UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Igor Melo Mesquita

Estudo de Caso do Uso de um Sistema de Gerenciamento de Banco de Dados para Detecção de Cyberbullying

Uberlândia, Brasil

2026

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Igor Melo Mesquita

Estudo de Caso do Uso de um Sistema de Gerenciamento de Banco de Dados para Detecção de Cyberbullying

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção título de Bacharel em Sistemas de Informação.

Orientador: Profa. Dra. Maria Camila Nardini Barioni

Universidade Federal de Uberlândia – UFU

Faculdade de Computação

Bacharelado em Sistemas de Informação

Uberlândia, Brasil

2026

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
Faculdade de Computação
Av. João Naves de Ávila, 2121, Bloco 1B - Bairro Santa Mônica, Uberlândia-MG, CEP 38400-902
Telefone: (34) 3239-4393 - www.facom.ufu.br - facom@ufu.br

ATA DE DEFESA - GRADUAÇÃO

Curso de Graduação em:	Bacharelado em Sistemas de Informação				
Defesa de:	FACOM31802 - Trabalho de Conclusão de Curso 2				
Data:	28/07/2026	Hora de início:	10:30	Hora de encerramento:	11:39
Matrícula do Discente:	12221BSI206				
Nome do Discente:	Igor Melo Mesquita				
Título do Trabalho:	Estudo de caso do uso de um sistema de gerenciamento de banco de dados para detecção de cyberbullying				

A carga horária curricular foi cumprida integralmente?	(X) Sim ( ) Não

Reuniu-se na Sala 1B126, Campus Santa Mônica, da Universidade Federal de Uberlândia, a Banca Examinadora, designada pelo Colegiado do Curso de Graduação em Bacharelado em Sistemas de Informação, assim composta: Professores: Elaine Ribeiro de Faria Paiva - FACOM/UFU; Marcelo Zanchetta do Nascimento - FACOM/UFU; Maria Camila Nardini Barioni - FACOM/UFU, orientadora do candidato.

Iniciando os trabalhos, a presidente da mesa, Dr(a). Maria Camila Nardini Barioni, apresentou a Comissão Examinadora e o candidato, agradeceu a presença do público, e concedeu ao discente a palavra, para a exposição do seu trabalho. A duração da apresentação do discente e o tempo de arguição e resposta foram conforme as normas do curso.

A seguir a senhora presidente concedeu a palavra, pela ordem sucessivamente, aos examinadores, que passaram a arguir o candidato. Ultimada a arguição, que se desenvolveu dentro dos termos regimentais, a Banca, em sessão secreta, atribuiu o resultado final, considerando o candidato:

(x) Aprovado Nota [100] (Somente números inteiros)

OU

( ) Aprovado sem nota.

Nada mais havendo a tratar foram encerrados os trabalhos. Foi lavrada a presente ata que após lida e achada conforme foi assinada pela Banca Examinadora.

Documento assinado eletronicamente por **Maria Camila Nardini Barioni**, **Professor(a) do Magistério Superior**, em 28/07/2026, às 11:39, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Elaine Ribeiro de Faria Paiva**, **Professor(a) do Magistério Superior**, em 28/07/2026, às 11:39, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Marcelo Zanchetta do Nascimento**, **Professor(a) do Magistério Superior**, em 28/07/2026, às 11:39, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://www.sei.ufu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **7499949** e o código CRC **51ACC67D**.

Referência: Processo nº 23117.048398/2026-09 SEI nº 7499949

A mim, e a tudo que esta conquista representa.

Agradecimentos

Agradeço, primeiramente, à minha mãe Rivane: mulher guerreira, resiliente e perseverante. Ela esteve presente em cada etapa da minha vida e, mesmo diante das dificuldades, nunca me deixou faltar nada, abrindo mão de si mesma inúmeras vezes para que eu e meu irmão tivéssemos um futuro brilhante. Se hoje eu chego até aqui, é porque ela nunca parou de lutar — mesmo quando lutar parecia impossível.

Ao meu irmão Caio, agradeço por ter sido, sempre, meu porto seguro. Passamos juntos por inúmeras mudanças: de escola, de casa, de cidade, de rotina. E, no meio de tanta instabilidade, ele foi a única constante que eu sempre pude segurar. Hoje seguimos lado a lado, e sei que, aconteça o que acontecer, sempre teremos um ao outro.

Ao meu pai Prêntice Júnior e à minha irmã Eliza, que, mesmo distantes, despertam em mim a vontade de um futuro melhor.

Aos meus avós maternos — José Antônio e Maria *(in memoriam)* — e paternos — Prêntice e Helena, e a toda minha família, agradeço por todo o carinho e suporte recebidos ao longo da minha vida.

