UNIVERSIDADE FEDERAL RURAL DO RIO DE JANEIRO
INSTITUTO MULTIDISCIPLINAR

NICOLAS MAGALHÃES SILVA
YAN CARLOS DE FIGUEIREDO MACHADO

Detecção e Monitoramento de Discurso de Ódio em Redes Sociais na Era do Big Data

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

Nova Iguaçu, dezembro de 2023

Detecção e Monitoramento de Discurso de Ódio em Redes Sociais na Era do Big Data

Nicolas Magalhães Silva

Yan Carlos de Figueiredo Machado

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

Apresentado por:

_____

Nicolas Magalhães Silva

_____

Yan Carlos de Figueiredo Machado

Aprovado por:

_____

Prof. Filipe Braida do Carmo, D.Sc.

_____

Prof. Leandro Guimarães Marques Alvim, D.Sc.

_____

Prof. Bruno José Dembogurski, D.Sc.

NOVA IGUAÇU, RJ - BRASIL

dezembro de 2023

Agradecimentos

Nicolas Magalhães Silva

Antes de tudo, preciso agradecer aos meus pais, Jenyffer e Valdir, e avós, Carmem e José Cícero. Sem todo apoio e sacrifício de vocês ao longo desses anos eu não estaria aqui agora. Obrigado por tudo, espero estar deixando vocês orgulhosos. Em especial ao meu avô, gostaria que ainda estivesse aqui nesse momento com a gente.

Gostaria também de agradecer ao Yan Figueiredo, meu parceiro nesse trabalho e um dos meus melhores amigos. Sou extremamente grato pela sua amizade. Fizemos praticamente todos os trabalhos juntos ao longo da faculdade e não poderíamos deixar essa tradição acabar no último! Acredito de coração que você indiretamente me ajudou a alcançar minha melhor versão como aluno na Ciência da Computação. Te admiro demais como pessoa e profissional, espero que essa amizade dure para sempre.

Preciso agradecer também ao grupo de amigos que fiz durante o curso. Softawii, vocês são incríveis. Durante a pandemia, a amizade de vocês foi essencial para manter o foco, seja pelas noites jogando jogos juntos ou pelos grupos de estudo que formávamos para entender as matérias durante o ensino remoto. Estar com vocês fez a experiência inteira da graduação ser mais divertida.

Agradeço também ao meu orientador, Filipe Braida, não fiz muitas matérias contigo, mas o apoio durante todo o curso, com a iniciação científica no ATISLabs, sendo nosso técnico e incentivando a participação em maratonas de programação e a liberdade de te chamar para conversar sobre o curso e pedir opiniões sobre assuntos acadêmicos e profissionais fez toda a diferença. Além disso, agradeço também à

professora Juliana Zamith, que durante o começo da pandemia, quando o curso estava parado, incentivou a participação em eventos de programação paralela e a entrada no ATISLabs.

Por fim agradeço aos meus amigos, Duda, João Matheus, Lucas e Manu. Obrigado pela amizade de vocês durante todo esse tempo e pelo apoio que me dão sempre.

Yan Carlos de Figueiredo Machado

Em primeiro lugar, expresso meus sinceros e profundos agradecimentos aos meus pais, Leide Jane e Carlos Antônio. Todo cuidado e carinho que recebi ao longo da minha vida, assim como a atenção e incentivo para perseguir meus sonhos, foram os elementos que me impulsionaram além de todas as barreiras e desafios até o ponto em que me encontro hoje. Não consigo imaginar um mundo sem o apoio de vocês. Muito obrigado por acreditarem em mim, mesmo quando eu não acreditava, foi graças a isso que consegui alcançar a graduação e muitas outras conquistas. Agradeço também por me amarem. Espero continuar dando orgulho para vocês.

Agradeço à minha irmã, Maria Clara. Você me fez conhecer o melhor da vida, e todos os dias agradeço por poder te chamar de irmã, por testemunhar o seu crescimento e apoiar cada passo seu. Você não sabe o quanto eu te amo e o quanto sua presença me ajudou a continuar nessa jornada.

Gostaria de expressar meus sinceros agradecimentos ao Nicolas Magalhães, meu amigo ao longo de toda a jornada acadêmica e parceiro de trabalho no projeto final. Iniciamos nossa colaboração na primeira disciplina do curso, desenvolvendo um joguinho para a disciplina de Computação 1, e agora estamos encerrando essa fase com mais uma colaboração bem-sucedida, entre muitas. Quero agradecer profundamente pela amizade que construímos e pelas experiências compartilhadas dentro e fora da UFRRJ. Sou imensamente grato por essa amizade ao longo de todo esse período acadêmico, e espero que ela perdure por toda a vida. Muito obrigado!

Agradeço a todos os membros da rede de apoio, também conhecida como Softawii, que foi construída nesta universidade. Estivemos juntos desde o primeiro período e, agora, cinco anos depois, continuamos mantendo a mesma amizade de sempre. Estudamos juntos, jogamos juntos, surtamos juntos, crescemos juntos e aprendemos juntos. É um enorme prazer fazer parte disso, e agradeço por cada conversa, cada partida e cada momento compartilhado! Digdins, vocês são brabos!

Agradeço a UFRRJ, em especial aos professores do DCC que contribuíram para a minha formação. Todo conhecimento adquirido durante o curso é fundamental para

minha formação profissional, pessoal e, especialmente, para o desenvolvimento deste trabalho. Expresso meu agradecimento especial ao professor Filipe Braida. Obrigado por apoiar nosso projeto, mesmo nos momentos difíceis, você estava sempre nos motivando. Agradeço pela oportunidade de crescimento no ATISLabs e pelo apoio em todas as disciplinas em que tive o prazer de participar. Obrigado também, Juliana Zamith, por catalisar a minha entrada na Iniciação Científica. Foi fundamental para a minha construção profissional e pela minha jornada acadêmica.

Agradeço aos meus amigos que cresceram comigo fora da faculdade. Ronan Rocha, Miguel Conforto, Bruno Conforto, Jonathan Senhor, Ester Muniz, e Ana Paula Prolungatti, quero expressar minha gratidão. A companhia de vocês ao longo desses anos sempre me proporcionou a força e a energia necessárias para me tornar a melhor versão de mim mesmo. Vocês são amigos incríveis, e desejo apenas o melhor em todos os caminhos que escolherem trilhar.

Obrigado, Deus, por estar ao meu lado em toda a minha vida e por sempre ouvir as minhas orações. Agradeço por sua proteção constante e por me guiar sempre que precisei. Sou grato por todas as alegrias que você me proporcionou ao longo da minha vida.

RESUMO

Detecção e Monitoramento de Discurso de Ódio em Redes Sociais na Era do Big Data

Nicolas Magalhães Silva e Yan Carlos de Figueiredo Machado

dezembro/2023

Orientador: Filipe Braida do Carmo, D.Sc.

A transição para a Web 2.0 no início do século XXI gerou uma produção massiva de dados em tempo real, dando origem ao conceito de *Big Data*. Dessa forma, surge a necessidade de moderação de conteúdo online, principalmente interações entre usuários em redes sociais e fóruns, pois acabam se tornando um ambiente propício para o discurso de ódio. Este trabalho propõe um sistema de detecção e uma plataforma de monitoramento para lidar com esse fenômeno em redes sociais, com o objetivo central de permitir aos usuários monitorar o Discord e o Twitter. A sistema proposto envolve uma aplicação *web* para o Twitter, facilitando análises de perfis problemáticos e monitoramento do discurso de ódio em discussões específicas. Para o Discord, a aplicação inclui a criação de um bot para moderação, capturando estatísticas sobre mensagens ofensivas e contribuindo para a manutenção de um ambiente saudável.

ABSTRACT

Detecção e Monitoramento de Discurso de Ódio em Redes Sociais na Era do Big Data

Nicolas Magalhães Silva and Yan Carlos de Figueiredo Machado

dezembro/2023

Advisor: Filipe Braida do Carmo, D.Sc.

The transition to Web 2.0 in the early 21st century led to a massive production of real-time data, giving rise to the concept of Big Data. Consequently, the need for online content moderation arises, especially in user interactions on social media and forums, as these platforms often become conducive to hate speech. This work proposes a detection system and monitoring platform to address this phenomenon on social media, with a central goal of enabling users to monitor Discord and Twitter. The proposed system involves a web application for Twitter, facilitating the analysis of problematic profiles and monitoring hate speech in specific discussions. For Discord, the application includes the creation of a moderation bot, capturing statistics on offensive messages and contributing to maintaining a healthy environment.

Lista de Figuras

Lista de Tabelas

Lista de Códigos

Lista de Abreviaturas e Siglas

DCC Departamento de Ciência da Computação

UFRRJ Universidade Federal Rural do Rio de Janeiro

PLN Processamento de Linguagem Natural

BoW Bag of Words

TF-IDF Term Frequency-Inverse Document Frequency

IDF Inverse Document Frequency

TF Term Frequency

Word2Vec Word to Vector

CBOW Continuous Bag-of-Words Model

BOW Bag-of-Words

GloVe Global Vectors for Word Representation

LSTM Long Short-Term Memory

SVM Support Vector Machine

RNN Recurrent Neural Network

API Application Programming Interface

REST Representational State Transfer

SP Stored Procedure

LLM Large Language Model

MVC Model-View-Controller

ORM Object-Relational Mapping

json JavaScript Object Notation

ORM Objeto Relational Mapping

W2V Word2Vec

Sumário

Capítulo 1

Introdução

A popularização do acesso a computadores pessoais e a internet no final do século XX e começo do século XXI, aliado aos novo conceito de *web* como plataforma, gerou uma grande mudança de paradigma na forma como as pessoas se organizam e se comunicam *online*. A partir disso, usuários da internet passaram a migrar e se concentrar em alguns poucos sites gerenciados por grandes empresas, e isso iniciou a produção de alta quantidade de dados em tempo real por esses usuários.

Esse grande volume de dados tornou-se praticamente impossível de ser consumido por seres humanos em tempo real (ELGENDY; ELRAGAL, 2014). Com isso, tornou-se necessário a utilização de ferramentas automatizadas para conseguir processar, analisar e gerar métricas para permitir um entendimento geral dos dados.

Entretanto, o grande volume de dados pode se tornar um problema quando o conteúdo dele são comentários e interações entre usuários que precisariam ser moderados. Devido a um certo grau de anonimidade gerado no contexto online e a impossibilidade de moderar todo conteúdo gerado, redes sociais e fóruns acabam se tornando um ambiente propício ao surgimento do discurso de ódio na internet.

A partir dessa problemática surgiu a necessidade de implementar um sistema capaz de detectar o discurso de ódio para monitorar as plataformas e manter o ambiente controlado. Assim, surge a proposta desse trabalho de desenvolver um sistema de detecção e monitoramento do discurso de ódio em diferentes redes sociais.

1.1 Objetivo

O objetivo desse trabalho é propor e implementar um sistema que permita a detecção da presença de discurso de ódio em publicações de usuários nas redes sociais Discord e Twitter, possuindo uma aplicação diferente para cada uma. A partir disso, possibilitar o usuário do sistema a monitorar de forma ampla essas redes.

No sistema proposto, além de fornecer o monitoramento de mensagens na rede social Twitter através de uma aplicação, também será possível coletar publicações de usuários a partir de termos especificados para filtrar a coleta. Dessa forma, permitindo a análise de perfis problemáticos, ou acompanhar o teor de discurso de ódio presente em discussões de assuntos específicos.

No contexto do Discord, a aplicação proposta envolve a criação de um bot que busca facilitar a moderação por parte dos administradores dos servidores. Esse bot permitirá a captura de estatísticas sobre mensagens contendo discurso de ódio de usuários individuais, além de gerar listas destacando os principais ofensores. Dessa forma, agindo como um facilitador do processo de moderação, auxiliando os moderadores a manterem um mais ambiente saudável.

Além disso, também será desenvolvido um modelo de aprendizado de máquina responsável pela classificação da presença de discurso de ódio nos textos. Será utilizado também um modelo de detecção de discurso de ódio como serviço como alternativa para classificação. O objetivo do trabalho é ser modular e permitir a utilização de qualquer tipo de modelo de classificação, seja ele treinado internamente ou provido por um serviço. Os detalhes da arquitetura desse trabalho serão discutidos ao longo dos capítulos, que serão descritos a seguir.

1.2 Principais Contribuições

- *Uso de inteligência artificial para detecção de discurso de ódio em redes sociais*

  Esse trabalho contribuiu como material de estudo sobre a utilização de inteligência artificial ativamente na classificação de dados publicados em redes

sociais em tempo-real. Foi explorado o uso de algoritmos leves e também do uso de *Transformers* para a classificação, reforçando pontos positivos, análises e incertezas.

- *Desenvolvimento de um sistema web para visualização de dados analisados das redes sociais*

  O desenvolvimento do sistema web para visualização dos dados é uma referência para futuras contribuições em projetos com objetivo de fornecer visualização para dados. Apresentando soluções e tecnologias eficientes e de rápido desenvolvimento.

- *Desenvolvimento de um bot para monitoramento de mensagens no Discord*

  O desenvolvimento do bot sugere uma implementação que pode servir como referência futura para vários trabalhos relacionados a bots no Discord ou em outras redes sociais. Neste estudo, apresentamos como é possível integrar diferentes tecnologias para fornecer dados úteis para os usuários.

1.3 Organização do Trabalho

Esse trabalho será organizado da seguinte forma:

- **Capítulo 2**: Será discutido de forma teórica os conhecimentos necessários para a compreensão técnica do trabalho.
- **Capítulo 3**: Será apresentada de forma mais detalhada a motivação do trabalho, será comentado sobre os trabalhos relacionados e iremos apresentar a arquitetura proposta para o sistema e a explicação de cada elemento que o compõe.
- **Capítulo 4**: Será abordado o desenvolvimento da arquitetura proposta no capítulo anterior e serão apresentados os resultados dessas implementações.
- **Capítulo 5**: Serão apresentadas as considerações finais dos autores, abordando as limitações, possíveis alterações e continuações que podem partir deste

trabalho.

Capítulo 2

Fundamentação

Nesse capítulo serão discutidos de forma teórica os conceitos de Discurso de Ódio, Mineração de Dados, Processamento de Linguagem Natural, Análise de Sentimento e Classificação de Texto que serão utilizados nesse trabalho.

2.1 Discurso de Ódio

A definição de discurso de ódio ainda é um conceito em discussão. Não existe uma definição formal sob a lei universal dos direitos humanos que abranja esse termo de forma concreta. Apesar disso, as Nações Unidas[1] propõe três atributos importantes ao discurso de ódio que serão detalhados a seguir.

O discurso de ódio se manifesta em qualquer forma de expressão, seja *online* ou *offline*. O discurso de ódio é discriminatório ou pejorativo a um indivíduo ou grupo. Além disso, o discurso de ódio tem como alvo característica de um indivíduo ou grupo, tal como: religião, etnia, nacionalidade, raça, cor, descendência, gênero, origem social, orientação sexual, além de muitos outros.

[1] https://www.un.org/en/hate-speech/understanding-hate-speech/what-is-hate-speech

2.2 Processamento de Linguagem Natural

As Linguagens Naturais são expressões linguísticas utilizadas pelos seres humanos com o propósito primordial de facilitar a comunicação entre indivíduos. Qualquer forma de linguagem empregada com esse intuito e aprendida naturalmente através do ambiente é considerada uma Linguagem Natural (KHURANA et al., 2022). Usualmente, linguagens naturais são utilizadas para expressar conhecimento, emoções e para fornecer respostas para quem está ao redor.

Embora seja de uso intuitivo para os seres humanos, a compreensão dessa linguagem representa um desafio significativo para os computadores. Dessa forma, surgiu o Processamento de Linguagem Natural (PLN), que é um campo de estudo que se encontra na interseção entre ciência da computação e linguística, com o propósito de possibilitar a comunicação entre computadores e a linguagem humana (TORFI et al., 2021).

Através do PLN é possível a extração de características textuais, o que permite a automação de tarefas de análise textual. Nesse contexto, técnicas capazes de processar a morfologia, fonologia, semântica e pragmática de uma entrada surgiram (RESHAMWALA; MISHRA; PAWAR, 2013). Assim sendo, serão definidos a seguir cada um desses campos de estudo.

A Morfologia é a primeira etapa da análise de uma entrada, ela é responsável por analisar os morfemas, as unidades mínimas de uma palavra. Os morfemas são utilizados para entender como esses componentes participam da construção do significado e suas variações em suas respectivas palavras (RESHAMWALA; MISHRA; PAWAR, 2013).

A semântica é o campo da linguística que se concentra em estudar o significado. Através da análise da representação de objetos e ações, e também do entendimento de como os adjetivos, advérbios, verbos e preposições são capazes de representar conceitos, sentimentos e relações (KHURANA et al., 2022) é possível determinar qual era a intenção do autor de um texto.

Apesar disso, linguagens naturais são intrinsecamente ambíguas, dessa forma, é

necessária uma análise de ambiguidade. A pragmática busca entender quando uma frase pode ter mais de uma interpretação, caracterizando uma a ambiguidade semântica (KHURANA et al., 2022). Esse é um problema desafiador para o computador, pois, os computadores operam com base em regras e algoritmos específicos, sendo necessários tratamentos para esse tipo de situação.

A fonologia lida com a interpretação de sons de fala, identificando sons de palavras, variação em pronunciação e entonação. Modelos de Processamento de Linguagem Natural que utilizam sons como entrada, processam e codificam as ondas sonoras em sinais digitais que podem ser interpretados por diferentes conjuntos de técnicas (RESHAMWALA; MISHRA; PAWAR, 2013). Portanto, será explicado como esses campos de estudo são aplicados no campo da Mineração de Texto.

2.3 Mineração de Texto

É necessário entender como coletar e avaliar os dados brutos fornecidos para ser possível estruturar e extrair informações relevantes de cada publicação. Cerca de 90% dos dados públicos globais estão em um formato bruto e desestruturado (DANG, 2015), sendo necessário, desenvolver um ferramental que possibilite coletar o significado de maneira automatizada, visto que, em grande escala, é inviável analisar manualmente.

Dessa forma, pode-se definir Mineração de Texto como a aplicação de um conjunto de algoritmos que irão converter dados desestruturados em estruturados, permitindo, assim, a extração e análise dos dados improcessáveis anteriormente. O uso pode variar para categorização, sumarização, recuperação e agrupamento de informação. Sendo assim, é possível identificar uma multidisciplinaridade entre Mineração de Texto, PLN, Recuperação de Informação, etc. (DANG, 2015).

O uso desses algoritmos está sendo refletido no dia-a-dia dos usuários, já que, o uso de redes sociais está em uma crescente[2]. Dessa forma, diversas aplicações de análise de texto em rede social foram implementadas buscando entender e filtrar o

[2]https://ourworldindata.org/rise-of-social-media

comportamento dos usuários. Impedindo ataques, discurso de ódio ou postagens indevidas[3].

