor atribuído pela árvore de decisão para a entrada X .

A predição final é dada pela votação (para classificação) dos resultados das árvores:

$$\hat{y} = \text{mode}(f_1(X), f_2(X), \dots, f_m(X))$$

Onde:

- $f_i(X)$ é a predição da i -ésima árvore.
- mode é o número total de árvores na floresta.
- $\hat{y}$ é a predição final.

2.2.6.4 Gradient Boosting Machines (GBM)

GBM é um algoritmo baseado na técnica de *boosting*, que constrói uma sequência de modelos (geralmente árvores de decisão) de forma incremental. Cada novo modelo corrige os erros dos modelos anteriores, ajustando-se ao gradiente do erro em relação à função

objetivo. O principal objetivo do GBM é minimizar a função de perda (por exemplo, erro quadrático médio para regressão ou *log-loss* para classificação) através de uma sequência de predições de modelos simples, combinados de forma ponderada (Friedman, 2001).

O GBM funciona da seguinte maneira:

- ❑ **Modelo inicial:** Começa com uma predição inicial (geralmente a média dos valores, para regressão, ou a classe mais frequente, para classificação).
- ❑ **Iteração:** Em cada iteração, um novo modelo é treinado para prever os resíduos (erros) do modelo anterior, ou seja, as diferenças entre as predições e os valores reais. Esse novo modelo ajusta-se aos erros do modelo anterior.
- ❑ **Atualização:** O modelo final é uma combinação dos modelos anteriores, com uma atualização ponderada.

A predição final do GBM é dada pela soma das predições de todos os modelos gerados no processo:

$$f(X) = F_0 + \sum_{t=1}^T \eta \cdot h_t(X)$$

Onde:

- ❑ $f(X)$ é a predição final para o ponto X .
- ❑ F_0 é a predição inicial (geralmente a média dos valores ou a classe mais frequente).
- ❑ $h_t(X)$ é o modelo (geralmente uma árvore de decisão) na t -ésima iteração.
- ❑ η é a taxa de aprendizado, que controla a contribuição de cada modelo.
- ❑ T é o número total de iterações (número de modelos treinados).

O erro ou resíduo $r_t(X)$ na iteração t é calculado como a diferença entre o valor real e a predição anterior:

$$r_t(X) = y - F_{t-1}(X)$$

Onde:

- ❑ y é o valor real (rótulo ou valor contínuo).
- ❑ $F_{t-1}(X)$ é a predição do modelo até a iteração anterior.

O ajuste incremental dos erros e a combinação ponderada dos modelos tornam o GBM uma técnica poderosa e eficaz para tarefas de classificação e regressão.

2.3 Processamento de Linguagem Natural

O PLN é uma área interdisciplinar da ciência da computação que estuda e desenvolve métodos que viabilizam a interação entre a linguagem de máquina e a linguagem humana, unindo princípios de linguística, inteligência artificial e ciência de dados.

Segundo Jurafsky and Martin (2009), o PLN é um campo que usa algoritmos e modelos matemáticos para permitir que sistemas computacionais compreendam e processem a linguagem natural em várias tarefas como reconhecimento de fala, tradução automática e análise de sentimentos. Esse ponto de vista destaca a importância de combinar técnicas estatísticas e computacionais para modelar a linguagem humana.

A óptica de Church and Mercer (1993) é baseada na probabilidade. O autor propõe que o PLN seja entendido como uma questão de predição baseada em dados reais e na relação estatística presentes nesses dados. Nesse sentido, o PLN explora as probabilidades de ocorrência de palavras e frases para resolver problemas complexos de linguagem através de algoritmos e modelos com *n-grams*.

Já para Allen (1988), o PLN busca modelar o comportamento linguístico humano, de modo a criar sistemas capazes de entender, interpretar e gerar linguagem de forma eficiente, para realizar tarefas complexas como tradução automática, resumo de textos e análise de sentimentos.

O início do PLN, na década de 1950, foi marcado por abordagens simbólicas baseadas em regras manuais e gramáticas formalizadas. Chomsky (2002), em seu trabalho sobre a Gramática Transformacional, propôs um modelo matemático para descrever a estrutura das línguas naturais, influenciando fortemente os primeiros sistemas de PLN, que utilizavam regras sintáticas rígidas para processar sentenças. Posteriormente, nas décadas de 1980 e 1990, com o avanço tecnológico e o crescimento de grandes bases de dados, as abordagens estatísticas começaram a emergir e os métodos baseados em AM e modelos probabilísticos, como os *n-gramas* de Shannon (1948), permitiram uma maior flexibilidade no tratamento da ambiguidade da linguagem humana.

O avanço da computação suscitou uma nova fase ao PLN, marcada pelas técnicas de aprendizado profundo (*deep learning*). Modelos neurais como *Long Short-Term Memory* (LSTM) (Memória de Longo e Curto Prazo) (Hochreiter, 1997), o Transformer (Vaswani, 2017) e o *Bidirectional Encoder Representations from Transformers* (BERT) (Representações de Codificadores Bidirecionais a partir de Transformers) (et al., 2019) tornaram-se o estado da arte, permitindo que os sistemas não apenas entendessem a estrutura sintática das sentenças, mas também capturassem nuances semânticas e contextuais complexas.

Conforme destacado por Manning and Schutze (1999), "o PLN tornou-se uma das áreas mais dinâmicas da IA, com aplicações que vão desde sistemas de atendimento automatizado até assistentes virtuais e análise de dados em mídias sociais". Essas inovações têm sido fundamentais para a aplicação do PLN em tarefas mais sofisticadas, como a detecção

de discurso ofensivo em redes sociais, o que será abordado ao longo deste trabalho.

2.3.1 Principais Abordagens de Processamento de Linguagem Natural

A evolução do PLN ao longo dos anos se deu por meio de três principais abordagens: a simbólica, a estatística e, mais recentemente, a abordagem baseada em aprendizado profundo.

2.3.1.1 Abordagens Simbólicas

As abordagens simbólicas baseiam-se na criação manual de regras gramaticais e lógicas para descrever a estrutura sintática e semântica da linguagem. De acordo com essa abordagem, a compreensão da linguagem é possível por meio de regras formais, implementadas em sistemas de análise linguística. O trabalho de Chomsky (2002), por meio de sua teoria da Gramática Transformacional, propôs que a linguagem humana pode ser descrita matematicamente. Ele introduziu a ideia de que as sentenças possuem duas estruturas: uma estrutura profunda, de natureza conceitual, e uma estrutura superficial, a forma que a frase assume em uma língua, conectadas por um conjunto de regras de transformação.

De acordo com Jurafsky and Martin (2000), os sistemas simbólicos funcionam através das seguintes etapas:

- ❑ **Análise sintática:** faz uso de gramáticas formais para analisar a estrutura das frases. Essa fase mapeia a relação entre os diferentes componentes de uma sentença, como sujeito, verbo e objeto.
- ❑ **Análise semântica:** a partir da análise sintática, essa fase aplica regras para interpretar o significado da frase, identificando entidades, relações e atributos e convertendo o conceito sintático em uma estrutura semântica compreensível.
- ❑ **Inferência Lógica:** faz uso de regras lógicas, proposicionais ou de primeira ordem, para fazer inferências a partir das informações já disponíveis. Dessa forma, é possível extrair informações que não estão explicitamente mencionadas no texto.
- ❑ **Geração de Linguagem:** por fim, é possível gerar respostas ou frases em linguagem natural aplicando as regras definidas para assegurar que o resultado esteja de acordo com as normas gramaticais da língua.

Winograd (1972), em 1972, desenvolveu o sistema SHRDLU, que buscava "compreender" a linguagem natural dentro de ambientes restritos, utilizando regras gramaticais rígidas. O SHRDLU operava em um "micromundo" limitado, onde objetos geométricos podiam ser manipulados e descritos em linguagem natural. O sistema era capaz de interpretar comandos ao aplicar regras sintáticas e semânticas previamente definidas.

A abordagem simbólica é amplamente utilizada em tarefas como tradução automática, que emprega gramáticas específicas; assistentes virtuais, que utilizam técnicas simbólicas para interpretar a intenção do usuário; e análise de sentimentos, que se baseia em regras que definem a relação entre palavras e suas significações (Jurafsky and Martin, 2000). Apesar de ter contribuído significativamente para o avanço do PLN, esse sistema era incapaz de lidar com a ambiguidade da linguagem humana, como polissemia, ambiguidade estrutural e variabilidade contextual. Além disso, a manutenção e a expansão de regras manuais para cobrir todos os casos possíveis revelaram-se impraticáveis, dada a dinamicidade da linguagem, que não pode ser antecipada por regras fixas (Jurafsky and Martin, 2000).

2.3.1.2 Abordagens Estatísticas

O aumento do poder computacional e a oferta de grandes bases de dados textuais viabilizaram o avanço das abordagens estatísticas, que utilizam métodos probabilísticos e de AM para processar a linguagem. Nessa abordagem, as máquinas aprendem automaticamente, capturando padrões e regularidades da linguagem a partir de grandes volumes de textos. Sob o olhar estatístico, a linguagem é tratada como um conjunto de eventos probabilísticos, onde a frequência de palavras, frases e suas ocorrências pode ser utilizada para inferir significados e relações (Jurafsky and Martin, 2000).