Ao meu padrasto Wagner, agradeço pela relação de respeito que construímos ao longo dos anos. Ele apoiou minha mãe de maneira contínua, e esse cuidado com ela se reflete também em mim.

Aos meus amigos de longa data — Hanny, Lucas, Amanda e Guilherme, e aos meus amigos que fiz na faculdade, em especial Maria Eduarda, Jaiany e Giovanna, agradeço por tornarem esses anos de graduação mais leves e por dividirem comigo tanto as dificuldades quanto as boas lembranças que levo do curso.

Aos meus professores da Faculdade de Computação, agradeço pelo conhecimento compartilhado ao longo da graduação. Em especial, agradeço aos professores dr. Marcelo Zanchetta do Nascimento e dra. Elaine Ribeiro de Faria Paiva, pelas aulas estimulantes de Ciência de Dados, que instigaram a minha vontade de conhecimento em uma das áreas desta pesquisa.

À minha antiga professora do Instituto Federal do Triângulo Mineiro, dra. Crícia Zilda Felício Paixão, agradeço por sempre ter me instruído com zelo e incentivado a minha trajetória na área de tecnologia — principalmente em banco de dados — durante o meu ensino médio técnico.

À minha orientadora, profa. dra. Maria Camila, agradeço pela orientação dedicada durante todo o desenvolvimento deste trabalho.

A mim, agradeço por persistir, mesmo nos momentos mais difíceis.

"Aquilo que não me mata, só me fortalece."
— Friedrich Nietzsche

Resumo

A facilidade de acesso às redes sociais intensificou episódios de *cyberbullying* entre os usuários, prática que afeta negativamente a vida das vítimas. Diante dessa problemática, este trabalho tem como objetivo avaliar a aplicabilidade do PGVector (uma extensão para criar SGBDVs) em processos de detecção de mensagens potencialmente associadas ao *cyberbullying*, unindo busca por similaridade semântica a um modelo supervisionado de aprendizado de máquina. Com esse propósito, dados da plataforma X (antigo *Twitter*) foram coletados e submetidos a seis etapas: pré-processamento textual, representação vetorial, amostragem, armazenamento no PostgreSQL com a extensão PGVector, classificação e avaliação dos resultados. Na etapa de classificação, foram comparados três modelos de geração de *embeddings* — SBERT, BERT e *FastText* —, cada um combinado ao algoritmo kNN executado diretamente no SGBDV. Os resultados demonstraram que o SBERT apresentou o melhor desempenho ($k = 9$, *f1-score* de 0,77), superando o BERT ($k = 7$, *f1-score* de 0,65) e o *FastText* ($k = 5$, *f1-score* de 0,59), confirmando que *embeddings* otimizados para comparação por similaridade são mais adequados a esse tipo de tarefa. Conclui-se que o PostgreSQL, por meio do PGVector, é capaz de atuar como um ambiente completo de classificação por similaridade, validando sua aplicabilidade para a detecção de *cyberbullying*.

Palavras-chave: *cyberbullying*, PGVector, *embeddings*, similaridade semântica, aprendizado de máquina supervisionado.

Abstract

The ease of access to social media has intensified episodes of cyberbullying among users, a practice that negatively affects victims' lives. Given this problem, this work aims to evaluate the applicability of PGVector (an extension for creating VDBMSs) in processes for detecting messages potentially associated with cyberbullying, combining semantic similarity search with a supervised machine learning model. To this end, data from the X platform (formerly Twitter) were collected and submitted to a method structured in six steps: textual preprocessing, vector representation, sampling, storage in PostgreSQL with the PGVector extension, classification, and evaluation of results. In the classification step, three embedding generation models were compared — SBERT, BERT, and FastText —, each combined with the kNN algorithm executed directly in the VDBMS. The results showed that SBERT achieved the best performance ($k = 9$, *f1-score* of 0.77), outperforming BERT ($k = 7$, *f1-score* of 0.65) and FastText ($k = 5$, *f1-score* of 0.59), confirming that embeddings optimized for similarity comparison are more suitable for this type of task. It is concluded that PostgreSQL, through PGVector, is capable of acting as a complete similarity-based classification environment, validating its applicability for cyberbullying detection.

Keywords: cyberbullying, PGVector, embeddings, semantic similarity, supervised machine learning.