A Mineração de Texto também pode ser usada para redirecionamento de anúncios, pois, através dos textos e opiniões de uma pessoa é possível entender qual categoria de consumidor ela se enquadra (DANG, 2015). Também tem crescido o interesse de monitorar opiniões sobre um governo ou eleição, como foi o caso da coleta de opiniões no Twitter sobre uma eleição da união europeia, que conseguiu capturar a presença de opiniões positivas e negativas sobre os candidatos em diversas línguas (YUE et al., 2019).

Eventos recentes demostram que redes sociais, como Twitter e Facebook, tem tido papel fundamental em acontecimentos sociopolíticos (EBNER, 2017), onde atuam como um ponto de encontro para organização política, notícias, discussão sobre conflitos globais, diplomacia, etc. Com a análise de texto é possível identificar informações sensíveis que poderiam ser importantes para a segurança de um país. Os casos mais notáveis são a Primavera Árabe e os tumultos na Inglaterra (YUE et al., 2019). Desta forma, com a definição da importância da Mineração de Texto é possível definir como o processamento irá acontecer.

O processo da Mineração de Texto pode ser dividido em cinco grandes etapas (SINOARA; ANTUNES; REZENDE, 2017). Conforme a figura 2.1, no primeiro passo é necessário definir um objetivo. Portanto, é necessário definir o escopo e também definir o conjunto de documentos para treinamento, formalmente definido como corpus. Após essas definições é possível definir qual será o objetivo e a finalidade da aplicação.

Em seguida, a segunda etapa, preparação dos dados, é executada. O objetivo da Mineração de Texto é identificar os padrões dos dados para executar a extração de conhecimento. Portanto, é nessa etapa que a transformação do dado bruto é feita para se adaptar ao padrão de entrada dos algoritmos de Mineração de Texto. Essa preparação é chamada de pré-processamento.

[3]https://support.discord.com/hc/pt-br/articles/4421269296535-FAQ-do-AutoMod

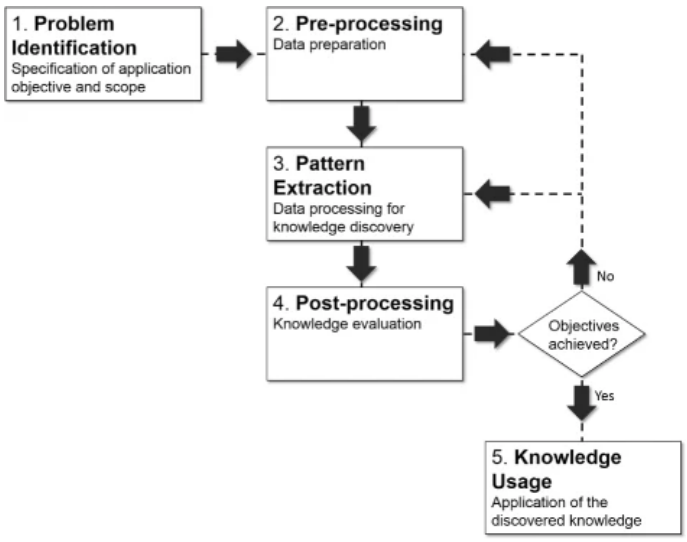

Figura 2.1: É possível observar os 5 passos do fluxo do processo de mineração de dados. O primeiro passo é a identificação do problema. O segundo passo é o pré-processamento dos dados. O terceiro passo é a extração de padrões. o Quarto passo é o pós-processamento, ou seja, avaliação do conhecimento. O quinto passo é o uso do conhecimento.

Fonte: (SINOARA; ANTUNES; REZENDE, 2017)

Após a preparação dos dados, é necessário executar a extração dos padrões. Nessa etapa é esperado que padrões implícitos dos dados sejam evidenciados pelo algoritmo. Esse algoritmo é escolhido com base em qual algoritmo lidará melhor para tratar o tipo de padrão de conhecimento que se deseja extrair, conforme especificado na primeira etapa.

Após a extração dos padrões é essencial avaliar os resultados, entender se os resultados obtidos são satisfatórios. Pois caso o objetivo não seja alcançado, uma nova iteração deve ser executada. Nessa fase diversas métricas são utilizadas para entender a qualidade do que foi produzido, pois, caso algum dos pontos não satisfaça, é necessário retornar a etapas anteriores. Dessa forma, após a aprovação do modelo, é possível começar a disponibilizar os recursos para os usuários, aplicando ao contexto para qual foi desenvolvido.

Assim sendo, é possível entrar em detalhes nas etapas mais complexas do processamento. Os dois passos mais cruciais para aplicações de extração de texto são o Pré-Processamento e a Representação Sintática do de Palavras.

2.3.1 Pré-Processamento de Texto

Quando se deseja analisar ou classificar texto é importante limpar os dados antes de repassar para o classificador. Dados textuais podem estar cheios de ruídos, capitalização distinta, gírias e abreviações, que podem prejudicar o desempenho do classificador (KOWSARI et al., 2019). Nessa seção serão apresentados e exemplificados os principais métodos utilizados durante o processo de tratamento de texto.

2.3.1.1 Tokenização

A tokenização é uma técnica onde documentos são transformados em tokens, ou seja, são separados em palavras, frases ou símbolos, essa ação permite que cada token possa ser processada individualmente, facilitando etapas posteriores do pré-processamento e do algoritmo (KOWSARI et al., 2019), como pode ser visto no exemplo a seguir:

> *Eu e meu amigo vamos ao show do Thundercat no prxoimo mês!!! MEUS AMIGOS!!! ESSE SHOW VAI SER BRABO D+!!!!!!!!*

Irá gerar os tokens:

> *"Eu", "e", "meu", "amigo", "vamo", "ao", "show", "do", "Thundercat", "no", "prxoimo", "mês!!!", "MEUS", "AMIGOS!!!", "ESSE", "SHOW", "VAI", "SER", "BRABO", "D+!!!!!!!!"*

2.3.1.2 Stop Words

Stop Words são palavras sem significado importante em um vocabulário. A língua possui conectivos e adjetivos que fazem sentido para um ser humano interpretar a sentença. Para uma máquina esses conectivos podem ser um dificultador para o processamento, pois são palavras comuns em todas as frases, então não representam nada importante para a classificação (KOWSARI et al., 2019).

A técnica mais comum para lidar com *Stop Words* é a remoção das mesmas. Após o processo de tokenização é feito uma detecção e remoção de todas as ocorrências. Dessa forma, palavras que possuem mais relevância serão mantidas ao texto, facilitando o processo de classificação. Dando continuidade ao exemplo acima, após as remoções serão mantidos os tokens:

> *"amigo", "vamos", "show", "Thundercat", "prxoimo", "mês!!!", "AMIGOS!!!", "SHOW", "BRABO", "D+!!!!!!!!"*

2.3.1.3 Capitalização

O computador interpreta letras maiúsculas e minúsculas como elementos diferentes. Deste modo, quando uma palavra é recebida em capitalizações distintas, ela será interpretada como palavras diferentes e independentes (KOWSARI et al., 2019). As palavras "Pão", "pão" e "PÃO" não serão agrupadas caso estejam com diferença de capitalização.

A técnica de capitalização busca converter todo o texto para minúsculo ou maiúsculo, dessa forma, as diferenças entre as palavras são resolvidas. Porém, problemas de ambiguidade podem surgir, por exemplo, "Aba", a borda do chapéu, é diferente de "ABA" (Associação Brasileira de Antropologia). Continuando o exemplo anterior, veja as manipulações abaixo:

> *"amigo", "vamos", "show", "thundercat", "prxoimo", "mês!!!" "amigos!!!", "show", "brabo", "d+!!!!!!!!"*

2.3.1.4 Abreviações e Gírias

O uso de gírias e abreviações é muito comum em textos em ambientes informais. Além disso, o ambiente da *internet* possui seu próprio conjunto de formas diferentes de escrever palavras e expressar reações e sentimentos.

Dessa forma, processar essa linguagem de forma algorítmica é uma ação complexa, já que uma palavra pode ter diferentes significados dependendo do contexto, também

depende da gíria que está sendo utilizada e também palavras abreviadas podem perder o significado (KOWSARI et al., 2019).

Dado esse problema, a frase a seguir exemplifica o surgimento de técnicas onde essas expressões podem ser substituídas pela linguagem formal, mantendo a intenção da frase e permitindo que ela seja melhor processada:

> *"amigo", "vamos", "show", "thundercat", "prxoimo", "mês!!!", "amigos!!!", "show", "legal", "demais!!!!!!!!"*

2.3.1.5 Remoção de Ruído

Semelhante as *Stop Words*, a remoção de ruído se refere a remoção de pontuação, *emojis* e outros caracteres especiais desnecessários. Esses caracteres não contribuem para o entendimento da máquina. Dessa forma, a remoção é recomendada. Como pode ser observada no exemplo a seguir, a frase resultante contém apenas as palavras.

> *"amigo", "vamos", "show", "thundercat", "prxoimo", "mês" "amigos", "show", "legal", "demais"*

2.3.1.6 Correção ortográfica

A correção ortográfica dos documentos é uma etapa importante do pré-processamento. Ela garante a padronização das palavras. Em alguns conjuntos de dados, os erros de escrita são muito comuns e os algoritmos podem processar essas palavras erradas como palavras novas e inferir outro significado, diferente do original pretendido, ou ser classificado como ruído (KOWSARI et al., 2019). Existem diversas técnicas que encontram a palavra correta a partir do erro de escrita. Esse problema é comum em contextos online como as redes sociais. No exemplo seguinte, podemos ver a correção da palavra "próximo", que estava incorreta nos exemplos anteriores.

> *"amigo", "vamos", "show", "thundercat", "próximo", "mês", "amigos", "show", "legal", "demais"*

2.3.1.7 Stemming e Radicalização

Dois conceitos semelhantes são o *Stemming* e a Radicalização. Ambos buscam reduzir todas as formas de uma palavra em uma única forma, ou seja, ocorrerá a remoção de plural, flexão de gênero, tempos verbais e outras variações (KOWSARI et al., 2019).

Uma palavra como "correr", "correndo", "correria" e "corre" podem ser resumidas em uma única palavra "corre" sem prejudicar o sentido da frase. Dessa forma, o classificador irá unificar o sentido de todas as variações, facilitando uma melhoria de desempenho.

A diferença entre cada uma das técnicas é que o *Stemming* não garante que a palavra raiz exista. Enquanto a radicalização, também chamada de *lemmatization*, irá necessariamente reduzir para uma palavra raiz existente. A frase a seguir demonstra um exemplo de *lemmatization*:

> *"amigo", "ir", "show", "thundercat", "próximo", "mê", "amigo", "show", "legal", "demais"*

2.3.2 Extração de Características

Na seção anterior foram apresentados métodos para limpar e tratar os dados textuais, a partir desse ponto é possível aplicar técnicas que transformam o texto pré-processado em uma entrada compatível com algoritmos de Mineração de Texto. Nessa seção serão introduzidas algumas técnicas de extração de características, sendo elas BoW, *N-Grams* e *Word Embeddings*.

2.3.2.1 Bag of Words

BoW é uma estrutura que busca reduzir e simplificar a representação de um texto para facilitar o entendimento pelo algoritmo. Essa técnica pode se basear na ocorrência de palavras ou frequência das palavras (KOWSARI et al., 2019).

O BoW irá construir um vetor contendo todos os termos únicos presentes no

corpus, dessa forma, a dimensão de um BoW é diretamente proporcional ao número de termos únicos do conjunto de dados. Para BoW a relação sintática e a ordem das palavras são ignoradas, portanto, é apenas capaz de identificar padrões de ocorrência de palavras.

Por exemplo, quando a palavra "Pão" aparecer pela primeira vez, ela será adicionada ao vetor. As próximas ocorrências irão ser computadas no mesmo local que a primeira. Dessa forma, a sintaxe será completamente perdida, visto que não terá garantia de ordem. Na tabela 2.3.2.1 é possível entender como será tratado no caso de existirem as frases "meu pai comprou pão na padaria do meu bairro" e "a padaria do outro bairro foi assaltada".

meu	pai	comprou	pão	na	padaria	do	bairro	a	outro	foi	assaltada
2	1	1	1	1	1	1	1	0	0	0	0
0	0	0	0	0	1	1	1	1	1	1	1

Tabela 2.1: Exemplo de formação do vetor de um BoW usando as frases "meu pai comprou pão na padaria do meu bairro" e "a padaria do outro bairro foi assaltada".

2.3.2.2 N-Gram

O N-*Gram* é uma forma de representar um texto por meio de uma sequência contínua de N elementos. Dessa forma, por exemplo, em uma frase cada *token* será gerado a partir N palavras. Assim, é possível inferir de forma mais precisa o significado original da frase (KOWSARI et al., 2019).

Além disso, a representação em *Bag of Words* vista anteriormente pode ser classificada como 1-*gram*. Exemplos de representação usando N-*gram*.

> Eu e meu amigo vamos ao show do Thundercat.
>
> 2-*gram*: *{"Eu e", "e meu", "meu amigo","amigo vamos", "vamos ao", "ao show","show do", "do Thundercat"}*
>
> 3-*gram*: *{"Eu e meu","e meu amigo", "meu amigo vamos", "vamos ao show", "ao show do","show do Thundercat"}*

2.3.2.3 TF-IDF

Um dos problemas da abordagem BoW é que termos muito frequentes nos documentos são geralmente pouco relevantes ao serem usadas para extração de significado ou alguma classificação. Por exemplo, ao usar uma base de dados de *e-mails* para classificar *spam*, termos como: bom, dia, noite, o, a, etc. aparecem em um grande número *e-mails* da base, tornando a relevância desses termos no objetivo de classificar *spam* muito baixa.

A proposta de JONES (1972) para o Inverse Document Frequency (IDF) é que termos implicitamente comuns em todo o *corpus* tenham efeito reduzido. Sendo assim, a combinação do Term Frequency (TF) com o IDF irá levar em consideração o contexto do *corpus* para definir o peso dos termos em cada documento, a fórmula do TF-IDF pode ser definida conforme a equação 2.1.

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

Conforme a explicação anterior, o termo $\mathrm{W}(d,t)$ representa o peso do termo t no documento d. $\mathrm{TF}(d,t)$ representa a frequência do termo t no documento d. $|N|$ representa o número total de documentos no *corpus* e $df(t)$ representa a frequência do termo t no total de documentos. Sendo assim, quanto mais documentos contiverem essa palavra específica, menos significativa ela será para a sentença.

O problema dessas abordagens anteriores é que cada token ocupa uma dimensão de um vetor de forma ortogonal. Portanto, mesmo que em algumas abordagens exista uma preocupação em preservar a sintaxe, a semântica pode ser perdida.

A técnica de *Word Embeddings* tem o objetivo de mapear a similaridade semântica de tokens de entrada em um vetor real N-dimensional que possa ser compreendido ao nível de máquina. *Word Embedding* utiliza Machine Learning para traduzir os tokens em uma entrada para o modelo, sendo as propostas mais populares *Word2Vec*, *GloVe* e *FastText* que serão discutidas a seguir.

2.3.2.4 Word2Vec

O Word to Vector (Word2Vec) é uma técnica de representação vetorial de palavras proposta por Tomas Mikolov em 2013 (MIKOLOV et al., 2013). A técnica utiliza *Shallow Neural Networks* para criar vetores densos, contínuos e de alta dimensionalidade para cada palavra, capazes de conter informações semânticas e contextuais.

O Word2Vec irá mapear palavras em vetores, de forma que, palavras semanticamente relacionadas sejam posicionadas próximas uma das outras, permitindo que operações vetoriais aconteçam entre as palavras. Dessa forma, é possível realizar as operações vetor(King) − vetor(Man) + vetor(Woman) que irão resultar em um vetor próximo ao vetor que representa a palavra Queen conforme exemplificado em (MIKOLOV et al., 2013). Na figura 2.2 é possível visualizar duas arquiteturas para aprendizado de representações de palavras.

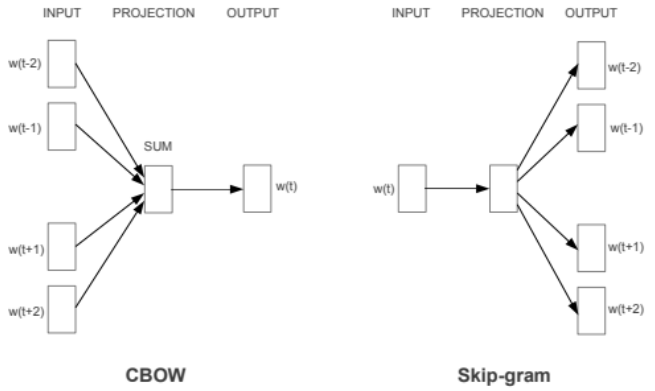

Figura 2.2: Arquiteturas de representações de palavras. O CBOW irá prever a palavra atual com base no contexto. O Skip-gram irá predizer as palavras ao redor com base na palavra atual.

Fonte: (MIKOLOV et al., 2013)

Continuous Bag-of-Words Model

O Continuous Bag-of-Words Model (CBOW) irá prever uma palavra com base em seu contexto, onde o contexto são as palavras que estão ao redor de uma palavra-alvo. O tamanho do contexto é um parâmetro variável, sendo a faixa entre quatro e cinco palavras mais comum. A base do CBOW é uma rede neural *FeedForward* onde se projeta para aprender a combinação de vetores frequentes da palavra-alvo

(MIKOLOV et al., 2013).

Através desse treino é possível aprender representações densas e contínuas que são capazes de representar sintática e semanticamente. Porém, o fato do modelo não considerar a ordem das palavras e produzir representações fixas o tornam uma má opção para tarefas maiores e mais complexas.

Continous Skip-Gram Model

A arquitetura do *Continuous Skip-Gram* é um pouco diferente, ele irá prever as palavras de contexto da palavra-alvo. Isso significa que ele irá tentar entender quais palavras são mais comuns entre si, criando uma representação vetorial densa para as palavras, onde a semântica estará vetorialmente próxima (MIKOLOV et al., 2013). Porém, sua maior desvantagem está em capturar contextos complexos e identificar contexto de palavras não frequentes.

2.3.2.5 GloVe

Global Vectors for Word Representation (GloVe) é uma técnica de *word embedding* similar ao *Word2Vec*, onde as palavras serão representadas em um vetor de alta dimensionalidade. Entretanto, o cálculo dos *embeddings* é realizado de uma forma diferente, a equação procura encontrar a frequência de co-ocorrência entre duas palavras em todo o *corpus*, dessa forma gerando uma matriz de similaridades que encontra a semântica em um *corpus* de texto (PENNINGTON; SOCHER; MANNING, 2014).

