Universidade Estadual de Londrina

JOÃO VITOR FERREIRA

DETECÇÃO DE DISCURSOS AGRESSIVOS EM TWEETS COM O USO DE MODELOS DE MACHINE LEARNING

LONDRINA

2023

JOÃO VITOR FERREIRA

DETECÇÃO DE DISCURSOS AGRESSIVOS EM TWEETS COM O USO DE MODELOS DE MACHINE LEARNING

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Ciência da Computação do Departamento de Computação da Universidade Estadual de Londrina, como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof(a). Dr(a). Helen C. de Mattos Senefonte

LONDRINA

2023

Ficha de identificação da obra elaborada pelo autor, através do Programa de Geração Automática do Sistema de Bibliotecas da UEL

F383d Ferreira, João Vitor.
Detecção de Discursos Agressivos em Tweets com o Uso de Modelos de Machine Learning / João Vitor Ferreira. - Londrina, 2023.
62 f. : il.

Orientador: Helen C. de Mattos Senefonte.
Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) - Universidade Estadual de Londrina, Centro de Ciências Exatas, Graduação em Ciência da Computação, 2023.
Inclui bibliografia.

1. Ciência da Computação - TCC. 2. Discurso de Ódio - TCC. 3. Machine Learning - TCC. 4. Processamento de Linguagem Natural - TCC. I. Senefonte, Helen C. de Mattos. II. Universidade Estadual de Londrina. Centro de Ciências Exatas. Graduação em Ciência da Computação. III. Título.

CDU 519

JOÃO VITOR FERREIRA

DETECÇÃO DE DISCURSOS AGRESSIVOS EM TWEETS COM O USO DE MODELOS DE MACHINE LEARNING

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Ciência da Computação do Departamento de Computação da Universidade Estadual de Londrina, como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA

Orientador: Prof(a). Dr(a). Helen C. de Mattos Senefonte
Universidade Estadual de Londrina

Prof. Me. Lucas Busatta Galhardi
Universidade Estadual de Londrina

Prof. Me. José Luiz Villela
Universidade Estadual de Londrina

Londrina, 3 de maio de 2023.

Este trabalho é dedicado às crianças adultas
que, quando pequenas, sonharam em se
tornar cientistas.

AGRADECIMENTOS

Eu gostaria de expressar meus sinceros agradecimentos a todas as pessoas que me ajudaram ao longo desta jornada acadêmica e me apoiaram na conclusão deste trabalho.

Em primeiro lugar, agradeço a Deus, por me dar a oportunidade de estudar na UEL e a cada dia fortalecer para chegar ao fim do curso. Também gostaria de agradecer a minha orientadora, Helen C. de Mattos Senefonte, pela orientação e aconselhamento valiosos que me foram oferecidos. Seus conselhos e críticas construtivas foram fundamentais para a conclusão deste trabalho.

Gostaria de agradecer também aos professores do curso de Ciência da Computação, por compartilharem seus conhecimentos e experiências comigo. Eu sou muito grato(a) a todos vocês.

Minha família e amigos foram uma grande fonte de apoio e encorajamento ao longo deste processo. Seu amor e suporte foram inestimáveis, e eu não poderia ter feito isso sem vocês.

Por último, mas não menos importante, quero agradecer a todas as pessoas que participaram da minha pesquisa e colaboraram com informações valiosas. Sem a ajuda de todos vocês, este trabalho não seria possível.

Muito obrigado(a) a todos!

"Não vos amoldeis às estruturas deste mundo, mas transformai-vos pela renovação da mente, a fim de distinguir qual é a vontade de Deus: o que é bom, o que Lhe é agradável, o que é perfeito.
(Bíblia Sagrada, Romanos 12, 2))

FERREIRA, J. V.. **Detecção de Discursos Agressivos em Tweets com o Uso de Modelos de Machine Learning**. 2023. 62f. Trabalho de Conclusão de Curso – Versão Preliminar (Bacharelado em Ciência da Computação) – Universidade Estadual de Londrina, Londrina, 2023.

RESUMO

Com o aumento do uso das redes sociais, novas oportunidades de pesquisa vêm surgindo, as pessoas expõem suas informações pessoais, seus contatos e também fatos e informações sobre suas vidas. Apesar do lado positivo de se manter essas relações virtuais, a exposição desses dados pessoais traz também pontos negativos, como por exemplo a utilização deles para invadir contas pessoais de seus proprietários, localizar pessoas com fins maliciosos, utilizar números telefônicos para falsos sequestros, ataque de *haters*[1], os quais denigrem a imagem de muitas pessoas. A detecção precoce e a identificação de tais eventos podem conter a ameaça dessa prática antiética. O objetivo desse trabalho é propor um modelo capaz de detectar essas ameaças e auxiliar o processo de prevenção do *cyberbullying*.

Palavras-chave: Cyberbullying. Redes Sociais. Haters.

[1] Termo inglês (em tradução direta significa: "Odiadores") utilizado para definir aqueles que praticam bullying virtual.

FERREIRA, J. V.. **Detection of Aggressive Speeches in tweets using machine learning models**. 2023. 62p. Final Project – Draft Version (Bachelor of Science in Computer Science) – State University of Londrina, Londrina, 2023.

ABSTRACT

As social network usage continues to surge, new research opportunities are unfolding. People are divulging their personal information, contacts, and even aspects of their lives. While there are positive aspects to maintaining these virtual connections, the exposure of personal data also brings about negative consequences. These include the exploitation of data to infiltrate personal accounts, locate individuals with malicious intent, employ phone numbers for false abductions, and the presence of haters who defame many people. Detecting and identifying such events early on can effectively counter the threat of this unethical practice. The objective of this work is to propose a model that can detect these threats and assist in the prevention of cyberbullying. In this endeavor, I leveraged sentence transformers and employed machine learning models such as Support Vector Machines (SVM), Decision Trees, and Self-Organizing Maps (SOM) to propose a model capable of detecting these threats and bolstering the prevention process of cyberbullying.

Keywords: Cyberbullying. Social Networks. Haters

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ML	*Machine Learning*
SVM	*Support Vector Machine*
BERT	*Bidirectional Encoder Representations from Transformers.*
PCA	*Principal Component Analysis*
SOM	*Self Organizing Map*
KNN	*K-Nearest Neighbors*
SARSA	*State–action–reward–state–action*
RNN	*Recurrent Neural Networks*
LSTM	*Long Shot Term Memory*
BiLSTM	*Bidirectional Long Shot Term Memory*
CNN	*Convolutional Neural Network*
HS	*Hate Speech*
LR	*Logistic Regression*
SGD	*Stochastics Gradient Descent*
LSTM	*Long short-term memory*

SUMÁRIO

1 INTRODUÇÃO

Atualmente existe um grande problema social: o bullying. Com o avanço da tecnologia e com sua repercussão em várias áreas do cotidiano das pessoas, o bullying não se limita apenas ao mundo real, mas se estende para o mundo virtual, conhecido como *cyberbullying*. Além das agressões sofridas presencialmente, várias pessoas sofrem assédio em redes sociais, seja por meio de mensagens, fotos e até vídeos ameaçando-as. O *cyberbullying* [2] pode surgir de várias motivações diferentes, como aparência física, racismo, sexismo, inteligência, entre outros.

Com o aumento do uso de plataformas digitais para fins de educação, lazer, político, econômico, ocorre também o aumento nos incidentes de *cyberbullying* [2]. Parece inevitável, pois os alunos que sofrem bullying estão mais propensos ao *cyberbullying*, então ocorre, paralelamente, um uso cada vez maior no uso de redes sociais [3]. O bullying e a discriminação com base em raça, religião, sexo, casta e credo podem causar um efeito adverso na saúde mental da vítima, levando eventualmente à ansiedade, depressão e até mesmo ao aumento dos casos de suicídio.

Os estudiosos do fenômeno do bullying costumam analisar e descrever características distintas dos diferentes indivíduos envolvidos. Os "autores"ou "agressores"são aqueles que praticam o bullying. Eles são geralmente descritos como possuindo uma boa autoestima, sendo insensíveis aos sentimentos alheios, apresentando força física e emocional, tendo dificuldade no controle de impulsos, demonstrando um elevado grau de agressividade e um desejo de dominação [4]. Além disso, alguns agressores percebem a violência como uma qualidade. [4]. De acordo com G. R. Gredler e D. Olweus [5] enfatiza que uma discrepância de força física entre agressores e vítimas costuma estar presente nestas relações, mas considera que esta não é o fator determinante, pois juntamente com a força deve combinar-se um padrão agressivo de reação. Esse autor lembra ainda que a prática de bullying apresenta relação com fatores da criação familiar, como conflitos frequentes, discórdia e discussões abertas entre os pais, os quais podem criar relações inseguras para a criança. Além desses aspectos, a presença de violência doméstica e de excesso de permissividade constituem padrões familiares que têm sido relacionados à prática do bullying [6].

Atualmente, Redes Neurais Convolucionais (CNNs) e Redes Neurais Recorrentes (RNNs), como *Long Short-Term Memory* (LSTM) e *Gated Recurrent Units (GRUs)*, têm sido exploradas para a detecção de *cyberbullying* [7]. Esses modelos são capazes de aprender representações complexas dos textos e capturar informações contextuais importantes para a classificação. Utilizar modelos de linguagem pré-treinados, como o BERT (*Bidirectional Encoder Representations from Transformers*), tem se mostrado promissor

na detecção de *cyberbullying* [8]. Esses modelos pré-treinados podem ser ajustados para tarefas específicas de classificação, como a detecção de conteúdo ofensivo e abusivo.

Detectar termos agressivos e identificar quem o fez não é uma tarefa simples e na literatura atual, existem algumas técnicas e conceitos para executar essa árdua tarefa. Em geral são utilizados modelos de *Machine Learning* (ML), tais como *Naive Bayes* e *Support Vector Machine* (SVM), além de modelos de Processamento de Linguagem Natural (PLN) como o BERT (*Bidirectional Encoder Representations from Transformers*).

Através de um grande volume de comentários capturados em redes sociais, este trabalho tem como objetivo identificar e classificar comentários considerados bullying, utilizando técnicas e conceitos de *Machine Learning*, a fim de classificá-los de forma assertiva e definir um nível de bullying cometido.

O capítulo 2, está retratando o estado da arte e conceitos, técnicas e informações do tema. No capítulo 3, mostra a base de dados utilizada e a análise exploratória dos dados. No capítulo 4, tem-se a a limpeza e a representação numérica dos textos retirados da base. Já no capítulo 5, há a exposição de como foram aplicados os métodos e quais métricas de avaliação foram usadas. Por fim, no capítulo 6, conclui-se todo o trabalho os resultados e a sugestão de trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICO-METODOLÓGICA E ESTADO DA ARTE

Nessa seção, é apresentado o levantamento bibliográfico realizado dentre os trabalhos correlatos, visando direcionar as investigações desse projeto. Também será apresentado os fundamentos, conceitos, técnicas e modelos que serão abordados.

2.1 Trabalhos Correlatos

Dentre os trabalhos que serão apresentados, haverá aqueles em que dão embasamento para a importância deste trabalho. Haverão também, em sua grande maioria, aqueles que trazem como a literatura atual lida com detecção e classificação do discurso de ódio.

A fim de entender mais profundamente o problema, no livro [9], Moura define todos os âmbitos do problema, sua origem e como ele ocorre em redes sociais. No trabalho tem-se uma compreensão de como ele afeta a vida das vítimas e algumas forma que evitá-lo. No trabalho de Rothenburg [10] o artigo pretende discutir os limites que precisam ser traçados para enfrentar o discurso do ódio intensificado pela utilização da internet e das redes sociais que reduzem, por um lado, a interação social direta entre os atores que passam a ser produtores de mensagens e não apenas receptores, e por outro, potencializam o anonimato e permitem a publicação instantânea de conteúdos com uma velocidade gigantesca

Nos trabalhos a seguir, irá se comentar trabalhos sobre como ocorre a detecção e a classificação do discurso de ódio. Em [11], os autores apresentam uma abordagem para detecção não supervisionada de *cyberbullying*. O modelo proposto consiste em dois componentes principais: uma *Representation Learning Network* que codifica a sessão de mídia social explorando recursos multimodais, por exemplo, texto, rede e tempo; e a *Multi-task Learning Network* que ajusta simultaneamente os tempos entre chegadas de comentários e estima a probabilidade de bullying com base em um modelo de mistura gaussiana. Foi utilizado uma estrutura de detecção de *cyberbullying* não supervisionada chamada UCD (*Unsupervised Cyberbullying Detection*) via *Time-Informed Gaussian Mixture Model*, resultados experimentais em dois conjuntos de dados do mundo real corroborar a eficácia da UCD.

Em [12, 13] é abordado o problema da análise de sensibilidade de conteúdo direto. Através de *deep learning* e *sequential deep neural network models*, vários posts textuais são analisados e o usuário é classificado como sendo ou não *cyberbullying*.

Em [7] foi proposto a primeira abordagem multitarefa que aproveita o conhe-

cimento afetivo compartilhado para detectar discurso de ódio em tweets em espanhol, usando *transformer-based model*. Os resultados mostram que a combinação de polaridade e conhecimento emocional ajudam a detectar discursos de ódio com mais precisão nos conjuntos de dados.

Em [14], os autores focam na utilização de modelos de detecção de HS (*Hate Speech*) para aplicar na linguagem Roman Urdu, através de dicionario de palavras, mapeamento de gírias e tokenização [1]. A detecção foi realizada através da implementação de modelos de RNN-LSTM (*Recurrent Neural Networks - Long Shot Term Memory*), RNN-BiLSTM (*Recurrent Neural Networks - Bidirectional Long Shot Term Memory*) e CNN (*Convolutional Neural Network*).

Em [15] os autores propõem um sistema de detecção de *cyberbullying* multilíngue para detecção de *cyberbullying* em duas línguas indianas: Hindi e Marathi. O artigo propõe uma solução utilizando LR (*Logistic Regression*) a qual, funciona bem para classificação de usuários e melhora a medida que o tamanho dos dados de entrada aumentam. No artigo também é utilizado SGD (*Stochastics Gradient Descent*), na qual executa mais rápido que a solução usando LR, porém em LR, quanto maior a base de dados, menor é o erro.

Em [16] o artigo apresenta uma abordagem para detectar *cyberbullying* em *streams* árabes do Twitter. A metodologia proposta lê as mensagens do twitter em tempo real, processa e limpa os tweets do ruído, detecta mensagens ofensivas e as classifica de acordo com sua força atribuindo pesos a cada mensagem de bullying.

Em [17] o trabalho tem como objetivo propor um léxico de *cyberbullying* para mídias sociais e se concentra no *cyberbullying* de exclusão e propõe um léxico de *cyberbullying* de exclusão usando uma abordagem ontológica. O léxico proposto pode ser usado como um dicionário para os usuários nas mídias sociais.

Em [18] o artigo descreve as técnicas e recursos utilizadas na detecção do *cyberbullying*. São revisadas as fontes de dados disponíveis, os recursos e as técnicas de classificação utilizadas. NLP (*Natural Language Processing* ou, em português, Processamento de Linguagem Natural) e *Machine Learnnig* são as famosas abordagens usadas para identificar palavras-chave de bullying dentro do corpus.

Em [19] o artigo se concentra na detecção de cyberbullying em páginas com termos em espanhol e utiliza técnicas de classificação de sentimento, conjunto de termos pejorativos, na análise são utilizadas técnicas de mineração de dados para gerar um dicionário. No artigo são mencionadas algumas plataformas (como por exemplo o Mr. Tweet e o Sentimento140) para essas etapas e classifica quais são mais flexíveis para otimizar o trabalho de detecção. Foi concluído que para detectar o cyberbullying é necessário analisar o contexto e os padrões semânticos de cada frase.