Declaração sobre o uso de inteligência artificial

Eu, **IGOR MELO MESQUITA**, discente do curso de Bacharelado **SISTEMAS DE INFORMAÇÃO** da Faculdade de Computação da Universidade Federal de Uberlândia, regularmente matriculado sob o número **12221BSI206**, declaro para os devidos fins que as informações prestadas a seguir refletem de forma verídica e completa o uso ou não uso de ferramentas de Inteligência Artificial neste trabalho, em atenção às recomendações para o uso e desenvolvimento ético e responsável de inteligência artificial na Universidade Federal de Uberlândia, versão 7 de 2025.

SELECIONE UMA DAS OPÇÕES A SEGUIR:

☐ **NÃO HOUVE USO DE FERRAMENTAS DE INTELIGÊNCIA ARTIFICIAL**

(Se marcar esta opção, não é necessário preencher os campos 1 a 4)

Declaro que não utilizei, em nenhuma etapa do desenvolvimento do presente trabalho, ferramentas de Inteligência Artificial para qualquer finalidade e assumo integral responsabilidade pelo conteúdo.

☒ **HOUVE USO DE FERRAMENTAS DE INTELIGÊNCIA ARTIFICIAL**

(Preencher os campos obrigatórios abaixo, assinalando obrigatoriamente os itens 3 e 4)

Declaro que utilizei, durante o desenvolvimento do presente trabalho, as seguintes ferramentas de Inteligência Artificial e informações descritas nos itens de 1 a 4.

1. Ferramenta(s) e versão(ões) predominante(s):

Claude Sonnet 5, Consensus, Gemini 3.5 Flash e GPT-5.3.

2. Propósito(s) (marque todos os que se aplicam):

☐ Exploração inicial de ideias.

☒ Busca de referências bibliográficas.

☒ Revisão de linguagem (por exemplo, melhoria de gramática ou correção de ortografia).

☒ Tradução técnica (conversão do texto ou partes dele para outro idioma).

☐ Geração automatizada de tabelas, gráficos ou figuras a partir de dados previamente analisados pelos autores.

☒ Programação: sugestão, depuração e documentação de códigos computacionais.

☐ Outros (especificar): _____

3. Autoria e validação humana:

☒ Informo que a autoria intelectual do manuscrito, bem como sua supervisão e validação, são de minha responsabilidade e que o uso de ferramentas de IA foi exclusivamente para os propósitos acima declarados.

4. Princípios éticos:

☒ Declaro que a utilização das ferramentas de IA seguiu práticas éticas, não incluindo atividades que comprometam a integridade científica tais como a geração de conteúdo original, manipulação de dados, plágio, interpretações ou análises pela IA.

☒ Declaro que respeitei direitos autorais, licenças, confidencialidade e políticas editoriais.

☒ Declaro que não enviei dados inéditos ou sensíveis à serviços que utilizam conteúdos para treinamento de modelos, exceto em plataformas institucionais ou com garantias contratuais de confidencialidade e não retenção, assegurando conformidade com a LGPD (Lei nº 13.709/2018) e demais normas de proteção de dados.

Uberlândia, 23 de Agosto de 2026

gov.br Documento assinado digitalmente
IGOR MELO MESQUITA
Data: 23/08/2026 14:07:27-0300
Verifique em https://validar.iti.gov.br

Igor Melo Mesquita

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ANN	*Approximate Nearest Neighbors*
BERT	*Bidirectional Encoder Representations from Transformers*
CNN	Rede Neural Convolucional
DER	Diagrama de Entidade e Relacionamento
FN	Falso Negativo
FP	Falso Positivo
HNSW	*Hierarchical Navigable Small World*
HTML	*HyperText Markup Language*
IA	Inteligência Artificial
kNN	K-vizinhos mais próximos
L1	Distância de Manhattan
L2	Distância euclidiana
LSTM	*Long Short-Term Memory*
PLN	Processamento de Linguagem Natural
RoBERTa	*Robustly Optimized BERT Approach*
SBERT	*Sentence-BERT*
SGBD	Sistema de Gerenciamento de Banco de Dados
SGBDV	Sistema de Gerenciamento de Banco de Dados Vetorial
SQL	*Structured Query Language*
SVM	Máquina de Vetores de Suporte
UMAP	*Uniform Manifold Approximation and Projection*
URL	*Uniform Resource Locator*
USE	*Universal Sentence Embeddings*
VN	Verdadeiro Negativo
VP	Verdadeiro Positivo

Sumário

1 Introdução

Com o aprimoramento da tecnologia e a expansão das redes sociais online, a interação entre humanos ficou cada vez maior, uma vez que não é mais necessário que ambas as partes estejam presentes no mesmo ambiente (SUBRAMANIAN, 2017). Junto a isso, surge a problemática do *cyberbullying*, que muitos pesquisadores definem como o uso de tecnologias eletrônicas de comunicação para fazer *bullying* com os outros — ainda que sua definição envolva particularidades, devido à diversidade de contextos e formas em que pode ocorrer (KOWALSKI et al., 2014).