Além disso, é possível a utilização de modelos pré-treinados em grandes bases de dados, como a Wikipedia e o Twitter, permitindo uma melhor qualidade na geração dos *embeddings* em diferentes dimensionalidades, como 50, 100, 200 ou 300 dimensões conforme o tamanho do *corpus*.

2.3.2.6 FastText

O *FastText* visa levar em consideração a morfologia de palavras ao gerar os *embeddings*, em outros métodos, cada palavra é associada a um vetor distinto no

espaço, entretanto, na abordagem *FastText*, cada palavra é representada como um conjunto de caracteres em um n-grama, por exemplo, dado a palavra "*abaixando*", e $n = 3$, será obtido:

ab, aba, bai, aix, ixa, and, ndo, do

Cada n-grama é representado como um vetor de números reais, e a representação de uma palavra inteira se dá por uma função de pontuação calculada usando todos n-gramas que a compõe. Dessa forma, palavras com morfologias similares tendem a ficar próximas, já que serão compostas por alguns dos mesmos n-gramas (BOJANOWSKI et al., 2017).

2.3.2.7 Representação Contextualizadas de Palavras

Essa técnica utiliza o *context2vec*, uma forma de *embeddings* baseada no CBOW, porém, substitui a forma de modelagem de contexto, que utiliza uma janela fixa, por uma rede neural mais robusta, fazendo o uso de uma rede Long Short-Term Memory (LSTM) bidirecional.

Dessa forma, é possível extrair características complexas das palavras, como sintaxe e semântica, além de distinguir diferentes significados de uma mesma palavra com base no contexto atual (MELAMUD; GOLDBERGER; DAGAN, 2016).

2.4 Classificação de Texto

A classificação de texto é uma área que se dedica à categorização de documentos em classes. Dado que o foco deste estudo é a detecção de discurso de ódio, a técnica de classificação de texto adotada será a análise de sentimentos.

2.4.1 Análise de Sentimento

Análise de Sentimento ou Mineração de Opinião é uma aplicação dos conceitos mencionados anteriormente que busca entender opiniões, atitudes e emoções de uma

entidade (MEDHAT; HASSAN; KORASHY, 2014). No caso específico desse trabalho as entidades são usuários de uma rede social, porém, podem ser empresas, eventos, tópicos, etc.

A Análise de Sentimento pode ser definida como a classificação de um documento, sentença ou aspecto para definição de polaridade (YUE et al., 2019). Essas avaliações precisas podem ser úteis em diversas áreas, por exemplo, identificação da opinião de usuários sobre um produto, identificação da polarização de campanhas políticas, etc. (MEDHAT; HASSAN; KORASHY, 2014).

A classificação de texto, em termos técnicos, se concentra na classificação de um documento textual d_i que pertence a um conjunto de documentos D, a uma categoria específica em um conjunto predeterminado de categorias, representado como $C = \{c_1, c_2, c_3, \ldots, c_n\}$. O objetivo principal é empregar métodos de aprendizado de máquina, PLN ou processamento estatístico para realizar essa atribuição com base nas características semânticas, sintáticas e contextuais do documento (ABIKOYE; OMOKANYE; ARO, 2018).

2.4.2 Algoritmos e Aplicações

Nesta subseção serão explorados técnicas de classificação de documentos, de forma, que será possível entender os pontos fortes e fracos de cada abordagem. Serão discutidos métodos como *Naive Bayes*, *Support Vector Machine (SVM)* e *Random Forest* que são tradicionalmente adotados pela literatura. Também serão descritas técnicas baseadas em redes neurais profundas, como LSTM e Transformers (KOWSARI et al., 2019).

2.4.2.1 Naive Bayes

O *Naive Bayes* é um classificador de texto baseado no teorema de Bayes. Por isso, a classificação se dá a partir do cálculo de probabilidades condicionais do conjunto de dados. O algoritmo observa a frequência de ocorrência de palavras e indica a probabilidade de uma sentença pertencer a uma classe pré-determinada (KOWSARI

et al., 2019).

Como esse método é puramente probabilístico, ele supõe que todos os elementos probabilísticos são independentes entre si (MANNING; RAGHAVAN; SCHüTZE, 2008), o que na maioria das vezes não é verdade. Apesar disso, o *Naive Bayes* é um método bastante utilizado, já que, possui um desempenho em um conjunto elevado de tarefas e não depende de treinamento prévio (KIM et al., 2006).

2.4.2.2 Support Vector Machine

O SVM é originalmente um algoritmo de classificação que maximiza uma função de decisão linear visando dividir uma amostra em dois grupos distintos (EVGENIOU; PONTIL, 2001). Entretanto, há variações onde é possível utilizar funções não-lineares e classificar em mais de dois grupos (KOWSARI et al., 2019).

Diferente do *Naive Bayes* o SVM precisa de treinamento prévio, otimizando a função e encontrando parâmetros que melhor dividem os grupos de dados. Após isso, ao adicionar novos dados ao conjunto, essa função de decisão previamente calculada pode ser re-utilizada.

2.4.2.3 Random Forest

O algoritmo de classificação *Random Forest* envolve a construção de múltiplas árvores de decisão, cada uma delas sendo construída com uma amostra aleatória dos dados de treinamento. Além disso, a escolha das variáveis em cada nó também é realizada de maneira aleatória (HO, 1995). Essa abordagem tem o objetivo de mitigar o *overfitting* e melhorar a acurácia, problemas comuns nas árvores de decisão.

O *Random Forest* mantém a velocidade de treinamento, facilidade de entendimento e usabilidade das árvores de decisão. A classificação do método é definida por uma função que avalia a decisão de cada árvore e realiza uma média ou regressão.

2.4.2.4 Long Short Term Memory (LSTM)

Recurrent Neural Network (RNN) é uma arquitetura que pode ser utilizada em mineração de texto e classificação de sequências, como *tweets* e séries temporais. A premissa é baseada na influência das informações obtidas nos dados anteriores na classificação atual, de uma maneira sofisticada a arquitetura irá fazer uma análise semântica da estrutura do conjunto de dados (KOWSARI et al., 2019).

A LSTM é um tipo de RNN que é capaz de armazenar a dependência dos elementos de entrada de maneira mais eficiente. Pode ser extremamente útil em séries temporais ou com influência histórica (HOCHREITER; SCHMIDHUBER, 1997). Essa arquitetura conta com uma camada de entrada, que será a representação sintática da frase, como o *Word Embedding*. As camadas ocultas e uma camada de saída que terá o número de classes possíveis para a classificação, conforme pode ser visualizado na figura 2.3.

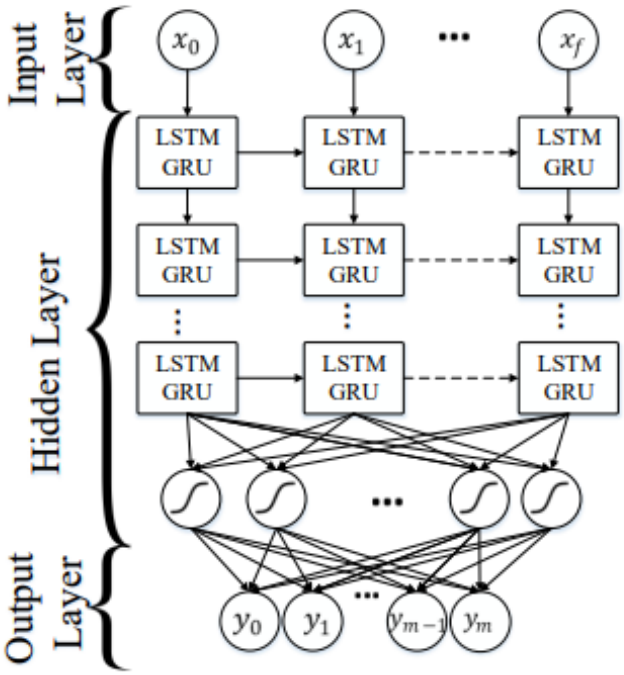

Figura 2.3: Desenho geral da arquitetura de um LSTM/GRU

Fonte: (KOWSARI et al., 2019)

Por conta da estrutura em cadeia, característica de RNNs, e da adição dos múltiplos *gates* é possível regular como as sequências estão influenciando os próximos

estados, conforme a figura 2.4. Com a entrada contínua da sequência, a rede neural será capaz de guardar e entender as informações das palavras anteriores e irá utilizar no processo de classificação.

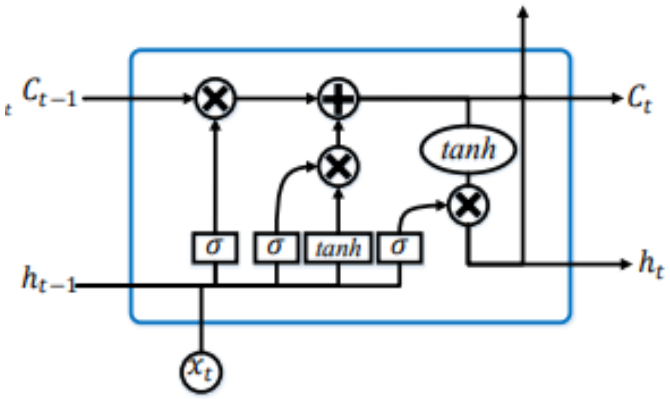

Figura 2.4: A figura acima representa uma célula LSTM

Fonte: (KOWSARI et al., 2019)

Apesar das diversas vantagens para análise mais precisa da semântica dos textos, as LSTMs podem não desempenhar tão bem a análise de dependência a longo prazo. Textos muito longos podem ser ineficientes para as LSTMs incapazes de manter o contexto durante toda a sentença.

2.4.2.5 Transformers

A arquitetura Transformer surgiu representando um avanço no estado da arte para PLN sendo utilizado em diversas aplicações, sendo algumas notáveis o GPT-2 (RADFORD et al., 2019), GPT-3 (BROWN et al., 2020), BERT (DEVLIN et al., 2019), AlphaStar (MATHIEU et al., 2023) e RoBERTa (LIU et al., 2019). A arquitetura é capaz de obter uma alta desempenho para tarefas envolvendo texto, como tradução instantânea, extração de informação e análise de sentimentos.

A proposta inicial do Transformer foi produzir um modelo capaz de alcançar um novo estado da arte na qualidade de traduções e conseguir aumentar o desempenho algorítmico em ambiente paralelo (VASWANI et al., 2023). Consequentemente, modelos derivados do Transformer alcançaram o estado da arte em outras tarefas

como visão computacional e processamento de áudio (LIN et al., 2021).

O Transformer tem como conceito principal o mecanismo de *self-attention*, ele processa os *embeddings* e permite que o modelo analise partes específicas do texto, definindo a importância de cada *token*. Dessa forma, relações complexas e o contexto serão considerados na análise textual (LIN et al., 2021). Além disso, o *self-attention* permite que todos *embeddings* sejam processados juntos e de forma que a sequência temporal não seja necessária. Por isso, é possível acelerar o tempo de processamento por meio da computação distribuída.

Capítulo 3

Proposta

Este capítulo irá discorrer sobre a motivação para o desenvolvimento desse trabalho, apresentando questões sobre o crescente uso de redes sociais, e como essas podem influenciar as emoções e o processo de decisão das pessoas. Ele também irá contar discussões sobre trabalhos relacionados e por fim a proposta para solucionar esse problema.

3.1 Motivação

Segundo a W3C (2011), consórcio internacional de padronização da internet, a fase da Web 2.0 teve seu início em 2004. Conforme o relatório do IBGE (2019), esse marco na história da internet acompanha um movimento de democratização do acesso a computadores e consequentemente aumento do número de usuários na internet (ROSER; RITCHIE; ORTIZ-OSPINA, 2015) fruto do barateamento de dispositivos eletrônicos e incentivos governamentais para expansão da internet[1]. Desde então, a quantidade de usuários na internet passou de aproximadamente 2.6 milhões no ano 1990, para 4.8 bilhões de usuários em 2021(KEMP, 2021; ROSER; RITCHIE; ORTIZ-OSPINA, 2015).

A Web 2.0 trouxe características muito distintas do seu predecessor. Os sites

[1]https://www.state.gov/declaration-for-the-future-of-the-internet

simples, estáticos e pessoais das décadas anteriores foram ofuscados por redes sociais e *e-commerces* gerenciados por grandes corporações com objetivo de produzir interações em tempo real para seu número crescente de usuários. A partir de então, a interação massiva entre usuários foi facilitada, permitindo que cada utilizador possa emitir sua própria opinião e participar de discussões e conversas instantaneamente, seja com tom positivo ou negativo.

Nesse contexto surgiram sites como MySpace, YouTube, Facebook e 4chan, onde, o nível de autenticação era baixo ou nulo, permitindo um pseudoanonimato de seus usuários, facilitando a ocorrência de crimes virtuais, como *cyberbullying* e roubo de dados. Atualmente, das 7.9 bilhões de pessoas no mundo, aproximadamente 4.6 bilhões estão nas redes sociais, que representa cerca de 59% da população mundial (KEMP, 2021).

Conforme descrito por Elgendy e Elragal (2014), a crescente de usuários resultou no fenômeno do *Big Data*, que consiste numa coleção de dados extensos, variados e gerados em alta velocidade, dessa maneira, é impossível que humanos ou softwares tradicionais processem essa quantidade de dados em busca de crimes e anomalias, permitindo que pessoas consigam expor opiniões criminosas, prejudiciais e até conteúdos sensíveis[2] sem detecção imediata e de foma anonima através de contas falsas. Por exemplo, a rede social Twitter produz diariamente mais de 500 milhões de publicações[3], sendo comum encontrar discursos de ódio nesses dados.

Na internet é possível mascarar a verdadeira identidade e localização de um usuário. Dessa forma, a anonimidade aliada ao grande volume de dados produzidos contribui diretamente para a existência de conteúdo de discurso de ódio nas redes sociais (VIDGEN et al., 2019).

Conforme as circunstâncias apresentadas, a questão do crime virtual se tornou mais relevante quando ataques cibernéticos começaram a se tornar mais frequentes. Segundo o relatório sobre *cyberbullying* e discurso de ódio do Ditch The Label[4], o

[2]https://www.washingtonpost.com/technology/2019/07/25/social-media-companies-are-outsourcing-their-dirty-work-philippines-generation-workers-is-paying-price/

[3]https://www.internetlivestats.com/twitter-statistics/

[4]https://www.ditchthelabel.org/research-papers/cyberbullying-and-hate-speech/

ódio direcionado se intensificou durante a pandemia de 2020.

Os relatórios de 2021[5] apontam que uma publicação contendo ódio racial foi publicada em média a cada 1.7 segundos durante o período pandêmico. Tal problemática está preocupando diversas empresas, que buscam desenvolver soluções para bloquear o ódio antes mesmo que seja publicado, como o caso do AutoMod do Discord [6], porém, ainda não existe uma solução definitiva para o problema.

Uma maneira de detectar alvos de toxicidade, é monitorar os assuntos em alta de uma rede social em busca de publicações contendo ódio e reportá-los em uma plataforma externa. Por exemplo, o Bot Sentinel foi criado para monitorar publicações no Twitter, conhecida por ser uma rede social que confere liberdade aos usuários publicarem o que desejam com pouca verificação.

Ao visualizar essa problemática no contexto do Brasil, é possível perceber que internet e as redes sociais estão mais presente na vida dos brasileiros, atualmente existem 171.5 milhões de usuários nas redes sociais, representando, numa proporção direta, 79% da população dos pais (KEMP, 2022). Contudo, apesar de diversas aplicações e pesquisas, pouco tem sido analisado em língua portuguesa, sendo necessário o monitoramento e análise dos espaços sociais virtuais para promover um ambiente mais seguro.

3.2 Trabalhos Relacionados

O Bot Sentinel surgiu em 2018 visando combater a desinformação e assédio direcionado dentro do Twitter (BOUZY, 2018). O Sentinel utiliza modelos preditivos em sua classificação e treina baseado nas regras de conduta do Twitter, tendo com alvo contas que violavam essas regras repetidas vezes, monitorando usuários e classificando entre quatro categorias, sendo elas Normal, Satisfatório, Disruptivo e Problemático. Também classifica os assuntos em alta, movendo-as para estado de alerta em caso de atividade suspeita, como, por exemplo, assuntos com alta presença

[5]https://www.ditchthelabel.org/research-papers/hate-speech-report-2021/
[6]https://discord.com/blog/automod-launch-automatic-community-moderation

de contas inautênticas e usuários tóxicos. O sentinela classifica em contas **ativas** ou **suspensas** e na tabela 3.2 é possível observar a quantidade de contas suspensas crescendo conforme o nível de problemas que representam, e também é possível visualizar a tela principal do sistema na figura 3.1

	Normal	Satisfactory	Disruptive	Problematic
Total	1.471.636	548.010	379.336	223.105
Suspensos	11.53%	23.31%	32.03%	41.52%

Tabela 3.1: Tabela de proporção de contas suspensas

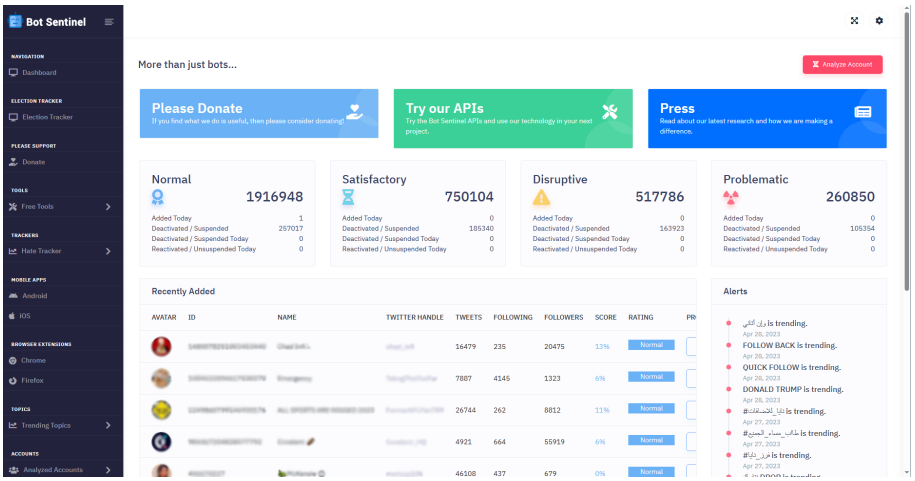

Figura 3.1: Tela inicial do *website* do Bot Sentinel

O PEGABOT[7] atua identificando contas controladas por robôs no Twitter, nasceu da necessidade da identificação de redes de bots maliciosos, pois, estes ajudam a promover *phishing*, *spam* e fraudes. O PEGABOT fornece a informação crua, para que assim seja possível fazer análises mais profundas a partir dos interesses específicos, muito útil para jornalistas, órgãos governamentais e outras entidades especializadas, a interface do sistema pode ser visualizada na figura 3.2.