[1] O processo de tokenização gera *tokens* de correspondência que são usados subsequentemente pelo processo de correspondência para identificar os registros de objeto base candidatos para correspondência.

Em [20] é construído um modelo de classificação com ótima precisão na identificação de conversas de cyberbullying usando o método *Naive Bayes* e SVM (*Support Vector Machine*). Utilizando n-gram de 1 a 5 para o número de classes 2, 4, 11 Naive Bayes produz uma precisão média de 92,81%, SVM com um poly kernel produz uma precisão média de 97,11%.

Em [1] foi utilizado uma abordagem com *Logistic Regression (LR), Naive Bayes (NB) e Random Forest Decision Tree (RFDT)* para a detecção e classificação de discurso de ódio em *tweets* indonésios. No trabalho, a metodologia tem resultados de 94.9% de *accuracy F1-score* para a tarefa de classificação utilizando regressão logística. Na base de dados presente o artigo possui 13189 *tweets*, no qual apenas 7.1% dos dados carregam códigos de ódio.

Em [21] o artigo concentra-se na detecção de *cyberbullying* em linguagens com troca de código, como acontece na Índia, por exemplo,na qual tem-se uma sociedade multilíngue. São utilizados diferentes algoritmos de Machine Learning, como (*Support Vector Machine*) e (*Logistic Regression*) e *Deep Learnnig* (*Multilayer Perceptron, Convolution Neural Network, BiLSTM, BERT*) para detectar cyberbullying de inglês-hindi (En-Hi) texto comutado por código.O modelo proposto tem resultado de 0,93 na pontuação F1com média macro.

Em [3] o trabalho propõe uma estrutura de *deep learning* que avaliará tweets ou postagens de mídia social em tempo real, bem como identificará corretamente qualquer conteúdo de *cyberbullying* neles. No artigo diz que em estudos recentes mostraram que as abordagens baseadas em redes neurais profundas são mais eficazes do que as técnicas convencionais na detecção de textos de *cyberbullying*. CNN sozinha só pode treinar características locais a partir de n-grams de palavras, com sua camada LSTM, a CNN-BiLSTM também pode aprender recursos globais e dependências de longo prazo. CNN (*Convolutional Neural network*) sozinha só pode treinar características locais a partir de n-grams de palavras, com sua camada LSTM (*Long short-term memory*), a CNN-BiLSTM (*Bidirectional Long Short-term Memory*) também pode aprender recursos globais e dependências de longo prazo. Os resultados mostrados utilizando modelos de redes neurais (CNN-BiLSTM) tem a melhor precisão.

Em [22, 11, 23] a discussão é centrada no desafio de modelar padrões temporais de comportamento de cyberbullying. Investiga como a informação temporal dentro de uma sessão de mídia social, que tem uma estrutura inerentemente hierárquica (por exemplo, palavras formam um comentário e comentários formam uma sessão), podem ser aproveitadas para facilitar a detecção de cyberbullying.

Em [24] foi apresentado uma nova aplicação do BERT (*Bidirectional Encoder Representations from Transformers*) para identificação de *cyberbullying*. Um modelo de classificação simples usando BERT é capaz de alcançar resultados de última geração em três

corpora do mundo real: Formspring, Twitter e Wikipedia. Os resultados experimentais demonstram que o modelo utilizando BERT teve os seguintes resultados na pontuação F1: 0.96, 0.94, 0.94; para os seguintes conjuntos de dados: Twitter, Wikipédia e FormSpring respectivamente. Ele alcança melhorias significativas em relação aos trabalhos existentes, em comparação com os modelos de redes neurais profundas baseados em slots ou atenção.

Em [25] são usadas CNN's (*Convolutional Neural Network*) personalizadas para o processamento de comentários de usuários do Instagram através de um sistema web. Os resultados mostram um acerto na detecção de 84,29% para *cyberbullying* e 83,08% para *cyberagressão*.

Em [26] é proposto o uso de dados de redes sociais baseadas em localização para prever e entender o comportamento de turistas em termos de padrões de mobilidade. A abordagem chamada PredicTour processa os "check-ins"dos usuários e utiliza técnicas de aprendizado de máquina para obter melhores resultados do que as abordagens tradicionais. Isso tem potencial para melhorar a experiência dos turistas e impulsionar a indústria do turismo. No trabalho é aplicado um método para extrair padrões gerais de perfis explorando informações mais abrangentes. O PredicTour primeiro utilizado o *Self organizing Maps* (SOM) nos descritores de mobilidade e logo após aplica o *Fuzzy C-Means* para o agrupamento das saídas fornecidas pelo SOM em diferentes perfis. Essa combinação é essencial para lidar com casos em zona cinzenta, isto é, perto dos limites do *cluster*.

2.2 *Machine Learning*

Tom M. Michell (1997, p. 3) [27] define *Machine Learning* como: "Campo de estudo que dá aos computadores a capacidade de aprender sem serem explicitamente programados". Um exemplo disso é um computador que aprende a jogar xadrez melhorando seu desempenho a partir das vitórias em cada partida.

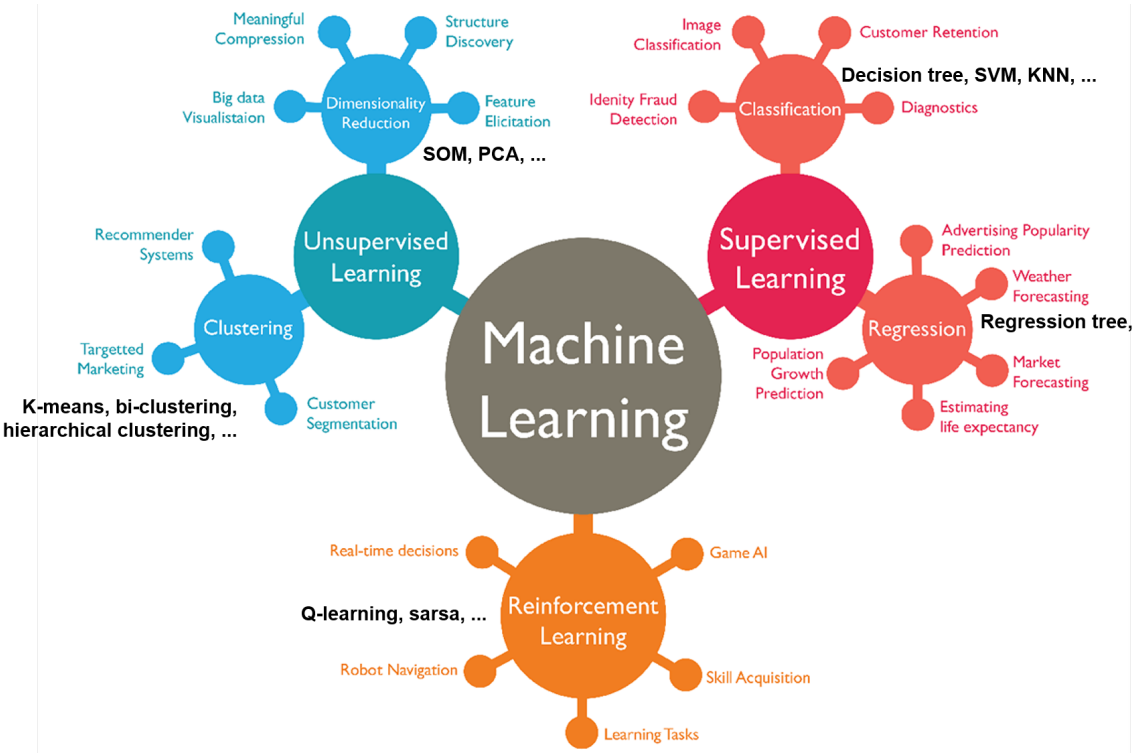
Aurélio Gerón(2019, p. 4) [28] define que "*Machine Learning* (ML) é a ciência (e arte) de programar computadores para que eles possam aprender com dados". Já Arthur Samuel (1959) afirma que "[*Machine Learning* é o] campo de estudo que dá aos computadores a capacidade de aprender sem serem explicitamente programados". Para que o computador aprenda deve haver uma inspeção no conjunto de dados, uma identificação de padrões, para que o computador possa tornar decisões sem tanta programação.

No mundo do ML, outro termo muito utilizado é *Data Mining*, na qual, neste contexto seria a aplicação de técnicas de ML para explorar grandes quantidades de dados podendo ajudar a descobrir padrões que não eram imediatamente aparentes [28].

Outro conceito importante para o desenvolvimento desse projeto é o de *Artificial Neural Networks*. Segundo Michell [27], "(...) o estudo das redes neurais artificiais (RNAs) foi inspirado em parte pela observação de que os sistemas de aprendizado biológico são

construídos de redes muito complexas de neurônios interconectados. Em analogia grosseira, as redes neurais artificiais são construídas a partir de um conjunto densamente interconectado de unidades simples, onde cada unidade recebe um número de entradas de valor real (possivelmente as saídas de outras unidades) e produz uma única saída de valor real (que pode se tornar a entrada para muitas outras unidades)" (Tom M. Michell, 1997).

Figura 1 – Diagrama que apresenta os diferentes tipos de Machine Learning



Fonte: Rajesh Khadka (2017) [29]

Na figura 1, *Machine Learning* (o circulo maior) está subdividida em 3 categorias (os outros 3 círculos de cores ciano, magenta e alaranjado), de acordo com a quantidade e o tipo de supervisão que recebe [28], que são: *supervised learning, unsupervised learning* e *reinforcement learning*. *Supervised learning* o algoritmo é treinado com dados factíveis sobre, por exemplo, uma bicicleta, e ele mapeia uma função de entrada e saída [28]. Ou seja, são definidos atributos e seus valores sobre essa bicicleta (bicicleta possui um quadro que mede 60 cm, rodas de aro 18, pedais de ferro, ...). Um clássico exemplo de *supervised learning* é o filtro de spam dos e-mails. Ele é treinado com muitos e-mails, que podem ser classificados como spam. *Unsupervised learning* o algoritmo define padrões da entrada de dados, ainda que não haja nenhuma informação fornecida previamente, isto é, o algoritmo tenta aprender sem que tenha quem o ensine[30]. Nesta categoria temos uma representação mais informativa e simples dos dados, condensando a informação em pontos

mais relevantes. Um exemplo prático hoje, ocorrem no site Linkedin [2], onde conexões são sugeridas a partir de dados do usuário. No *reinforcement learning*, existe um sistema de aprendizagem chamado de agente, na qual ele seleciona uma ação a ser executada e receber uma recompensa ou punição em troca. Logo, o agente aprende qual a melhor ação a se tomar e cria uma política de quais ações tomar em cada situação [28]. Um exemplo disso são as recomendações de vídeos do Youtube [3], onde ele sugere um vídeo e, se você o assisti-lo todo, o agente aprende que o video é relevante para o usuário (recompensa), porém, caso você, comece o video e logo troque de video, o agente assume que você não gostou do video (punição) e não o recomenda mais.

A figura 1 também ilustra os tipos de *Machine Learning* e os modelos mais populares de cada abordagem. Alguns dos modelos são brevemente descritos a seguir:

- **PCA**: *Principal Component Analysis (PCA)* ou, em português Analise de Componentes Principais, é um algoritmo estatístico usado para transformar um conjunto de variáveis possivelmente correlacionadas em um conjunto de recombinações lineares não correlacionadas dessas variáveis chamadas de componentes principais[28].

- **SOM**: *Self Organizing Maps (SOM)* ou, em português Mapas Auto-organizados, é uma técnica de dimensionamento multidimensional que constrói uma aproximação da função de densidade de probabilidade de algum conjunto de dados subjacente, que também preserva a estrutura topológica desse conjunto de dados.

- ***Decision Tree***: Em português, Árvore de Decisão, é uma estrutura de árvore semelhante a um fluxograma, onde cada nó interno denota um teste em um atributo, cada ramo representa um resultado do teste e cada nó folha (nó terminal) contém um rótulo de classe [28].

- **SVM**: *Support Vector Machine* ou, em português Máquina de Vetores de Suporte, é um algoritmo de *Machine Learning* que analisa dados para classificação e análise de regressão. O SVM é um método de aprendizado supervisionado que analisa os dados e os classifica Um SVM gera um mapa dos dados classificados com as margens entre os dois o mais distantes possível. SVMs são usados na categorização de texto, classificação de imagens, reconhecimento de escrita e nas ciências [30].

- **KNN**: *K-Nearest Neighbors* ou, em português K-vizinhos Mais Próximos, é um dos muitos algoritmos (de aprendizagem supervisionada) usado no campo de *Data Mining* e *Machine Learning*, ele é um classificador onde o aprendizado é baseado "no quão similar" é um dado (um vetor) do outro. O treinamento é formado por vetores de n dimensões [28].

[2] Rede social para profissionais que estão a procura de emprego
[3] Plataforma de vídeos online.

- ***K-means***:Em português em tradução direta seria os k-meios, seria um algoritmo de aprendizado não supervisionado que avalia e clusteriza os dados de acordo com suas características, buscando o "meio"entre elas. Na prática ele tenta separar os dados em k *clusters*, os dados geralmente têm que estar na forma de vetores numéricos. Estritamente falando, o método funcionará desde que você tenha uma maneira de calcular a média de um conjunto de pontos de dados e a distância euclidiana entre eles [28].

- ***Bi-clustering***: Em português, Agrupamento Binário, são tarefas de mineração de dados capazes de extrair informações relevantes dos dados aplicando critérios de similaridade simultaneamente a linhas e colunas de matrizes de dados. Algoritmos utilizados para realizar essas tarefas agrupam simultaneamente objetos e atributos, possibilitando a descoberta de biclusters[28].

- ***Hierarchical Clustering***: Em português, Agrupamento Hierárquico, trata-se de um algoritmo que agrupa objetos semelhantes em grupos chamados *clusters*. O resultado final é um conjunto de *clusters*, onde cada um é distinto do outro e os objetos dentro de cada *cluster* são amplamente semelhantes entre si. [28].

- ***Regression Tree***: Em português, Árvore de Regressão, é construída por meio de um processo conhecido como particionamento recursivo binário, que é um processo iterativo que divide os dados em partições ou ramificações e continua dividindo cada partição em grupos menores à medida que o método avança em cada ramificação [28].

- ***Q-learning***: Em português, Aprendizagem Q Uma política de aprendizado de reforço que encontrará a próxima melhor ação, dado um estado atual. Ele escolhe essa ação aleatoriamente e visa maximizar a recompensa. É um aprendizado de reforço fora da política, sem modelo, que encontrará o melhor curso de ação, dado o estado atual do agente. Dependendo de onde o agente estiver no ambiente, ele decidirá a próxima ação a ser tomada [30].

- **SARSA**: É um algoritmo, no qual seu nome é uma sigla de *State–action–reward–state–action* ou, em tradução direta para o português "Estado-Ação-Recompensa-Estado-Ação"é uma pequena variação do popular algoritmo Q-Learning e um dos algoritmos de aprendizado por reforço que aprende com o conjunto atual de estados e ações e aprende com a mesma política de destino. O principal ponto que diferencia o algoritmo SARSA do algoritmo Q-learning é que ele não maximiza a recompensa para o próximo estágio de ação a ser realizado e atualiza o valor Q para os estados correspondentes [30].