O *cyberbullying* é um tema de extrema relevância e que deve ser combatido. Em um estudo realizado com 351 estudantes, cerca de 56,1% afirmaram já ter sofrido *cyberbullying*, experienciando sentimentos de raiva, impotência, medo e tristeza (HOFF; MITCHELL, 2009). Em outro estudo envolvendo 16.292 pessoas, cerca de 49,57% disseram sofrer de depressão e 9,84% relataram já ter pensado em cometer suicídio por *cyberbullying* (MAURYA et al., 2022).

A detecção de *cyberbullying* pode ocorrer de maneiras distintas, mas metodologicamente relacionadas. Alguns autores propuseram modelos para classificar mensagens entre *bullying* ou não *bullying*, como o de *autoencoder* semântico (ZHAO; MAO, 2017) e outro de *fine-tuning* de *embeddings Bidirectional Encoder Representations from Transformers* (BERT) (GUTIÉRREZ-BATISTA; GÓMEZ-SÁNCHEZ; FERNANDEZ-BASSO, 2024). Embora esses estudos foquem na classificação e mostrem bons resultados, com o crescente uso de *embeddings* — que são uma representação de dados não estruturados, como textos e imagens (MIKOLOV et al., 2013a), torna-se essencial buscar soluções eficientes de armazenamento e recuperação dos dados (LEWIS et al., 2020).

Nesse sentido, trabalhos sobre a recente ferramenta PGVector (WARD, 2025; WANG et al., 2025; RIBEIRO et al., 2025) exploram o uso de bancos de dados vetoriais para armazenamento de *embeddings* e para buscas eficientes por similaridade utilizando algoritmos da família *Approximate Nearest Neighbors* (ANN), aplicando métricas como similaridade de cosseno, produto interno e distância euclidiana.

O objetivo geral deste trabalho foi avaliar a aplicabilidade do PGVector — que é uma extensão do Sistema de Gerenciamento de Banco de Dados (SGBD) PostgreSQL — em processos de detecção de mensagens potencialmente associadas ao *cyberbullying* por meio de busca por similaridade semântica em *embeddings*. Como objetivos específicos, teve-se: (i) demonstrar o uso de *embeddings* em Sistemas de Gerenciamento de Banco de Dados Vetoriais (SGBDVs); (ii) implementar o algoritmo de classificação kNN diretamente no SGBDV, por meio de funções nativas do PostgreSQL; e (iii) analisar resultados obtidos

no armazenamento e recuperação das representações vetoriais.

Este trabalho utilizou ferramentas computacionais para o pré-processamento textual, a geração de *embeddings*, o armazenamento vetorial e a busca por similaridade semântica, além de métricas de avaliação comumente empregadas na literatura da área. Dessa forma, foram utilizados: (i) linguagem de programação Python com bibliotecas específicas; (ii) SGBD PostgreSQL com extensão PGVector; e (iii) conjunto de dados público contendo amostras textuais associadas ao *cyberbullying*.

Posto isso, este trabalho se caracteriza como científico e, também, tecnológico, já que envolve a investigação de conceitos de Processamento de Linguagem Natural (PLN) e recuperação da informação, além da aplicação prática de ferramentas computacionais para análise semântica e busca vetorial no contexto de detecção de *cyberbullying*. Trata-se, portanto, de um estudo de caso de caráter exploratório, uma vez que não busca elaborar uma teoria nem apresentar resultados estatisticamente generalizáveis, mas sim explorar um cenário específico (WAZLAWICK, 2020).

Este trabalho está ordenado em seis capítulos, contando com esta introdução. No Capítulo 2, discute-se os fundamentos teóricos necessários para a compreensão da pesquisa, contemplando cyberbullying, conceitos de PLN e embeddings, e a extensão PGVector. Já o Capítulo 3 traz um panorama dos trabalhos correlatos, situando esta pesquisa diante do que já foi produzido na área. O Capítulo 4 descreve, etapa por etapa, o método adotado neste estudo de caso, enquanto os resultados obtidos são discutidos no Capítulo 5. Encerrando o trabalho, o Capítulo 6 traz as considerações finais, as limitações encontradas ao longo da pesquisa e possíveis direções para trabalhos futuros.