Claude Shannon, considerado o fundador da teoria da informação, introduziu conceitos de entropia, redundância e compressão, aplicando-os ao contexto de análise de linguagem natural. A entropia mede a instabilidade de uma fonte de informação, sendo fundamental para o processo de modelagem de linguagem. Através dela os modelos estatísticos conseguem atribuir probabilidades às sequências de palavras com base em sua frequência observada (Shannon, 1948). A teoria de Shannon é aplicada em modelos de *n-gram*, que representam sequências de n palavras e calculam a probabilidade de cada palavra com base nas palavras anteriores. Por exemplo, em um modelo de trigramas, a probabilidade de uma palavra w_i é condicionada às duas palavras anteriores (w_{i-2}, w_{i-1}). Isso é formalizado como $P(w_i \mid w_{i-2}, w_{i-1})$. O trabalho de Shannon estabeleceu a base matemática para essa estrutura probabilística, permitindo uma modelagem mais precisa e prática do comportamento linguístico.

Os modelos probabilísticos também são utilizados no campo da tradução automática, servindo para estabelecer probabilidades de tradução entre diversas línguas. A equipe de Brown et al. (1993) fez contribuições significativas nesse sentido, criando o “modelo ruído-canal”, que é formalizado pela equação $P(e \mid f) = P(f \mid e) \cdot P(e)$, onde e representa a sequência de palavras na língua alvo, e f refere-se à sequência na língua fonte. O conceito é tratar o processo de tradução como um problema de maximização de probabilidade, onde, a partir de um texto em uma língua fonte, busca-se identificar a tradução mais provável em uma língua alvo. Sendo uma base fundamental para a criação de sistemas

de tradução, esse modelo combina uma distribuição de probabilidade sobre palavras na língua fonte (modelo de linguagem) com uma probabilidade condicional para pares de palavras ou expressões equivalentes (Brown et al., 1993).

A abordagem estatística no PLN depende de técnicas de AM que incluem:

- ❑ **Modelagem Probabilística de Linguagem:** modelos de linguagem estatísticos processam a probabilidade de uma sequência de palavras ocorrer em determinado contexto. Os modelos *n-gram*, por exemplo, calculam a probabilidade de uma palavra ser precedida por $n - 1$ palavras, usando uma função de probabilidade condicionada. Essa abordagem é amplamente descrita e utilizada em PLN, como detalhado por Jurafsky and Martin (2000).
- ❑ **Métodos de Inferência Estatística e Algoritmos de Aprendizado de Máquina:** técnicas de aprendizado supervisionado e não supervisionado, como árvores de decisão, regressão logística, SVM e redes neurais, permitem a construção de modelos que generalizam a partir dos dados. Na classificação de texto, por exemplo, o SVM é usado para categorizar documentos em classes predefinidas, como no reconhecimento de sentimentos ou na filtragem de spam (Learning, 1997).
- ❑ **Tradução Automática e Extração de Informação:** Modelos estatísticos avançados são aplicados em tarefas complexas como a tradução automática e a extração de informações de textos. A tradução automática, conforme abordada por Brown et al. (1993), depende de grandes bases bilíngues, que permitem aos sistemas "aprender" as probabilidades de mapeamento entre diferentes línguas. A extração de informações, por outro lado, utiliza modelos estatísticos para identificar e colher entidades e fatos de textos, aplicando técnicas de classificação e *clustering* para interpretar automaticamente o conteúdo.

2.3.1.3 Abordagem Baseada em Redes Neurais

As técnicas de AM, especialmente as Redes Neurais Artificiais (RNA), permitem que modelos de linguagem aprendam padrões linguísticos diretamente a partir de grandes conjuntos de dados, eliminando a necessidade de definir manualmente regras formais para todos os aspectos da linguagem. Como resultado, as redes neurais conseguem capturar nuances e complexidades na interpretação de dados linguísticos com precisão, modelando padrões complexos com fluidez e adaptatividade (Jurafsky and Martin, 2000), (Goldberg, 2016).

As RNA são modelos computacionais que se inspiram na estrutura e funcionamento do cérebro humano. São compostas por camadas de neurônios artificiais, organizadas para transformar, aprender e transmitir informações ao longo de camadas de processamento, possibilitando a modelagem de relações complexas entre as palavras e o contexto (Good-

fellow, 2016). No PLN, as RNAs aplicam-se a várias arquiteturas, cada uma otimizada para diferentes tarefas linguísticas:

- ❑ **Redes Neurais Recorrente (RNN):** as RNN são programadas especificamente para lidar com dados sequenciais, o que as torna ideais para modelar padrões de linguagem em que a ordem das palavras afeta diretamente o significado. Essas redes mantêm informações de contexto ao longo da sequência, armazenando estados ocultos que representam palavras anteriores na cadeia de entrada. RNN evoluíram para incluir unidades como LSTM e *Gated Recurrent Units* (GRU) (Unidades Recorrentes com Portas), que melhoram a capacidade da rede de reter informações de longo prazo (Hochreiter, 1997), (Sutskever et al., 2014).
- ❑ **Redes Neurais Convolucionais (CNN):** embora CNN sejam amplamente utilizadas para tarefas de visão computacional, elas também são aplicadas no PLN, em particular para classificação de texto e extração de características de *n-grams* em dados linguísticos (Kim, 2014).
- ❑ **Transformadores e Modelos de Atenção:** os transformadores introduzem um mecanismo de atenção que permite ao modelo focar em diferentes partes da entrada de maneira flexível e adaptativa. Esse mecanismo possibilita que o transformador capture dependências de longo alcance sem a limitação de sequências lineares das RNN. Modelos como BERT e *Generative Pre-trained Transformer* (GPT) (Transformador Pré-treinado Generativo), baseados em transformadores, são amplamente utilizados para tarefas como tradução e compreensão de textos, mostrando alta eficácia em capturar contexto bidirecional (Vaswani, 2017), (et al., 2019).

As RNAs se destacam principalmente em tarefas como:

- ❑ **Classificação de Sentimentos:** modelos como RNN e CNN têm sido aplicados para analisar sentimentos, usando grandes conjuntos de dados anotados para que o modelo aprenda a associar palavras e frases com emoções específicas. A eficácia dessa abordagem foi demonstrada por Tang et al. (2015), que combinaram LSTM com atenção para capturar contexto e nuances em tarefas de análise de sentimentos.
- ❑ **Tradução Automática Neural:** com o advento dos transformadores, a *Neural Machine Translation* (NMT) (Tradução Automática Neural) tornou-se mais eficiente e precisa. Ao contrário dos métodos estatísticos, os modelos NMT, como o modelo *seq2seq*, utilizam codificadores e decodificadores para processar e traduzir texto, aprendendo diretamente de grandes conjuntos de dados bilíngues (Vaswani, 2017).
- ❑ **Modelos de Linguagem Pré-Treinados:** modelos como BERT e GPT são treinados em grandes volumes de dados para aprender o contexto bidirecional de uma

palavra em uma frase. No caso do BERT, a técnica de pré-treinamento permite que ele capture uma variedade de padrões e aplicabilidades, o que o torna eficaz em tarefas como perguntas e respostas e geração de texto (et al., 2019).

As três abordagens – simbólica, estatística e RNA – refletem a evolução do PLN . Conforme ressaltado por Jurafsky and Martin (2009), "cada abordagem contribuiu de maneira fundamental para o avanço do PLN , sendo que as técnicas modernas frequentemente combinam elementos de diferentes escolas de pensamento para alcançar resultados mais robustos". Hoje, a maioria das aplicações de PLN utilizam uma combinação de aprendizado supervisionado, modelos neurais e grandes bases de dados anotados, consolidando a evolução das abordagens anteriores.

2.3.2 Ferramentas e Técnicas de Pré-Processamento

O pré-processamento de texto é indispensável para garantir a qualidade e consistência dos dados de entrada em modelos de PLN , pois prepara os dados textuais para serem manipulados e analisados por modelos de AM. Esse processo reduz a complexidade e a variabilidade linguística, além de manter as informações mais relevantes, o que refina o desempenho e a precisão dos modelos de PLN . Técnicas como *Tokenização*, *Limpeza* e *Normalização de Texto*, *Stemming* e *Lematização* transformam dados brutos em um formato estruturado e homogêneo, facilitando a extração de padrões e a modelagem de informações.

2.3.2.1 Tokenização

A Tokenização é uma técnica que fragmenta um texto em menores unidades de análise chamadas tokens, que podem ser palavras, sentenças ou até mesmo caracteres, dependendo da granularidade necessária. Esse processo permite que os algoritmos identifiquem e manipulem cada elemento linguístico individualmente, facilitando a análise sintática, semântica e estatística. No PLN , o texto T é frequentemente representado como um conjunto de tokens $\{t_1, t_2, \dots, t_n\}$, onde cada t_i representa um termo individual no texto (Jurafsky and Martin, 2000). Esse processo simplifica a análise, especialmente em métodos de contagem ou vetorização, onde cada token é tratado como uma unidade estatística. Isso é particularmente útil em modelos baseados em *n-grams*, onde combinações de tokens são analisadas para capturar contextos específicos, como em traduções automáticas e análise de sentimentos. Esse processo permite que o modelo capture a sequência e o relacionamento entre as palavras, o que é essencial em tarefas de PLN (Jurafsky and Martin, 2000).

2.3.2.2 Limpeza e Normalização de Texto

Limpeza e normalização de texto são processos que garantem a padronização do conteúdo textual, removendo caracteres e estruturas desnecessárias como pontuações, números e caracteres especiais, enquanto transforma o texto para uma forma padronizada, convertendo o texto para letras minúsculas. Esse processo minimiza a variabilidade lexical que pode surgir de diferentes formas de escrita para a mesma palavra ou expressão, o que pode prejudicar a identificação de padrões. Por exemplo, palavras como “Vovô” e “vovô” seriam normalizadas para uma única representação: “vovo”. Isso evita que o modelo trate essas palavras como entidades diferentes, aumentando a consistência na análise dos dados. Ademais, a limpeza pode envolver a remoção de palavras irrelevantes, conhecidas como *stopwords* (como “e”, “de”, “o”) que, em muitos casos, não carregam informações semânticas relevantes e podem ser removidas para reduzir o ruído nos dados (Manning and Schütze, 2008).