2.3 *Supervised Learning*

2.3.1 *Support Vector Machine*

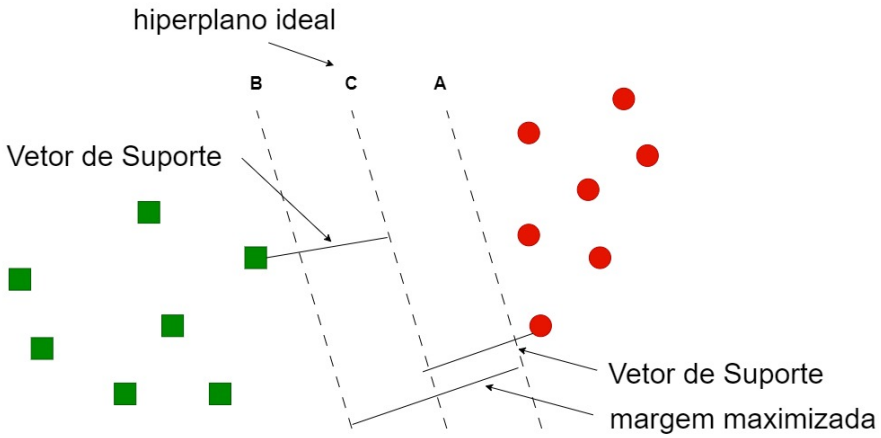
O SVM é um algoritmo de aprendizado supervisionado usado para classificação e regressão. O objetivo do SVM é encontrar um hiperplano que separe duas classes de dados de maneira ótima. O hiperplano é uma superfície que divide o espaço em duas regiões: uma região para cada classe.

Vamos considerar o caso de classificação binária, onde temos dois conjuntos de dados, o conjunto de dados de classe positiva e o conjunto de dados de classe negativa. Cada conjunto é representado por um conjunto de pontos no espaço bidimensional, onde cada ponto tem duas características (por exemplo, altura e peso). O objetivo do SVM é encontrar um hiperplano que divida esses dois conjuntos de dados de maneira ótima.

O hiperplano ideal é aquele que maximiza a margem entre as duas classes. A margem é a distância perpendicular do hiperplano até os pontos mais próximos de cada classe, chamados vetores de suporte [31]. Esses vetores de suporte são os pontos mais próximos do hiperplano, ou seja, aqueles pontos que estão mais próximos do hiperplano e que pertencem às classes opostas.

O SVM encontra esse hiperplano ótimo resolvendo um problema de otimização convexa. O problema é formulado de tal forma que minimiza a norma do vetor de pesos do hiperplano, sujeito a uma restrição de que todos os pontos estejam do lado correto do hiperplano. A norma do vetor de pesos é uma medida da complexidade do hiperplano e a restrição garante que o hiperplano separa as duas classes corretamente.

Figura 2 – Plano Cartesiano SVM



Fonte: Elaborado pelo o autor (2023)

Em resumo, o SVM é um algoritmo de aprendizado supervisionado que encontra o hiperplano que separa duas classes de dados de maneira ótima, maximizando a margem

entre as classes. O hiperplano é encontrado resolvendo um problema de otimização convexa que minimiza a norma do vetor de pesos sujeito a uma restrição de que todos os pontos estejam do lado correto do hiperplano.

2.3.2 *Decision Tree*

Decision tree, ou, em português, Árvore de decisão é um algoritmo de aprendizado de máquina supervisionado que busca construir um modelo de classificação ou regressão representado em forma de árvore. [32]

O algoritmo constrói a árvore a partir de um conjunto de dados de treinamento, que consiste em exemplos rotulados (ou seja, instâncias com uma classe conhecida). Cada nó interno da árvore corresponde a um teste em uma das variáveis preditoras (ou características) dos dados. Os ramos que saem de cada nó representam o resultado do teste (sim ou não) e levam a outros nós ou folhas.

Para construir a árvore, o algoritmo busca a variável preditora que melhor separa os dados em subconjuntos com classes distintas. A separação é feita utilizando alguma métrica de impureza, como a entropia ou o índice Gini. A cada passo, o algoritmo seleciona a variável que resulta em maior ganho de informação ou redução de impureza.

O processo continua recursivamente até que as folhas da árvore contenham apenas exemplos de uma mesma classe, ou até que um critério de parada seja satisfeito (por exemplo, um número mínimo de exemplos em cada nó ou profundidade máxima da árvore).

Uma vez construída, a árvore pode ser usada para fazer previsões em novos dados. Basta percorrer a árvore seguindo o caminho correspondente às características do exemplo, até chegar a uma folha que representa a classe prevista.

Árvores de decisão são relativamente fáceis de interpretar e explicar, mas podem ser suscetíveis a *overfitting* [4] e a erros de classificação. Diversas técnicas foram desenvolvidas para lidar com esses problemas, incluindo a poda da árvore, o uso de variantes como florestas aleatórias e *gradient boosting* (algoritmo de aprendizado de máquina que cria um modelo preditivo a partir da combinação de vários modelos simples e fracos, geralmente árvores de decisão, em uma abordagem de aprendizado baseado em conjunto. O algoritmo é chamado de "*Gradient Boosting*"porque usa um gradiente descendente para minimizar o erro do modelo durante o treinamento) e a aplicação de técnicas de pré-processamento e seleção de características.

[4] Overfitting é um problema comum em aprendizado de máquina, onde um modelo é ajustado excessivamente aos dados de treinamento e perde sua capacidade de generalização para novos dados.

2.3.3 *Logistic Regression*

A *Logistic Regression* ou regressão logística é um modelo de classificação utilizado para prever a probabilidade de uma observação pertencer a uma determinada classe. É amplamente utilizado em problemas de análise de dados em que a variável dependente é categórica, ou seja, quando queremos prever uma resposta binária (0 ou 1) ou uma resposta categórica com mais de duas categorias.

O modelo de regressão logística é baseado na função logística, que transforma a soma ponderada das variáveis independentes em uma probabilidade de pertencer à classe 1. A função logística é definida como:

$$p = \frac{1}{1+e^{-z}}$$

onde p é a probabilidade de pertencer à classe 1, z é a soma ponderada das variáveis independentes e e é a constante matemática conhecida como número de Euler.

A equação do modelo de regressão logística é dada por:

$$\mathit{logit}(p) = \ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$$

Onde p é a probabilidade de um evento binário ocorrer, $\mathit{logit}(p)$ é a transformação logit da probabilidade p, β_0 é o intercepto da reta, $\beta_1, \beta_2, ..., \beta_n$ são os coeficientes dos n preditores (ou features) e $x_1, x_2, ..., x_n$ são os valores desses preditores para a amostra de entrada.

O objetivo da regressão logística é encontrar os valores dos coeficientes que maximizam a verossimilhança dos dados. Isso é feito usando um algoritmo de otimização, como o método dos gradientes conjugados ou o método de Newton-Raphson.

Uma vez que os coeficientes são estimados, o modelo pode ser usado para fazer previsões para novos dados. A probabilidade de pertencer à classe 1 é calculada usando a função logística e um limite de decisão pode ser escolhido para classificar as observações como pertencentes à classe 1 ou classe 0.

Em resumo, a regressão logística é um modelo de classificação que usa a função logística para prever a probabilidade de pertencer a uma determinada classe. É usado em problemas de análise de dados em que a variável dependente é categórica e o objetivo é prever uma resposta binária ou categórica com mais de duas categorias.

2.4 *Unsupervised Learning*

2.4.1 *Self Organizing Maps*

Uma Rede Neural de Mapas Auto-Organizáveis (Self-Organizing Maps - SOM) é uma técnica de aprendizado não supervisionado utilizada para transformar dados complexos e de alta dimensionalidade em uma representação visual de baixa dimensionalidade. A técnica foi introduzida por Teuvo Kohonen em 1982 e é amplamente utilizada em áreas como reconhecimento de padrões, análise de dados e visualização de dados.

A ideia principal por trás de uma SOM é a de que ela aprende a mapear um conjunto de dados de entrada em uma grade de neurônios em duas dimensões. Cada neurônio da grade representa um ponto no espaço de entrada e está conectado a todos os neurônios vizinhos. A rede ajusta os pesos de cada neurônio para que ele responda aos padrões de entrada de forma organizada. Quando a SOM é treinada, cada neurônio tem um vetor de pesos associado a ele, que representa uma parte do espaço de entrada.

O primeiro passo no processo de aprendizado de mapas auto-organizados é a inicialização de todos os pesos nas conexões. Depois disso, uma amostra aleatória do conjunto de dados é usada como entrada para a rede [33]. A rede então calcula os pesos de quais neurônios são mais parecidos com os dados de entrada (vetor de entrada). Para tanto, utiliza-se esta fórmula:

$$distancia^2 = \sum_{i=0}^{n} (entrada_{\mathrm{i}} - peso_{\mathrm{i}})^2$$

Onde n é o número de conexões (pesos). O neurônio mapeado com o melhor resultado é chamado de Best Matching Unit ou BMU. Em essência, isso significa que o vetor de entrada pode ser representado com esse neurônio de mapeamento. [33]Agora, os mapas auto-organizados não estão apenas calculando esse ponto durante o processo de aprendizado, mas também tentam torná-lo “mais próximo” dos dados de entrada recebidos.

Isso significa que os pesos nesta conexão são atualizados de forma que a distância calculada seja ainda menor. Ainda assim, essa não é a única coisa que é feita. Os pesos dos vizinhos da BMU também são modificados para que fiquem mais próximos desse vetor de entrada. É assim que todo o mapa é ’puxado’ para este ponto. [34] Para isso, temos que saber o raio dos vizinhos que serão atualizados. Este raio é inicialmente grande, mas é reduzido a cada iteração (época). Portanto, a próxima etapa no treinamento de mapas auto-organizados é calcular o valor do raio mencionado. A seguinte fórmula é aplicada:

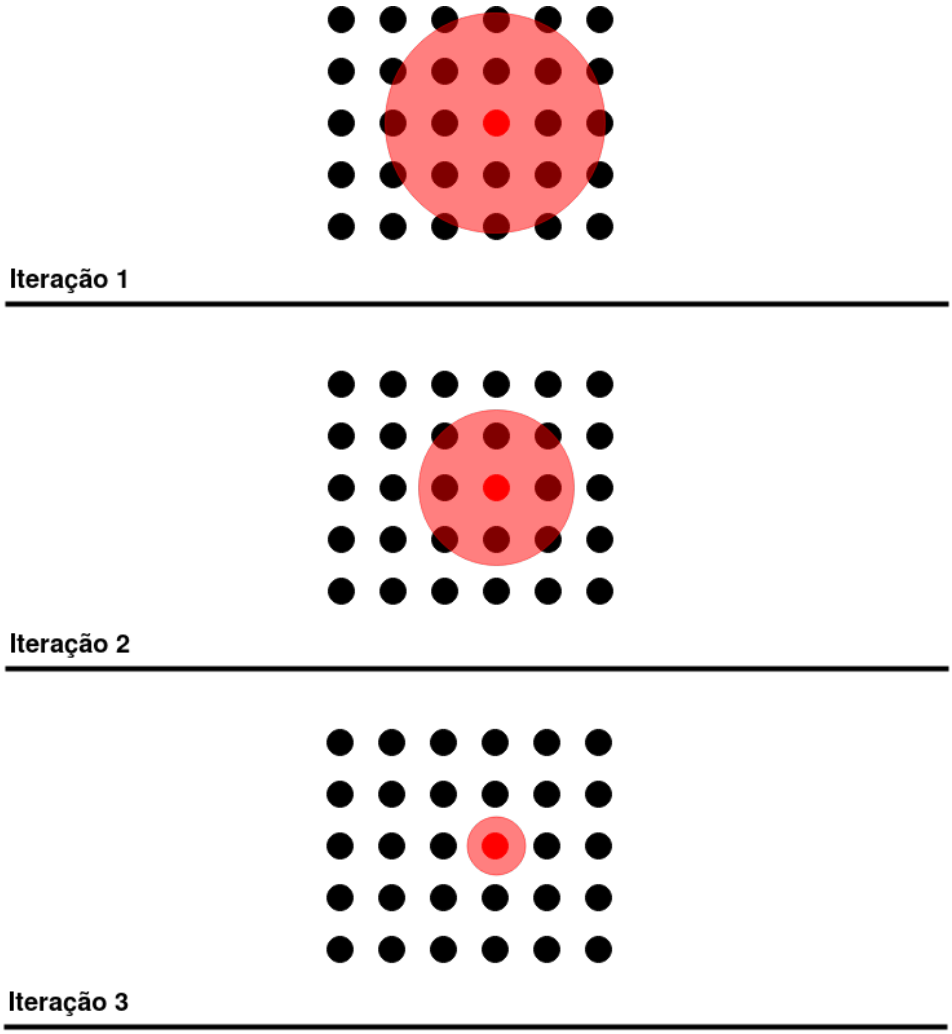
$$\sigma(t) = \sigma_0 e^{-\frac{t}{\lambda}}$$

Onde t é a iteração atual, σ_0 é o raio do mapa. O λ na fórmula é definido como:

$$\lambda = \frac{k}{\sigma_0}$$

onde k é o número de iterações. Essa fórmula utiliza decaimento exponencial, tornando o raio menor à medida que o treinamento avança, que era o objetivo inicial. Em poucas palavras, isso significa que cada iteração pelos dados trará pontos relevantes mais próximos dos dados de entrada. O mapa auto-organizado é ajustado como na Figura 3.

Figura 3 – Exemplo das iterações no SOM



Fonte: Elaborado pelo o autor (2023)

Quando o raio da iteração atual é calculado, os pesos de todos os neurônios dentro do raio são atualizados. [34] Quanto mais próximo o neurônio estiver da BMU, mais seus pesos serão alterados. Isso é obtido usando esta fórmula:

$$peso(t+1) = peso(t) + \Theta(t)L(t) \cdot (entrada(t) \text{ - } peso(t))$$

Essa é a principal fórmula de aprendizado e tem alguns pontos importantes que devem ser discutidos. O primeiro é L(t) que representa a taxa de aprendizado. Da mesma

forma que a fórmula do raio, ela está utilizando decaimento exponencial e está diminuindo a cada iteração:

$$L(t) = L_0 e^{-\frac{t}{\lambda}}$$

Além disso, mencionamos que o peso do neurônio será mais modificado se esse neurônio estiver mais próximo da BMU. Na fórmula, isso é tratado com $\Theta(t)$. Este valor é calculado assim:

$$\Theta(t) = e^{-\text{distBMU}/2\sigma(t)^2}$$

Caso o neurônio estiver mais próximo da BMU, distBMU é menor, e com isso o valor de $\Theta(t)$ fica mais próximo de 1 [35]. Isso significa que o valor do peso de tal neurônio será mais alterado. Todo este procedimento é repetido várias vezes.

Para resumir, estas são as etapas mais importantes no processo de aprendizado de mapa auto-organizado:

1. Inicialização de peso.
2. O vetor de entrada é selecionado do conjunto de dados e usado como entrada para a rede.
3. BMU é calculado.
4. O raio dos vizinhos que serão atualizados é calculado.
5. Cada peso dos neurônios dentro do raio é ajustado para torná-los mais parecidos com o vetor de entrada.
6. As etapas de 2 a 5 são repetidas para cada vetor de entrada do conjunto de dados.

Existem muitas variações das equações apresentadas usadas no processo de aprendizagem de mapas auto-organizados. [35]De fato, muitas pesquisas foram feitas tentando obter os valores ideais para o número de iterações, a taxa de aprendizado e o raio da vizinhança. O inventor, Teuvo Kohonen [33], sugeriu que esse processo de aprendizagem fosse dividido em duas fases. Durante a primeira fase, a taxa de aprendizado seria reduzida de 0,9 para 0,1 e o raio de vizinhança de metade do diâmetro da rede para os nós imediatamente adjacentes.