2 Fundamentação Teórica

Neste capítulo, são apresentados os conceitos teóricos fundamentais para o desenvolvimento deste trabalho. A princípio, introduz-se a problemática do *cyberbullying*, motivação prática desta pesquisa. Posteriormente, abordam-se fundamentos sobre dados estruturados e não estruturados, além de técnicas de PLN para tratamento de textos, incluindo pré-processamento e representação vetorial. Em seguida, discute-se o armazenamento e as técnicas de recuperação desses dados, destacando o uso do PostgreSQL com a extensão PGVector. Na sequência, expõem-se técnicas de aprendizado de máquina para a classificação dos dados. Por fim, apresentam-se técnicas para avaliação e visualização dos resultados obtidos pelo algoritmo.

2.1 *Cyberbullying*

Nesta seção, é melhor apresentada a problemática do *cyberbullying*. Muitos pesquisadores o definem como o uso de tecnologias eletrônicas de comunicação para fazer *bullying* com os outros — ainda que sua definição envolva particularidades, devido à diversidade de contextos e formas em que pode ocorrer (KOWALSKI et al., 2014). De acordo com Groover e Raju (2023), o *cyberbullying* vem se tornando um grande problema na vida de muitas pessoas, principalmente adolescentes, resultando em comportamentos imprevisíveis e efeitos negativos em suas vidas, uma vez que possui um peso psicológico prejudicial significativo nas vítimas.

O Núcleo de Informação e Coordenação do Ponto BR (2025) introduz o termo "*cyberbullying*" como conteúdos de risco de natureza agressiva para crianças e adolescentes em ambientes digitais. A pesquisa evidencia, ainda, como essa temática afeta a vida dessas pessoas, de modo a violar as suas privacidades, criar riscos para a saúde física e mental (como sedentarismo, estilo de vida, isolamento e ansiedade) e disseminar desigualdades e discriminação entre os usuários.

2.2 Dados Estruturados *vs* Dados Não Estruturados

Os dados estruturados são aqueles organizados em esquemas fixos em um SGBD, possuindo atributos bem definidos, o que possibilita a fácil recuperação destes por meio de consultas estruturadas em *Structured Query Language* (SQL). De forma oposta, os dados não estruturados são aqueles que não possuem um esquema fixo, como imagens, documentos e áudios. Devido a essa natureza, torna-se inadequado utilizar consultas estruturadas, criando-se a necessidade de converter esses dados em vetores por meio de

modelos de *embedding* para a sua recuperação, que se dá por meio de busca por similaridade (PAN; WANG; LI, 2023). Este trabalho se baseia em uma solução para a complicada manipulação de dados não estruturados no contexto de detecção de *cyberbullying*.

2.3 Processamento de Linguagem Natural

O PLN é o estudo da interpretação e geração automática da linguagem natural. Isto é, um texto é tomado como objeto de análise que, através de estágios dentro do PLN, um significado intencional é descoberto (MARTINS; LENZ; SILVA, 2020). No contexto deste trabalho, o PLN foi utilizado para o pré-processamento em conjuntos de textos curtos de plataformas de comunicação, como redes sociais online, a fim de classificá-los posteriormente.

O pré-processamento do objeto de análise ou, em suma, de um texto, baseia-se na limpeza e preparação deste texto para a sua classificação. É uma etapa primordial do PLN, em razão de que os caracteres, palavras e frases identificadas nessa fase são responsáveis por todo o processamento e entendimento final da máquina (JURAFSKY; MARTIN, 2009). A seguir, são detalhadas as técnicas de pré-processamento utilizadas em PLN no contexto da presente pesquisa.

2.3.1 Limpeza do Texto

De acordo com Martins, Lenz e Silva (2020), textos *online* costumam conter muita "sujeira" e partes pouco informativas, como caracteres especiais, *tags HyperText Markup Language* (HTML), *scripts* e anúncios. Para além, algumas palavras não impactam de forma significativa a semântica do texto; e mantê-las dificulta a classificação deste. Dessa forma, a limpeza do texto consiste em retirar essa "sujeira", visando uma melhor predição no classificador.

2.3.2 Normalização do Formato das Palavras

A normalização do formato das palavras refere-se à tarefa de colocar palavras em formas padrões (por exemplo: "tudooo" ou "tuudoo" em "tudo") (JURAFSKY; MARTIN, 2009). Martins, Lenz e Silva (2020) afirmam que esse processo é muito importante em casos de textos com muitas gírias e abreviações, situação cotidiana em redes sociais.

2.3.3 Lematização

A técnica de lematização é utilizada visando a redução do vocabulário de um texto, abstraindo o significado das palavras de forma mais fácil (MARTINS; LENZ; SILVA,

2020). O processo ocorre transformando a palavra em sua origem básica, à forma masculina e singular. Por exemplo, as palavras “gato”, “gatas”, “gata” e “gatos” seriam limitadas ao lema “gato”. No caso de verbos, o lema ficaria no infinitivo, ou seja, verbos como “responde” e “respondendo” são formas do mesmo lema “responder”.