2.3.2.3 Stemming e Lematização

O *stemming* e a lematização são técnicas que têm como foco a flexibilização de palavras a uma raiz básica comum. Porém, enquanto o *stemming* reduz à morfologia, a lematização reduz ao lema.

Stemming é a técnica que reduz a palavra a sua raiz ou base morfológica, removendo sufixos e prefixos, sem se preocupar com a correção gramatical. Por exemplo, as palavras “trabalho”, “trabalhando” e “trabalhou” são abreviadas a uma mesma base “trabalh” que, embora não seja uma palavra real, ainda representa um mesmo conceito. Este processo é realizado por meio de algoritmos, como o de Porter (1980), amplamente utilizado no PLN. O *stemming* é empregado em tarefas onde a fidelidade semântica de uma palavra individual não é crítica, como na recuperação de informações, que combina termos semelhantes para aprimorar a abrangência de resultados. Para tarefas que demandam precisão semântica, o *stemming* não é recomendável pois pode gerar ruídos, dado que palavras morfologicamente semelhantes podem ter significados diferentes (Manning and Schütze, 2008).

A lematização é uma técnica mais avançada e precisa, que reduz a palavra à sua forma dicionário ou lema, considerando tanto a análise morfológica quanto a estrutura sintática. Diferente do *stemming*, que executa truncamentos, a lematização identifica a forma correta da palavra com base no contexto para garantir que o sentido semântico seja preservado (Jurafsky and Martin, 2000). Como exemplo, temos os verbos “trabalhando”, “trabalhado”, “trabalharão”, que são normalizadas para “trabalhar”. A lematização é muito importante em tarefas como a análise de sentimentos e tradução automática, onde a precisão semântica é crucial, pois ela permite que o modelo capture melhor os significados subjacentes e as relações entre as palavras. Embora seja computacionalmente

mais complexa, a técnica é mais precisa para identificar o significado real de uma palavra dentro de uma frase (Jurafsky and Martin, 2000).

2.3.2.4 Ferramentas e Técnicas de Vetorização de Texto

A vetorização de texto é uma etapa fundamental do PLN . Essa técnica possibilita que o texto seja transformado em uma representação numérica que pode ser manipulada por algoritmos de AM. O PLN necessita identificar as palavras, suas relações e contextos. Nesse sentido, a vetorização funciona como uma etapa essencial ao registrar informações relevantes da linguagem de forma quantitativa, facilitando o processamento em modelos baseados em AM. Essas representações numéricas são cruciais para algoritmos que buscam vincular padrões linguísticos a respostas específicas, como na recuperação de informações, na detecção de *spam* ou em recomendações automáticas (Jurafsky and Martin, 2000; Manning and Schütze, 2008).

Técnicas clássicas de vetorização incluem o *CountVectorizer* e o TF-IDF, que transformam texto em vetores que representam a importância relativa de cada termo dentro de um documento ou conjunto de documentos. Elas permitem que os algoritmos captem relações e padrões no texto, facilitando a análise semântica e a compreensão contextual, com implicações diretas em tarefas como análise de sentimentos, classificação de documentos e sistemas de recomendação (Jurafsky and Martin, 2000; Manning and Schütze, 2008; Ramos et al., 2003).

CountVectorizer

O *CountVectorizer* cria um vetor com base na contagem das ocorrências de cada termo no documento. Essa abordagem cria um vetor onde cada termo corresponde a uma posição específica e o valor na posição indica quantas vezes o termo aparece no documento, sem ponderar sua relevância relativa em relação a outros documentos. Isso resulta em uma matriz esparsa, onde cada linha representa um documento e cada coluna um termo, permitindo identificar termos frequentes e padrões básicos no texto (Manning and Schütze, 2008). Formalmente, seja D um conjunto de documentos e T o vocabulário total, o vetor de contagem v_d de um documento d terá uma dimensão $|T|$ onde cada elemento v_{d_i} é dado por:

$$v_{d_i} = \text{count}(t_i, d)$$

onde t_i é o termo no vocabulário T e $\text{count}(t_i, d)$ é o número de ocorrências de t_i em d . Essa técnica é amplamente aplicada em tarefas de PLN , como classificação de documentos e análise de sentimentos (Manning and Schütze, 2008; Jurafsky and Martin, 2000).

TF-IDF

O TF-IDF aprimora o *CountVectorizer* ao considerar não apenas a frequência de um termo em um documento, mas também a frequência do termo em todo o conjunto de documentos, permitindo reduzir a influência de termos muito comuns, como "e" ou "de", que carregam pouca informação contextual. O peso TF-IDF de um termo t em um documento d é definido por:

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \cdot \log \frac{|D|}{\text{DF}(t)}$$

onde $\text{TF}(t, d)$ é a frequência do termo t no documento d , $|D|$ é o número total de documentos e $\text{DF}(t)$ é o número de documentos em que t aparece. O TF-IDF é particularmente útil em sistemas de recomendação, motores de busca e tarefas de recuperação de informações, onde a relevância de um termo específico para um documento deve ser ponderada (Ramos et al., 2003; Manning and Schütze, 2008; Jurafsky and Martin, 2000).

A combinação de técnicas de pré-processamento de texto e vetorização oferece uma base sólida para diversas aplicações em PLN. O pré-processamento estrutura o texto, enquanto a vetorização converte essa estrutura em representações que podem ser interpretadas por modelos de AM.

2.4 Ferramentas Utilizadas

Esta sessão aborda as ferramentas utilizadas ao longo do desenvolvimento deste trabalho. Foram utilizados a *API YouTube Data* para coleta de dados, o ambiente *Google Colab* e a biblioteca *Scikit-learn* para o processo de modelagem e implementação do sistema de AM. A seguir está a descrição detalhada dessas ferramentas, suas funcionalidades e os benefícios de sua aplicação neste trabalho.

2.4.1 API YouTube Data

A *API YouTube Data* é uma interface de programação de aplicativos desenvolvida pelo *Google* que permite aos desenvolvedores acessar e gerenciar conteúdos do *YouTube* de forma programática. Esta API permite, por exemplo, a obtenção de dados de vídeos, canais, playlists e, especialmente relevante para este trabalho, os comentários em vídeos do *YouTube*. Por meio da API, é possível extrair informações como o conteúdo textual dos comentários, identificadores de usuários, datas de postagem e metadados que auxiliam na análise de conteúdo e comportamento. A API é acessada via requisições HTTP e está disponível em várias bibliotecas cliente, como *Python*, facilitando sua integração em projetos de AM e PLN. A API permite a automação de processos de coleta em grande escala, facilitando a obtenção de amostras substanciais e variadas de dados, além da

possibilidade de configurar filtros e parâmetros de pesquisa, o que permite uma coleta de dados personalizada e orientada para as necessidades específicas. Essas funcionalidades são fundamentais para o tratamento inicial dos dados e para garantir que o conjunto de dados resultante seja de alta qualidade para o treinamento de modelos de detecção de discurso ofensivo (YouTube Developers, 2023).

2.4.2 Google Colab

O *Google Colaboratory*, conhecido como *Google Colab*, é um ambiente de desenvolvimento integrado que permite a execução de código *Python* diretamente em um navegador. Desenvolvido pelo *Google*, ele fornece uma estrutura para o desenvolvimento de projetos que exigem alto poder computacional, como tarefas de AM e PLN. Entre seus principais recursos, o ambiente disponibiliza Unidades de Processamento Gráfico (GPUs) e Unidades de Processamento Tensorial (TPUs), permitindo a execução de modelos de aprendizado profundo em alta velocidade (Bisong, 2019). Adicionalmente, ele é integrado ao *Google Drive*, facilitando o armazenamento e a manipulação de arquivos. O uso do *Google Colab* é vantajoso, pois elimina a necessidade de recursos locais robustos, oferecendo a possibilidade de treinar modelos em grandes conjuntos de dados sem custo adicional. Além disso, o *Google Colab* é altamente acessível e colaborativo, permitindo que múltiplos usuários acessem o mesmo *notebook*, uma característica que facilita a troca de informações e acelera o processo de desenvolvimento.

2.4.3 Biblioteca Scikit-learn

O *Scikit-learn* é uma biblioteca de aprendizado de máquina em *Python* que fornece uma ampla gama de ferramentas para modelagem preditiva. Ela é conhecida por sua interface simples e intuitiva, o que a torna acessível tanto para iniciantes quanto para pesquisadores experientes (Pedregosa et al., 2011). Desenvolvida para trabalhar com os dados processados no ambiente *Python*, o *Scikit-learn* oferece uma série de algoritmos de classificação, regressão e *clustering*, que são amplamente usados na área de aprendizado de máquina. Entre as funcionalidades disponíveis, destacam-se as ferramentas de pré-processamento de dados, como tokenização e vetorização, que são essenciais para o tratamento de dados textuais. A biblioteca também oferece métodos para ajuste de hiperparâmetros, permitindo otimizar o desempenho dos modelos, e métricas para avaliação, como precisão e *F1-score*, essenciais para validar a eficácia do sistema desenvolvido.

2.5 Trabalhos Correlatos

Fortuna and Nunes (2018) fazem uma extensa revisão das abordagens automatizadas para a detecção de discurso de ódio em textos, organizando, analisando e comparando

métodos existentes de PLN e AM. Os autores estruturam a revisão em categorias de métodos e técnicas, cobrindo desde técnicas mais tradicionais até métodos mais avançados de *deep learning*. O estudo classifica as abordagens de detecção de discurso de ódio em três categorias principais: Métodos Baseados em Regras, Métodos de Aprendizado Supervisionado e Métodos de Deep Learning. O estudo traz os diversos desafios associados à detecção de discurso de ódio, incluindo: Ambiguidade Linguística (palavras ou expressões que podem ser ofensivas em certos contextos, mas em outros não); Falta de Bases de Dados Padronizadas (ausência de bases de dados de qualidade e rotuladas especificamente para diferentes idiomas e contextos culturais) e Diversidade Linguística e Cultural (adaptação de modelos para diferentes idiomas e culturas exige o desenvolvimento de abordagens específicas). Os autores apostam em abordagens híbridas que combinem regras com AM e em técnicas mais robustas para lidar com ambiguidades e regionalismos.