Na segunda fase, a taxa de aprendizado seria ainda mais reduzida de 0,1 para 0,0. No entanto, haveria o dobro ou mais iterações na segunda fase e o valor do raio da vizinhança deveria permanecer fixo em 1, significando apenas o BMU. Isso significa que a primeira fase seria usada para aprendizado e a segunda fase seria usada para ajuste fino. [33]

2.4.2 *k-means*

O *K-Means* é um algoritmo de otimização que visa minimizar a soma das distâncias quadradas entre os pontos de dados e os centroides dos *clusters* atribuídos. Essa função de custo é chamada de "inércia"[36]. O objetivo é encontrar a configuração dos *clusters* que minimize a inércia. Formalmente, a inércia (I) é definida como:

$$I = \sum_{i=0}^{n} \sum_{j=0}^{n} \|x(i) - v(j\|^2$$

onde x(i) é um ponto de dado, v(j) é o centróide do *cluster* j ao qual x(i) está atribuído, e a soma é feita sobre todos os pontos de dados e todos os *clusters*.

O algoritmo *K-means* segue os seguintes passos:

1. Inicialização:

   a) Escolha o número de *clusters* desejado, chamado de K.

   b) Selecione aleatoriamente K pontos de dados como centroides iniciais. Esses centroides serão os representantes iniciais de cada *cluster*.

2. Atribuição de Pontos de Dados aos *Clusters*:

   a) Calcule a distância entre cada ponto de dado e todos os centroides. A distância euclidiana entre um ponto de dado x(i) e um centróide v(j) é definida como:

   $$distancia(x(i), v(j)) = \sqrt{\sum_{i=0}^{n} (x(i,k) - v(j,k))^2}$$

   Onde x(i,k) é o valor da k-ésima dimensão do ponto de dado x(i), e v(j,k) é o valor da k-ésima dimensão do centróide v(j). A soma é feita sobre todas as dimensões dos dados.

   b) Atribua cada ponto de dado ao *cluster* cujo centróide está mais próximo. Isso é feito usando uma medida de distância, comumente a distância euclidiana.

   c) Cada ponto de dado é atribuído a um único *cluster* com base na menor distância.

3. Atualização dos Centroides:

   a) Calcule os novos centroides para cada *cluster*. Isso é feito encontrando a média de todos os pontos de dados atribuídos a cada *cluster*. Supondo que o *cluster* j contenha n(j) pontos de dados, a nova posição do centróide v(j) é calculada como:

   $$v(j,k) = \frac{1}{n(j)} * \sum_{i=0}^{n} x(i,k)$$

Onde x(i,k) é o valor da k-ésima dimensão do ponto de dado x(i), e a soma é feita sobre todos os pontos de dados atribuídos ao *cluster* j.

b) A média é calculada para cada dimensão dos dados. Por exemplo, se os dados tiverem duas dimensões (x e y), o novo centróide terá um novo valor x e um novo valor y.

4. Repetição dos Passos 2 e 3:

   a) Repita os passos 2 e 3 até que os centroides não mudem significativamente de iteração para iteração ou até que um critério de parada seja atingido.

   b) Os pontos de dados são atribuídos aos *clusters* correspondentes com base nas distâncias calculadas durante o processo.

5. Cada peso dos neurônios dentro do raio é ajustado para torná-los mais parecidos com o vetor de entrada.

É importante ressaltar que o *K-Means* pode encontrar ótimos locais diferentes, dependendo da inicialização dos centroides. Portanto, é comum executar o algoritmo várias vezes com diferentes inicializações e selecionar o resultado com a menor inércia ou escolher o resultado que melhor atende aos critérios e objetivos específicos do problema[36].

2.4.3 *Fuzzy C-Means*

O *Fuzzy C-Means* (FCM) é um algoritmo de agrupamento *fuzzy* que permite agrupar dados com base na similaridade entre eles. Ele estende o algoritmo de *clustering* tradicional *K-Means*, permitindo que os pontos de dados pertençam a múltiplos *clusters* com diferentes graus de pertinência, em vez de pertencerem a um único *cluster* [37] .

O FCM utiliza técnicas de lógica *fuzzy* para determinar a associação de cada ponto de dados a cada *cluster*. Em vez de atribuir um valor binário de 0 ou 1 para a associação, o FCM atribui graus de pertinência para cada ponto de dados em relação a cada *cluster* [37]. Esses graus de pertinência representam a probabilidade de um ponto de dados pertencer a um determinado *cluster*.

O algoritmo do FCM pode ser resumido em cinco etapas principais:

1. Inicialização: Defina o número de *clusters* desejados (c) e atribua valores iniciais para os centros dos *clusters* e os graus de pertinência.

2. Atualização dos Centros dos *Clusters*: Calcule os centros dos *clusters* atualizados com base nos graus de pertinência dos pontos de dados.

3. Atualização dos Graus de Pertinência: Calcule os graus de pertinência atualizados para cada ponto de dados, com base na distância entre os pontos de dados e os centros dos *clusters*.

4. Verificação de Convergência: Verifique se os centros dos *clusters* e os graus de pertinência convergiram. Se sim, vá para a etapa 5. Caso contrário, retorne à etapa 2 e repita o processo.

5. Fim: Os centros dos *clusters* finais e os graus de pertinência fornecem as informações sobre os agrupamentos resultantes.

Os cálculos no FCM envolvem principalmente a distância entre os pontos de dados e os centros dos *clusters*, bem como a atualização dos centros e dos graus de pertinência. A distância entre um ponto de dado (x) e o centro de um *cluster* (v) pode ser calculada usando uma medida de distância, como a distância euclidiana:

$$v(a,b) = \sqrt{\sum_{i=0}^{n} (a_{\mathrm{i}} - b_{\mathrm{i}})^2}$$

Os graus de pertinência são atualizados usando a fórmula:

$$u(i,j) = \frac{1}{\sum_{i=0}^{n} ((distancia(a_{\mathrm{i}},b_{\mathrm{i}})/(distancia(a_{\mathrm{i}},b_{\mathrm{k}}))^{\frac{2}{m-1}}}$$

Onde u(i, j) representa o grau de pertinência do ponto de dado i ao *cluster* j, m é um parâmetro que controla a "fuzzificação"dos *clusters* (tipicamente definido como 2) e k é o índice dos *clusters* [37]. Esses cálculos são repetidos iterativamente até que os centros dos *clusters* e os graus de pertinência convergem para um estado de estabilidade.

3 COLETA DE DADOS E ANÁLISE EXPLORATÓRIA

Para treinar uma IA a fim de aprender algum padrão, é necessário uma base de dados, com conteúdo verídico e relevante para o problema. Neste trabalho, inicialmente, o intuito era coletar periodicamente tweets através da API do Twitter, com a finalidade de usá-los como base de dados de treino para a IA. Entretanto, recentemente, o Twitter alterou sua política de privacidade quanto às informações que disponibilizaria via API, limitando assim o conteúdo disponibilizado. Sendo assim, as informações que o Twitter disponibilizaria não seriam suficientes para a finalidade deste trabalho. Desta forma, fez-se necessário a busca de alguma base de dados já coletada de plataformas disponíveis.

3.1 Base 1

Esta base foi encontrada na plataforma *Kaggle*, aonde existem varias base de dados relevantes para o trabalho, e a escolhida vamos chama-la de Base 1 [1], na qual contém 47.692 tweets. Abaixo, há uma tabela com um exemplo de como é a Base 1:

Tabela 1 – Exemplo Base 1

tweet_text	**cyberbullying_type**
...	...
Argh another round of instant restaurants....over it!!!! #mkr	not_cyberbullying
@ManhattaKnight I mean he's gay, but he uses gendered slurs and makes rape jokes	gender
RT @HellOnHeelsGirl: Dasani water is fucking disgusting and if you don't agree I literally can't even with you	other_cyberbullying
...	...

Fonte: Elaborado pelo o autor (2023)

A Base 1 dispõe-se em duas colunas: uma com o tweet, outra com a classificação de "not_cyberbullying", "gender", "religion", "age", "ethnicity" ou outro (Neste caso, tem-se o texto que foi detectado como bullying). Como esta base ja contém uma coluna com a classificação do comentário, pode-se usá-la para comparação com os resultados, ou seja, verificar se a detecção e classificação está funcionando adequadamente. O gráfico abaixo (Figura 4) demonstra a relação entre o número de tweets e o tipo de cyberbulying ocorrido no texto do tweet:

[1] <https://www.kaggle.com/code/ludovicocuoghi/detecting-bullying-tweets-pytorch-lstm-bert/data>

A partir do gráfico observa-se que o número de tweets para cada tipo de cyberbullying são bem próximos uns dos outros. O que demonstra que a Base 1 é ótima para treinar uma IA para classificação de cyberbullying, pois como não há muita discrepância do numero de tweets entre os tipos, a IA não irá tender a classificar um novo texto para o tipo de tweet com maior número.

3.2 Base 2

A outra base de dados, foi da plataforma chamado *data.wolrd*, o qual disponibiliza várias base de dados de diversas partes do mundo, com diferentes conteúdos. A base (vamos chamá-la de Base 2 (<https://data.world/data-society/twitter-user-data>)) contém várias informações sobre o *tweet*, o que agrega valor na classificação e detecção de um possível usuário malicioso. Pois, a partir dessas informações pode-se determinar os perfis de usuários com esse tipo de intenção. Na tabela 2 está a lista com todas as colunas da Base 2, com suas respectivas descrições.

Na Base 2 existem especificamente 20.050 linhas com seus tweets e classificação. Dentre esses tweets, como mostra Figura 5, 33,4% são de usuários masculinos, 30,9% são de usuários femininos, 29,6% são de usuários de outro gênero e 6,1% são de usuários que não identificaram seu gênero. Portanto, a Base 2 está muito bem dividida em questão de gênero. Exceto aqueles que não informaram seu gênero, a diferença entre os usuários de gênero diferente não ultrapassa os 4%, o que demonstra uma subdivisão adequada.

A partir das porcentagens da Figura 5, tem-se uma ideia de como se divide os perfis dos usuários do Twitter, pertencentes a Base 2. Contudo a proporção com que cada usuário utiliza a plataforma é diferente para cada gênero, como mostra o gráfico da figura Figura 6. No gráfico podemos observar que, os usuário que mais postam conteúdo no Twitter são usuários do gênero não tradicionais (masculino e feminino), postando 1,8 vezes mais que os usuários do gênero masculino, o segundo maior.

Quanto a localização de onde foram postados os tweets, na Base 2 existe o campo "user_timezone", que serve para identificar em qual fuso horário o tweet foi postado. Com este parâmetro, conseguimos também agrupar os tweets por essas zonas, a fim de determinar onde há a concentração maior de tweets.

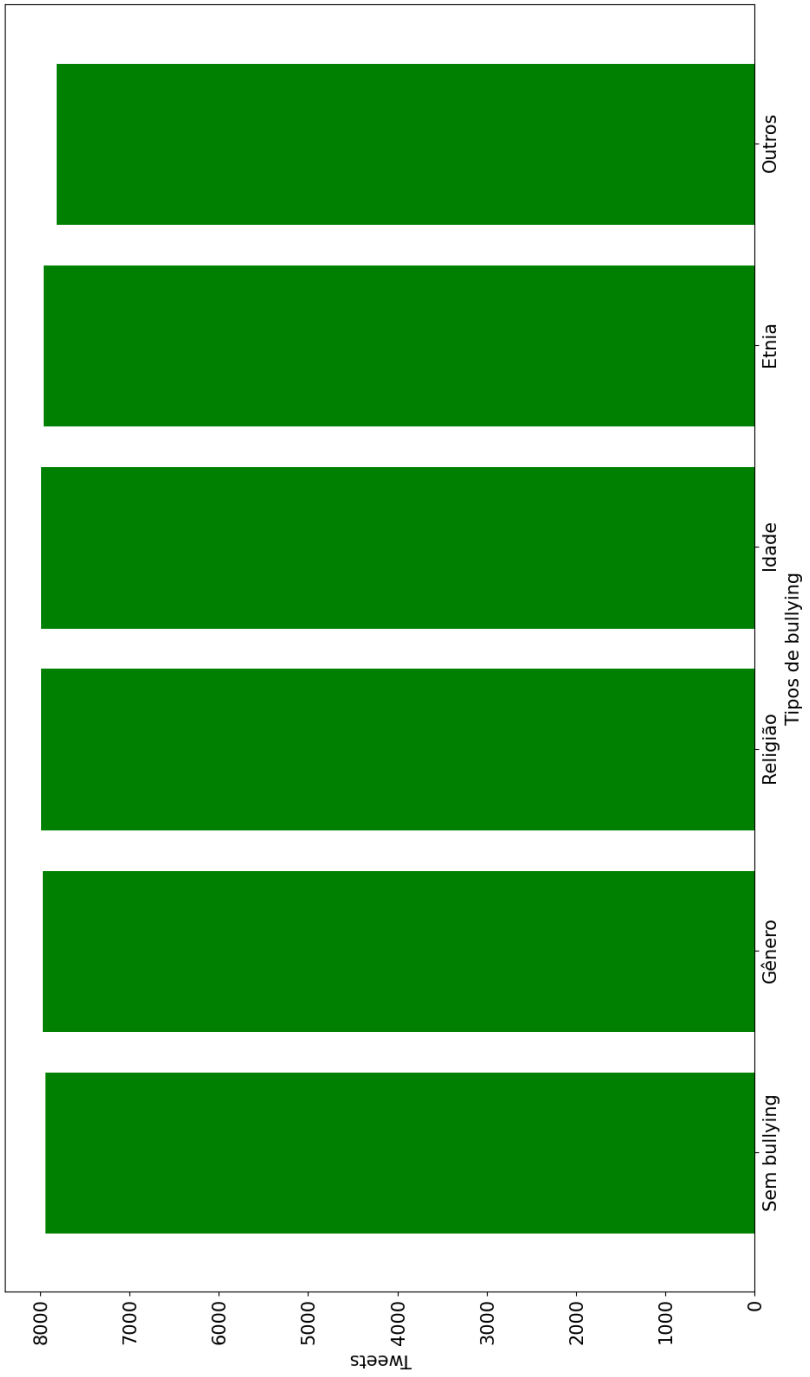
Além disso, após a classificação, saberemos onde mais ocorre o cyberbullying, na Base 2. Dessa forma, produzimos um gráfico (Figura 7) com os fuso horários com maior concentração de tweets.

Na Figura 7, a região de Eastern Time (US & Canada) é onde concentra-se a maior parte dos posts, com 2496 tweets. Observa-se também que a segunda (Pacific Time (US & Canada)) e a terceira (Central Time (US & Canada)) regiões de maiores quantidades de tweets são dos Estados Unidos, com 2106 e 1505 tweets, respectivamente. Logo, calcula-se

que 30,4% de todos os tweets da Base 2 são norte-americanos. Entretanto, 7798 tweets não foram classificados em nenhuma região do fuso horário, pois o campo estava em branco. O Restante das regiões (das que não aparecem no gráfico) tinham um número de tweets menor que 100, logo, não fariam impacto neste caso.

Na Figura 8, tem-se uma ideia do comportamento dos usuários do Twitter durante o dia. Percebe-se que há alguns picos de no gráfico, nos horários 8:00, 13:00, 19:00, que são os horários onde houve maior número de posts por período do dia (manhã, tarde e noite, respectivamente). também observa-se que no período das 8:00 às 15:00, foi quando ocorreram o maior número de posts do dia, ultrapassando 1000 tweets por hora. Já no período das 0:00 às 4:00, foi quando ocorrem menos posts do dia, chegando ao numero máximo de 515 tweets na maior hora.

Figura 4 – Gráfico do tipo de bullying pelo número de tweets da Base 1.