O MEE6[8] atua como um automoderador no Discord, um ambiente mais privado que o Twitter. O *bot* identifica padrões de *spam*, links suspeitos e insultos e aplica punições conforme a quantidade de vezes que determinada ação foi tomada, o painel de configuração do sistema pode ser visualizado na figura 3.3

[7] <https://itsrio.org/pt/projetos/pegabot/>
[8] <https://mee6.xyz/pt>

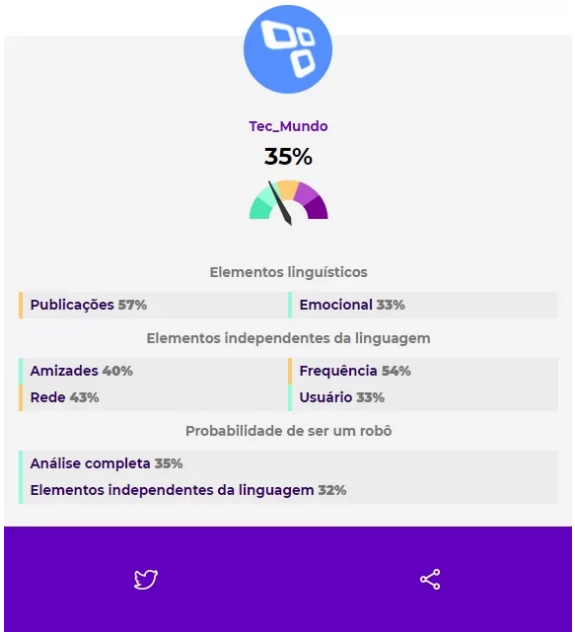

Figura 3.2: Interface de análise de perfis do PEGABOT

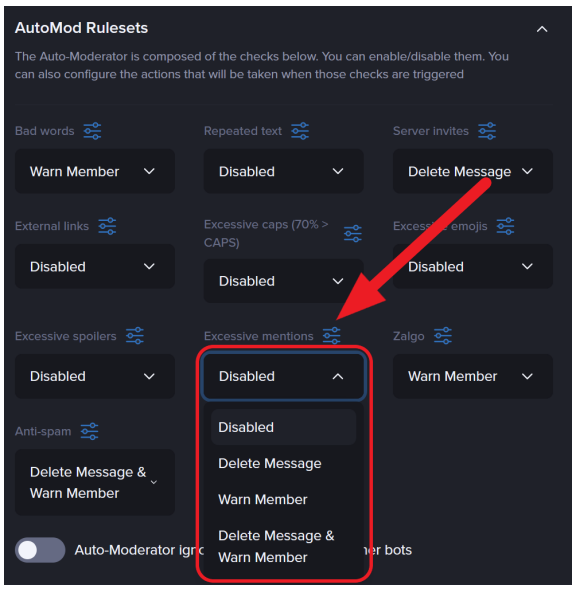

Figura 3.3: Painel de Configuração do AutoMod do MEE6

Adicionalmente, foram pesquisadas técnicas para detecção de comentários ofensivos. Conforme explicado na motivação, o objetivo do trabalho é detectar e analisar discurso de ódio publicações em língua portuguesa. Um dos trabalhos relacionados em destaque foi o Pelle (2021), onde, através da combinação de três técnicas de classificação textual, é produzido um classificador que realizará predições com base nos resultados dos três classificadores.

O primeiro classificador, *HateWord2Vec*, é baseado em um léxico e em uma lista de palavras ofensivas expandidas pelo *word embedding*. Através do uso de Word2Vec, o modelo irá produzir um vetor para cada palavra, que quando combinados irá comparar com a similaridade com as palavras ofensivas. O segundo classificador é um algoritmo de *Logistic Regression* usando *Doc2Vec* e como terceiro modelo adotado é um Bag-of-Words (BOW) em conjunto com um SVM.

Fortuna et al. (2019) fornece um conjunto de dados com 5668 tweets em português classificados com a presença de discurso de ódio. Além disso, foi disponibilizada uma versão desse conjunto de dados onde o ódio é classificado em diferentes categorias, como "racismo", "sexismo" e "homofobia". Dessa forma, é possível analisar de forma mais detalhada as publicações e gerar um modelo mais preciso.

3.3 Proposta

Conforme o cenário apresentado anteriormente, é evidenciada a necessidade de detectar discursos de ódio na internet rapidamente. Institutos como o The Alan Turing Institute[9] tem pesquisado formas de detectar e mitigar a disseminação de ódio virtual. De forma similar, o intuito do trabalho é desenvolver uma ferramenta de monitoramento de discurso de ódio em tempo-real, responsável pela identificação de possíveis focos de discurso de ódio e reportar os resultados em um painel de monitoramento.

Este trabalho foi estendido de um sistema *web* prévio, criado para o monitoramento de *tweets*. Dessa forma, a parte do trabalho que integra com o Twitter foi adaptada desse sistema, que pode ser visualizado na figura 3.4, onde é possível visualizar as integrações com o navegador e as rotinas de monitoramento com a rede social, que serão explicadas mais adiante. A partir disso, foi desenvolvida uma integração com esse projeto, conforme a figura 3.5, que foi dividida em dois módulos separados, um para exibição e monitoramento da rede social e um segundo para classificação dos dados recebidos.

[9] <https://www.turing.ac.uk/research/research-projects/hate-speech-measures-and-counter-measures>

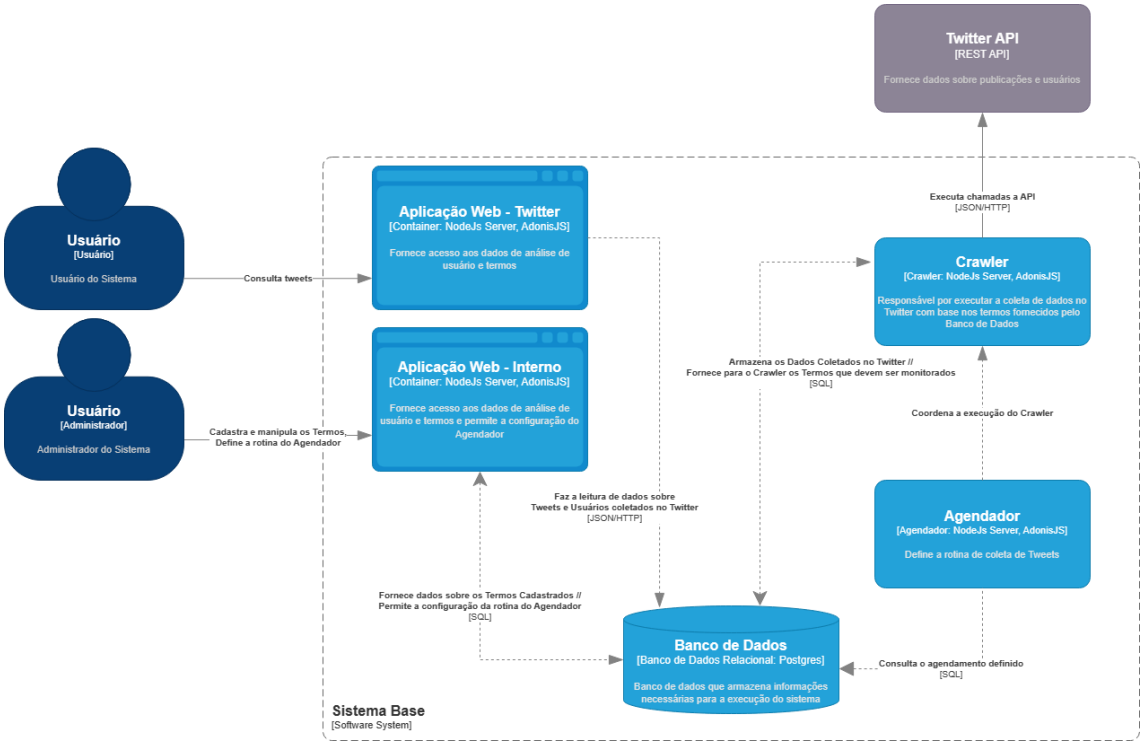

Figura 3.4: Arquitetura do sistema que foi utilizado com base para o sistema desse trabalho. Nele é possível visualizar as camadas de interface e integração com o Twitter

O primeiro módulo, originado da aplicação base, é um sistema *web* que tem como responsabilidade a interação com os usuários do aplicativo e a exibição das estatísticas sobre os dados coletados e classificados. Dessa forma, é possível realizar a inserção de novos tópicos para monitoramento, visualizar os resultados dos últimos monitoramentos e a configurar as rotinas da aplicação. Adicionalmente, essa aplicação terá a responsabilidade de executar a coleta de dados por meio de um *crawler* que utiliza a Application Programming Interface (API) do Twitter.

O segundo módulo dessa aplicação tem a responsabilidade realizar a classificação textual, ou seja, coletar os dados não classificados, identificar a presença de discurso de ódio e retorná-los para o consumo do sistema *web*. Dessa forma, será possível minimizar as alterações no sistema base, e executar a classificação dos dados de forma não bloqueante e independente.

Apesar disso, nos casos em que o sistema buscasse a integração entre diversas fontes de dados, ou seja, várias redes sociais, é recomendável que as estruturas de

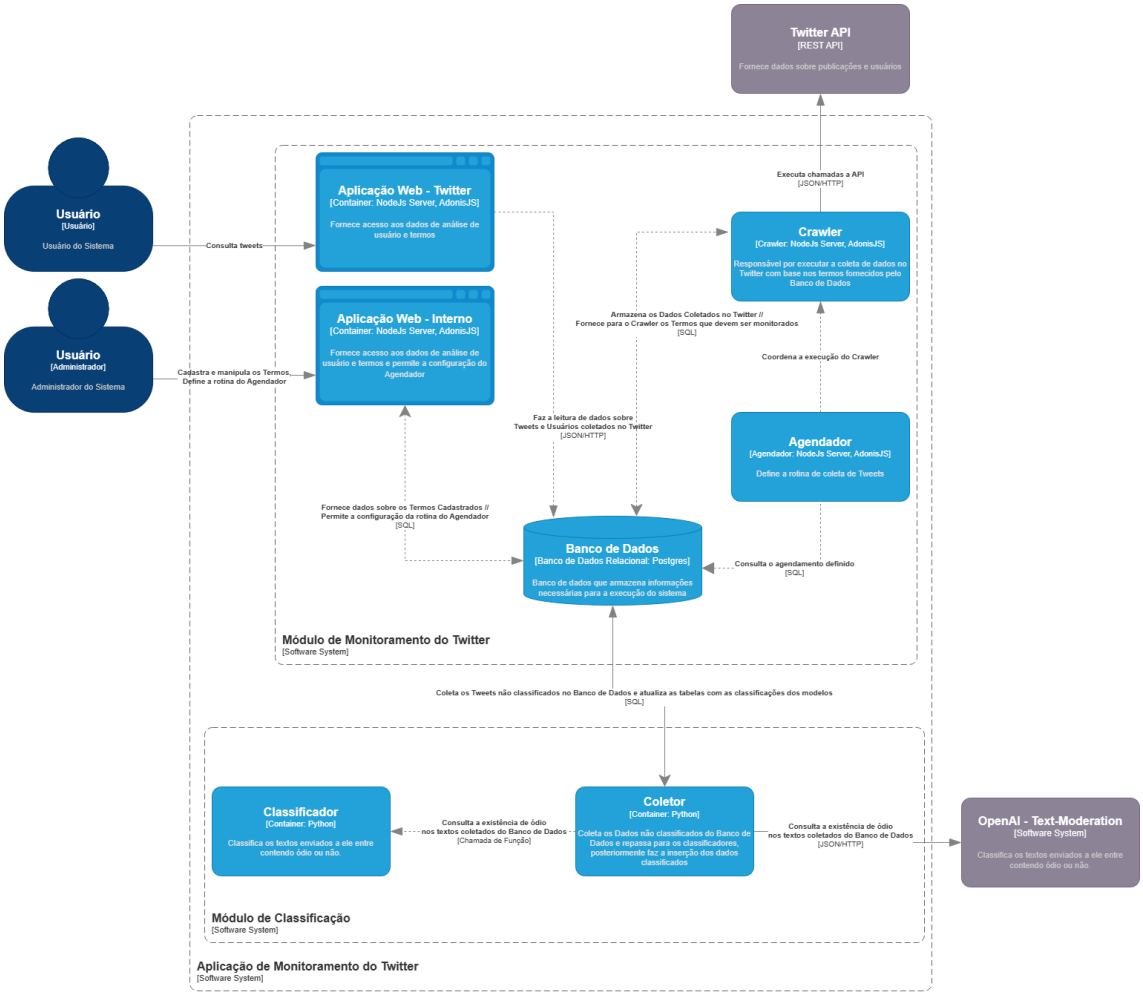

Figura 3.5: Diagrama do sistema base com a adição do Sistema de Classificação que faz a integração com a OpenAI.

coleta de dados sejam desacopladas do restante da aplicação, pois assim, será possível um maior controle e compartimentalização do processo, além, de limitar o escopo da aplicação *web* apenas para a exibição dos dados.

Conforme mencionado anteriormente, a estrutura da aplicação *web* original estava preparada apenas para a integração com o Twitter, dessa forma, foi adicionalmente desenvolvida uma integração com o Discord. Essa integração foi desenvolvida como uma nova aplicação, conectando-se exclusivamente aos modelos de classificação, conforme representado na figura 3.6.

Portanto, nas próximas seções do trabalho será feita uma análise de cada aplicação do sistema. Adicionalmente, a razão da escolha da rede social como objeto de estudo

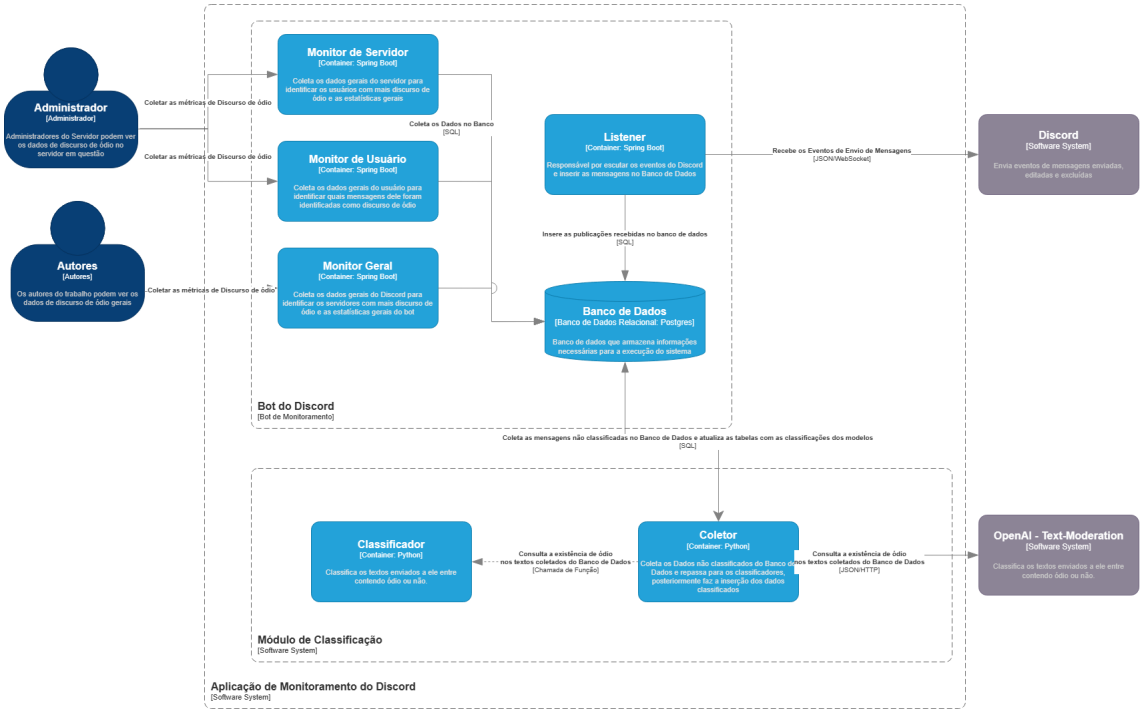

Figura 3.6: Diagrama de Integração do sistema proposto com o Discord e com a OpenAI

também será justificada.

3.3.1 Fonte dos Dados

A decisão de utilizar o Twitter como a rede social a ser monitorada se dá ao fato do formato majoritário da comunicação do Twitter ser textual, esta é uma característica que diferencia essa rede social de outras plataformas como Instagram, Facebook e TikTok. Essa particularidade torna o Twitter uma plataforma mais propícia para extração e análise de conteúdo textual, facilitando a identificação de temas e padrões de comportamento.

Além disso, o Twitter é amplamente utilizado como ferramenta de marketing e mobilização socio-política, atraindo milhões de usuários e promovendo uma grande heterogeneidade de discursos. Essas características tornam a base de dados do Twitter uma fonte de alta qualidade e relevante para a análise de opinião pública e comportamento social.

Porém, após a venda do Twitter em 2022[10] e a reformulação da marca para X, o nível de acesso a API sofreu uma série de limitações nos planos gratuitos e universitários. Contudo, antes desse evento, quando este trabalho começou a ser desenvolvido, o Twitter possuía uma API de fácil acesso, documentação robusta e plano gratuito para testes.

Outra rede social que apresenta um plano gratuito para pesquisa é o TikTok. Através da Research API[11] é possível coletar dados de publicações, como comentários, descrição e legendas. Porém, para ter acesso a esse plano é necessário fazer parte de um instituto acadêmico nos Estados Unidos ou na Europa.

A API fornecida pela Meta é mais limitada que a do Twitter para a escolha de dados, pois é necessário fornecer uma especificação sobre o que está buscando, portanto, foi considerada. Não é possível fazer busca de publicações sem fornecer uma *hashtag*, geolocalização ou usuário. Assim, caracteriza uma desvantagem para a proposta do trabalho, que busca monitorar dados de qualquer nova postagem na rede social.

A API do Discord foi definida como a segunda integração para o trabalho. O ecossistema do Discord permite a existência de usuários robôs para monitorar e executar ações na plataforma de forma gratuita[12]. Por consequência, a comunidade construiu um ambiente com muitas extensões e *wrappers* para desenvolvimento, como o JDA[13], discord.js[14] e o curupira[15].

Além disso, o Discord é uma opção relevante para a aplicação do trabalho devido à existência de um servidor do Departamento de Ciência da Computação (DCC) da Universidade Federal Rural do Rio de Janeiro (UFRRJ), nele ocorre a interação entre alunos e professores. Por isso, esse trabalho poderia ser usado como base de monitoramento nesse servidor e potencializar um automoderador personalizado. Dessa forma, será definido como o monitoramento e coleta de dados irão acontecer

[10] <https://www.bbc.com/portuguese/internacional-63422571>
[11] <https://developers.tiktok.com/products/research-api/>
[12] <https://discord.com/developers/docs/intro>
[13] <https://github.com/discord-jda/JDA>
[14] <https://discord.js.org/>
[15] <https://github.com/Softawii/curupira>

no Twitter e no Discord.