Em Muneer and Fati (2020), os autores compararam diversas técnicas de aprendizado de máquina aplicadas à detecção de cyberbullying na plataforma *Twitter*, visando identificar quais métodos são mais eficazes para classificar *tweets* como *cyberbullying* ou não. Foi utilizada uma base de dados composta por *tweets* manualmente anotados. Os dados passaram por etapas de limpeza, incluindo remoção de *stopwords*, normalização de texto, tokenização e vetorização utilizando o TF-IDF. Os algoritmos utilizados foram Naive Bayes, SVM, Random Forest, Logistic Regression e *K-Nearest Neighbors* (KNN). O SVM e Random Forest demonstraram os melhores resultados em termos de acurácia e F1, superando os demais métodos. A combinação de técnicas de pré-processamento e métodos robustos, como SVM, mostrou-se eficaz para lidar com a variação linguística e a natureza ambígua de muitos *tweets*. Os autores destacaram a importância de técnicas de pré-processamento e a necessidade contínua de aprimorar os algoritmos para acompanhar as mudanças linguísticas e contextuais das redes sociais.

O trabalho realizado por Guide (2022) desenvolveu e avaliou modelos de aprendizado de máquina capazes de identificar automaticamente discursos de ódio com teor punitivista em postagens de redes sociais. Foi compilado um corpus específico denominado Corpus de Discurso de Ódio Punitivista (DOP), contendo postagens de redes sociais brasileiras anotadas quanto à presença de discurso de ódio punitivista. Foram testados diversos modelos, incluindo o Gradient Boosting (XGB) e o BERT específico para detecção de discurso de ódio em português. O modelo XGB alcançou uma média F1 de 0,76, superando o modelo BERT, que obteve uma F1 de 0,49. Observou-se que o discurso de ódio punitivista apresenta comportamentos similares a outros tipos de discurso de ódio já estudados. Foi possível esboçar uma tipologia e identificar argumentos base recorrentes nesse tipo de discurso. O estudo concluiu que é viável detectar automaticamente discursos de ódio punitivistas em redes sociais utilizando modelos de aprendizado de máquina. A pesquisa também destacou a importância de monitorar e identificar comportamentos que incentivam ódio e violência nas redes sociais, contribuindo para o combate à disseminação

de conteúdos tóxicos e agressivos.

O estudo de Pelle and Moreira (2017) abordou a criação de um conjunto de dados anotados de comentários ofensivos em português, coletados de notícias na web brasileira, e apresentou resultados iniciais de classificação utilizando algoritmos padrão. Os comentários foram extraídos de notícias das seções de política e esportes do site G1, devido à alta incidência de comentários ofensivos nessas categorias. Três juízes independentes classificaram os comentários como ofensivos ou não, considerando categorias como racismo, sexismo, homofobia, xenofobia, intolerância religiosa e uso de palavrões. Foram criadas duas versões do conjunto de dados: OFFCOMBR-2, que incluiu comentários com concordância de pelo menos dois juízes, e OFFCOMBR-3, que incluiu apenas comentários com concordância unânime dos três juízes. Algoritmos de classificação padrão foram aplicados aos conjuntos de dados e apresentaram desempenho variável, servindo como referência para trabalhos futuros na detecção de discurso ofensivo em português. O estudo destacou a viabilidade de criar conjuntos de dados anotados para a detecção de comentários ofensivos em português e estabeleceu uma base para o desenvolvimento de modelos automatizados de filtragem de conteúdo ofensivo na web brasileira.

O artigo de Pitsilis et al. (2018) investiga o uso de redes neurais profundas para identificar linguagem ofensiva na rede social *Twitter*. Os autores propõem um modelo de aprendizado profundo que utiliza RNN com LSTM para capturar o contexto das palavras em mensagens curtas e complexas, como *tweets*. O modelo foi treinado com um grande conjunto de dados de *tweets* rotulados como ofensivos ou não, permitindo a aprendizagem de padrões linguísticos associados a ofensas. O modelo LSTM alcançou uma precisão de cerca de 90% na detecção de linguagem ofensiva em *tweets*, superando métodos tradicionais como SVM e *Naive Bayes*, que ficaram na faixa de 70% a 80%. O estudo destaca o potencial das redes neurais profundas para moderação automática de conteúdo, contribuindo para a segurança e o bem-estar dos usuários em plataformas de redes sociais.

As abordagens exploradas nos artigos apresentam desafios comuns, como lidar com as nuances linguísticas, abreviações e gírias específicas das redes sociais, fatores que tornam a tarefa de detecção automática ainda mais complexa. Outro ponto em destaque é a importância de conjuntos de dados de alta qualidade, essenciais para o desempenho eficaz dos modelos de aprendizado de máquina. Inspirado por essas pesquisas, este trabalho segue uma trajetória semelhante ao focar na detecção de comentários ofensivos em português brasileiro no *YouTube*. Para isso, utiliza uma base de dados com comentários rotulados como ofensivos ou não ofensivos, buscando contribuir para o avanço do estado da arte no tema.

Materiais em Métodos

Este capítulo descreve o conjunto de dados empregado, desde a coleta e pré-processamento até a rotulação dos dados e análise exploratória. Além disso, detalha o desenho experimental e as estratégias de treinamento dos classificadores, incluindo a divisão dos dados, as técnicas de vetorização e os algoritmos empregados.

3.1 Conjunto de Dados

A coleta dos dados foi realizada por meio da *API YouTube Data*, com o objetivo de construir um conjunto representativo de comentários em língua portuguesa, oriundos de vídeos do *YouTube* Brasil. Foram selecionados cinco vídeos de diferentes temas, escolhidos pela sua grande repercussão e pelo fato de serem amplamente conhecidos e polêmicos. Justamente por serem vídeos de alta visibilidade, eles atraem um grande volume de comentários, incluindo um número expressivo de comentários negativos e ofensivos, o que enriquece a análise. De cada vídeo, foram extraídos 1000 comentários, totalizando 5000 mil comentários. Para garantir a qualidade e a relevância dos dados textuais, uma série de etapas de pré-processamento foram executadas, visando reduzir o ruído e preservar a estrutura linguística dos comentários:

- ❑ Remoção de caracteres especiais: eliminando *emojis*, *links* e símbolos para focar no conteúdo textual.
- ❑ Filtragem de onomatopeias: expressões como ‘kkkk’, ‘haha’, e ‘ksks’ foram removidas.
- ❑ Exclusão de nomes próprios: com o intuito de evitar vieses que poderiam afetar o modelo.
- ❑ Filtragem de idioma: apenas os comentários em português brasileiro foram mantidos.

- ❑ Eliminação de texto não inteligível: exemplos como ‘hdjk’ foram descartados.
- ❑ Conversão para minúsculas: padronizando o texto para melhorar a consistência dos dados.

Após o pré-processamento, a base de dados final passou a ser composta por 4139 comentários, mantendo uma amostra relevante e tratada para a análise.

Os comentários foram rotulados manualmente, com base em uma série de critérios que representam comportamentos ofensivos e inadequados. A cada comentário, foi atribuída uma classe binária, onde 1 indica comentário ofensivo e 0 indica comentário não ofensivo. Os critérios de rotulação incluem:

- ❑ Linguagem inapropriada
- ❑ Discurso de ódio
- ❑ Ameaças
- ❑ Assédio ou *bullying*
- ❑ Conteúdo discriminatório
- ❑ Desrespeito
- ❑ Incitação à violência
- ❑ Intenções maliciosas
- ❑ Expressão excessivamente negativa

3.2 Análise Exploratória dos Dados

A análise exploratória foi conduzida no *Google Colab*, utilizando a biblioteca *Spacy*. O conjunto de dados é composto por 4139 documentos, dos quais 2683 (cerca de 64,82%) foram classificados como não ofensivos e 1456 (aproximadamente 35,18%) como ofensivos. Esse desbalanceamento é um aspecto relevante para a modelagem, pois demanda técnicas de ajuste para evitar viés em favor da classe majoritária (não ofensivo). A análise exploratória indicou um total de 48495 palavras no conjunto de dados, com uma média de 12 palavras por documento.

A Tabela 1 traz alguns exemplos de comentários presentes na base de dados e suas respectivas classes.

A Figura 3 traz um gráfico de n-grams, com os 3-grams mais frequentes na base de dados.

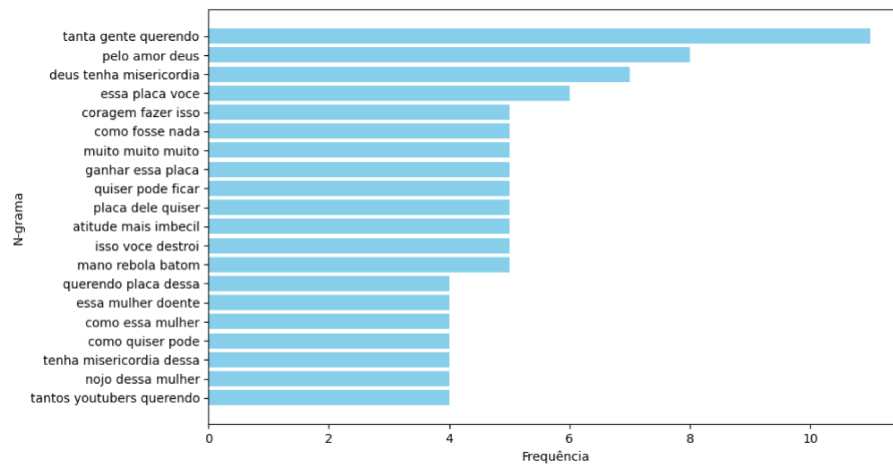


Figura 3 – Gráfico dos 3-grams mais comuns.