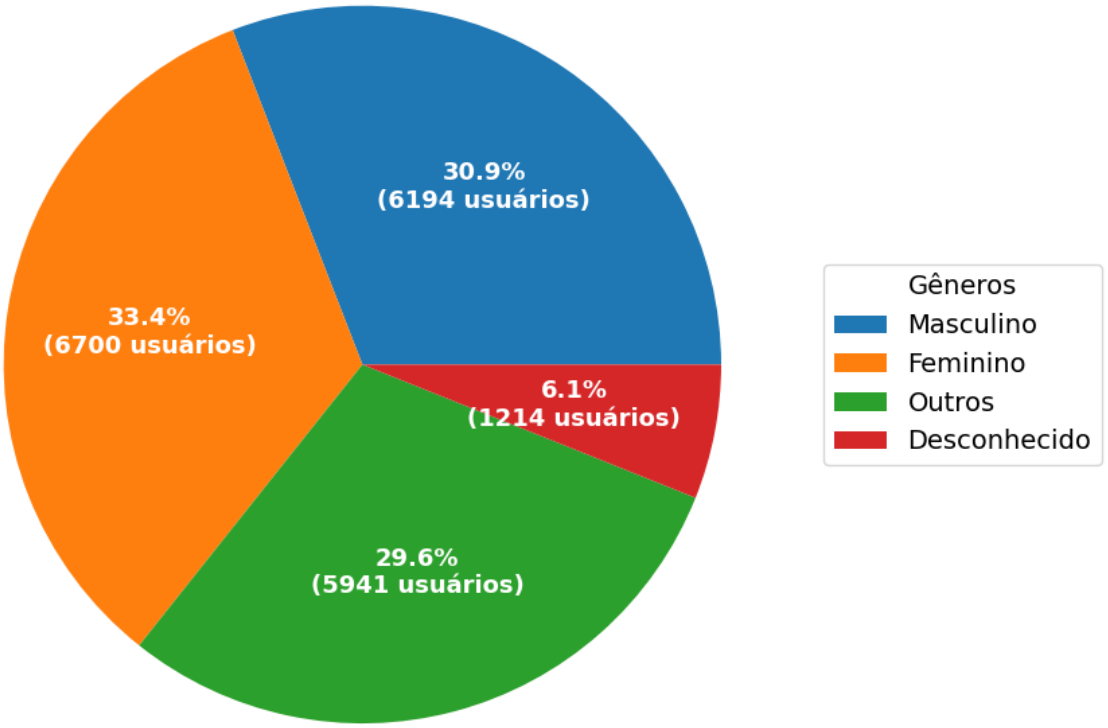
Fonte: Elaborado pelo o autor (2023)

Tabela 2 – Descrição dos campos da Base 2

Campo	Descrição
_unit_id	Um id exclusivo para o usuário.
_golden	Se o usuário foi incluído no padrão ouro para o modelo; TRUE ou FALSE.
unit_state	Estado da observação; "finalized"para o avaliado pelo colaborador ou "golden"para observações padrão-ouro.
_trusted_judgments	número de julgamentos confiáveis (int); sempre 3 para não-ouro, e o que pode ser um id único para observações
_last_judgment_at	data e hora do último julgamento do colaborador; em branco para observações padrão-ouro
gender	"male", "female", ou "brand"(para perfis não humanos)
gender:confidence	Um float representando a confiança no gênero fornecido
profile_yn	"no"aqui significa que o perfil deveria fazer parte do conjunto de dados, mas não estava disponível quando os contribuidores foram julgá-lo
profile_yn:confidence	confiança na existência/inexistência do perfil.
created	Data e hora em que o perfil foi criado.
description	A descrição do perfil do usuário
fav_number	Número de tweets que o usuário adicionou como favorito
gender_gold	Se o perfil for golden, qual é o gênero?
link_color	A cor do link no perfil, como um valor hexadecimal.
name	O nome do usuário.
profile_yn_gold	Se o valor do perfil s/n é dourado.
profileimage	Um link para a imagem do perfil.
retweet_count	Número de vezes que o usuário retweetou (ou possivelmente foi retweetado).
sidebar_color	Cor da barra lateral do perfil, como um valor hexadecimal.
text	texto de um tweet aleatório do usuário.
tweet_coord	Se o usuário tiver a localização ativada, as coordenadas como uma string com o formato "[latitude , longitude]".
tweet_count	Número de tweets que o usuário postou.
tweet_created	Quando o tweet aleatório (na de texto) foi criado coluna.
tweet_id	O id do tweet aleatório.
tweet_location	Localização do tweet; parece não ser particularmente normalizado.
user_timezone	O fuso horário do usuário.

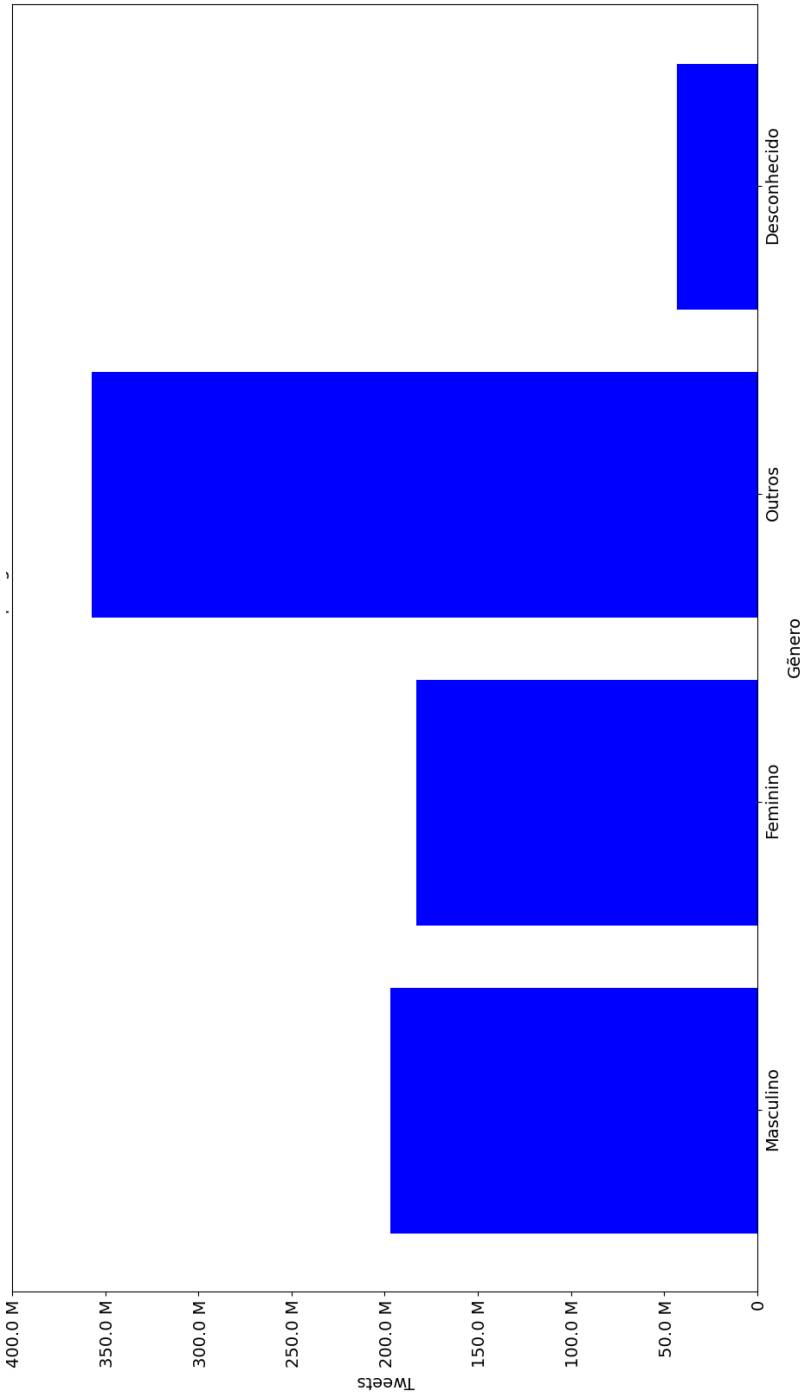
Fonte: Elaborado pelo o autor (2023)

Figura 5 – Gráfico da porcentagem de usuários separados por gênero da Base 2.



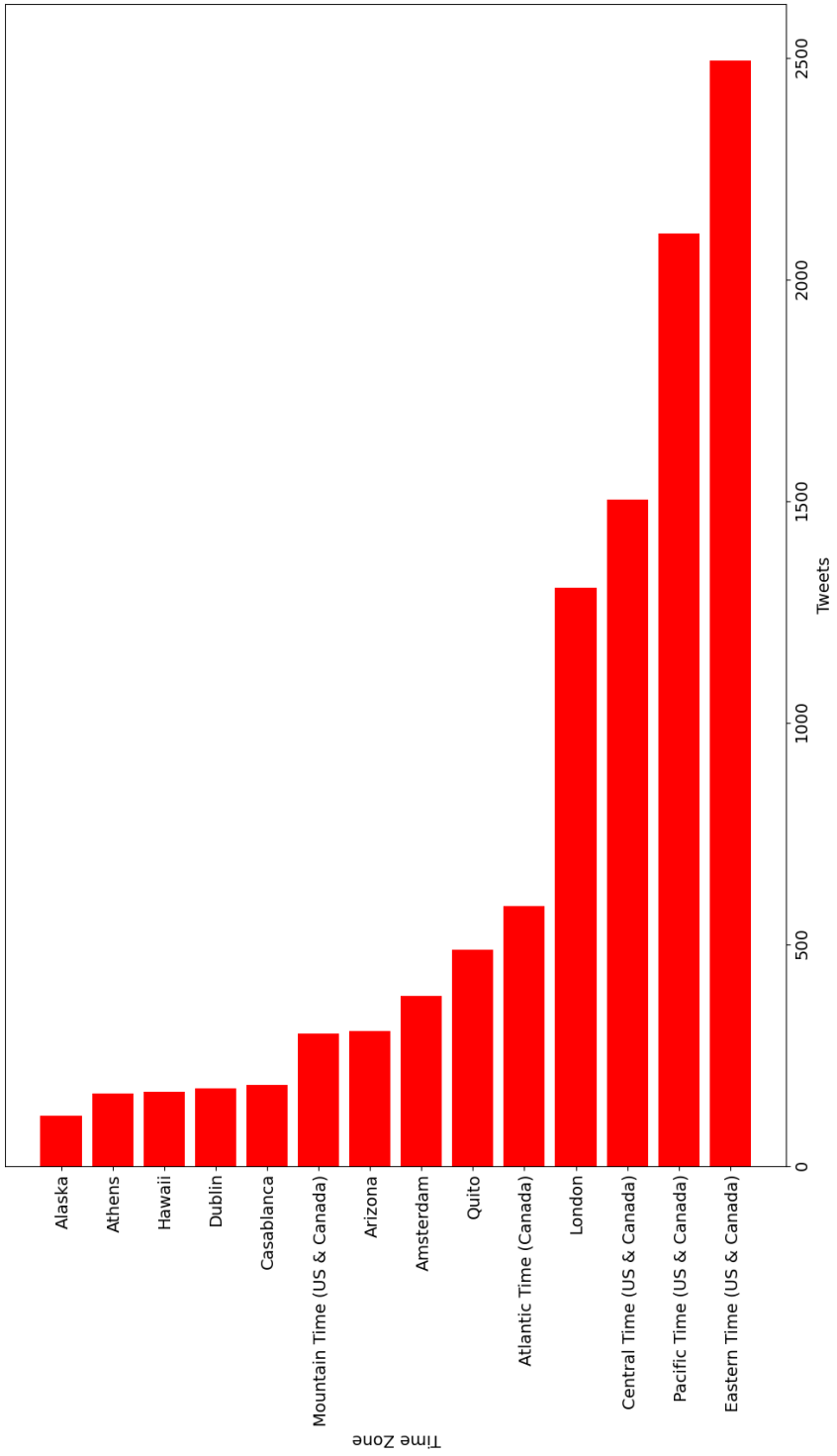
Fonte: Elaborado pelo o autor (2023)

Figura 6 – Gráfico do número de tweets por gênero da Base 2.



Fonte: Elaborado pelo o autor (2023)

Figura 7 – Gráfico da Região (do fuso horário) do post pela quantidade de tweets da Base 2 .



Fonte: Elaborado pelo o autor (2023)

Figura 8 – Gráfico do número de tweets por cada hora do dia da Base 2.

Fonte: Elaborado pelo o autor (2023)

4 PRÉ-PROCESSAMENTO DOS DADOS

O pré-processamento de dados é uma etapa fundamental no processo de análise de dados que envolve a preparação dos dados brutos coletados para que possam ser usados em análises posteriores. É uma etapa crítica para garantir que os dados estejam prontos para serem usados em algoritmos de aprendizado de máquina e outras técnicas de análise de dados.

4.1 Limpeza dos dados

A limpeza dos dados é uma das etapas mais importantes do pré-processamento de dados e envolve a identificação e correção de problemas nos dados brutos coletados, com o objetivo de garantir que os dados estejam prontos para serem usados em análises posteriores [38]. Visto que os dados a serem analisados são textuais, alguns dos problemas comuns que podem ser identificados durante a limpeza de dados. Neste trabalho foram aplicados as seguintes técnicas na limpeza dos dados:

4.1.1 Tratamento de abreviações e siglas

O tratamento de abreviações e siglas é uma etapa importante na limpeza de dados textuais, pois abreviações e siglas são comuns em textos e podem apresentar problemas para a análise de texto se não forem tratadas adequadamente. Abreviações e siglas devem ser padronizadas para que todas sejam representadas de maneira consistente no texto. Uma das técnicas comuns para o tratamento destes é a expansão de abreviações e siglas. Isso envolve a substituição da abreviação ou sigla por sua forma completa. Por exemplo, a abreviação "SOM"pode ser expandida para "Self Organizing Maps"e a sigla "PCA"pode ser expandida para "Principal Component Analysis". O tratamento de abreviações e siglas é importante porque pode melhorar a precisão da análise de texto, garantindo que todas as informações relevantes sejam incluídas no conjunto de dados. Além disso, pode ajudar a reduzir a ambiguidade e melhorar a legibilidade do texto para os usuários finais. Foram identificados nos textos esses termos e substituídos por uma *tag* chamada SIGLA.

4.1.2 Tratamento de Hashtags, Links e Usernames

Em alguns tweets, há menção a alguns desses termos, e neste caso, alguns termos podem trazer o significado errado a frase, como por exemplo: "@Firebomb173 @ANDAA-SONSAN not the first time it's happened. That was probably the worst though.". Neste exemplo, o nome do usuário "@Firebomb173", pode trazer um significado errado a frase, se levado em consideração o seu significado na frase. Sendo assim, para usuários, serão subs-

tituídos seus nomes por "user", assim deixando claro que se trata de um usuário. Vejamos o exemplo a seguir: "RT @Hanar_Marouf Daesh is distrbuting Stolen U.N. Humanitarian Aid W Its Own Logo!! #Iraq #twitterkurds http //t.co/kSkGLrh1jt http://t.chyrtj"nele percebemos que existem várias *hashtags*, algumas que agregam significado a frase, outras fazem referência a outros contextos. Sendo assim, optamos por substitui-las pelo termo "hashtag"na frase, e os links, que não agregam significado semântico a frase, substituímos por "link".

4.1.3 Expressões regulares

A utilização de expressões regulares é uma técnica muito poderosa para a limpeza de textos. As expressões regulares são sequências de caracteres que definem um padrão de busca em um texto. Elas são usadas para buscar e substituir padrões específicos de caracteres em um texto [39]. As expressões regulares são úteis para diversas tarefas de limpeza de texto, como a remoção de caracteres especiais, números, espaços em branco, tags HTML, entre outras coisas. Algumas das tarefas mais comuns que podem ser realizadas com expressões regulares incluem:

- **Remoção de caracteres especiais:** Expressões regulares podem ser usadas para remover caracteres especiais de um texto. Um exemplo de expressão regular para remover caracteres especiais é a seguinte: [^\w\s]. Essa expressão regular vai buscar por qualquer caractere que não seja alfanumérico (\w) ou espaço em branco (\s), e remover todos eles do texto.