3.3.2 Aplicação de Monitoramento do Twitter

O monitoramento acontecerá por meio de um *dashboard* de tweets e usuários, que possibilitará a visualização e automatização da coleta dos *tweets*. Além disso, o resultado da análise de sentimento das publicações, perfis ou palavras-chave também serão exibidos. O *dashboard* será estendido de um projeto prévio, e apenas adaptado para melhor servir o contexto atual.

A decisão de utilizar um *dashboard* foi tomada para ser possível centralizar as análises de discurso de ódio, a visualização dos dados e o monitoramento dos assuntos em um único lugar. Além disso, abre possibilidade para expansão e integração com qualquer plataforma, como o Discord e WhatsApp. Entretanto, para cada nova integração será necessária adaptação do código-fonte.

A coleta de dados do Twitter é feita através de uma API fornecida pela própria rede social. Nesse caso, será utilizado um *crawler* monitorar, coletar e filtrar publicações do Twitter que contenham termos cadastrados.

Além disso, o termo é uma ferramenta interna que permite agrupar publicações por meio de um conjunto de palavras especificado, no contexto desse trabalho, isso pode ser usado para monitorar assuntos perigosos ou em alta na plataforma. Para o *crawler* ser ativado existe um agendador integrado a aplicação, este irá permitir a execução de uma rotina de monitoramento, sua função é definir o momento que os termos serão coletados, para que assim, o Twitter não bloqueie a aplicação devido a um número excessivo de requisições em um curto período.

3.3.2.1 Agendador

Um agendador é um componente que permite programar a execução de tarefas em um momento futuro ou em intervalos regulares. No contexto do trabalho será feita a coleta de novas publicações para armazená-las em um banco de dados para análise posterior. Foi definido que o uso do agendador ao nível de aplicação seria utilizado,

pois permite maior flexibilidade na escolha do sistema operacional e descentraliza a necessidade de configurar o agendamento no sistema, como no crontab do Linux.

Muitas linguagens de programação, como Java, JavaScript e Python, por exemplo, oferecem pacotes nativos que permitem a programação de tarefas em intervalos regulares. Isso pode facilitar a implementação de tarefas agendadas diretamente na aplicação, sem a necessidade de um agendador de sistema, flexibilizando a escolha de tecnologia e sistema operacional.

Os pontos fortes do uso de um agendador incluem a facilidade de uso e a capacidade de automatizar a execução tarefas repetitivas, como a coleta de publicações. Isso permite que o foco do desenvolvimento esteja em partes mais importantes da aplicação. Além disso, o uso de um agendador pode ajudar a melhorar a escalabilidade da aplicação, permitindo que ela processe um grande volume de dados sem afetar o desempenho da aplicação principal, visto que o processamento será feito em intervalos regulares em um contexto separado.

No entanto, o uso de um agendador também apresenta alguns pontos fracos. O agendador pode não ser adequado para tarefas que precisam ser executadas em resposta a eventos em tempo real, como a detecção imediata de publicações ofensivas. Nesses casos, outras abordagens de processamento em tempo real, como o processamento de fluxo de dados, podem ser mais adequadas. Porém, a aplicação deste trabalho não tem objetivo de banir contas automaticamente e sim fornecer análises periódicas do que está acontecendo na rede social. A seguir, será desenvolvido a maneira quais dados serão coletados e como os dados serão armazenados durante as rotinas de coleta.

3.3.2.2 Banco de Dados e Armazenamento

O banco de dados irá possuir tabelas para armazenar detalhadamente publicações, perfis, usuários e também o resultado da análise de sentimento. Dessa forma, será possível avaliar individualmente a presença de ódio em determinados perfis ou termos.

A classificação do sentimento e discurso de ódio de um texto é realizada por

meio de um modelo de aprendizado de máquina alocado em outra aplicação. Dessa forma, a aplicação irá fazer consultas assíncronas para atualizar as tabelas, assim, será possível evitar uma sobrecarga de processamento de publicações.

A escolha de alocar uma aplicação dedicada para o modelo de aprendizado de máquina foi tomada devido à facilidade de encapsular, desacoplar e abstrair. Isso permite que a estrutura interna do modelo seja alterada sem necessidade de refatorar o código do programa principal. Permitindo assim, alterar o modelo utilizado a qualquer momento.

Cada nova publicação capturada pelo *crawler* deverá ser adicionado ao banco de dados e ficará pendente de avaliação até que a rotina de comunicação com a API seja executada. Portanto, se faz necessário a criação de uma interface que terá a responsabilidade de entender os textos, classificá-los, e devolver os resultados para o banco de dados.

3.3.2.3 Interface do Classificador

O classificador pode interagir com o restante do ambiente de diversas maneiras. A primeira possibilidade é criar uma arquitetura de *pooling* com o banco de dados através de uma Stored Procedure (SP) que irá fornecer os dados não classificados, dessa forma, a aplicação irá classificar os dados e retornar para o banco de dados. Essa forma, permite que o classificador seja independente do *crawler* e do *dashboard*, caracterizando uma vantagem para o desacoplamento da aplicação. Além, de ser mais uma solução limpa para a manutenção do projeto.

Uma segunda possibilidade seria a integração do *crawler* diretamente com o modelo de aprendizado de máquina, nesse caso, a tecnologia escolhida para essa interface seria a Representational State Transfer (REST), devido à sua simplicidade e quantidade de material gratuito disponível. Porém, a sobrecarga do protocolo *HTTP*[16] e o custo de decodificação do formato *JSON*[17] podem causar lentidão no processamento de solicitações e respostas.

[16]https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Overview
[17]https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/JSON

A avaliação de textos seria realizada por meio de uma requisição *GET*, que irá retornar significado do texto. Porém, para cada nova integração com uma fonte de dados seria necessário a re-implementação, dessa forma, caracteriza um ponto negativo para a escalabilidade da implementação.

Além disso, outro problema é um *overhead* na disponibilidade dos dados para o usuário do *dashboard*. As publicações seriam disponibilizadas após a coleta de dados e a classificação, o contrário da proposta de criar um *pooling* com o banco de dados, onde todas as publicações seriam disponibilizadas para consumo imediatamente, apenas tendo a classificação dos dados preenchida de forma assíncrona.

Uma solução elegante, porém mais complexa, é a utilização de um *streaming* de dados, por exemplo, através do uso do Apache Kafka[18] ou o Amazon Kinesis[19]. Dessa forma, a classificação dos dados será feita em tempo-real, não sendo necessário esperar que a rotina de classificação seja executada. Portanto, para a simplicidade do fluxo para a prova de conceito do trabalho, a solução mais simples e eficiente é o uso da arquitetura de *pooling*.

3.3.3 Aplicação de Monitoramento no Discord

O monitoramento do Discord será feito através da API fornecida pela própria plataforma. A coleta de mensagens é feita a partir do repasse de eventos do *WebSocket* do Discord e são posteriormente armazenadas no banco de dados da aplicação. A partir desse momento, a classificação dos dados fica a cargo do classificador externo, semelhante ao processo que acontece com o Twitter.

A aplicação não foi construída visando automoderar o servidor, ela irá coletar e classificar os dados, para que assim, seja possível fornecer dados estatísticos sobre os usuários aos administradores. Porém, com a validação de um modelo eficiente é possível adaptar a aplicação para aplicar punições automaticamente.

[18] <https://kafka.apache.org/>
[19] <https://aws.amazon.com/pt/kinesis/>

3.3.4 Classificação Textual

A classificação textual é a etapa do trabalho onde será identificado o discurso de ódio nos textos utilizando modelos de inteligência artificial. Tendo em vista que a arquitetura proposta procura desacoplar essa etapa e gerar maior flexibilidade para diferentes implementações, planejou-se a implementação de duas variações de modelo de classificação para a experimentação do trabalho.

A primeira variação é obtida com base nas experimentações do trabalho de ANDRADE (2019), que forneceu base teórica e algorítmica para facilitar o processo da implementação de modelos. A segunda variação de implementação será a utilização do modelo *Text-Moderation* (MARKOV et al., 2023) da OpenAI[20], visto que, modelos que fazem uso de *Transformers* tem obtido grande sucesso para a classificação textual.

Seguindo os resultados de desempenho utilizando a métrica F1, foi decidido que para a representação dos dados seria utilizado TF-IDF em conjunto com *Logistic Regression*, pois obteve resultados semelhantes aos melhores desempenhos no trabalho de ANDRADE (2019) e também por conta da simplicidade da implementação. Portanto, não foi necessário a implementação de Word2Vec (W2V).

Além disso, foi tomada a decisão de utilizar um detector de ódio como serviço como uma forma alternativa de classificar discurso de ódio. Com ele será possível realizar a comparação de desempenho entre os modelos treinados no conjunto de dados utilizados no trabalho de ANDRADE (2019) e um modelo Large Language Model (LLM) que é treinado utilizando um conjunto de dados extenso e que engloba variedades de contextos distintos.

[20] <https://openai.com/blog/openai-api>

Capítulo 4

Experimentos

Neste capitulo será discutido de forma técnica a implementação da arquitetura proposta no capítulo anterior. Dessa forma, entraremos em detalhes sobre as tecnologias utilizadas durante o desenvolvimento e também detalharemos o processo da implementação de cada componente apresentado na arquitetura.

Para a implementação da arquitetura do sistema é essencial que cada componente esteja utilizando uma tecnologia adequada para cada funcionalidade implementada. A interface com o usuário final, a coleta de dados e a avaliação textual foram desenvolvidas considerando as vantagens de cada linguagem. Portanto, essa seção irá apresentar as tecnologias e as decisões por trás da escolha de cada uma.

4.1 Coleta de Dados

Nesta subseção será comentado o desenvolvimento e tecnologias utilizadas para coleta de dados em rede social, etapa fundamental para permitir a verificação de comentários e perfis com teor odioso, possibilitando o rastreamento e monitoramento de contas que tenham comportamento suspeito. Para esse trabalho foram propostas a coleta de dados no Twitter e no Discord. Além disso, será abordado como foi realizado o armazenamento dos dados coletados utilizando um banco de dados e como foram organizadas as tabelas que armazenam esses dados.

4.1.1 Twitter

A coleta de dados no Twitter é feita a partir de uma API REST, isso é, é necessário iniciar o processo de coleta do lado da aplicação, não sendo possível receber as atualizações em formato de evento. O *endpoint*[1] para coleta fornece uma resposta com informações sobre cada tweet recebido, conforme pode ser observado no código 4.1, é possível coletar informações sobre os usuários, publicações e respostas.

O Twitter foi a rede social selecionada em primeiro momento por conta da API gratuita, porém, devido às mudanças internas na organização empresarial, o uso do sistema foi limitado para usuários gratuitos, não sendo possível fazer um teste de carga da aplicação.

Código 4.1: Relatório em JSON

```
{
  "data": [
    {
      "id": "1460323737035677698",
      "text": "tweet example",
      "edit_history_tweet_ids": [
        "1460323737035677698"
      ]
    },
    {
      "id": "1519781379172495360",
      "text": "bom dia pessoal",
      "edit_history_tweet_ids": [
        "1519781379172495360"
      ]
    },
    {
      "id": "1519781381693353984",
      "text": "hello world": [
```

[1] <https://developer.twitter.com/en/docs/twitter-api/tweets/lookup/api-reference/get-tweets>

```
20          "1519781381693353984"
21        ]
22      }
23    ]
24  }
```

4.1.2 Discord

A API fornecida pelo Discord faz envio ativo de eventos. Um desses eventos é o envio de mensagens em servidores que o *bot* é membro. Dessa forma, é possível desenvolver uma aplicação que faça a coleta automática das publicações dos clientes e popule o banco com essas informações.

Nesse caso, o uso do JDA[2] é recomendado, pois, fornece um número elevado de funcionalidades e com alta abstração, não sendo necessário o consumo manual dos eventos fornecidos[3]. Além disso, a familiaridade dos autores com a ferramenta também favorecem o uso da ferramenta. Segue no código 4.2 um exemplo de mensagem fornecida pela API do Discord.

Código 4.2: Resposta do Discord em JSON

```
1  {
2    "reactions": [
3      {
4        "count": 1,
5        "count_details": {
6           "burst": 0,
7           "normal": 1
8        },
9        "me": false,
10       "emoji": {
11          "id": null,
12          "name": "       "
```

[2] <https://github.com/discord-jda/JDA>
[3] <https://discord.com/developers/docs/topics/gateway-events>

```
      },
      "burst_colors": []
    }
  ],
  "attachments": [],
  "tts": false,
  "embeds": [],
  "timestamp": "2017-07-11T17:27:07.299000+00:00",
  "mention_everyone": false,
  "id": "334385199974967042",
  "pinned": false,
  "edited_timestamp": null,
  "author": {
    "username": "Mason",
    "discriminator": "9999",
    "id": "53908099506183680",
    "avatar": "a_bab14f271d565501444b2ca3be944b25"
  },
  "mention_roles": [],
  "content": "Supa␣Hot",
  "channel_id": "290926798999357250",
  "mentions": [],
  "type": 0
}
```


4.1.3 Banco de Dados

O PostgreSQL foi escolhido como ferramenta para gerenciar o banco de dados utilizado para armazenar as publicações monitoradas. Ele é um gerenciador de banco de dados relacional que possui um sistema robusto, confiável, escalável e *open-source*. Além disso, possui uma comunidade extensa e ativa que contribui com diversos recursos e compatibilidades que serão utilizadas, como a conexão com a linguagem Python, por exemplo.

Além disso, o projeto do qual foi estendido a interface possuía *schemas* de tabelas relacionadas a coleta de tweets prontas para serem migradas. A partir disso, apenas foi necessário adicionar colunas relacionadas a detecção de discurso de ódio. Ademais, pela familiaridade dos autores do trabalho com o PostgreSQL, ele se tornou uma escolha sensata.

Ao todo, sete tabelas serão usadas para a integração com o Twitter. As tabelas de termos, sinônimos e categorias permitem uma coleta filtrada de publicações, a tabela de *tweets* irá conter as informações sobre tweets e também sobre a presença de discurso de ódio em cada um. Ademais, foi criada a tabela auxiliar "geos" para armazenar informações de geolocalização das publicações. Para realizar a filtragem por usuários foram criadas duas tabelas para identificar e centralizar os *tweets* de cada usuário.

Além disso, foram criadas outras tabelas que permitem o gerenciamento de componentes do sistema, como uma tabela de usuários, cache, estatísticas de coleta e sessões de usuários ativas como pode ser vista na figura 4.1.

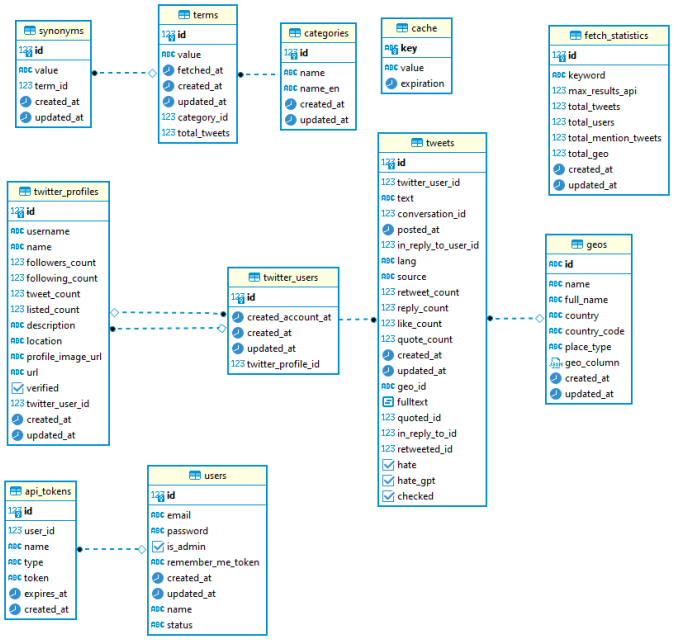

Figura 4.1: Diagrama de Relações do Banco de Dados utilizados para integração do Twitter.

Para o armazenamento de dados do Discord foi necessário apenas a criação de

uma única tabela. A biblioteca JDA abstrai o carregamento de usuários e servidores, não sendo necessário armazenamento no banco de dados. A tabela criada, conforme a figura 4.2, contêm informações necessárias para filtragem por servidor, usuário e também o cálculo da presença de ódio por usuário, servidor e globalmente.

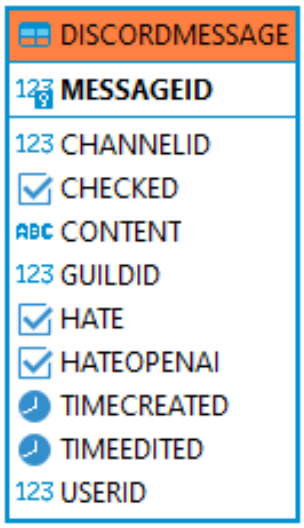

Figura 4.2: Diagrama de Relações do Banco de Dados para integração do Discord.

4.2 Detecção de Discurso de Ódio

A avaliação de publicações foi feita a partir um componente dedicado para essa funcionalidade. A aplicação contêm uma integração com a API da OpenAI para a classificação dos textos e também um modelo de classificação local. A aplicação irá requisitar um lote de publicações para o banco de dados a cada período e irá fazer uma classificação gradual de todos os novos dados que estão sendo inseridos no banco de dados.

Dessa forma, fica evidente que o objetivo desse trabalho não é avaliar o melhor modelo de aprendizado de máquina, como feito por ANDRADE (2019), e sim, demostrar como é possível implantar de forma desacoplada esses componentes em uma aplicação real. Portanto, serão implementados duas variações de classificador, a aplicação *Self-Hosted* e a aplicação como de detecção de discurso de ódio como

serviço.

4.2.1 Detecção de Discurso de Ódio *Self-Hosted*

O desenvolvimento do modelo de aprendizado de máquina utilizado para a classificação do texto localmente, foi feito utilizando o Python aliado das bibliotecas: Pandas, scikit-learn, pickle e nltk. Dessa forma, é possível focar mais na experimentação e no resultado do que na implementação dos modelos e métricas que já estão disponíveis nessas bibliotecas.

O Python foi escolhido por ser uma linguagem que pode ser compilada e interpretada, favorecendo a prototipação rápida, além disso, como mencionado anteriormente, possui uma grande número de bibliotecas e *frameworks* voltados para aprendizado de máquina e manipulação de tabelas. Deste jeito, utilizando essas bibliotecas, é possível importar implementações de algoritmos de aprendizado de máquina, importar e manipular os conjuntos de dados de treino e validação.