3.3 Design Experimental

Os experimentos foram realizados no *Google Colab*, que oferece recursos computacionais adequados para o processamento e treinamento dos modelos. A biblioteca *Scikit-learn* foi utilizada por seu robusto conjunto de algoritmos de AM e ferramentas para pré-processamento e avaliação de modelos.

Para transformar os textos em representações numéricas, foram utilizadas duas técnicas de vetorização: *CountVectorizer* e TF-IDF. Ambas as abordagens são amplamente aplicadas em tarefas de PLN e AM. Essa abordagem permite avaliar o impacto de cada técnica no desempenho dos classificadores.

O conjunto de dados foi dividido em treinamento e teste, com 80% dos dados (3311 documentos) para treinamento e 20% (828 documentos) para teste. Esse particionamento visa a generalização dos modelos, reduzindo a chance de sobreajuste.

Os algoritmos selecionados foram *Naive Bayes*, SVM, *Random Forest* e GBM. Esses métodos foram escolhidos por sua eficácia em problemas de aprendizado supervisionado e em tarefas de PLN.

As métricas empregadas para avaliar o desempenho dos modelos foram a acurácia e a medida F1. A acurácia mede a proporção de previsões corretas, enquanto a medida F1 considera o equilíbrio entre precisão e sensibilidade, sendo particularmente relevante para cenários de desbalanceamento de classes.

3.4 Etapas de Treinamento dos Classificadores

O treinamento foi realizado em quatro etapas distintas, variando o ajuste de hiperparâmetros e a técnica de balanceamento:

- ❑ **Treinamento Padrão:** (Etapa 1) Na primeira fase de treinamento, os algoritmos foram executados em sua forma padrão, ou seja, sem ajuste ou otimização explícita de seus hiperparâmetros.
- ❑ **Busca em Grade:** (Etapa 2) Na segunda etapa, foi realizada uma busca em grade para otimização dos hiperparâmetros. Esta técnica envolve testar todas as combinações de valores de hiperparâmetros pré-definidos para identificar a configuração que maximize, no caso deste trabalho, a acurácia. Ainda nessa fase os melhores hiperparâmetros encontrados foram aplicados no treinamento.
- ❑ **Subamostragem (Undersampling):** (Etapa 3) Na terceira fase, os algoritmos foram reajustados com os hiperparâmetros encontrados na segunda fase, e a técnica de subamostragem foi aplicada aos dados de treinamento. Para essa subamostragem, utilizou-se o método de *resample* da biblioteca *Scikit-learn*, configurado para reduzir a classe majoritária (classe 0) ao mesmo tamanho da classe minoritária (classe 1), a fim de minimizar o impacto do desbalanceamento. A implementação do código foi feita com *randomstate* = 42 para assegurar a reprodutibilidade dos resultados.
- ❑ **Sobreamostragem (Oversampling):** (Etapa 4) Na quarta fase, os algoritmos foram ajustados com os hiperparâmetros encontrados na segunda fase, e a técnica de sobreamostragem foi aplicada aos dados de treinamento. A sobreamostragem incrementou a classe minoritária (classe 1) ao duplicar amostras, de modo a equilibrar a quantidade de exemplos entre as classes. Para isso, utilizou-se o método de *resample* da biblioteca *Scikit-learn* configurado com *randomstate* = 42 para reprodutibilidade e *replace* = *True* afim de aumentar o número de exemplos da classe minoritária com a geração de amostras com reposição.

Em todas as fases de treinamento foram aplicadas as técnicas de vetorização *CountVectorizer* e TF-IDF. Ou seja, em cada uma das quatro fases de treinamento, os algoritmos foram inicialmente submetidos ao *CountVectorizer* e posteriormente ao TF-IDF, permitindo uma análise mais detalhada do impacto de cada técnica no desempenho final dos classificadores.

Na próxima seção são apresentados e discutidos os resultados dos experimentos realizados com os algoritmos e os respectivos parâmetros citados acima.

Experimentos e Análise dos Resultados

Neste capítulo é apresentada a realização dos experimentos, bem como os resultados obtidos pelos algoritmos aplicados. Também é apresentado um sumário dos melhores resultados por algoritmo e o algoritmo com melhor desempenho.

4.1 Resultados por algoritmo

As seções abaixo apresentam os resultados para cada algoritmo nas quatro etapas do experimento: Etapa 1 (simple), Etapa 2 (*GridSearch*), Etapa 3 (*GridSearch* + *undersampling*) e Etapa 4 (*GridSearch* + *oversampling*). Para cada etapa, comparou-se as técnicas de vetorização *CountVectorizer* e TF-IDF, avaliando as métricas de acurácia e medida F1.

O ajuste realizado nos hiperparâmetros dos algoritmos e na base de dados foi conduzido da seguinte maneira:

- ❑ **Etapa 1 (Configuração Simples):** Nessa fase inicial, nenhum hiperparâmetro foi alterado; os algoritmos foram executados com seus valores padrão. A base de dados foi apenas normalizada, sem quaisquer ajustes específicos, mantendo a configuração básica de cada modelo.
- ❑ **Etapa 2 (Busca em Grade):** Nesta etapa, um espaço de busca para os hiperparâmetros foi definido, e a técnica de *GridSearch* foi aplicada para otimização. Os melhores valores de hiperparâmetros identificados para *CountVectorizer* e TF-IDF foram então aplicados ao treinamento desta etapa e mantidos para as etapas seguintes. A base de dados permaneceu apenas normalizada, mas agora com os hiperparâmetros ajustados para melhor desempenho.
- ❑ **Etapa 3 (Subamostragem):** Aqui, a técnica de subamostragem foi utilizada para reduzir o desbalanceamento da base de dados. A classe majoritária (classe 0 - não ofensivo) foi reduzida ao tamanho da classe minoritária (classe 1 - ofensivo), que

possui 1456 amostras. Assim, a classe 0 também passou a ter 1456 amostras. Nesta fase, a base de dados estava equilibrada pela subamostragem e os hiperparâmetros já ajustados foram mantidos.

- **Etapa 4 (Sobreamostragem):** Por fim, foi aplicada a técnica de sobreamostragem à classe minoritária, aumentando-a para igualar o tamanho da classe majoritária por meio de duplicações aleatórias. Com essa abordagem, a classe minoritária (ofensivo) passou a ter 2683 amostras, igualando o número de amostras da classe majoritária. Nessa etapa, a base de dados estava equilibrada por sobreamostragem, com os hiperparâmetros ajustados.

4.1.1 Naive Bayes

A Tabela 2 traz o resultado do experimento realizado com o algoritmo *Naive Bayes*.

Naive Bayes					
Etapa	Configuração	CountVectorizer		TF-IDF	
		Acurácia	F1	Acurácia	F1
1	Simple	0.82	0.82	0.83	0.82
2	GridSearch	0.83	0.83	0.80	0.79
3	GridSearch + undersampling	0.78	0.78	0.81	0.81
4	GridSearch + oversampling	0.82	0.82	0.84	0.84

Tabela 2 – Acurácia média do algoritmo Naive Bayes com diferentes configurações

□ Resultados *CountVectorizer*

Na configuração simples com *CountVectorizer*, o *Naive Bayes* obteve uma acurácia e medida F1 de **0.82**. Nessa etapa, o hiperparâmetro α foi mantido em seu valor padrão de 0.1. Ao aplicar a busca em grade, o valor de α foi ajustado para um intervalo de valores [0.1,0.5,1.0,1.5,2.0], e o melhor desempenho foi obtido com $\alpha = 1.5$, resultando em uma ligeira melhora nas métricas, ambas atingindo **0.83**.

Esses resultados indicam que um ajuste do α contribui para uma maior estabilidade do modelo. No que diz respeito ao balanceamento de dados, a subamostragem causou uma redução nas métricas, sugerindo que a remoção de exemplos da classe majoritária (não ofensivos) teve um impacto negativo no modelo. Já a sobreamostragem foi mais eficaz, mantendo as métricas em **0.82**.

□ Resultados TF-IDF

Na configuração com TF-IDF, o *Naive Bayes* obteve uma acurácia de **0.83** e uma medida F1 de **0.82**, com α inicialmente em 0.1. Durante a busca em grade, com o espaço de busca definido em [0.1,0.5,1.0,1.5,2.0], o valor de α foi ajustado para 0.5, provocando uma ligeira queda na acurácia e na medida F1.

A técnica de subamostragem com TF-IDF mostrou-se um pouco mais estável do que a subamostragem com *CountVectorizer*. Por outro lado, a sobreamostragem apresentou o melhor desempenho entre todas as configurações, com uma acurácia e medida F1 de **0.84**, destacando-se como a configuração mais vantajosa para este modelo.

Os resultados gerais indicam que a técnica de sobreamostragem foi a mais eficaz em ambos os métodos de vetorização, especialmente com TF-IDF, onde o modelo alcançou seu melhor desempenho. A redução da importância de termos frequentes, característica do TF-IDF, parece auxiliar o modelo na identificação de padrões relevantes para a classificação de comentários ofensivos. A combinação de TF-IDF com sobreamostragem e o ajuste de α para 0.5 mostraram-se a configuração ideal, proporcionando o melhor desempenho.