- **Remoção de números:** Se os números não são relevantes para a análise de texto, as expressões regulares podem ser utilizadas para removê-los. Um exemplo de expressão regular para remover números é a seguinte: \d+. Essa expressão regular vai buscar por um ou mais dígitos (\d) e remover todos eles do texto.

- **Remoção de espaços em branco:** Espaços em branco podem ser removidos com expressões regulares. Um exemplo de expressão regular para remover espaços em branco é a seguinte: \s+. Essa expressão regular vai buscar por um ou mais espaços em branco (\s) e removê-los do texto.

- **Remoção de tags HTML:** Se o texto contém tags HTML, é possível utilizar expressões regulares para removê-las. Um exemplo de expressão regular para remover tags HTML é a seguinte: <.*?>. Essa expressão regular vai buscar por qualquer tag HTML, incluindo os caracteres < e > que as delimitam, e remover tudo o que estiver entre eles do texto.

- **Substituição de padrões de texto:** As expressões regulares podem ser usadas para substituir padrões de texto por novas sequências de caracteres. Por exemplo,

se houver a necessidade de substituir todas as ocorrências da palavra "amigo"por "colega", é possível usar a seguinte expressão regular: amigo e substituir por colega. Dessa forma, todas as ocorrências de "amigo"serão substituídas por "colega". No trabalho foi necessário substituir alguns textos com erros de digitação detectáveis com expressões regulares, por exemplo, palavras com substituição de letras por dígitos, como em: "r3m41n", que seria "remain", mas as vogais foram substituídas por dígitos.

4.1.4 Tratamento de *Stop Words*

A remoção de *stop words* é uma técnica bastante simples e efetiva de pré-processamento de dados em processamento de linguagem natural [38]. O objetivo é remover palavras que ocorrem frequentemente em um texto, mas que geralmente não contribuem para compreensão do modelo usado. Essas palavras são chamadas de "*stop words*". As *stop words* geralmente incluem pronomes, artigos, preposições e outras palavras comuns em um idioma. Exemplos de *stop words* em inglês incluem *"a", "an", "the", "and", "in", "on", "of", "to", "that", "this", "is", "are", "was", "were", "be", "being", "been"*, entre outras.

A remoção de *stop words* pode ser feita de diferentes maneiras, dependendo do objetivo da análise de texto. Uma maneira comum é usar uma lista de *stop words* predefinida para o idioma em questão. Existem várias listas de *stop words* disponíveis na internet para diferentes idiomas. Ao remover *stop words*, é importante ter em mente que nem sempre é a melhor opção em todas as situações. Em algumas tarefas, como classificação de texto, a remoção de *stop words* pode melhorar a precisão do modelo, enquanto em outras tarefas, como análise de sentimentos, a remoção de *stop words* pode prejudicar a precisão do modelo. [39]

Além disso, em alguns casos, as *stop words* podem ser relevantes para a análise. Por exemplo, em uma análise de sentimentos, as palavras "não"e "mas"podem alterar significativamente o sentido de uma frase, então é importante considerá-las na análise. Em resumo, a remoção de *stop words* é uma técnica simples e útil de pré-processamento de dados em processamento de linguagem natural, mas deve ser usada com cuidado e avaliada de acordo com o contexto da análise de texto.

O *Natural Language Toolkit (NLTK)* é uma biblioteca de processamento de linguagem natural em Python, a qual foi utilizada no projeto. Ela oferece várias ferramentas para a limpeza e pré-processamento de dados de texto, incluindo a remoção de *stop words*. Para usar o NLTK para remover *stop words* em um texto, é necessário primeiramente importar a biblioteca e baixar as *stop words* para o idioma desejado.

4.1.5 Tokenização

Tokenização é o processo de dividir um texto em unidades menores, chamadas de *tokens*. Os *tokens* podem ser palavras, pontuação ou outras sequências de caracteres que tenham algum significado no contexto do texto [38]. A tokenização é uma etapa importante no pré-processamento de dados de texto, pois ela transforma um texto em uma sequência de elementos que podem ser processados posteriormente.

Existem várias maneiras de realizar a tokenização de um texto, dependendo do tipo de dados de texto e das necessidades do projeto. Uma abordagem comum é dividir o texto em palavras individuais, removendo espaços em branco, pontuação e caracteres especiais. Isso pode ser feito usando uma expressão regular ou uma biblioteca de processamento de linguagem natural, como o NLTK (*Natural Language Toolkit*) do Python.

Por exemplo, considerando a seguinte frase:

"O gato está dormindo na janela."

A tokenização dessa frase poderia resultar na seguinte lista de *tokens*:

["O", "gato", "está", "dormindo", "na", "janela", "."]

Esses *tokens* podem então ser processados posteriormente para análise de texto, como por exemplo contagem de frequência de palavras, identificação de entidades nomeadas, criação de modelo de linguagem, entre outros.

A tokenização é uma etapa fundamental em muitas aplicações de processamento de linguagem natural, e pode ser realizada de diversas formas, dependendo do objetivo e do contexto do projeto.

Para realizar a tokenização das frase no trabalho, foi utilizada uma biblioteca de processamento de linguagem natural, como o NLTK (*Natural Language Toolkit*) do Python. Através do NLTK, podemos dividir a frase em uma lista de palavras individuais, removendo espaços em branco, pontuação e caracteres especiais.

4.1.6 *Stemming e Lemmatization*

Stemming (do inglês, "derivação") e *lemmatization* (do inglês, lematização) são técnicas de processamento de linguagem natural utilizadas para reduzir uma palavra ao seu núcleo ou raiz, a fim de normalizar e simplificar o texto e, assim, facilitar a análise e o processamento de dados de texto.

A diferença básica entre as duas técnicas é que o *stemming* corta a palavra ao seu núcleo usando regras simples, enquanto a lematização usa um dicionário para obter a forma básica da palavra, levando em consideração sua classe gramatical. Vamos entender cada técnica com mais detalhes:

- ***Stemming*:** é um processo de transformação de palavras em suas formas básicas, ou "stems". A ideia é que diferentes formas de uma palavra possam ser contabilizadas como uma só palavra, mesmo que apresentem variações devido a conjugação, gênero ou número [38]. Por exemplo, as palavras "corrida", "corredor"e "corremos"seriam reduzidas a "corr"usando um algoritmo de *stemming*. Vamos ver outro exemplo:

  Palavras: "computação", "computador", "computadores"*Stemming*: "comput"

  Note que o *stemming* não leva em conta a classe gramatical da palavra e, portanto, pode gerar resultados imprecisos ou até mesmo palavras que não existem. No entanto, é uma técnica útil para reduzir o tamanho do vocabulário e facilitar a análise de dados de texto.

- **Lematização:** é uma técnica mais sofisticada que leva em conta a classe gramatical da palavra para encontrar sua forma básica, ou "lema". Isso significa que a lematização pode gerar resultados mais precisos do que o *stemming*, pois considera a estrutura gramatical da frase [38]. Por exemplo, a palavra "corrida"seria reduzida a "correr"usando um algoritmo de lematização, pois ambas são formas verbais do mesmo verbo. Vamos ver outro exemplo:

  Palavras: "corri", "correu", "corrida", "correr"Lematização: "correr"

  Note que a lematização pode gerar resultados mais precisos do que o *stemming*, pois leva em conta a estrutura gramatical da frase e a relação entre as palavras. No entanto, é uma técnica mais complexa do que o *stemming* e pode ser mais lenta de ser executada.

Ambas as técnicas são úteis para normalizar o texto e reduzir as variações nas palavras, permitindo uma análise mais precisa dos dados de texto. No entanto, a escolha de qual técnica usar depende do contexto do projeto e do tipo de análise que está sendo realizada. Neste trabalho foi utilizado a técnica de lematização, oriunda a biblioteca NLTK do Python. Nela tem-se a função lemmatize que aplica este método.

4.2 Representação Numérica de Textos

Após feito todo o processo de limpeza dos dados, neste caso, textos, tem-se uma base de dados menor, porém, ainda com muita informação. Como resultado, obteve-se uma base de dados onde cada linha tem um vetor de palavras. Contudo, tratar esse vetor de palavras computacionalmente, não é tão simples e pode se tornar um grande problema para o trabalho. Os modelos computacionais esperam dados numéricos, logo, como próximo etapa, faz-se necessário representar esses vetores de palavras de forma numérica.

Existem algumas técnicas de representar estes vetores de palavras como valores numéricos. As mais comuns são: *Bag of Words* e *Word Embedding*.

4.2.1 *Bag of Words*

O *Bag of Words (BoW)* é uma técnica usada em processamento de linguagem natural para representar documentos de texto como vetores numéricos. Ele é baseado na ideia de que o significado de um documento pode ser capturado pela frequência das palavras contidas nele. [40]

O primeiro passo do BoW é criar um vocabulário de palavras únicas a partir de um corpus de texto, que é um conjunto de documentos usados como base para a análise. Em seguida, cada documento é representado como um vetor numérico, onde cada posição no vetor corresponde a uma palavra no vocabulário e o valor numérico dessa posição corresponde à frequência dessa palavra no documento.

Essa técnica é chamada de "*Bag of Words*"(saco de palavras) porque ela não leva em conta a ordem das palavras no documento, mas apenas sua frequência. Em outras palavras, o BoW trata cada documento como um "saco"de palavras, sem levar em conta a ordem ou a estrutura gramatical. Dado as sentenças em inglês: *"welcome to great learning, now start learning"* e *"learning is a good practice"*. Vamos chamá-las de S1 e S2, respectivamente. Abaixo tem-se uma tabela que representa como seria a composição de um BoW:

Tabela 3 – *Bag Of Words*

	welcome	**great**	**learning**	**now**	**start**	**good**	**practice**
Sentença 1	1	1	2	1	1	0	0
Sentença 2	0	0	1	0	0	1	1

Fonte: Elaborado pelo o autor (2023)

Uma vez que os documentos são representados como vetores numéricos, eles podem ser usados como entradas para algoritmos de aprendizado de máquina, como classificadores ou agrupadores. Por exemplo, o vetor numérico de um documento pode ser usado para classificar o documento em uma categoria pré-definida ou para agrupar documentos semelhantes em um conjunto de documentos.

Embora o BoW seja uma técnica simples e eficaz para representar documentos de texto, ele tem algumas limitações. Uma das limitações é que ele não leva em conta a semântica das palavras, ou seja, palavras com significados semelhantes podem ter vetores numéricos diferentes. Além disso, ele pode ser sensível a palavras muito comuns ou muito raras, que podem não ter um impacto significativo na representação do documento.

Apesar de suas limitações, o BoW é amplamente utilizado em processamento de linguagem natural e aprendizado de máquina, principalmente em tarefas de classificação

de texto e agrupamento de documentos. Ele também pode ser combinado com outras técnicas, como a redução de dimensionalidade, para melhorar a precisão e a eficiência do modelo.

4.2.2 *Word Embedding*

Word Embedding é uma técnica de processamento de linguagem natural usada para representar palavras como vetores numéricos densos em um espaço de alta dimensão. [41] Essa técnica permite que as palavras sejam representadas em um espaço semântico, onde palavras semelhantes são próximas umas das outras e palavras com significados diferentes são distantes umas das outras.

4.2.2.1 Word2Vec

Existem várias técnicas de *Word Embedding*, mas a mais comum é o modelo Word2Vec. O Word2Vec é um modelo de aprendizado de máquina que usa uma rede neural para aprender a representação vetorial de palavras. O modelo é treinado em um corpus de texto grande, onde as palavras são representadas como vetores numéricos densos em um espaço de alta dimensão. O objetivo do modelo é prever a probabilidade de uma palavra aparecer em uma determinada posição em uma sentença, dada a palavra anterior.

Ao treinar o modelo Word2Vec, as palavras que aparecem frequentemente juntas são agrupadas em regiões próximas umas das outras no espaço de alta dimensão. Além disso, o modelo também captura relações semânticas entre as palavras, como a relação entre "rei"e "rainha", que são representados como vetores que apontam em direções semelhantes no espaço.

Com a representação vetorial de palavras, as operações matemáticas podem ser aplicadas aos vetores para obter resultados semânticos significativos. Por exemplo, a diferença entre os vetores de "rei"e "homem", somado ao vetor de "mulher", resulta em um vetor que é próximo ao vetor de "rainha".

A técnica de *Word Embedding* tem sido amplamente utilizada em tarefas de processamento de linguagem natural, como análise de sentimento, classificação de texto, tradução automática e resumo automático de texto. Com a representação vetorial de palavras, as tarefas de processamento de linguagem natural se tornam mais precisas e eficientes.

4.2.3 *Sentence Embedding*

Sentence embedding é uma técnica de processamento de linguagem natural (NLP) que tem como objetivo representar uma sentença em um espaço vetorial contínuo de alta dimensão, onde cada dimensão representa uma característica da sentença. A representação

vetorial da sentença é obtida por meio de um modelo de aprendizado de máquina que mapeia a sentença para um espaço vetorial contínuo. [8]

Uma abordagem comum é usar modelos de linguagem pré-treinados, como o BERT [42] e o GPT-2 [43], que foram treinados em grandes conjuntos de dados para capturar a semântica de sentenças e palavras. Esses modelos pré-treinados são capazes de gerar *embedding* de sentenças de alta qualidade que podem ser usados em tarefas específicas de NLP, como classificação de texto, resumo automático de texto e análise de sentimentos.

Outra abordagem mais recente e avançada para a geração de *embeddings* de sentenças é o uso de redes neurais de codificação de sentenças. Esse modelo de codificação de sentenças usa redes neurais para mapear sentenças em vetores de alta dimensão, que capturam a semântica e a estrutura da sentença. Esses *embeddings* podem ser usados para tarefas como classificação de texto, agrupamento de sentenças semelhantes e geração de resumos de texto.

Em resumo, *sentence embedding* é uma técnica útil para representar sentenças em um espaço vetorial de alta dimensão, permitindo que os modelos de aprendizado de máquina capturem a semântica e a estrutura da sentença. Existem várias abordagens para gerar *embeddings* de sentenças, desde a simples contagem de palavras até o uso de modelos pré-treinados e redes neurais de codificação de sentenças.

4.2.3.1 *Transformers*

O conceito principal dos *Transformers* é o uso de mecanismos de atenção para capturar as relações entre as palavras de uma sentença. Antes dos *Transformers*, a maioria das arquiteturas de NLP usava redes neurais recorrentes (RNNs) ou convolucionais (CNNs), que lidam com as entradas sequencialmente. No entanto, essas arquiteturas têm algumas limitações, como dificuldades para capturar dependências a longo prazo e custos computacionais elevados.

Os *Transformers* resolvem essas limitações ao utilizar um mecanismo de atenção que permite que o modelo preste mais atenção às palavras que são relevantes para a tarefa em questão. O mecanismo de atenção é aplicado em cada camada do *Transformer* e permite que o modelo pondere a importância de cada palavra em relação às outras palavras na mesma sequência.

A arquitetura *Transformer* é composta de camadas de auto-atendimento (*self-attention*) e camadas totalmente conectadas (*feedforward*)[44]. Na camada de auto-atendimento, cada palavra em uma sequência é processada em paralelo e é atribuído um peso para cada palavra em relação às outras palavras na mesma sequência. Em seguida, a saída do mecanismo de atenção é passada para uma camada totalmente conectada, que é responsável por realizar transformações lineares nos dados. [8]

4.2.3.2 *Sentence Transformers em Python*

Sentence Transformers[1] é uma biblioteca em Python[2] para aprendizado de representações de frases (também conhecidas como vetores de sentença) usando técnicas de aprendizado profundo[45]. Essa biblioteca utiliza a arquitetura *Transformer*, que é uma das técnicas mais populares para aprendizado de representações de texto, e foi pré-treinada em grandes conjuntos de dados de texto.