O *Logistic Regression* é utilizado em problemas de classificação binária. Para a classificação textual, o algoritmo irá transformar os recursos de entrada em uma escala contínua entre 0 ou 1, que representa a probabilidade daquela entrada pertencer à classe especificada.

Porém, a entrada textual não é aceita em algoritmos matemáticos, dessa forma, é imperativo produzir um ferramental que converta os dados textuais originais em um formato aceitável para o modelo. Nesse sentido, serão detalhados todos os procedimentos para ser realização dessa transformação.

Para o treinamento do modelo local foi utilizado a base de dados fornecida por Fortuna et al. (2019), onde, foi utilizado uma descriminação binária que descrimina cada entrada entre positivo e negativo. A base de dados é composta por 5.668 publicações, onde, 31.1% possuem presença de discurso de ódio.

O primeiro passo para a execução do modelo é realizar o pré-processamento dos dados, removendo os ruídos, caracteres e palavras indesejadas conforme descrito no capítulo 2. No caso deste trabalho, as técnicas de pré-processamento utilizadas foram

a tokenização, remoção de acentos, pontuação e números. Além disso, foi utilizada a remoção de *stopwords* por meio da biblioteca de *stopwords* provida pelo nltk.

A fim de realizar uma transformação textual capaz de converter a entrada de texto em um formato adequado para o modelo de classificação, optou-se por empregar a técnica TF-IDF. O *scikit-learn* disponibiliza uma classe para a criação de TF-IDF chamada *TfidfVectorizer*. Dessa forma, após a fase de treinamento, o conversor de entrada já estava pronto para ser aplicado em novas entradas.

Através do *scikit-learn* foi possível treinar um modelo de *Logistic Regression*, que foi construído com base na experimentação construída por ANDRADE (2019). Para evitar o re-treino do modelo foi utilizado uma funcionalidade para salvar o estado do modelo utilizando o *pickle*[4].

4.2.2 Detecção de Discurso de Ódio Como Serviço

A OpenAI disponibiliza uma API que permite que os desenvolvedores tenham acesso a diversas funcionalidades como serviço. Uma das funcionalidades que a API fornece é a moderação de conteúdo através do modelo *text-moderation-stable* oferecido de forma gratuita por meio de um *endpoint*.

A classificação é feita por meio de uma requisição ao *endpoint*. O texto a ser classificado é inserido dentro do corpo da requisição. A resposta recebida é composta por um JavaScript Object Notation (json) contendo uma classificação detalhada da presença de diferentes tipos de conteúdo que possam ser problemáticos e violam políticas de uso da OpenAI, como pode ser visto no código 4.3.

Código 4.3: Resposta do Modelo text-moderation em JSON

```
1 {
2   "id": "modr-XXXXX",
3   "model": "text-moderation-005",
4   "results": [
5     {
6       "flagged": true,
```

[4]https://docs.python.org/3/library/pickle.html

```
      "categories": {
        "sexual": false,
        "hate": false,
        "harassment": false,
        "self-harm": false,
        "sexual/minors": false,
        "hate/threatening": false,
        "violence/graphic": false,
        "self-harm/intent": false,
        "self-harm/instructions": false,
        "harassment/threatening": true,
        "violence": true,
      },
      "category_scores": {
        "sexual": 1.2282071e-06,
        "hate": 0.010696256,
        "harassment": 0.29842457,
        "self-harm": 1.5236925e-08,
        "sexual/minors": 5.7246268e-08,
        "hate/threatening": 0.0060676364,
        "violence/graphic": 4.435014e-06,
        "self-harm/intent": 8.098441e-10,
        "self-harm/instructions": 2.8498655e-11,
        "harassment/threatening": 0.63055265,
        "violence": 0.99011886,
      }
    }
  ]
}
```


A API[5] fornece detalhes sobre o tipo de conteúdo que está sendo violado, ou seja, é possível filtrar qual tipo de situação será considerada. A API fornece descriminação

[5]https://platform.openai.com/docs/guides/moderation/quickstart

para conteúdo contendo ódio, assédio, conteúdo sexual, textos depressivos e violentos. Apesar disso, o classificador não está completamente preparado para lidar com línguas diferentes de inglês, logo, é possível ter um desempenho abaixo do esperado.

A API impõe restrições à quantidade de textos avaliados, permitindo até 32 avaliações por requisição e 1000 requisições por minuto. Portanto, o projeto não enfrenta preocupações quanto aos limites de requisições, já que dentro do contexto do projeto é viável avaliar até 32.000 textos por minuto.

4.2.3 Comparação Entre as Duas Abordagens

A fim de calcular os resultados e realizar a comparação da forma mais justa entre o modelo local de *Logistic Regression* e o modelo Text-Moderation da OpenAI, foi decidido classificar o conjunto de teste do treinamento do modelo local utilizando o modelo Text-Moderation, dessa forma poderíamos calcular e comparar métricas tendo o mesmo conjunto como base.

O desempenho do modelo local obteve o valor da métrica F1 de 0.65 e acurácia de 0.74. Considerando o desbalanceamento do conjunto de dados, é possível perceber, conforme a figura 4.3, que o modelo acerta 80% dos textos sem presença de ódio, porem acerta 64% dos textos com presença de discurso de ódio.

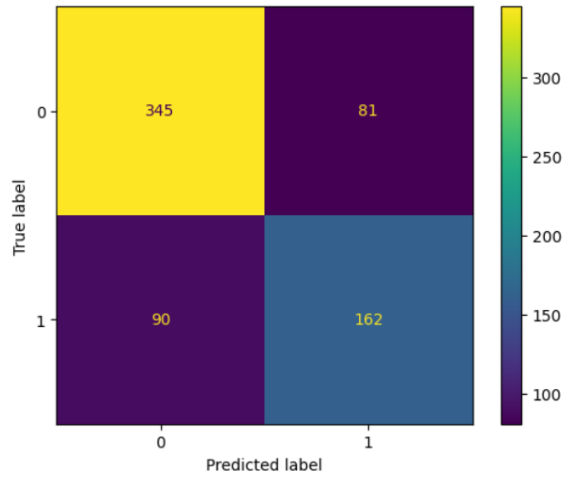

Figura 4.3: Matriz de Confusão do Modelo Local que utiliza Logistic Regression e TF-IDF

O modelo Text-Moderation teve um desempenho bem diferente do modelo local, o valor da métrica F1 foi de 0.41 e a acurácia 0.68. Conforme a figura 4.4, podemos perceber que o modelo classificou 87% do conjunto de dados inteiro, como sem presença de discurso de ódio. Dessa forma, o modelo acerta 90% dos valores onde realmente não existe discurso de ódio, porém acerta apenas 30% dos textos com presença de discurso de ódio.

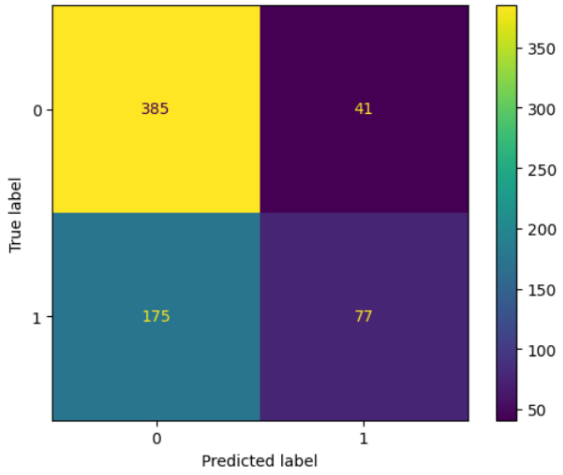

Figura 4.4: Matriz de Confusão do Modelo Text-Moderation

É possível perceber uma discrepância significativa entre os modelos, entretanto, como pode ser visto tabela 4.1, ao gerar um exemplo de textos para a classificação, podemos perceber que o modelo Text-Moderation é capaz de generalizar melhor o contexto e a intenção da frase, evitando classificar como ódio frases com palavras de baixo calão, mas num contexto positivo.

Texto	Ódio (LR)	Ódio (TM)
vai se fuder seu pau no cu	true	true
eu acho o joao um cara muito foda vai tomar no cu amo muito ele	true	false
jesus ama a todos nos amem igreja	false	false
5 bananas, 8 ovos, 1 scoop de whey de torta de limao e fé	false	false
ola, boa noite	false	false
fodase, eu quero mais é amar muito	true	false

Tabela 4.1: Exemplos para comparar a classificação do Logistic Regression e do Text-Moderation

Por isso, foi decido investigar o conjunto de treinamento disponibilizado por Fortuna et al. (2019) e diversas inconsistências surgiram ao comparar com classi-

ficações do modelo Text-Moderation. Em determinados casos, exemplificados na tabela 4.2, observou-se que o conjunto de dados designava erroneamente a categoria de "ódio", enquanto nossa avaliação manual revelou discordância com essa classificação. Consequentemente, a confiabilidade do conjunto de dados para as atribuições de classificação é questionável, influenciando negativamente nas métricas geradas para os modelos.

Publicação	Dataset	OpenAI
a avaliar pela quantidade de malta conhecida que morreu em 2016, o in memoriam deste ano é coisa para durar umas 9 horitas na boa oscars	VERDADEIRO	FALSO
acabei de pagar 20 reais em um carregador pq esqueci o meu. já começando o dia se sentindo a mulher mais burra do brasil	VERDADEIRO	FALSO
Qndo falo ir em mulher me refiro a meter a rola nela	FALSO	VERDADEIRO
Assista! Aqui esta o documento que prova q o jornalista canalha da istoe requenta a noticia p me denegrir!	FALSO	VERDADEIRO

Tabela 4.2: Exemplo de inconsistência entre a classificação de Fortuna et al. (2019) e da OpenAI

Apesar disso, a implementação de um modelo com alto desempenho não foi o foco do trabalho. Dessa forma, é possível obter resultados melhores ao utilizar mais *datasets*, como (CASELLI et al., 2021) e com uma melhor experimentação do modelo. Adicionalmente, a OpenAI avisa sobre o desempenho do modelo não ser satisfatório para línguas diferentes da inglesa.

Conforme mencionado por Markov et al. (2023), a utilização de dados públicos deve ser feita com cautela. As diferenças de taxonomia, contexto e ruído das classificações baseado nas crenças de quem classificou podem influenciar em divergências entre o rótulo dos dados. Ou seja, o fato do *Transformer*, previamente treinado, estar tendo seus resultados comparados com um *dataset* externo possibilitam divergências na taxonomia da classificação, dessa forma, o desempenho do modelo pode ser inferior ao esperado.

4.3 Agendadores

O sistema necessita executar rotinas para coletar dados com o Twitter e para classificar os dados pendentes no banco de dados. Para a coleta de dados no Twitter foi utilizada uma rotina acoplada a aplicação *web*. Para a classificação de tweets,

pela aplicação ser dedicada para esse processo, foi utilizado um simples *sleep* da biblioteca *timer* do Python.

O AdonisJS possui uma extensão para agendamento de tarefas[6]. Desta maneira, é possível agendar a execução da rotina de coleta de dados diretamente da aplicação *web*. Porém, existem preocupações sobre sobrecarregar o servidor *web* com a rotina de coleta. Logo, é recomendado separar a coleta de dados em uma aplicação dedicada.

O *crawler* realiza coletas com base em palavras-chave específicas, isto é, ao especificar uma palavra no sistema, um termo e seus sinônimos, ela é monitorada em conjunto com outros termos relevantes. Dessa forma, é possível manter o sistema atualizado com as últimas postagens, porém, o conteúdo massivo publicado no Twitter impede a coleta integral dos dados.

A Rotina classificação dos dados irá ser executada a cada intervalo de segundos especificado, nesse caso, foi decidido permitir a classificação de até 32.000 textos a cada 60 segundos. Sendo o fator limitante a carga máxima permitida pelo Text Moderation da OpenAI.

4.4 Interface de Monitoramento

O Node.js[7] foi escolhido para o desenvolvimento da aplicação do lado do servidor para unificar a tecnologia utilizada em todo o ecossistema do site, o que facilita a manutenção tanto do código *client-side* como *server-side*. O Node.js inclui benefícios como o suporte multiplataforma, arquitetura baseada em eventos e suporte a operações assíncronas.

O *framework* AdonisJS[8] foi adotado para o desenvolvimento do *website* responsável por apresentar os dados do monitoramento. O AdonisJS fornece uma estrutura completa para facilitar a criação e integração do site com qualquer API ou sistema externo.

[6] <https://www.npmjs.com/package/adonis5-scheduler>
[7] <https://nodejs.org/en>
[8] <https://adonisjs.com/>

Uma das vantagens do AdonisJS é seu padrão arquitetural. Por seguir o padrão Model-View-Controller (MVC), é possível estruturar o código de forma mais organizada e com a delegação de responsabilidade mais eficiente. Dessa forma, a leitura e manutenção do código é facilitada.

A integração com Objeto Relational Mapping (ORM), Lucid, é uma parte essencial para o projeto, pois simplifica a integração com o banco de dados. Essa parte é importante, pois, deixa mais intuitivo o tratamento dentro do próprio JavaScript. Outra parte importante é a existência de *Migrations*, que permite a gerência automática das tabelas do banco de dados pelo próprio Framework.

Além disso, o AdonisJS é desenvolvido em *TypeScript*, dessa forma, é recomendado o uso dessa linguagem por conta da tipagem estática, que facilita a manutenção e depuração do código. Ademais, o AdonisJS fornece recursos integrados para automatização da autenticação e controle de sessão, facilitando o cadastro e login de usuários.

4.4.1 Interface com o Usuário

A interação na aplicação *web* foi dividida em três telas. Uma tela para adição de termos para monitoramento, uma tela para monitoramento geral de discurso de ódio nos perfis da base de dados e uma tela de monitoramento de usuário que especifica a quantidade de publicações do usuário que foram avaliadas, a porcentagem de publicações contendo ódio e discrimina cada *tweet* na sua devida categoria.

4.4.1.1 Tela para Criação de Categorias

Foi criada uma tela para criação de categorias, essas categorias permitem a definição de grupos. Esses grupos têm a função de agrupar os termos que serão criados em cada categoria, dessa forma podemos definir diferentes tipos de conteúdo de ódio como pode ser visto na figura 4.5. A partir disso, podemos clicar em uma categoria, ela redirecionará para a tela de termos utilizando um filtro para visualizar apenas os termos cadastrados sob essa categoria.

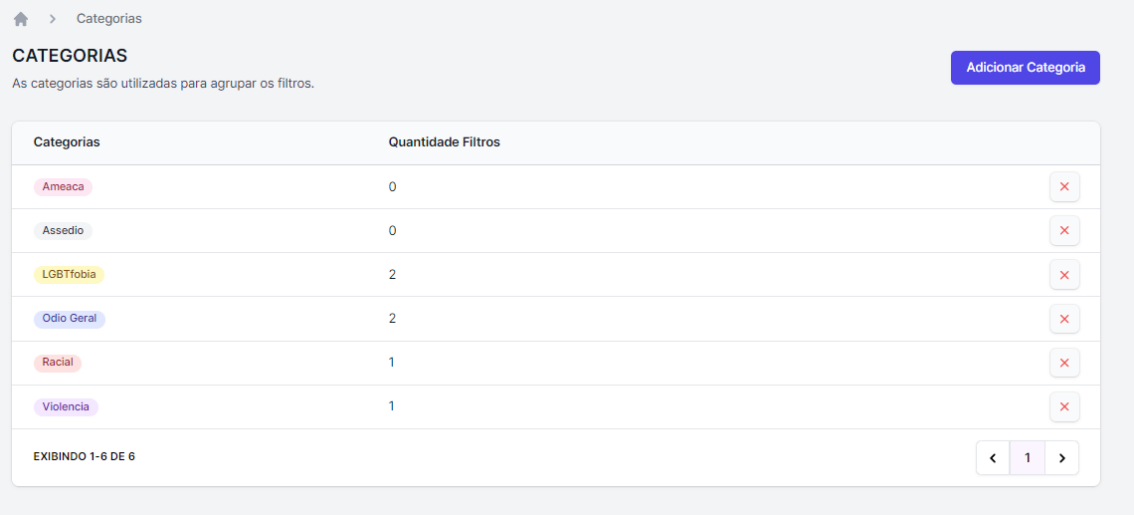

Figura 4.5: Tela de Categorias de Termo

4.4.1.2 Tela de Adição de Termos

A tela de adição de termos permite a criação de novas palavras a serem rastreadas, que podem ser, por exemplo, algumas palavras-chave utilizadas em certos discursos de ódio ou frases e palavras que estão em alta em alguma discussão no momento, permitindo acompanhar o índice de discurso de ódio em tempo real. Nessa tela, a criação de um termo novo permite a associação a alguma categoria pré-definida, além de adicionar sinônimos que serão tratados como um mesmo termo.

Além disso, podemos acessar a tela de um termo cadastrado e verificar os *tweets* que foram encontrados e correspondem com aquele termo. Nessa página podemos visualizar a presença de discurso de ódio dos tweets capturados pelo termo como visto na figura 4.7, e a avaliação individual de cada *tweet* capturado como na figura 4.8.

4.4.1.3 Tela para Monitoramento Geral

A tela de monitoramento geral irá exibir informações com poucos detalhes sobre os usuários, porém, com o objetivo de fornecer uma visão geral sobre os usuários da base de dados, conforme pode ser visualizado na figura 4.9. Essa tela, assim como a de usuários, irá classificar os usuários entre quatro categorias.