4.1.2 SVM

A Tabela 3 traz o resultado do experimento realizado com o algoritmo SVM. A Etapa 1 manteve os valores padrão dos hiperparâmetros em $C = 1.0$, $\gamma = scale$ (desde o *scikit-learn* 0.22) e $kernel = rbf$. Na Etapa 2 o espaço de busca foi definido como:

- $C = [0.1, 1, 10, 100]$
- $\gamma = [1, 0.1, 0.01, 0.001]$
- $kernel = ['linear', 'rbf', 'poly', 'sigmoid']$

Essa configuração assistiu às duas técnicas de vetorização utilizadas neste trabalho.

SVM					
Etapa	Configuração	CountVectorizer		TF-IDF	
		Acurácia	F1	Acurácia	F1
1	Simple	0.80	0.78	0.82	0.81
2	GridSearch	0.86	0.85	0.84	0.84
3	GridSearch + undersampling	0.83	0.83	0.82	0.82
4	GridSearch + oversampling	0.86	0.85	0.84	0.84

Tabela 3 – Acurácia média do algoritmo SVM com diferentes configurações

□ Resultados *CountVectorizer*

Na primeira rodada utilizando o *CountVectorizer*, o SVM demonstra uma capacidade razoável em separar as classes ofensivas e não ofensivas, mas ainda abaixo do ideal. Após aplicar o melhor conjunto de parâmetros encontrado, com $C = 1$, $\gamma = 1$ e $kernel = linear$, os resultados melhoraram significativamente com acurácia

de **0.86** e medida F1 de **0.85**. Esse resultado indica que o ajuste de hiperparâmetros aumentou substancialmente a eficácia do modelo.

A técnica de subamostragem reduziu a acurácia e a medida F1 mas ainda manteve o resultado robusto, sugerindo que a redução do desbalanceamento de classes teve um efeito moderado. Por fim, na sobreamostragem o modelo alcançou novamente **0.86** de acurácia e **0.85** de F1, igualando o desempenho máximo obtido sem o balanceamento, sugerindo que a sobreamostragem não prejudicou a capacidade de generalização do modelo.

□ Resultados TF-IDF

Com o TF-IDF, a configuração inicial proporcionou um resultado levemente superior à configuração simples com *CountVectorizer*. Ao realizar a busca em grade, o conjunto ideal de hiperparâmetros foi $C = 10$, $\gamma = 0.1$ e $kernel = rbf$, resultando em uma acurácia e medida F1 um pouco inferior se comparado ao *CountVectorizer*.

A subamostragem piorou o desempenho, ao contrário da sobreamostragem, que melhorou novamente os valores de acurácia e F1 para **0.84**, mostrando um impacto positivo da sobreamostragem.

O desempenho do SVM melhorou substancialmente com o ajuste de hiperparâmetros, tanto com *CountVectorizer* quanto com TF-IDF, sendo o *CountVectorizer* levemente superior, especialmente na Etapa 2. A sobreamostragem revelou-se mais eficaz que a subamostragem, aumentando o desempenho sem introduzir vieses significativos. Esses resultados sugerem que o SVM, com os parâmetros devidamente ajustados, é capaz de capturar padrões complexos de linguagem associados aos comentários ofensivos. A escolha dos parâmetros ideais ($kernel$, C e γ) foi determinante para o sucesso do modelo em contextos desbalanceados.

4.1.3 Random Forest

A Tabela 4 traz o resultado do experimento realizado com o algoritmo *Random Forest*. A etapa 1 manteve os valores padrão dos hiperparâmetros em $n_estimators = 100$, $max_depth = none$, $min_samples_split = 2$ e $min_samples_leaf = 1$. A Etapa 2 definiu o espaço de busca como:

- $n_estimators = [50, 100, 200]$
- $max_depth = [None, 10, 20]$
- $min_samples_split = [2, 5, 10]$
- $min_samples_leaf = [1, 2, 4]$

Essa configuração assistiu às duas técnicas de vetorização utilizadas neste trabalho.

Random Forest					
Etapa	Configuração	CountVectorizer		TF-IDF	
		Acurácia	F1	Acurácia	F1
1	Simple	0.86	0.86	0.84	0.83
2	GridSearch	0.86	0.86	0.85	0.85
3	GridSearch + undersampling	0.83	0.83	0.81	0.81
4	GridSearch + oversampling	0.73	0.74	0.83	0.83

Tabela 4 – Acurácia média do algoritmo Random Forest com diferentes configurações

❑ Resultados CountVectorizer

Na configuração inicial, utilizando o *CountVectorizer*, o modelo *Random Forest* apresentou uma acurácia e medida F1 de **0.86**, demonstrando um desempenho inicial robusto e refletindo uma capacidade sólida de generalização do modelo, mesmo sem ajustes específicos de hiperparâmetros. A busca em grade permitiu a identificação dos melhores parâmetros, que permaneceram idênticos aos parâmetros padrão, reforçando que a configuração inicial do modelo já estava bem ajustada para essa aplicação. Isso sugere que, para esse conjunto de dados e configuração de vetorização, o *Random Forest* consegue capturar padrões relevantes de maneira satisfatória, mesmo sem a necessidade de ajustes complexos.

Com a subamostragem, a acurácia e a medida F1 caíram, sugerindo que a redução do desbalanceamento de classes através da remoção de amostras da classe majoritária impactou negativamente a capacidade do modelo de generalizar adequadamente. E ao aplicar a sobreamostragem, houve uma queda ainda mais acentuada no desempenho, com acurácia em **0.73** e medida F1 em **0.74**. Esse resultado indica que a sobreamostragem pode ter introduzido ruído, levando o modelo a uma tendência de *overfitting*, onde ele aprendeu padrões específicos ao invés de características generalizáveis.

❑ Resultados TF-IDF

Utilizando o TF-IDF, na Etapa 1 o modelo obteve uma acurácia de 0.84 e medida F1 de 0.83. Embora esses valores sejam ligeiramente inferiores aos obtidos com *CountVectorizer*, eles indicam um desempenho competitivo, mostrando que o TF-IDF também fornece uma representação útil para a classificação. Na Etapa 2, os melhores hiperparâmetros encontrados foram: $n_estimators = 100$, $max_depth = none$, $min_samples_split = 10$ e $min_samples_leaf = 1$, elevando a acurácia e a medida F1 para **0.85** cada. O ajuste em $min_samples_split$ para um valor maior (10) indica que o modelo beneficiou-se de árvores ligeiramente menos complexas, o que pode ter contribuído para uma redução de *overfitting*, melhorando a estabilidade dos resultados em dados de teste.

A subamostragem resultou em uma perda de desempenho, possivelmente por reduzir a quantidade de dados da classe majoritária, afetando a diversidade das árvores na floresta. Em contrapartida sobreamostragem elevou a acurácia e F1 para **0.83**, mostrando que, embora a sobreamostragem tenha reequilibrado as classes, o desempenho final ainda foi inferior ao da configuração sem balanceamento. Assim como no *CountVectorizer*, a sobreamostragem parece ter adicionado alguma variabilidade desnecessária, levando o modelo a capturar padrões específicos das amostras duplicadas em vez de manter uma visão generalizável.

Os resultados do *Random Forest* indicam que a configuração padrão já proporciona um desempenho robusto, especialmente com *CountVectorizer*, sem que a busca em grade traga melhorias significativas. Os resultados também mostram que a aplicação de técnicas de balanceamento, em particular a sobreamostragem, pode introduzir ruído e reduzir a capacidade de generalização do modelo, especialmente em dados textuais complexos.

Essa análise sugere que, para o modelo *Random Forest*, o uso de vetores de contagem (*CountVectorizer*) sem balanceamento de classes oferece a melhor combinação de desempenho e estabilidade, enquanto a sobreamostragem pode ser evitada para minimizar o risco de *overfitting* em dados desbalanceados. O ajuste do parâmetro *min_samples_split* mostrou-se relevante para TF-IDF, indicando que a complexidade das árvores deve ser ajustada com cautela para evitar sobreajustes.

4.1.4 GBM

A Tabela 5 traz o resultado do experimento realizado com o algoritmo GBM. A Etapa 1 manteve os valores padrão dos hiperparâmetros em $n_estimators = 100$, $learning_rate = 0.1$, $min_samples_split = 2$ e $max_depth = 3$. A Etapa 2 definiu o espaço de busca como:

- $n_estimators = [50, 100, 150]$
- $learning_rate = [0.05, 0.1, 0.2]$
- $min_samples_split = [3, 5, 7]$
- $max_depth = [2, 5, 10]$

Essa configuração assistiu às duas técnicas de vetorização utilizadas neste trabalho.

GBM					
Etapa	Configuração	CountVectorizer		TF-IDF	
		Acurácia	F1	Acurácia	F1
1	Simple	0.82	0.80	0.82	0.80
2	GridSearch	0.85	0.84	0.84	0.83
3	GridSearch + undersampling	0.83	0.83	0.82	0.81
4	GridSearch + oversampling	0.85	0.84	0.84	0.83

Tabela 5 – Acurácia média do algoritmo GBM com diferentes configurações

□ Resultados CountVectorizer

Na Etapa 2, utilizando o *CountVectorizer*, o GBM obteve seu melhor resultado, com acurácia de **0.85** e F1 de **0.84**. Nessa etapa, os melhores parâmetros para o GBM foram aplicados: *learning_rate* = 0.2, *max_depth* = 7, *min_samples_split* = 10, e *n_estimators* = 150. A combinação de um *learning_rate* mais alto e maior profundidade (*max_depth* = 7) sugere que o modelo aprendeu melhor os padrões complexos dos dados, enquanto um número maior de árvores (150) e uma divisão mínima de amostras mais elevada (*min_samples_split* = 10) ajudaram a evitar *overfitting*, favorecendo a generalização.