A ideia por trás do *Sentence Transformers* é que, ao invés de tratar as frases como simples sequências de palavras, é possível representá-las como vetores de alta dimensionalidade em um espaço semântico. Esses vetores são criados por meio de uma rede neural que recebe as frases como entrada e retorna um vetor que representa o seu significado semântico.

Uma vez que a rede foi pré-treinada em grandes conjuntos de dados de texto, os vetores de sentença gerados podem ser usados para diferentes tarefas de processamento de linguagem natural, como classificação de texto, *clustering* de frases semelhantes, recuperação de informações, entre outras.

O *Sentence Transformers* também oferece vários modelos pré-treinados que podem ser baixados e usados diretamente para gerar os vetores de sentença. Alguns desses modelos são BERT, RoBERTa, DistilBERT, entre outros, que foram pré-treinados em grandes conjuntos de dados de texto para gerar representações de frases de alta qualidade.[45]

A arquitetura do *Sentence Transformers* é baseada no *Transformer*, que é uma arquitetura de rede neural desenvolvida pela Google em 2017 para processamento de linguagem natural. Essa arquitetura tem sido muito bem-sucedida em tarefas de linguagem natural, como tradução automática, sumarização de texto, entre outras. O *Transformer* utiliza atenção multi-cabeça para permitir que a rede se concentre em diferentes partes da entrada de forma independente, o que ajuda a melhorar a qualidade da representação de texto gerada.[45]

O *Sentence Transformers* utiliza o *Transformer* para gerar vetores de sentença. Esses vetores são gerados a partir da codificação da entrada de texto em um espaço vetorial contínuo. Essa codificação é feita através de múltiplas camadas de codificadores, onde cada camada adiciona uma nova representação das informações de entrada. A saída da última camada é uma representação de alta qualidade da entrada de texto, que pode ser usada para várias tarefas de processamento de linguagem natural.[45]

Os modelos pré-treinados disponíveis no *Sentence Transformers* foram treina-

[1] https://www.sbert.net/

[2] Python é uma linguagem de programação de alto nível, interpretada, interativa e orientada a objetos. Foi criada por Guido van Rossum e lançada pela primeira vez em 1991. Python é amplamente utilizada em vários campos, como desenvolvimento web, análise de dados, aprendizado de máquina, automação de tarefas, entre outros.

dos em grandes conjuntos de dados de texto usando a técnica de aprendizado auto-supervisionado. Esse método de treinamento envolve o uso de grandes quantidades de dados de texto para aprender a representação de sentenças de alta qualidade. O modelo pré-treinado é então usado para gerar representações de sentenças para novas frases.[45]

Ao usar o *Sentence Transformers*, pode-se gerar vetores de sentença para um conjunto de frases. Esses vetores de sentença são vetores de alta dimensionalidade, onde cada dimensão representa um aspecto diferente do significado da frase. Esses vetores podem então ser usados em várias tarefas de processamento de linguagem natural, como classificação de texto, *clustering* de frases semelhantes, recuperação de informações, entre outras.[45]

5 PROCEDIMENTOS E MÉTODOS

Como resultado da limpeza dos dados, tem-se as Bases 1 e 2 prontas para a aplicação dos modelos de detecção e classificação que serão utilizados neste trabalho. Para a tarefa de classficação, serão utilizados os modelos de SVM, *Decision Tree* e *Logistic Regression*. Para a tarefa de detecção, será utilizados o SOM acompanhado de *Fuzzy C-means*.

5.1 Modelos de Classificação Supervisionados

5.1.1 Métricas de Avaliação

As métricas de avaliação de modelo são uma forma de medir a qualidade e o desempenho do modelo de aprendizado de máquina em relação aos dados utilizados para treiná-lo e testá-lo. As principais métricas de avaliação de modelo são:

- **Acurácia (*accuracy*)**: é a medida da proporção de exemplos classificados corretamente pelo modelo em relação ao número total de exemplos no conjunto de dados. A acurácia é uma métrica geral e simples, mas pode não ser apropriada em todos os casos, especialmente quando o conjunto de dados está desbalanceado em termos de classe. O cálculo da acurácia é dado pela fórmula:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

  Onde:

  - *TP (*true positive*)*: exemplos corretamente classificados como positivos (verdadeiros positivos);
  - *TN (*true negative*)*: exemplos corretamente classificados como negativos (verdadeiros negativos);
  - *FP (*false positive*)*: exemplos incorretamente classificados como positivos (falsos positivos);
  - *FN (*false negative*)*: exemplos incorretamente classificados como negativos (falsos negativos).

- **Precisão (*precision*)**: é a medida da proporção de exemplos verdadeiros positivos (TP) em relação ao número total de exemplos classificados como positivos (TP + FP). A precisão é útil quando queremos evitar falsos positivos. O cálculo da precisão é dado pela fórmula:

$$precision = \frac{TP}{TP+FP}$$

- ***Recall* (sensibilidade ou revocação)**: é a medida da proporção de exemplos verdadeiros positivos em relação ao número total de exemplos verdadeiramente positivos (TP + FN). O *recall* é útil quando queremos evitar falsos negativos. O cálculo do *recall* é dado pela fórmula:

$$recall = \frac{TP}{TP+FN}$$

- **F1-score**: é a média harmônica entre precisão e *recall*, ou seja, é uma medida balanceada entre as duas. É uma métrica comum quando queremos uma avaliação mais equilibrada do modelo em relação à precisão e ao *recall*. O cálculo do F1-score é dado pela fórmula:

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision+recall}$$

- **Suporte (*support*)**: é o número de exemplos na classe correspondente no conjunto de dados.

5.1.2 Aplicações

A partir da da Base 1, foram realizados os métodos de limpeza de dados, os quais incluem a remoção de caracteres especiais, remoção de espaços em branco, remoção de *tags HTML*, substituição de padrões de texto, substituição de links, *usernames* e *hashtags*, remoção de *stop words*, lematização e tokenização, além de outros ajustes para a base de dados. Com esse processo de limpeza de dados, houve uma redução do tamanho da cadeia de caracteres, reduzindo consideravelmente o número de termos por linha da base de dados.

Uma das etapas mais lentas do processo de execução do trabalho foi o de *embedding*. O modelo de *embedding* utilizado para o trabalho foi o modelo de *Sentence Transformers* devido ao fato de ser uns dos modelos mais atuais e com melhor desempenho atual [8]. Além do fato do modelo capturar o significado do texto em nível de sentença e geram uma representação de alta dimensão que pode ser usada para comparar e medir a similaridade semântica entre sentenças.

Dentro da biblioteca do *Sentence transformers* existem vários modelos pré-treinados para se utilizar no *encoder* dos dados. O modelo pré treinado utilizado foi "all-MiniLM-L6-v2", pela sua alta performance e velocidade entre os modelos. Ainda assim, para a Base 1, o modelo demorou 3 horas e 47 minutos para executar.

Após a execução do *embedding*, a próxima etapa foi aplicar os modelos de *Machine Learning* na saída do *embedding*. Os modelos aplicados foram os: SVM, *decision tree* e

logistic regression. Em todos os modelos foram divididos 80% da base de dados para treino e 20% para teste.

5.1.3 Resultados

Abaixo tem-se o resultado para cada modelo (Perceba que *support* não é uma métrica, mas sim um valor que referência o número de instâncias da Base.):

Tabela 4 – Resultados do modelo SVM

Classes	Precisão	Recall	F1-score	suport
age	1.00	1.00	1.00	1603
ethnicity	1.00	1.00	1.00	1603
gender	1.00	1.00	1.00	1531
religion	1.00	1.00	1.00	1566
other_cyberbullying	1.00	1.00	1.00	1612
not_cyberbullying	1.00	1.00	1.00	1624
accuracy			1.00	9539
macro avg	1.00	1.00	1.00	9539
weighted avg	1.00	1.00	1.00	9539

Fonte: Elaborado pelo o autor (2023)

A tabela 4 apresenta os resultados do modelo de SVM, que foi aplicado em um conjunto de dados para prever seis classes diferentes. Os resultados apresentados indicam que o modelo teve uma precisão, *recall* e F1-score de 1.00 (ou seja, 100%) para cada uma das seis classes diferentes. Isso significa que o modelo foi capaz de classificar corretamente todas as instâncias de cada classe. Uma alta precisão indica que o modelo é muito preciso ao prever uma determinada classe. Um alto *recall* indica que o modelo é capaz de recuperar a maioria das instâncias de uma classe.

Tabela 5 – Resultados do modelo *Decision Tree*

Classes	Precisão	Recall	F1-score	suport
age	1.00	1.00	1.00	1603
ethnicity	1.00	1.00	1.00	1603
gender	1.00	1.00	1.00	1531
religion	1.00	1.00	1.00	1566
other_cyberbullying	1.00	1.00	1.00	1612
not_cyberbullying	1.00	1.00	1.00	1624
accuracy			1.00	9539
macro avg	1.00	1.00	1.00	9539
weighted avg	1.00	1.00	1.00	9539

Fonte: Elaborado pelo o autor (2023)

Os da tabela 5 resultados indicam que o modelo de *Decision Tree* obteve uma precisão, *recall* e *F1-score* de 1.00 para todas as classes, bem como para a média ponderada

(*weighted avg*) e média macro (*macro avg*), e uma acurácia de 1.00 para o conjunto de teste.

Esses resultados indicam que o modelo foi capaz de classificar corretamente todos os exemplos do conjunto de teste para todas as classes, sem cometer erros, o que sugere que o modelo se ajustou bem aos dados de treinamento e teste.

Tabela 6 – Resultados do modelo de *Logistic Regression*

Classes	**Precisão**	**Recall**	**F1-score**	**suport**
age	0.96	0.97	0.97	1603
ethnicity	0.99	0.99	0.99	1603
gender	0.98	0.97	0.97	1531
religion	0.99	0.97	0.98	1566
other_cyberbullying	0.96	0.96	0.96	1612
not_cyberbullying	0.98	0.99	0.98	1624
accuracy			0.97	9539
macro avg	0.98	0.97	0.97	9539
weighted avg	0.97	0.97	0.97	9539

Fonte: Elaborado pelo o autor (2023)

Os resultados da tabela 6 indicam que o modelo de *Logistic Regression* obteve precisão, *recall e F1-score* superiores a 0.95 para todas as classes, exceto para a classe "age"que obteve valores um pouco inferiores, mas ainda assim bastante razoáveis.

A acurácia do modelo foi de 0.97, indicando que ele classificou corretamente a maioria dos exemplos do conjunto de teste.

Os resultados da média macro (*macro avg*) e média ponderada (*weighted avg*) estão próximos, sugerindo que o modelo tem bom desempenho em todas as classes e que a distribuição de exemplos por classe é relativamente equilibrada.

No entanto, é importante notar que a precisão e *recall* para a classe "age"foram um pouco inferiores, o que sugere que o modelo pode ter mais dificuldade em distinguir entre exemplos dessa classe.

Em um trabalho similar, de Damayanti Elisabeth e Indra Budi [1], foi aplica uma abordagem de classificação a qual também utilizava-se *Logistic Regression*, além de aplicarem outras técnicas, como *Naive Bayes e Random Forest Decision Tree*. No trabalho, obteve-se um resultado bastante positivo, demonstrado na tabela 7.

5.2 Aplicação Para o Modelo Não Supervisionado

Neste trabalho o modelo não supervisionado escolhido foi o *Self Organizing Maps* (SOM). A partir da Base 2, que é uma base de dados não rotulada. Nela, foi aplicado o

Tabela 7 – Resultados do modelo de classificação em [1]

Metric	LR				NB				RFDT			
	Hate Code (%)		*Hate Code without Abusive Code (%)*		*Hate Code (%)*		*Hate Code without Abusive Code (%)*		*Hate Code (%)*		*Hate Code without Abusive Code (%)*	
	Word	*Phrase*	*Word*	*Phrase*	*Word*	*Phrase*	*Word*	*Phrase*	*Word*	*Phrase*	*Word*	*Phrase*
Accuracy	94.14	94.06	94.31	**94.44**	93.09	92.94	93.57	93.48	93.96	93.83	94.12	94.14
Precision	94.18	94.08	94.35	**94.47**	93.10	92.95	93.58	93.49	93.98	93.86	94.14	94.17
Recall	96.78	96.37	96.78	**96.94**	94.75	94.16	95.13	94.84	96.82	96.41	96.82	96.90
F1 Score	94.63	94.46	94.77	**94.90**	93.47	93.22	93.94	93.80	94.51	94.33	94.65	94.70

Fonte: *Hate Code Detection in Indonesian Tweets using Machine Learning Approach: A Dataset and Preliminary Study, table VI* [1]

modelo a fim de agrupar tweets com certos padrões, e esperava-se que um dos padrões encontrados fossem os padrões com algum tipo de agressão verbal. Na Base 2 foram aplicados os mesmos métodos de limpeza de dados aplicados na etapa anterior (remoção de caracteres especiais, remoção de espaços em branco, remoção de *tags HTML*, substituição de padrões de texto, substituição de links, *usernames e hashtags*, remoção de *stop words*, lematização e tokenização, etc). Após a limpeza, foi aplicado o método de embedding dos dados utilizando o *Sentence Transformers* que também foi aplicado na Base 1. A partir da limpeza e pré-processamento da base de dados, esta foi ao modelo SOM. No modelo SOM, uma etapa importante é definir a dimensionalidade do mapa de saída que será retornado e a partir das pesquisas feitas por Shalaginov em [34], o número ideal de nós S no modelo é definida por:

$$S = 5 \cdot \sqrt{N}$$

Onde N é o número de linhas da base de dados. Portanto as dimensões no mapa (m, n) são definidas por:

$$m = n = \lfloor \sqrt{S} \rfloor$$

A Base 2 contém 20050 linhas, então a dimensionalidade do mapa é:

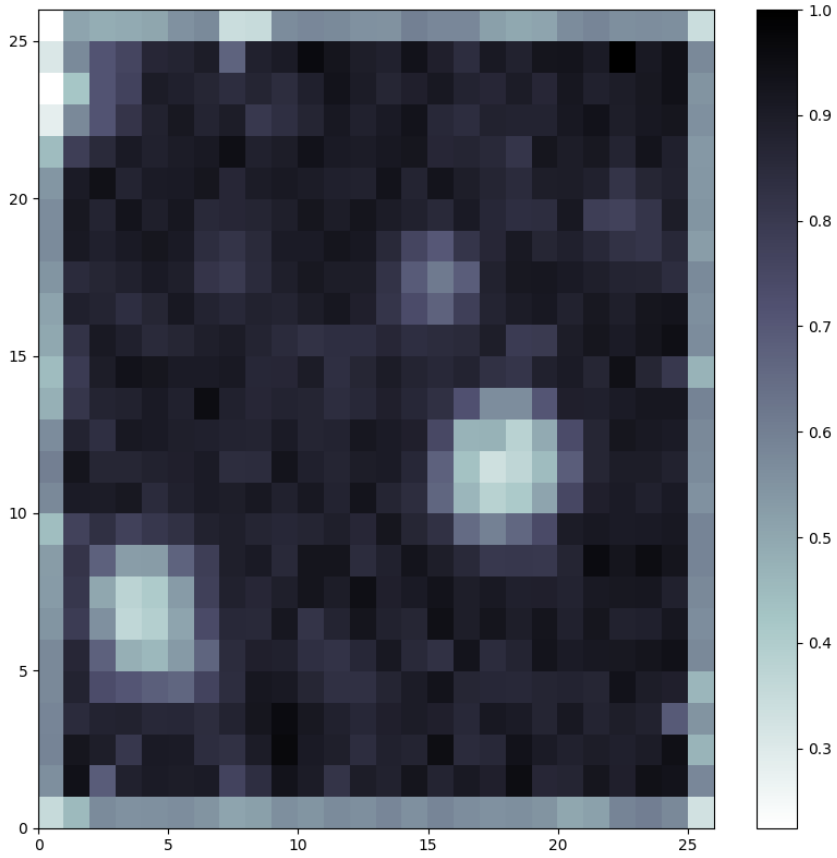
$$m = n = \lfloor \sqrt{5 \cdot \sqrt{20050}} \rfloor = 26$$

Portanto as dimensões do mapa são 26 X 26 para a Base 2.