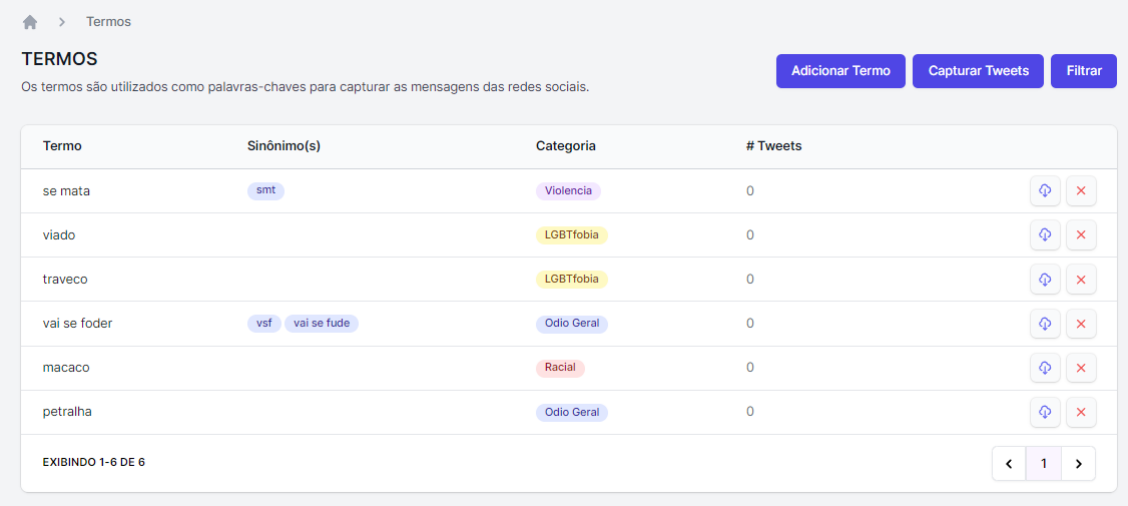

Figura 4.6: Tela de termos cadastrados

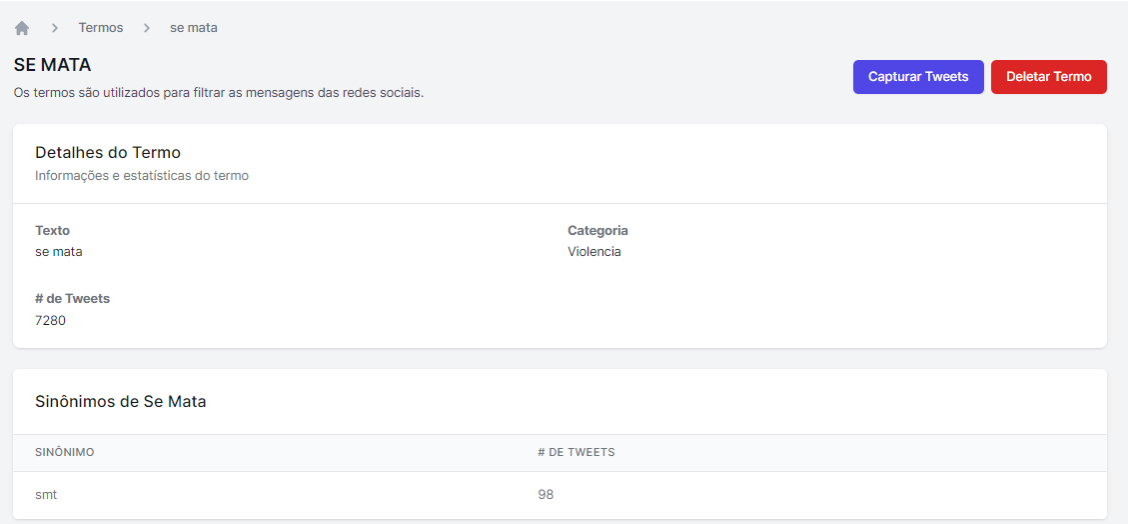

Figura 4.7: Tela de informações de um termo

Usuários com menos de 5% de discurso de ódio no perfil irão ser classificados como **respeitosos**, usuários com menos de 15% de ódio no perfil irão ser classificados como **normal**, usuários com menos de 25% de ódio no perfil serão classificados como **problemáticos** e para qualquer caso maior que 25% será identificado como **crítico**.

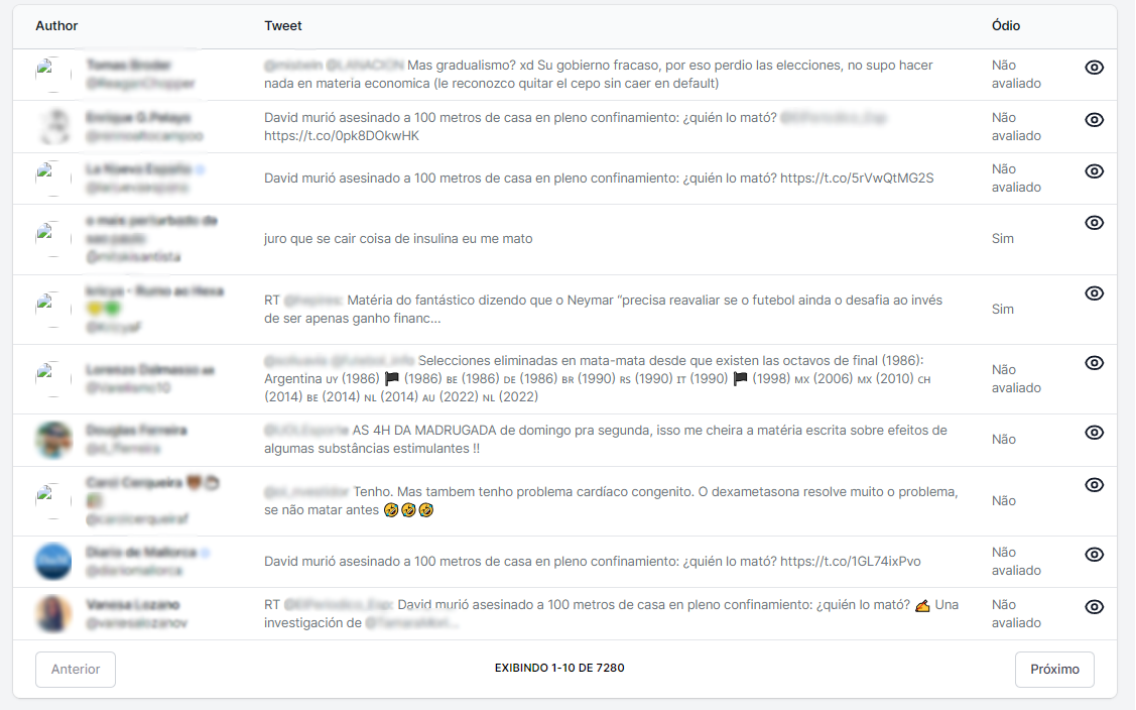

Figura 4.8: Tela de *tweets* de um termo

4.4.1.4 Tela para Monitoramento do Usuário

A tela de usuários do Twitter é usada para mostrar dados gerais do usuário e também para mostrar informações sobre o comportamento do usuário na plataforma. A figura 4.10 exibe os dados relevantes da conta, sendo eles a quantidade de seguidores, a descrição e quantidade de *tweets*. Adicionalmente, foram adicionados campos para mostrar a taxa de discurso de ódio encontrada naquele perfil. Na parte inferior da tela é possível analisar dados sobre a presença de ódio em cada *tweet*, e é possível visualizar um exemplo de *tweet* com discurso de ódio.

4.5 Monitoramento pelo Discord

A linguagem Java foi escolhida para o desenvolvimento do bot para o Discord devido a três fatores, o primeiro fator é a biblioteca JDA[9] que fornece uma abstração

[9]<https://jda.wiki/>

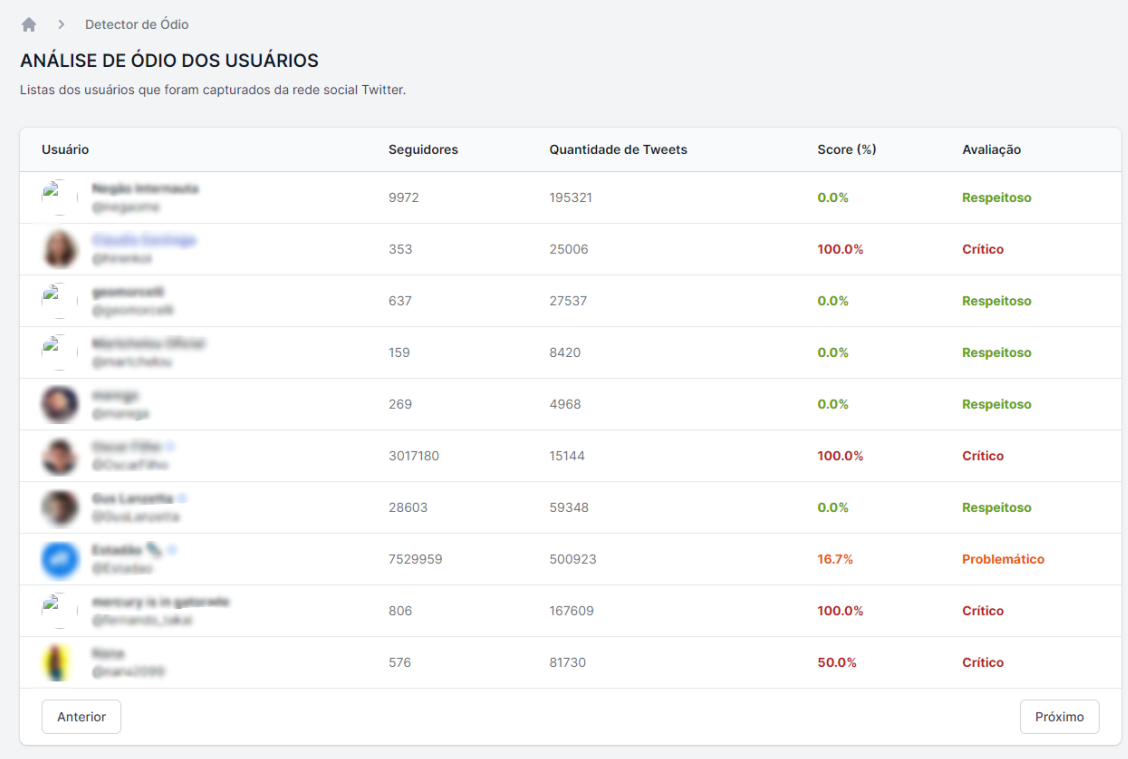

Figura 4.9: Tela de usuários do Twitter na aplicação *web*

limpa, efetiva e eficiente para comunicação com o Discord, fornecendo uma abstração para os eventos *web sockets* e a interface REST.

O segundo fator é a existência do Spring Data JPA[10], que se trata de um *framework* ORM que fornece uma abstração robusta para a persistência de objetos. Além disso, a biblioteca Curupira[11], desenvolvida pelos autores desse trabalho, abstrai o tratamento de eventos através do uso da API *Reflection* do Java para fornecer uma interface declarativa para o tratamento de *slash commands*, *menus*, *buttons* e *modals*.

As funcionalidades fornecidas para o Discord são desenvolvidas com objetivo de fornecer dados estatísticos sobre a presença de discurso de ódio em três níveis. Estatísticas globais, estatísticas por servidor e estatísticas por usuário. As estatísticas globais são fornecidas exclusivamente para os autores do trabalho, conforme a figura 4.11, caso um usuário externo tente utilizar o comando será bloqueado conforme

[10] <https://spring.io/projects/spring-data-jpa>
[11] <https://github.com/Softawii/curupira>

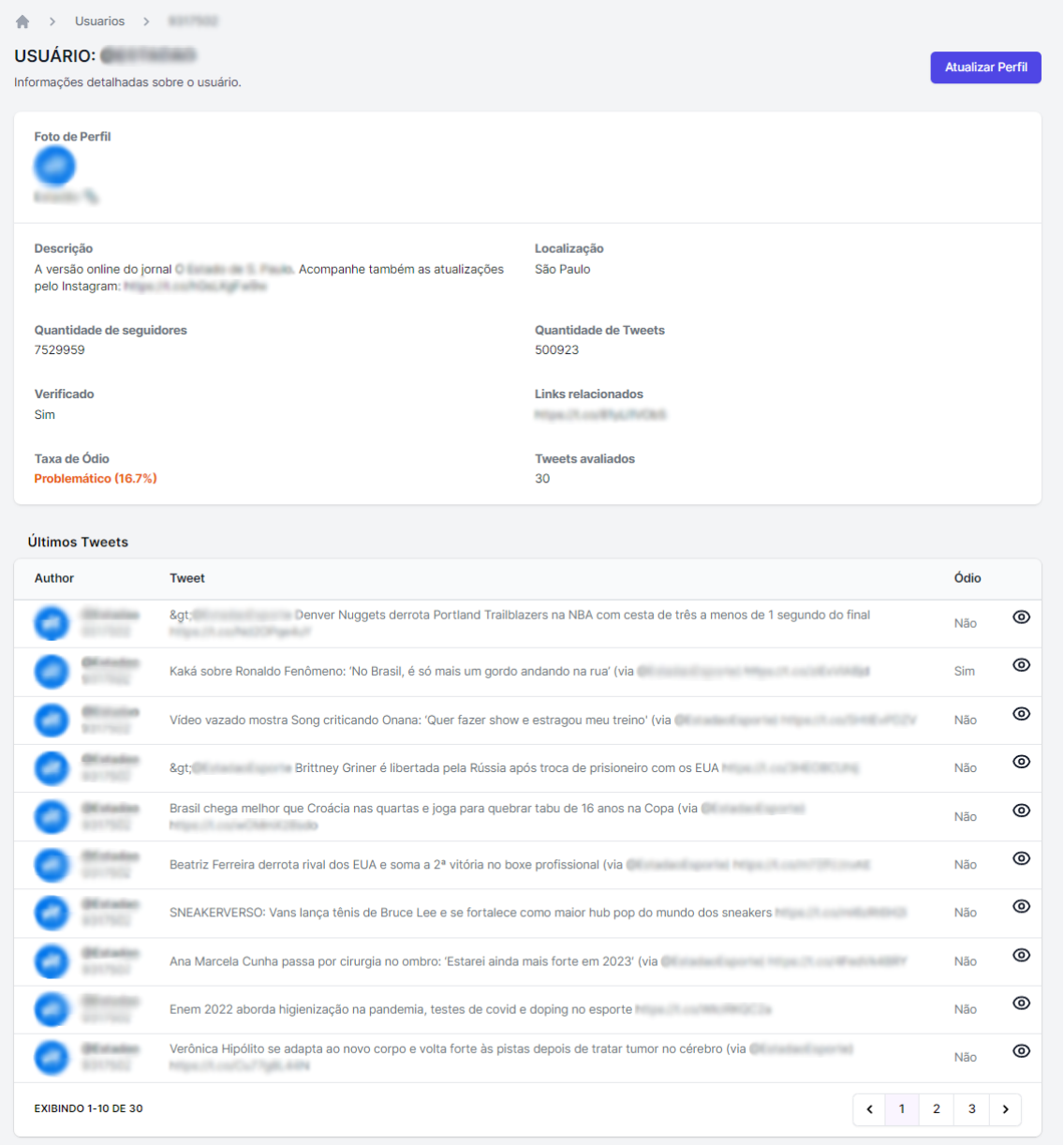

Figura 4.10: Tela de perfil de um usuário do Twitter com os *tweets* coletados

a figura 4.12.

Os administradores do servidor são capazes de analisar quais os usuários do servidor que possuem mais mensagens com a presença de discurso de ódio, conforme a figura 4.13. Dessa forma, caso necessário, é possível procurar as mensagens de discurso de ódio enviadas por usuário específicos, conforme a figura 4.14 e a figura 4.15.

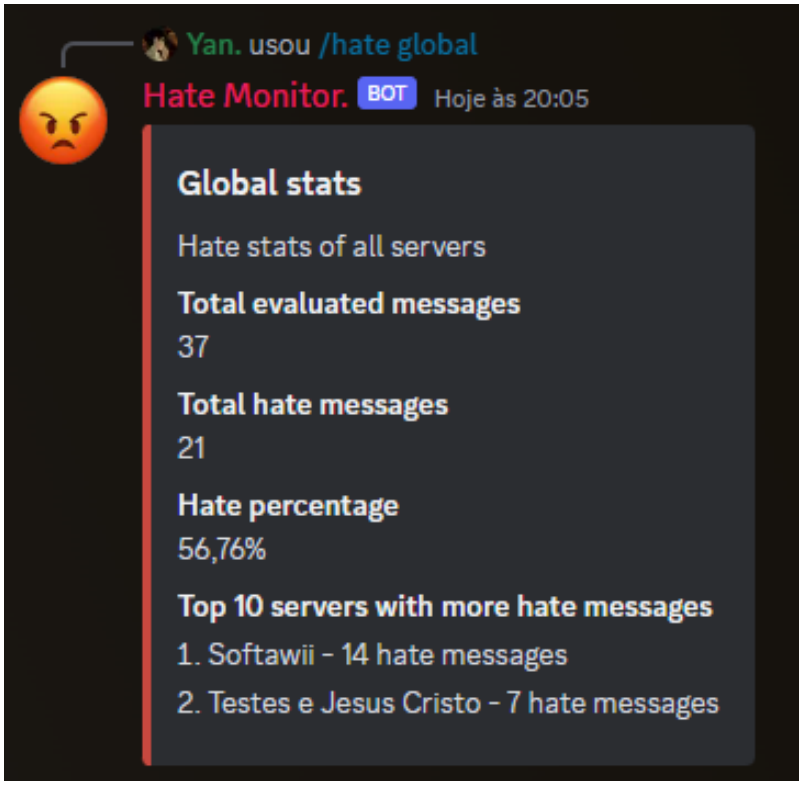

Figura 4.11: Resposta do comando *slash* para exibição de estatísticas globais coletadas pelo bot.

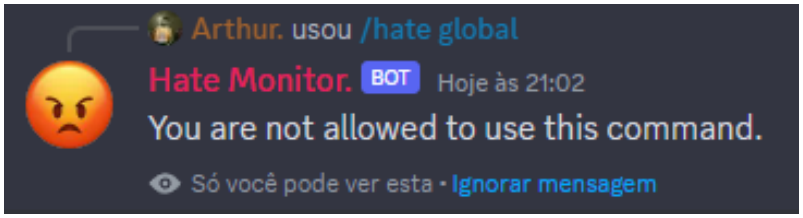

Figura 4.12: Resposta do comando *slash* para exibição de estatísticas globais coletadas pelo bot quando um usuário não autorizado executa.