Ao aplicar subamostragem o desempenho do modelo caiu indicando que, ao reduzir a quantidade de informação disponível na classe majoritária, reduz-se também a capacidade do modelo de aprender padrões de maneira robusta. Por outro lado, empregando a sobreamostragem juntamente com os melhores hiperparâmetros o desempenho torna-se o mesmo da Etapa 2, com acurácia e F1 em **0.85** e **0.84**, respectivamente. Isso sugere que a sobreamostragem balanceou as classes sem prejudicar a generalização do modelo.

□ Resultados TF-IDF

Na configuração inicial com TF-IDF, o GBM apresentou os mesmos resultados observados para o *CountVectorizer*. Esse desempenho estável entre as duas técnicas de vetorização indica que o modelo inicial estava razoavelmente adequado para ambas as representações de texto. Na Etapa 2, a busca em grade selecionou os parâmetros *learning_rate* = 0.1, *max_depth* = 7, *min_samples_split* = 2, e *n_estimators* = 150, elevando o desempenho para uma acurácia de **0.84** e F1 de **0.83**. A profundidade aumentada e o número maior de estimadores (150) reforçam a capacidade do modelo de aprender padrões detalhados sem comprometer a generalização. Aqui, o *learning_rate* mais conservador (0.1) parece ter favorecido uma convergência mais estável, contribuindo para o ganho de desempenho.

Na Etapa 3, a subamostragem com TF-IDF resultou em uma ligeira queda no desempenho. Esse comportamento é consistente com os resultados observados com

CountVectorizer, sugerindo que a redução da classe majoritária impactou negativamente a capacidade do GBM de capturar todos os padrões dos dados. Na Etapa 4, com a aplicação de sobreamostragem, o desempenho com TF-IDF foi restaurado. Esse resultado mostra que o balanceamento com sobreamostragem, quando combinado com os parâmetros ajustados, permite uma melhor generalização em relação ao modelo sem balanceamento, especialmente quando a representação TF-IDF é utilizada.

Os resultados do GBM mostram que, para ambos os vetorizadores, o ajuste de hiperparâmetros proporciona ganhos significativos em acurácia e F1. A combinação de uma maior profundidade e um número elevado de estimadores sugere que o GBM beneficia-se de configurações que permitem uma exploração mais detalhada dos padrões, especialmente com o *learning_rate* adequado para estabilizar a aprendizagem.

O impacto das técnicas de balanceamento foi diferente para cada abordagem: a sobreamostragem, em especial, demonstrou maior compatibilidade com o GBM, enquanto a subamostragem tendeu a reduzir a performance. Com isso, os melhores resultados foram obtidos na Etapa 2 (ajuste de hiperparâmetros sem balanceamento) e na Etapa 4 (ajuste de hiperparâmetros com sobreamostragem), indicando que o GBM é capaz de lidar eficientemente com desbalanceamento moderado quando bem ajustado, mas também se beneficia de técnicas de balanceamento que introduzam novas amostras da classe minoritária.

4.1.5 Sumário dos resultados

A Tabela 6 sumariza os melhores resultados obtidos para cada uma dos algoritmos considerados na sessão anterior. De modo geral, várias conclusões podem ser extraídas sobre o desempenho desses algoritmos e a influência das técnicas de vetorização e balanceamento. Algumas elas estão destacadas abaixo:

Sumário dos melhores resultados				
Classificador	Estratégia	Vetorização	Acurácia	F1
Randon Forest	Simple	CountVectorizer	0.86	0.86
Randon Forest	GridSearch	CountVectorizer	0.86	0.86
SVM	GridSearch	CountVectorizer	0.86	0.85
GBM	GridSearch	CountVectorizer	0.85	0.84
GBM	GridSearch + oversampling	CountVectorizer	0.85	0.84
Naive Bayes	GridSearch	TF-IDF	0.84	0.84

Tabela 6 – Melhores resultados dos algoritmos em termos de acurácia e medida F1

4.1.5.1 Melhores desempenhos

SVM e *Random Forest* foram os classificadores com maior desempenho, ambos alcançando **0.86** em acurácia e medida F1. Para o SVM, o desempenho máximo foi obtido com *CountVectorizer* após ajustes com a busca em grade (*SearchGrid*). No caso do *Random Forest*, esse resultado foi alcançado tanto com a configuração simples quanto com os hiperparâmetros ajustados. Isso sugere que o modelo é particularmente robusto para esse tipo de tarefa, conseguindo atingir alta precisão mesmo sem ajustes complexos.

4.1.5.2 Melhor técnica de vetorização

A vetorização por ***CountVectorizer*** foi dominante entre os melhores resultados, particularmente para SVM, *Random Forest* e GBM. Esse resultado pode indicar que a representação com frequências simples das palavras, ao invés da ponderação por TF-IDF, foi mais adequada para os algoritmos baseados em árvores e o SVM, talvez por fornecer uma estrutura de dados mais direta para a discriminação de classes.

Porém, cabe ressaltar que, diferentemente dos outros algoritmos, o *Naive Bayes* obteve seu melhor desempenho com TF-IDF, alcançando 0.84 em acurácia e medida F1 após o ajuste de hiperparâmetros com a busca em grade. Esse resultado é consistente com a natureza probabilística do *Naive Bayes*, que frequentemente se beneficia das pontuações TF-IDF, já que elas ajudam a suavizar a influência de palavras muito frequentes e não informativas.

4.1.5.3 Impactos da busca em grade e as técnicas de balanceamento

A busca em grade teve um papel significativo na melhoria de desempenho para *Naive Bayes*, SVM e GBM, demonstrando que a configuração ideal dos parâmetros pode influenciar consideravelmente a capacidade preditiva.

No caso do GBM, o uso de sobreamostragem juntamente com os hiperparâmetros ajustados levou a um desempenho maximizado, sugerindo que esse algoritmo se beneficia de dados balanceados para evitar o viés para a classe majoritária. Já para o *Random Forest*, os resultados indicam que o modelo é naturalmente mais resiliente ao desbalanceamento dos dados, dado que o modelo simples também apresentou um ótimo desempenho.

O GBM, mesmo com ajuste e sobreamostragem, alcançou uma acurácia de 0.85 e F1 de 0.84 com *CountVectorizer*, ficando ligeiramente atrás de SVM e *Random Forest*. Isso sugere que, embora o GBM tenha sido eficaz, ele é mais sensível aos ajustes e ao balanceamento dos dados do que o *Random Forest*, necessitando de otimizações específicas para atingir um desempenho competitivo. Esse comportamento reflete a natureza do GBM, que realiza aprendizado de maneira sequencial e é mais suscetível ao *overfitting*, especialmente em conjuntos de dados desbalanceados.

4.1.5.4 Conclusão

Os experimentos mostram que, para essa tarefa de classificação, SVM e *Random Forest* com *CountVectorizer* foram os modelos mais adequados, apresentando um excelente equilíbrio entre precisão e simplicidade. O GBM, embora eficiente, requer ajustes de hiperparâmetros e técnicas de balanceamento para obter um desempenho comparável, enquanto o *Naive Bayes*, com TF-IDF, se mostrou uma alternativa leve e eficiente.

Essa análise pode orientar decisões para futuras implementações, destacando que modelos como o Random Forest e o SVM são opções promissoras quando se busca alta performance com um custo de ajuste reduzido, ao passo que o GBM pode ser considerado quando há recursos para otimizações e um controle mais refinado do aprendizado.

Conclusão

A crescente presença de comentários ofensivos em redes sociais virtuais destaca a importância de métodos eficazes para a moderação de conteúdo, especialmente em plataformas de alta interação, como o *YouTube*. Este trabalho propôs uma abordagem sistemática para detectar e classificar automaticamente comentários ofensivos em português brasileiro, utilizando uma combinação de técnicas de PLN e AM.

A análise focou na comparação de 4 modelos clássicos de classificação de texto, sendo eles *Naive Bayes*, SVM, *Random Forest* e GBM, utilizando vetorização com *CountVectorizer* e TF-IDF. O desempenho máximo de acurácia e medida F1, de 86%, foi alcançado com o modelo *Random Forest*, demonstrando que é possível desenvolver classificadores com desempenho competitivo através de técnicas baseadas em aprendizado supervisionado, ainda que com algumas limitações. Esses resultados reforçam a hipótese inicial do trabalho, sugerindo que métodos de AM aplicados a dados textuais em português brasileiro podem, de fato, atingir um nível de desempenho que os torna promissores para a implementação em sistemas reais de moderação automatizada.

Embora os resultados obtidos comprovem a viabilidade da abordagem, a literatura indica que os algoritmos mais promissores para esse tipo de tarefa são redes neurais profundas, como CNN e RNN, além de transformadores como BERT, que demonstraram alto desempenho em tarefas de classificação de texto e têm sido amplamente aplicados em outros idiomas. A implementação desses modelos, embora mais complexa e exigente em termos computacionais, representa uma direção promissora para trabalhos futuros, com potencial para elevar a precisão da detecção de comentários ofensivos em português brasileiro.

Adicionalmente, a base de dados utilizada, composta de comentários de uma única plataforma, poderia ser expandida para incluir dados de outras redes sociais, como *Twitter* e *Facebook*, enriquecendo a diversidade linguística e contextual dos exemplos ofensivos. A ampliação do conjunto de dados não só ajudaria a capturar melhor as variações da linguagem ofensiva, mas também a mitigar possíveis vieses que uma amostra limitada pode introduzir nos modelos.

Outro ponto de aprimoramento seria a inclusão de técnicas de pré-processamento mais avançadas para o idioma português, como correção ortográfica automática e lematização, que podem melhorar a representação dos dados textuais. Além disso, experimentos com *embeddings* contextualizados, como *word embeddings* e *sentence embeddings* específicos para o português, poderiam ajudar a capturar nuances linguísticas que os vetores de contagem e TF-IDF, baseados em frequências de palavras, podem não representar adequadamente.