2.4 Representação Semântica dos Dados e *Embeddings*

Martins, Lenz e Silva (2020) definem os computadores como máquinas que trabalham com bases numéricas, por isso, eles tendem a converter a informação que os humanos veem para uma informação que a máquina seja capaz de compreender, ou seja, números. Dessa forma, para que seja possível trabalhar com aprendizado de máquina, é preciso que os textos sejam transformados em números, mais especificamente, em representações vetoriais (BENGFORT; BILBRO; OJEDA, 2018). Existem diversos algoritmos para que isso seja possível, um conjunto deles é o *word embedding*.

O contexto semântico das palavras é essencial para determinar o significado de um texto, sem que hajam ambiguidades. Para isso, foi proposta a técnica de *word embeddings*, que cria vetores semânticos — chamados *embeddings* — de múltiplas dimensionalidades e permite a manipulação matemática desses dados (JURAFSKY; MARTIN, 2009). Esses vetores criam relações aritméticas entre si, o que permite fazer operações entre palavras, uma vez que palavras similares tendem a apresentar atributos similares e, portanto, a se posicionar próximas no espaço de atributos (GOLDBERG; LEVY, 2014). Mikolov et al. (2013b) realizam uma analogia como exemplo dessas operações em que o vetor(“Rei”) - vetor(“Homem”) + vetor(“Mulher”) resultava em, aproximadamente, vetor(“Rainha”).

Existem vários modelos que fazem a geração de *embeddings*, cada um contrastando entre si. Wang et al. (2019) trazem em seu trabalho uma comparação de modelos clássicos, como o Word2Vec, GloVe e FastText. O modelo Word2Vec gera os *embeddings* ajustando os vetores durante o treinamento de modo que palavras que tenham semântica semelhante, fiquem próximas no espaço vetorial. O GloVe por sua vez, tenta ir além, captura não somente relações semânticas como sintáticas também, fazendo isso através de uma matriz de co-ocorrência entre palavras. De modo ainda mais aprimorado, o FastText retrata as palavras através de vetores de subpalavras, permitindo representações para palavras raras ou desconhecidas. Apesar das evoluções, todos esses modelos carecem em um ponto muito importante: as palavras não são representadas pelo contexto. Ou seja, palavras com mais de um significado, como “banco”, terão sempre o mesmo vetor, seja na frase “ele foi ao banco” ou “ele sentou no banco”.

Diante dessa problemática, Devlin et al. (2019) propuseram o BERT, um modelo que, diferentemente dos clássicos, é capaz de gerar vetores com representações dinâmicas, levando em conta não apenas a semântica, mas também o contexto do texto. O BERT

possui a arquitetura *Transformer*, que usa um mecanismo de atenção para calcular o quanto cada palavra deve se atentar nas demais palavras de um mesmo texto. Assim como os modelos clássicos, o BERT também possui uma limitação considerável: a sua construção o torna inadequado para busca de similaridade semântica (REIMERS; GUREVYCH, 2019).

Reimers e Gurevych (2019) criaram uma adaptação do BERT, o *Sentence-BERT* (SBERT) para resolver a incapacidade computacional citada. A ideia dos autores foi utilizar uma arquitetura siamesa, em que cada sentença é processada de forma independente. Com isso, os *embeddings* podem ser armazenados e comparados diretamente com similaridade do cosseno, sem a necessidade de reprocessamento conjunto, anteriormente exigida pelo BERT. Além disso, com essa nova técnica, o SBERT permite que sentenças sejam comparadas em larga escala de forma eficiente, indo de ∼65 horas com BERT para cerca de 5 segundos com SBERT (REIMERS; GUREVYCH, 2019).

2.5 Armazenamento e Recuperação de Dados Vetoriais

Dados não estruturados exigem um armazenamento diferente do comum, uma vez que são representados por vetores e não por atributos de um esquema fixo, como os dados estruturados. Dessa forma, introduz-se o SGBDV, capaz de fornecer recursos de um SGBD tradicional, como otimização de recursos, transações, escalabilidade e tolerância a falhas, porém para dados vetoriais, ou seja, dados não estruturados (PAN; WANG; LI, 2023).

A partir dos dados armazenados em um SGBDV, é possível realizar a recuperação destes para o processamento de consultas que buscam o *score* de similaridade entre dois vetores. O *score* de similaridade é calculado através de métricas de distância e são utilizados para verificar o quão semelhantes dois vetores são, sendo que valores próximos de 0 indicam grande similaridade. Existem diversas métricas de distância que podem ser utilizadas no cálculo da similaridade, dentre as principais, tem-se: Hamming, produto interno, similaridade do cosseno e Minkowski (PAN; WANG; LI, 2023).