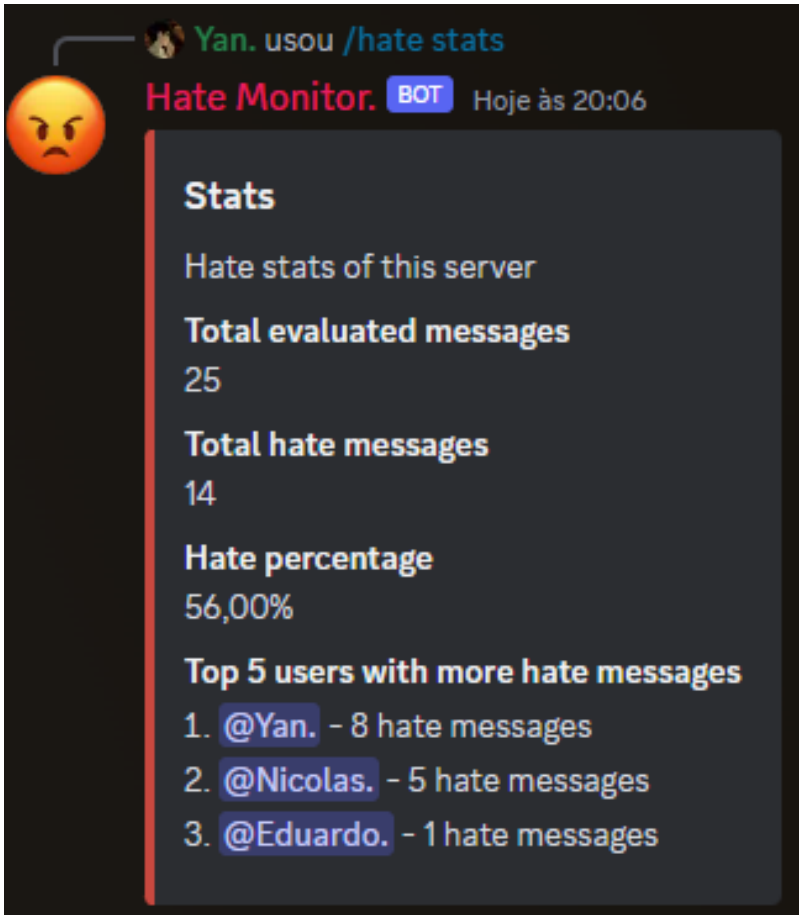

Figura 4.13: Resposta do comando *slash* para exibição de estatísticas para um servidor corrente

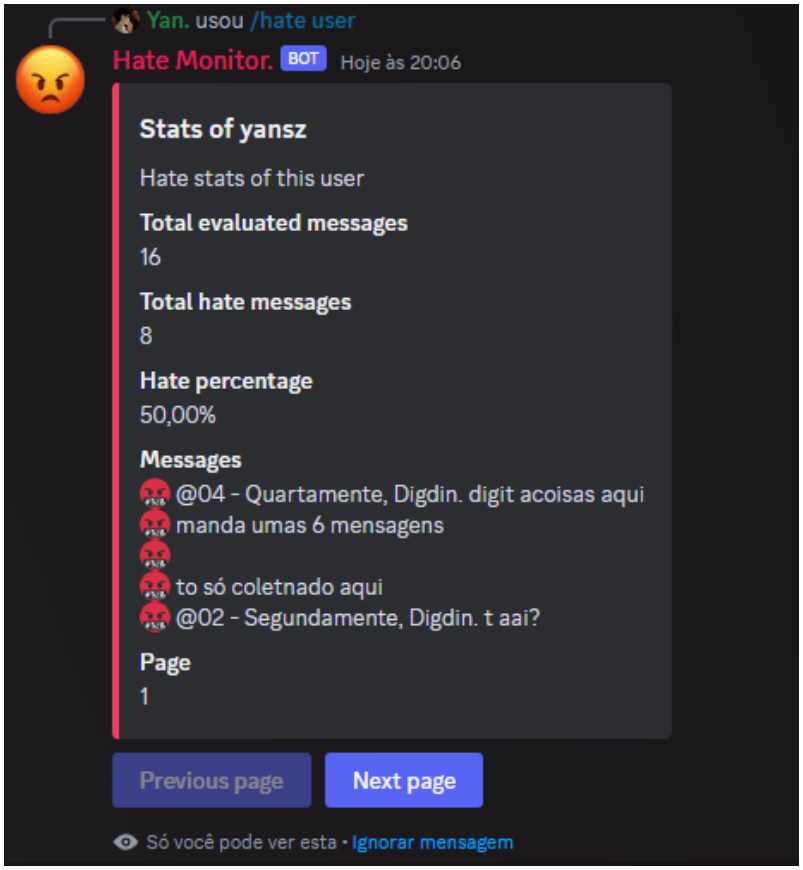

Figura 4.14: Resposta do comando *slash* para exibição das estatísticas de ódio por usuário (parte 1)

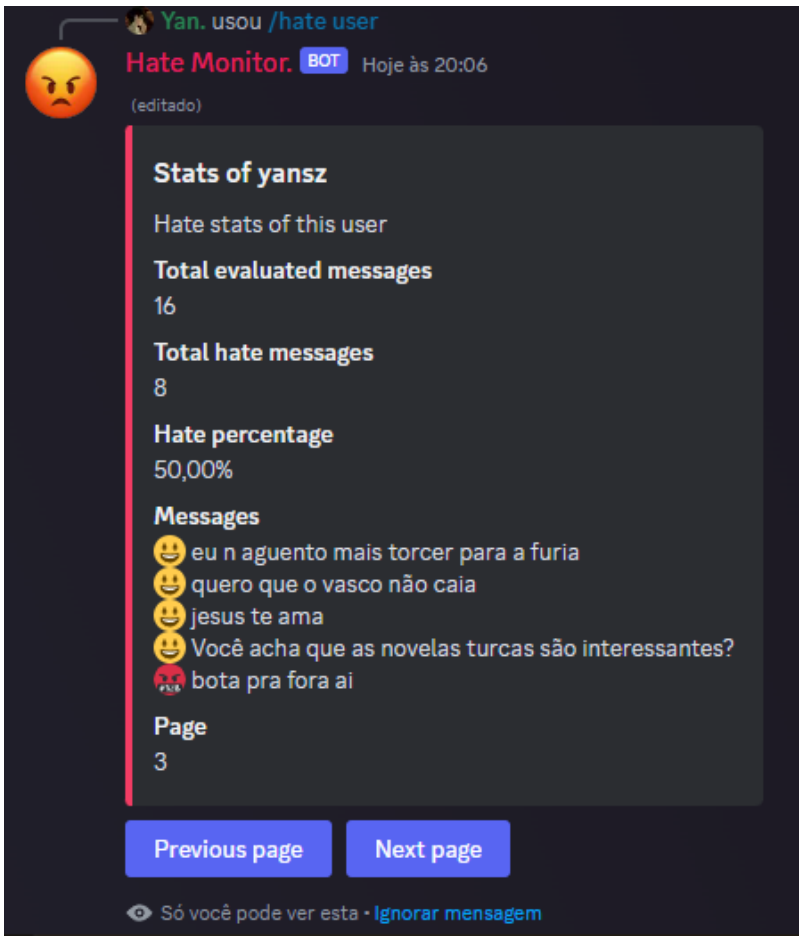

Figura 4.15: Resposta do comando *slash* para exibição das estatísticas de ódio por usuário (parte 2)

Capítulo 5

Conclusão

Neste capitulo será discutido a conclusão que os autores tiveram ao decorrer do desenvolvimento do trabalho. Apresentaremos nossas considerações finais, limitações encontradas ao longo do desenvolvimento e sugestões de trabalhos futuros que podem originar a partir desse.

5.1 Considerações finais

Ao longo deste trabalho foram desenvolvidos duas aplicações de monitoramento de discurso de ódio em redes sociais. O primeiro é uma aplicação *web* integrada ao Twitter, permitindo aos usuários coletar *tweets* por busca de termos ou capturar *tweets* de usuários. Esse sistema utiliza um agendador para programar a coleta de *tweets* em horários específicos, um *crawler* para efetuar a coleta dos dados através da API do Twitter, um banco de dados para armazenar informações do sistema e os dados coletados do Twitter.

A aplicação se conecta externamente com um classificador por meio de um *script* intermediário que captura as publicações escritas no banco de dados e as classifica quanto a presença de discurso de ódio. O classificador emprega dois métodos: um modelo baseado em regressão logística treinado localmente e outro utilizando o *endpoint* de moderação de texto da OpenAI.

A segunda aplicação foi desenvolvida como um bot integrado ao Discord, visando mitigar o discurso de ódio presente nas mensagens dos usuários. Essa aplicação utiliza os eventos disponibilizados pela API do Discord para monitorar e analisar interações dentro dos servidores. Sua principal finalidade é não apenas identificar expressões prejudiciais, mas também proporcionar recursos avançados, como a geração de relatórios sobre usuários e servidores. Essa abordagem visa simplificar o processo de moderação, concedendo aos administradores de servidor no Discord uma ferramenta eficaz para gerenciar o ambiente de forma mais proativa e responsiva.

Com isso, o objetivo proposto nesse trabalho foi alcançado, a arquitetura proposta é eficiente e de fato facilita o monitoramento do discurso de ódio, permitindo acompanhar de forma detalhada, seja num conjunto de assuntos e usuários do Twitter ou em servidores do Discord. A escalabilidade do projeto para um serviço real de monitoramento é viável com algumas mudanças e otimizações na forma que os dados são coletados e armazenados, como iremos discutir na seção seguinte.

5.2 Limitações e trabalhos futuros

Esta seção aborda as limitações identificadas durante a implementação do projeto. São discutidos aspectos como a proposta original do trabalho, o fluxo de dados, a acessibilidade à API do Twitter, melhorias na apresentação de dados na página de termos, a qualidade do *dataset*, o desempenho dos classificadores, e a integração do Discord com o servidor do DCC. Além disso, são exploradas perspectivas para a expansão da aplicação, incluindo a integração direta de outras plataformas na aplicação *web*, visando um monitoramento mais abrangente.

Inicialmente, a ideia desse trabalho era a implementação de um bot para o Twitter onde os usuários do site poderiam interagir, consultando a presença de ódio em *tweets*. Além disso, o bot seria responsável por acompanhar assuntos em alta na plataforma e verificar a presença de ódio nessas discussões. Entretanto, com as mudanças de acesso a API do Twitter, que será comentada seguir, essa ideia acabou sendo descartada durante o desenvolvimento.

Durante o desenvolvimento do projeto, enfrentamos restrições no acesso gratuito à API do Twitter, o que impossibilitou a coleta contínua de *tweets* relacionados ao contexto do trabalho. Apesar dessas limitações, conseguimos realizar a coleta de alguns *tweets* anteriormente, o que permitiu a condução de testes com base nos dados já disponíveis na base. Como perspectiva para trabalhos futuros, consideramos a exploração de outras redes sociais que permitam a coleta automatizada de dados, destacando, por exemplo, o potencial do Bluesky.

Uma das grandes limitações encontradas durante o desenvolvimento do trabalho foi o fluxo de dados durante todo o processo, por não ser linear e instantâneo, as aplicações podem ficar ociosas esperando as tarefas de coleta e manipulação dos dados serem disparadas. Dessa forma, todo o fluxo iria se beneficiar de um sistema de *streaming* de dados, ou uma arquitetura de fila, dessa forma, permite que os dados sejam processados conforme a demanda. Dessa forma, pode se tornar um trabalho futuro a adoção do *streaming* de dados junto ao Discord.

Um aprimoramento proposto para a página de termos envolve a apresentação da porcentagem de discurso de ódio associada a cada termo. A não implementação dessa funcionalidade decorreu da complexidade das *queries* que estavam sendo geradas. Devido à priorização de outras tarefas, essa funcionalidade foi designada como uma melhoria para o futuro, visto que demanda uma análise mais detalhada e um desenvolvimento cuidadoso para garantir a eficiência.

A precisão das classificações presentes no *dataset* disponibilizado por Fortuna et al. (2019) revelou-se insatisfatória. Ao conduzir verificações manuais, conforme descrito na figura 4.4, observou-se frequentemente a existência de inconsistências entre as opiniões expressas pelos autores, a classificação da OpenAI e as classificações atribuídas aos textos. Essa constatação aponta para a necessidade de um refinamento nas classificações existentes no conjunto de dados, com foco na importância de uma abordagem rigorosa na classificação dos *tweets* do *dataset*.

Embora a proposta central deste trabalho não tenha sido a análise e teste aprofundado de classificadores, evidenciou-se que a qualidade dos modelos destinados à língua portuguesa ainda não atende aos requisitos necessários para sua aplicação

em um ambiente real. Nos testes realizados, observou-se a ocorrência de *tweets* classificados erroneamente tanto pelo modelo local, que demonstrou uma sensibilidade excessiva a palavrões, quanto pelo modelo da OpenAI, o qual revelou limitações na compreensão plena do contexto.

A fim de evitar o uso inadequado ou desrespeitoso do servidor oficial no Discord do DCC por parte dos alunos ou de outros usuários, é importante a implementação de um moderador automático capaz de intervir independentemente da situação. A expansão da aplicação do bot representa uma perspectiva a ser explorada em trabalhos futuros, dessa forma, a extensão do escopo do bot para incluir funcionalidades de auto-moderação pode contribuir para manter um ambiente respeitoso e seguro dentro do servidor.

Um outra possível melhoria para o trabalho seria a integração direta de outras redes sociais a aplicação *web*, por exemplo, o próprio Discord. É interessante manter a existência do bot para o monitoramento direto pela própria plataforma, porém, permitir uma visualização global dos dados de todas as redes sociais no *dashboard* aumenta as possibilidades de monitoria.

Referências

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

ANDRADE, V. D. A. *Detecção Automática de Discurso de Ódio em Textos do Twitter*. 69 f. Monografia (Graduação) — Universide Federal Rural do Rio de Janeiro, Nova Iguaçu, Rio de Janeiro, 2019.

BOJANOWSKI, P. et al. *Enriching Word Vectors with Subword Information*. 2017.

BOUZY, C. *About Bot Sentinel*. 2018. Disponível em: <https://botsentinel.com/info/about>.

BROWN, T. B. et al. *Language Models are Few-Shot Learners*. 2020.

CASELLI, T. et al. HateBERT: Retraining BERT for abusive language detection in English. In: DAVANI, A. M. et al. (Ed.). *Proceedings of the 5th Workshop on Online Abuse and Harms (WOAH 2021)*. Online: Association for Computational Linguistics, 2021. p. 17–25. Disponível em: <https://aclanthology.org/2021.woah-1.3>.

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

DEVLIN, J. et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019.

EBNER, J. How germany's far right took over twitter – and tilted the election. *The Guardian*, setembro 2017. Disponível em: <https://www.theguardian.com/commentisfree/2017/sep/26/germany-far-right-election-afd-trolls>.

ELGENDY, N.; ELRAGAL, A. Big data analytics: A literature review paper. In: PERNER, P. (Ed.). *Advances in Data Mining. Applications and Theoretical Aspects*. Cham: Springer International Publishing, 2014. p. 214–227. ISBN 978-3-319-08976-8.

EVGENIOU, T.; PONTIL, M. Support vector machines: Theory and applications. In: . [S.l.: s.n.], 2001. v. 2049, p. 249–257. ISBN 978-3-540-42490-1.

FORTUNA, P. et al. A hierarchically-labeled Portuguese hate speech dataset. In: *Proceedings of the Third Workshop on Abusive Language Online*. Florence,

Italy: Association for Computational Linguistics, 2019. p. 94–104. Disponível em: <https://aclanthology.org/W19-3510>.

HO, T. K. Random decision forests. In: *Proceedings of 3rd International Conference on Document Analysis and Recognition*. [S.l.: s.n.], 1995. v. 1, p. 278–282 vol.1.

HOCHREITER, S.; SCHMIDHUBER, J. Long Short-Term Memory. *Neural Computation*, v. 9, n. 8, p. 1735–1780, 11 1997. ISSN 0899-7667. Disponível em: <https://doi.org/10.1162/neco.1997.9.8.1735>.

IBGE. *USO DE INTERNET, TELEVISÃO E CELULAR NO BRASIL*. 2019. Disponível em: <https://educa.ibge.gov.br/jovens/materias-especiais/20787-uso-de-internet-televisao-e-celular-no-brasil.html>.

JONES, K. S. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, MCB UP Ltd, v. 28, n. 1, p. 11–21, Jan 1972. ISSN 0022-0418. Disponível em: <https://doi.org/10.1108/eb026526>.

KEMP, S. *DIGITAL 2021 OCTOBER GLOBAL STATSHOT REPORT*. 2021. Disponível em: <https://datareportal.com/reports/digital-2021-october-global-statshot>.

KEMP, S. *DIGITAL 2022: BRAZIL*. 2022. Disponível em: <https://datareportal.com/reports/digital-2022-brazilbrazilabout>.

KHURANA, D. et al. Natural language processing: state of the art, current trends and challenges. *Multimedia Tools and Applications*, Springer Science and Business Media LLC, v. 82, n. 3, p. 3713–3744, jul 2022. Disponível em: <https://doi.org/10.1007%2Fs11042-022-13428-4>.

KIM, S.-B. et al. Some effective techniques for naive bayes text classification. *Knowledge and Data Engineering, IEEE Transactions on*, v. 18, p. 1457–1466, 12 2006.

KOWSARI et al. Text classification algorithms: A survey. *Information*, MDPI AG, v. 10, n. 4, p. 150, apr 2019. Disponível em: <https://doi.org/10.3390%2Finfo10040150>.

LIN, T. et al. *A Survey of Transformers*. 2021.

LIU, Y. et al. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. 2019.

MANNING, C. D.; RAGHAVAN, P.; SCHüTZE, H. *Introduction to Information Retrieval*. [S.l.]: Cambridge University Press, 2008.

MARKOV, T. et al. *A Holistic Approach to Undesired Content Detection in the Real World*. 2023.

MATHIEU, M. et al. *AlphaStar Unplugged: Large-Scale Offline Reinforcement Learning*. 2023.

MEDHAT, W.; HASSAN, A.; KORASHY, H. Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal*, v. 5, n. 4, p. 1093–1113, 2014. ISSN 2090-4479. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2090447914000550>.

MELAMUD, O.; GOLDBERGER, J.; DAGAN, I. context2vec: Learning generic context embedding with bidirectional LSTM. In: *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*. Berlin, Germany: Association for Computational Linguistics, 2016. p. 51–61. Disponível em: <https://aclanthology.org/K16-1006>.

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

PELLE, R. P. de. *Identificação de comentários ofensivos da Web*. 59 f. — Universidade Federal do Rio Grande do Sul, Rio Grande do Sul, Brasil, 2021.

PENNINGTON, J.; SOCHER, R.; MANNING, C. Glove: Global vectors for word representation. In: . [S.l.: s.n.], 2014. v. 14, p. 1532–1543.

RADFORD, A. et al. *Language models are unsupervised multitask learners*. 2019. Disponível em: <https://www.semanticscholar.org/paper/Language-Models-are-Unsupervised-Multitask-Learners-Radford-Wu/9405cc0d6169988371b2755e573cc28650d14dfe>.

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

ROSER, M.; RITCHIE, H.; ORTIZ-OSPINA, E. Internet. *Our World in Data*, 2015. Https://ourworldindata.org/internet.

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

TORFI, A. et al. *Natural Language Processing Advancements By Deep Learning: A Survey*. 2021.

VASWANI, A. et al. *Attention Is All You Need*. 2023.

VIDGEN, B. et al. Challenges and frontiers in abusive content detection. In: ROBERTS, S. T. et al. (Ed.). *Proceedings of the Third Workshop on Abusive Language Online*. Florence, Italy: Association for Computational Linguistics, 2019. p. 80–93. Disponível em: <https://aclanthology.org/W19-3509>.

W3C. *W3C Brasil - Sobre*. 2011. Disponível em: <https://www.w3c.br/Sobre/>.

YUE, L. et al. A survey of sentiment analysis in social media. *Knowledge and Information Systems*, v. 60, n. 2, p. 617–663, Aug 2019. ISSN 0219-3116. Disponível em: <https://doi.org/10.1007/s10115-018-1236-4>.