Enfim, este estudo oferece contribuições relevantes ao demonstrar a viabilidade de métodos de AM clássicos na detecção de comentários ofensivos em português brasileiro, além da construção e disponibilização de uma base de dados composta por 4139 comentários, voltada para a detecção de discurso ofensivo em português brasileiro. Esta base oferece uma representação diversificada e útil para o avanço de pesquisas futuras no campo, permitindo que outros pesquisadores possam explorar e desenvolver modelos para o idioma português com dados relevantes e consistentes. No entanto, a continuidade desta pesquisa com técnicas de redes neurais avançadas, expansão de dados e pré-processamento específico para o idioma se faz necessária para alcançar um desempenho ainda mais robusto. Trabalhos futuros nessa direção poderão fortalecer a moderação de conteúdo automatizada em redes sociais e promover um ambiente online mais inclusivo e seguro.

O dataset produzido por esse estudo pode ser acessado através do *link*: <<https://github.com/KellyAlves/detection-offensive-comments-dataset.git>>

Referências

aaaa, a. Nenhuma citação no texto.

aaaa, b. Nenhuma citação no texto.

aaaa, c. Nenhuma citação no texto.

James Allen. *Natural language understanding*. Benjamin-Cummings Publishing Co., Inc., 1988. Citado na página 28.

Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006. Citado 2 vezes nas páginas 18 e 24.

E. Bisong. *Google Colaboratory*. Apress, Berkeley, CA, 2019. Citado na página 37.

Da BBC News Brasil. Discurso de ódio na internet aumentou durante a pandemia, aponta pesquisa. 2022. URL <<https://www.bbc.com/portuguese/geral-59300051>>. Citado na página 12.

Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001. Citado na página 26.

Peter F Brown, Stephen A Della Pietra, Vincent J Della Pietra, and Robert L Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311, 1993. Citado 2 vezes nas páginas 30 e 31.

Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002. Citado na página 23.

Ying Chen, Yilu Zhou, Sencun Zhu, and Heng Xu. Detecting offensive language in social media to protect adolescent online safety. In *2012 international conference on privacy, security, risk and trust and 2012 international conference on social computing*, pages 71–80. IEEE, 2012. Citado na página 13.

Noam Chomsky. *Syntactic structures*. Mouton de Gruyter, 2002. Citado 2 vezes nas páginas 28 e 29.

Kenneth Church and Robert L Mercer. Introduction to the special issue on computational linguistics using large corpora. *Computational linguistics*, 19(1):1–24, 1993. Citado na página 28.

- Corinna Cortes. Support-vector networks. *Machine Learning*, 1995. Citado na página 25.
- Rogers de Pelle and Viviane Moreira. Offensive comments in the brazilian web: a dataset and baseline results. In *Anais do VI Brazilian Workshop on Social Network Analysis and Mining*, pages 510–519, Porto Alegre, RS, Brasil, 2017. SBC. doi: <10.5753/brasnam.2017.3260>. URL <<https://sol.sbc.org.br/index.php/brasnam/article/view/3260>>. Citado 2 vezes nas páginas 13 e 39.
- Jacob Devlin et al. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186, 2019. Citado 3 vezes nas páginas 28, 32 e 33.
- Daury Cesar Fabríz and Gabriel Heringer de Mendonça. O papel das plataformas de redes sociais diante do dever de combater o discurso de ódio no brasil. 2022. Citado na página 13.
- Paula Fortuna and Sérgio Nunes. A survey on automatic detection of hate speech in text. *ACM Computing Surveys (CSUR)*, 51(4):1–30, 2018. Citado na página 37.
- Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001. Citado na página 27.
- Yoav Goldberg. A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, 57:345–420, 2016. Citado na página 31.
- Ian Goodfellow. Deep learning, 2016. Citado 4 vezes nas páginas 18, 19, 20 e 31.
- Bruno Ferrari Guide. *Detecção automática de discurso de ódio punitivista em redes sociais*. PhD thesis, Universidade de São Paulo, 2022. URL <<https://www.teses.usp.br/teses/disponiveis/8/8139/tde-08122022-174035/>>. Citado na página 38.
- Claire Hardaker. “uh.... not to be nitpicky, but... the past tense of drag is dragged, not drug.”: An overview of trolling strategies. *Journal of Language Aggression and Conflict*, 1(1):58–86, 2013. Citado na página 12.
- Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009. Citado 4 vezes nas páginas 18, 19, 20 e 24.
- Haibo He and Edwardo A Garcia. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9):1263–1284, 2009. Citado na página 23.
- Susan C Herring. Gender and democracy in computer-mediated. *Computerization and controversy: Value conflicts and social choices*, 3(2):476–489, 1993. Citado na página 17.
- S Hochreiter. Long short-term memory. *Neural Computation MIT-Press*, 1997. Citado 2 vezes nas páginas 28 e 32.
- Nathalie Japkowicz and Shaju Stephen. The class imbalance problem: A systematic study. *Intelligent data analysis*, 6(5):429–449, 2002. Citado na página 22.

- Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall, 1st edition, 2000. ISBN 0130950696. Citado 7 vezes nas páginas 29, 30, 31, 33, 34, 35 e 36.
- Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall, 2nd edition, 2009. ISBN 9780131856216. Citado 2 vezes nas páginas 28 e 33.
- Daniel Jurafsky and James H. Martin. *Speech and Language Processing*. Prentice Hall, 3rd edition, 2019. ISBN 978-0-13-187321-6. Citado na página 13.
- Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, 2014. URL <<https://aclanthology.org/D14-1181/>>. Citado na página 32.
- K Hazel Kwon and Anatoliy Gruzd. Is offensive commenting contagious online? examining public vs interpersonal swearing in response to donald trump’s youtube campaign videos. *Internet Research*, 27(4):991–1010, 2017. Citado na página 17.
- Machine Learning. Tom mitchell. *Publisher: McGraw Hill*, 1997. Citado na página 31.
- Amanda Lenhart, Michele Ybarra, Kathryn Zickuhr, and Myeshia Price-Feeney. Online harassment, digital abuse, and cyberstalking in america. *Pew Research Center*, 2016. URL <<https://www.pewresearch.org/internet/2016/07/11/online-harassment-2016/>>. Citado na página 17.
- Christopher Manning and Hinrich Schütze. *Foundations of statistical natural language processing*. MIT press, 1999. Citado na página 28.
- Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, 2008. ISBN 0262133601. Citado 3 vezes nas páginas 34, 35 e 36.
- Andrew McCallum, Kamal Nigam, et al. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Madison, WI, 1998. Citado na página 24.
- What Is Data Mining. Data mining: Concepts and techniques. *Morgan Kaufmann*, 10 (559-569):4, 2006. Citado na página 24.
- Mehryar Mohri. *Foundations of machine learning*, 2018. Citado 2 vezes nas páginas 18 e 20.
- Amgad Muneer and Suliman Mohamed Fati. A comparative analysis of machine learning techniques for cyberbullying detection on twitter. *Future Internet*, 12(11):187, 2020. Citado na página 38.
- Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012. Citado 2 vezes nas páginas 19 e 20.

- Sünje Paasch-Colberg, Christian Strippel, Joachim Trebbe, and Martin Emmer. From insult to hate speech: Mapping offensive language in german user comments on immigration. *Media and Communication*, 9(1):171–180, 2021. Citado na página 16.
- F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. Citado na página 37.
- Whitney Phillips. Lolling at tragedy: Facebook trolls, memorial pages and resistance to grief online. *First Monday*, 2011. Citado na página 12.
- Georgios K Pitsilis, Heri Ramampiaro, and Helge Langseth. Detecting offensive language in tweets using deep learning. *arXiv preprint arXiv:1801.04433*, 2018. Citado na página 39.
- Martin F Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980. Citado na página 34.
- David MW Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *arXiv preprint arXiv:2010.16061*, 2020. Citado na página 21.
- Juan Ramos et al. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 29–48. Citeseer, 2003. Citado 2 vezes nas páginas 35 e 36.
- Sebastian Raschka and Vahid Mirjalili. *Python machine learning: Machine learning and deep learning with Python, scikit-learn, and TensorFlow 2*. Packt publishing ltd, 2019. Citado 2 vezes nas páginas 21 e 22.
- Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47, 2002. Citado 2 vezes nas páginas 19 e 21.
- Andrew Sellars. Defining hate speech. *Berkman Klein Center Research Publication*, (2016-20):16–48, 2016. Citado na página 17.
- Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948. Citado 2 vezes nas páginas 28 e 30.
- John Suler. The online disinhibition effect. *Cyberpsychology & behavior*, 7(3):321–326, 2004. Citado na página 17.
- Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 27, pages 3104–3112, 2014. URL <<https://papers.nips.cc/paper/2014/hash/30e1e3f1e34fcaaf9c1795b251d34a2f-Abstract.html>>. Citado na página 32.
- Duyu Tang, Bing Qin, and Ting Liu. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1422–1432, 2015. Citado na página 32.
- A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017. Citado 2 vezes nas páginas 28 e 32.

- Jeremy Waldron. *The harm in hate speech*. Harvard University Press, 2012. Citado na página 12.
- Terry Winograd. Understanding natural language. *Cognitive psychology*, 3(1):1–191, 1972. Citado na página 29.
- YouTube Developers. Youtube data api overview. <<https://developers.google.com/youtube/v3>>, 2023. Accessed: 2024-11-07. Citado na página 37.
- Ziqi Zhang and Lei Luo. Hate speech detection: A solved problem? the challenging case of long tail on twitter. *Semantic Web*, 10(5):925–945, 2019. Citado 2 vezes nas páginas 12 e 13.