- **Distância de Hamming**

$$d(\mathbf{a}, \mathbf{b}) = \sum_{i=1}^{n} \delta(a_i, b_i) \tag{2.1}$$

onde a e b são vetores e δ é o delta de Kronecker.

- **Produto Interno**

$$f(\mathbf{a}, \mathbf{b}) = \sum_{i=1}^{n} a_i b_i \tag{2.2}$$

onde a e b são vetores.

- **Similaridade do Cosseno**

$$f(\mathbf{a}, \mathbf{b}) = \frac{\langle \mathbf{a} \cdot \mathbf{b} \rangle}{\|\mathbf{a}\|\|\mathbf{b}\|} \tag{2.3}$$

onde a e b são vetores.

- **Distância de Minkowski**

$$d(\mathbf{a}, \mathbf{b}) = \left(\sum_{i=1}^{n} |a_i - b_i|^p \right)^{\frac{1}{p}} \tag{2.4}$$

onde a e b são vetores e p o parâmetro que define a métrica a ser utilizada no cálculo.

Se $p = 1$, calcula-se a distância de Manhattan,

Se $p = 2$, calcula-se a distância euclidiana.

2.5.1 PostgreSQL e PGVector

O PostgreSQL[1] é um SGBD que, ao adicionar a extensão PGVector, funciona como um SGBDV e, portanto, é capaz de trabalhar com dados não estruturados. Esse SGBDV funciona criando um esquema com tabelas que podem possuir colunas como *id* e *texto* e, adicionalmente, uma coluna do tipo *vector*() que armazena os *embeddings* dos textos. A partir desses *embeddings*, é possível calcular a similaridade dos vetores através das métricas de distância apresentadas por Pan, Wang e Li (2023). A extensão PGVector do PostgreSQL implementa as métricas de distância em operadores da seguinte maneira:

- <~> Distância de Hamming;
- <#> Produto Interno;
- <=> Similaridade do Cosseno;
- <+> Distância de Manhattan (L1);
- <-> Distância euclidiana (L2).

Esses operadores devem ser implementados em consultas SQL como mostrado na listagem 2.1.

Listagem 2.1 – Retornar os vizinhos mais próximos a um vetor usando L2

```
1  SELECT * FROM items ORDER BY embedding <-> '[3,1,2]' LIMIT 5;
```

[1] <https://github.com/pgvector/pgvector>

Fonte: Kane (2021).

Nesse exemplo, o vetor `[3,1,2]` representa o vetor de consulta, ou seja, o vetor que se deseja descobrir os vizinhos mais próximos. O operador `<->` calcula a distância euclidiana (L2) entre o vetor de consulta e os vetores armazenados na coluna *embedding*. As cláusulas `ORDER BY` e `LIMIT 5` são impostas para ordenar os resultados pela menor distância e retornar apenas os cinco vizinhos mais próximos, respectivamente.

O cálculo dessas distâncias entre os vetores a fim de encontrar a similaridade entre eles, é a aplicação prática do k-Vizinhos Mais Próximos (kNN), que é apresentado na seção 2.6 (Técnicas de Aprendizado de Máquina). Apesar do kNN ser apresentado como técnica para encontrar os vizinhos mais próximos, este algoritmo, que tem complexidade $O(DN)$ (onde D se refere à dimensionalidade do vetor e N à quantidade de vetores), não seria a melhor escolha em um cenário de dados não estruturados. Isto ocorre porque esses dados, representados por vetores multi-dimensionais, implicariam na maldição da dimensionalidade, fazendo as distâncias entre os vetores se igualarem de modo a deixá-las indiferentes e, portanto, tornando as similaridades imprecisas. Dessa maneira, introduz-se o *Hierarchical Navigable Small World* (HNSW), um índice que cria um grafo de múltiplas camadas para encontrar os k-vizinhos mais próximos de maneira mais eficiente, com uma complexidade $O(logN)$ (PAN; WANG; LI, 2023).

2.6 Técnicas de Aprendizado de Máquina

A partir de dados coletados, há uma premissa fundamental no ramo de Inteligência Artificial (IA) que sistemas podem aprender, identificar padrões e tomar decisões com mínima interferência humana. Para que isso seja feito, há dois principais modelos que o aprendizado de máquina pode seguir: o supervisionado e o não supervisionado (MARTINS; LENZ; SILVA, 2020).