Após definido as dimensões do mapa que o modelo retorna, outro parâmetro importante é o número de nós de entrada que o modelo receberá. Visto que após o processo de *embedding*, cada linha da Base 2, tem-se 384 colunas, então o número de nós de entrada para o modelo também será 384. A partir dessa configuração foi-se aplicado o modelo SOM na Base 2, a qual a partir deste teve-se o mapa a seguir:

Na figura 9 tem-se um mapa de calor que representam os neurônios no modelo SOM, onde cada neurônio do mapa é ligado com cada valor de entrada. Neste caso cada

Figura 9 – Mapa de distancias entre os neurônios do SOM



Fonte: Elaborado pelo o autor (2023)

neurônio é conectado com 384 valores de entrada. A cada iteração do algoritmo, os pesos de cada conexão são atualizados as distancias entre neurônios. Portanto, a partir do gráfico, as regiões mais claras possuem distâncias menores entre os neurônios o que representam padrões na Base 2.

Após a aplicação do modelo SOM, obteve-se 248 *clusters* não vazios, os quais foram submetidos a técnica de busca de padrões *frequent pattern mining* a qual não obteve resultados significativos. Teve-se como intuito que a partir da técnica de *frequent pattern mining* encontrar relações entre os termos nos *clusters*, que indicassem o uso de palavras ofensivas, resultando em uma clusterização bem sucedida. Entretanto não foi o resultado esperado. Esta metodologia expôs termos comuns da língua inglesa, como em "*found*", que se relaciona com "'*split*', '*department*', '*work*', '*different*', '*crazy*', '*monday*', '*manager*', '*team*', '*moving*'"no *cluster* 240. Foi utilizado para esta técnica a biblioteca do

Python **apriori_python** [1] e **prefixspan**[2] para a aplicação de *frequent pattern mining*.

Devido a clusterização mal sucedida aplicando o modelo SOM, foi utilizado o método visto em [26], a qual após aplicar o SOM para mapear as *features* do problema, aplicou consecutivamente o *Fuzzy C-Means*, a fim de agrupar as regiões mais próximas. Mas, ainda assim não houve correlação como se vê na tabela 8. Nela, observa-se que há uma divisão igualitária entre os elementos, indicando assim que não ocorreu uma agrupamento eficiente. No exemplo da tabela mostram apenas até 10 *clusters*, porém o numero aplicado no algoritmo variou entre 2 até n, sendo n o número de elementos de saído do SOM. Ainda assim não ocorreu uma clusterização esperada.

Tabela 8 – Resultados do modelo de *Fuzzy C-Means*

№Clusters	**Exemplo do valor de Pertinência**	**Valor de Pertinência Máximo**
2	[0.500 0.499]	0.500
3	[0.333 0.333 0.333]	0.333
4	[0.250 0.249 0.250 0.249]	0.250
5	[0.200 0.199 0.199 0.200 0.200]	0.200
6	[0.166 0.166 0.166 0.166 0.166 0.166]	0.166
7	[0.145 0.147 0.142 ... 0.145 0.142 0.147]	0.142
8	[0.125 0.125 0.124 ... 0.129 0.120 0.124]	0.125
9	[0.111 0.111 0.1111 ... 0.111 0.111 0.111]	0.111
10	[0.0999 0.0999 0.100 ... 0.100 0.099 0.100]	0.100

Fonte: Elaborado pelo autor (2023)

[1] https://github.com/chonyy/apriori_python
[2] https://github.com/chuanconggao/PrefixSpan-py

6 CONCLUSÃO

O objetivo do trabalho era identificar e classificar *cyberbulling* em comentário do twitter. Foram utilizados técnicas de limpeza e pré-processamento de dados textuais, como tokenização e *embeddings*, e aplicação de modelos de *machine learning*. Estas que neste trabalho foram divididos em técnicas supervisionadas, SVM, *decision tree* e *logistic regression*, e não supervisionadas, como o SOM.

Os resultados dos três modelos de classificação SVM, *Decision Tree* e *Logistic Regression* obtiveram 99.9%, 99.9% e 97.3%, respectivamente, de *accuracy F1-score* em média de todas as classes. Isso indica um bom desempenho na tarefa de classificação de discurso de ódio.

Em particular, os modelos SVM e *Decision Tree* obtiveram resultados muito próximos aos da *Logistic Regression*, sugerindo que diferentes modelos podem ter desempenhos semelhantes nessa tarefa.

Já as técnicas utilizadas para clusterização, *Self Organizing Maps e Fuzzy C-Means* e busca de padrões frequentes, *frequent mining patterns*, não se mostram eficientes para a identificação de *cyberbullying*.

Os resultados sugerem que os modelos supervisionados são promissores para a tarefa de classificação de *cyberbullying*, mas é importante avaliar seu desempenho em outras situações e considerar ajustes adicionais caso necessário. Ainda assim, com o resultado positivo, o trabalho traz uma metodologia eficaz para a classificação de *tweets* contendo discurso de ódio.

No trabalho, uma das limitações encontradas foi na etapa de *embedding* a qual exigia uma *hardware* potente, e ainda assim, o processamento não foi tão rápido. Também foi identificado a limitação dos modelos não supervisionados, que não tiveram a resposta esperada.

Como trabalho futuro, espera-se aplicar outras abordagens não supervisionadas, ou seja, modelos de CNN ou como em [14], RNN-LSTM (*Recurrent Neural Networks - Long Shot Term Memory*) ou RNN- BiLSTM (Recurrent Neural Networks - Bidirectional Long Shot Term Memory).

REFERÊNCIAS

[1] ELISABETH, D.; BUDI, I.; IBROHIM, M. O. Hate code detection in indonesian tweets using machine learning approach: A dataset and preliminary study. In: *2020 8th International Conference on Information and Communication Technology (ICoICT)*. [S.l.: s.n.], 2020. p. 1–6.

[2] SMITH, P. K. et al. Cyberbullying: its nature and impact in secondary school pupils. *Journal of Child Psychology and Psychiatry*, v. 49, n. 4, p. 376–385, 2008. Disponível em: <https://acamh.onlinelibrary.wiley.com/doi/abs/10.1111/j.1469-7610.2007.01846.x>.

[3] RAJ, M. et al. An application to detect cyberbullying using machine learning and deep learning techniques. *EPJ Data Science*, Springer Nature, 2022.

[4] PEARCE, J. B.; THOMPSON, A. E. Practical approaches to reduce the impact of bullying. *Archives of Disease in Childhood*, BMJ Publishing Group Ltd, v. 79, n. 6, p. 528–531, 1998. ISSN 0003-9888. Disponível em: <https://adc.bmj.com/content/79/6/528>.

[5] GREDLER, G. R. Olweus, d. (1993). bullying at school: What we know and what we can do. malden, ma: Blackwell publishing, 140 pp., $25.00. *Psychology in the Schools*, v. 40, n. 6, p. 699–700, 2003. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/pits.10114>.

[6] NETO, A. A. L. Bullying: comportamento agressivo entre estudantes. *Jornal de Pediatria*, Sociedade Brasileira de Pediatria, v. 81, n. J. Pediatr. (Rio J.), 2005 81(5) suppl, Nov 2005. ISSN 0021-7557. Disponível em: <https://doi.org/10.1590/S0021-75572005000700006>.

[7] PLAZA-DEL-ARCO, F. M. et al. A multi-task learning approach to hate speech detection leveraging sentiment analysis. *IEEE Access*, v. 9, p. 112478–112489, 2021.

[8] VASWANI, A. et al. Attention is all you need. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2017. (NIPS'17), p. 6000–6010. ISBN 9781510860964.

[9] MOURA, M. A. *O Discurso do Ódio em Redes Sociais*. São Paulo - 2016: Lura Editorial, 2023.

[10] ROTHENBURG, W. C.; STROPPA, T. Liberdade de expressÃo e discurso do Ódio: O conflito discursivo nas redes sociais. In: *2020 3° Congresso Internacional de Direito e Contemporaneidade*. [S.l.: s.n.], 2015. p. 1–15.

[11] CHENG, L. et al. Unsupervised cyberbullying detection via time-informed gaussian mixture model. In: *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. New York, NY, USA: Association for Computing Machinery, 2020. (CIKM '20), p. 185–194. Disponível em: <https://doi.org/10.1145/3340531.3411934>.

[12] BIOGLIO, L.; PENSA, R. G. Analysis and classification of privacy-sensitive content in social media posts. *EPJ Data Science*, Springer Nature, 2021.

[13] BERRIMI, M. et al. Attention-based networks for analyzing inappropriate speech in arabic text. In: *2020 4th International Symposium on Informatics and its Applications (ISIA)*. [S.l.: s.n.], 2020. p. 1–6.

[14] DEWANI, A.; MEMON, M. A.; BHATTI, S. Cyberbullying detection: advanced preprocessing techniques & deep learning architecture for roman urdu data. In: *2020 4th International Symposium on Informatics and its Applications (ISIA)*. [S.l.]: Journal of Big Data, 2020.

[15] PAWAR, R.; RAJE, R. R. Multilingual cyberbullying detection system. In: *2019 IEEE International Conference on Electro Information Technology (EIT)*. [S.l.: s.n.], 2019. p. 040–044.

[16] MOUHEB, D. et al. Real-time detection of cyberbullying in arabic twitter streams. In: *2019 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*. [S.l.: s.n.], 2019. p. 1–5.

[17] HANG, O. C.; DAHLAN, H. M. Cyberbullying lexicon for social media. In: *2019 6th International Conference on Research and Innovation in Information Systems (ICRIIS)*. [S.l.: s.n.], 2019. p. 1–6.

[18] ALI, W. N. H. W.; MOHD, M.; FAUZI, F. Cyberbullying detection: An overview. In: *2018 Cyber Resilience Conference (CRC)*. [S.l.: s.n.], 2018. p. 1–3.

[19] TAPIA, F.; AGUINAGA, C.; LUJE, R. Detection of behavior patterns through social networks like twitter, using data mining techniques as a method to detect cyberbullying. In: *2018 7th International Conference On Software Process Improvement (CIMPS)*. [S.l.: s.n.], 2018. p. 111–118.

[20] NOVIANTHO; ISA, S. M.; ASHIANTI, L. Cyberbullying classification using text mining. In: *2017 1st International Conference on Informatics and Computational Sciences (ICICoS)*. [S.l.: s.n.], 2017. p. 241–246.

[21] PAUL, S.; SAHA, S.; SINGH, J. P. Covid-19 and cyberbullying: deep ensemble model to identify cyberbullying from code-switched languages during the pandemic. *Springer Science*, Springer Nature, 2021.

[22] CHENG, L. et al. Hierarchical attention networks for cyberbullying detection on the instagram social network. In: _____. *Proceedings of the 2019 SIAM International Conference on Data Mining (SDM)*. Society for Industrial and Applied Mathematics, 2019. p. 235–243. Disponível em: <https://epubs.siam.org/doi/abs/10.1137/1.9781611975673.27>.

[23] CHENG, L. et al. Modeling temporal patterns of cyberbullying detection with hierarchical attention networks. *Society for Industrial and Applied Mathematics*, Association for Computing Machinery, New York, NY, USA, v. 2, n. 2, 2021. Disponível em: <https://doi.org/10.1145/3441141>.

[24] PAUL, S.; SAHA, S. Cyberbert: Bert for cyberbullying identification. *Springer Nature*, Springer International Publishing, 2020.

[25] ZHONG, H.; MILLER, D. J.; SQUICCIARINI, A. Flexible inference for cyberbully incident detection. In: BREFELD, U. et al. (Ed.). *Machine Learning and Knowledge Discovery in Databases*. Cham: Springer International Publishing, 2019. p. 356–371.

[26] SENEFONTE, H. C. d. M. Predicting mobility patterns based on profiles of social media users: tourists case study. *Universidade Tecnológica Federal do Paraná*, 2022.

[27] MITCHELL, T. M. *Machine Learning*. [S.l.]: McGraw-Hill Science/Engineering/Math, 1997. v. 1.

[28] GéRON, A. *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow*. [S.l.]: McGraw-Hill Science/Engineering/Math, 2019. v. 3.

[29] KHADKA, R. *Machine Learning Types n2 - Towards Data Science*. Towards Data Science, 2017. Disponível em: <https://towardsdatascience.com/machine-learning-types-2-c1291d4f04b1>.

[30] RUSSEL, S.; NORVIG, P. *Artificail Intelligence, A modern Approach*. [S.l.]: Pearson, 2020. v. 2.

[31] YU, S. et al. The ocs-svm: An objective-cost-sensitive svm with sample-based misclassification cost invariance. *IEEE Access*, v. 7, p. 118931–118942, 2019.

[32] HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. *The elements of statistical learning: Data mining, inference, and prediction*. 2. ed. [S.l.]: Springer, 2009.

[33] KOHONEN, T. Self-organized formation of topologically correct feature maps. *Biological cybernetics*, Springer, v. 43, n. 1, p. 59–69, 1982.

[34] SHALAGINOV, A.; FRANKE, K. A new method for an optimal som size determination in neuro-fuzzy for the digital forensics applications. In: ROJAS, I.; JOYA, G.; CATALA, A. (Ed.). *Advances in Computational Intelligence*. Cham: Springer International Publishing, 2015. p. 549–563. ISBN 978-3-319-19222-2.

[35] Self-organizing mapsself-organizing maps. In: KIRCH, W. (Ed.). *Encyclopedia of Public Health*. Dordrecht: Springer Netherlands, 2008. p. 1290–1290. ISBN 978-1-4020-5614-7. Disponível em: <https://doi.org/10.1007/978-1-4020-5614-7_3141>.

[36] CHI, D. Research on the application of k-means clustering algorithm in student achievement. In: *2021 IEEE International Conference on Consumer Electronics and Computer Engineering (ICCECE)*. [S.l.: s.n.], 2021. p. 435–438.

[37] WANG, W. et al. The global fuzzy c-means clustering algorithm. In: *2006 6th World Congress on Intelligent Control and Automation*. [S.l.: s.n.], 2006. v. 1, p. 3604–3607.

[38] INDURKHYA, N.; DAMERAU, F. J. *Handbook of Natural Language Processing*. [S.l.]: Chapman and Hall/CRC, 2010.

[39] KLEENE, S. C. et al. Representation of events in nerve nets and finite automata. *Automata studies*, v. 34, p. 3–41, 1956.

[40] HARRIS, Z. S. Distributional structure. *WORD*, v. 10, n. 2-3, p. 146–162, 1954.

[41] BOJANOWSKI, P. et al. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, v. 5, p. 135–146, 2017.

[42] DEVLIN, J. et al. Bert: Pre-training of deep bidirectional transformers for language understanding. In: *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*. [S.l.: s.n.], 2018.

[43] RADFORD, A. et al. Language models are unsupervised multitask learners. *OpenAI Blog*, 2019.

[44] CONNEAU, A. et al. Infersent: A sentence encoder by translation. In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. [S.l.: s.n.], 2017.

[45] REIMERS, N.; GUREVYCH, I. Sentence-bert: Sentence embeddings using siamese bert-networks. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2019. Disponível em: <https://arxiv.org/abs/1908.10084>.