O modelo supervisionado — também chamado de classificação — é aquele que tem como objetivo tarefas de predição, ou seja, classificar um resultado específico com base em características conhecidas dos dados. Esse modelo implementa a técnica *hold-out*, que divide o conjunto de dados em duas partes: treinamento e teste. O conjunto de treinamento é utilizado para encontrar e aprender as características e padrões dos dados, que estão rotulados com alguma classe específica. Já no conjunto de teste, os dados têm seus rótulos omitidos para que o algoritmo tente classificá-los; a partir do resultado, é possível verificar o desempenho do algoritmo aplicado. Existem várias técnicas de *hold-out* e algoritmos de classificação, podendo diferentes métodos resultar em um desempenho melhor ou pior, sendo crucial testar diferentes algoritmos para comparar estes resultados (JIANG; GRADUS; ROSELLINI, 2020).

Uma das técnicas mais comuns para a realização do *hold-out* é a amostragem simples aleatória. Neste método, os dados de um conjunto são selecionados aleatoriamente para compor o treino e o teste, sem considerar a distribuição de seus grupos. Ou seja, é possível que haja x quantidades da classe 1 e $2x$ da classe 2, criando desproporcionalidades nos resultados da classificação. Para contornar esse problema, usa-se a amostragem estratificada, em que as amostras são selecionadas de modo a garantir que haja homogeneidade no conjunto de treino e de teste (MAHMUD et al., 2020).

Quanto aos algoritmos de classificação, pode-se citar o kNN, que opera com um embasamento geométrico e não-paramétrico: dado um valor k e um ponto x, o algoritmo calcula a distância do ponto x a todos os outros pontos do conjunto de treino e retorna os k pontos mais próximos; a partir deles, é feita uma votação majoritária para classificar o ponto x com base nas classes de seus vizinhos. Dessa forma, o kNN não constrói um modelo durante o treinamento, uma vez que os dados são armazenados em memória e o processamento ocorre completamente no momento da predição; o que o torna conhecido por "aprendizado preguiçoso" (SYRIOPOULOS et al., 2025).

A regressão logística é um aprendizado de máquina supervisionado estatístico explicado por Sperandei (2014) que calcula a probabilidade de ocorrência de um evento binário, ou seja, uma predição com apenas dois resultados possíveis (por exemplo: verdadeiro ou falso). Este algoritmo transforma a saída através de uma função *logit*, que seria o logaritmo da razão entre a probabilidade de o evento ocorrer e a de não ocorrer, o que garante uma classificação sempre entre 0 e 1. O cálculo do *logit* é dado pela fórmula abaixo:

$$log(\frac{\pi}{1-\pi}) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_m x_m \tag{2.5}$$

onde π indica a probabilidade de um evento, β_i representam os coeficientes de regressão e x_i as variáveis explicativas.

O *random forest* é também um modelo supervisionado que classifica os dados através da técnica *ensemble*, isto é, ele combina as previsões de múltiplos modelos mais simples para produzir o resultado final com maior precisão. O algoritmo se baseia no modelo simples de árvore de decisão, que divide os dados repetidamente por meio de perguntas binárias sobre os preditores até chegar em nós folhas rotulados. São criadas várias árvores de decisão, cada uma treinada em uma amostra aleatória com reposição dos dados. Ao final, é realizada uma votação majoritária entre todas as árvores para definir o resultado real. (SALMAN; KALAKECH; STEITI, 2024)

2.7 Avaliação e Visualização dos Dados

Ao final da execução de técnicas de aprendizado de máquina, é necessário realizar análises e avaliações do desempenho destas a fim de verificar seus desempenhos. Para isso, Obi (2023) discute métricas que identificam diferentes aspectos no resultado do algoritmo. Dentre as principais métricas de avaliação, têm-se a acurácia, que indica o número correto de classificações em todo o conjunto de teste; a precisão, que calcula a proporção de verdadeiros positivos em relação ao total de positivos observados (o que inclui os falsos positivos, erroneamente identificados como positivos); o *recall*, que identifica a habilidade do modelo de identificar corretamente verdadeiros positivos e o *F1-score*, que seria a média harmônica entre precisão e *recall*.

Para entender o que cada métrica explica, é necessário primeiro saber sobre a matriz de confusão. Um algoritmo de classificação pode gerar quatro tipos de predições: Verdadeiro Positivo (VP), Falso Positivo (FP), Verdadeiro Negativo (VN) e Falso Negativo (FN). Considere, por exemplo, um algoritmo que classifica textos como `cyberbullying` ou `not_cyberbullying`. O VP ocorre quando um texto é realmente `cyberbullying` e o algoritmo o classifica corretamente como `cyberbullying`. O FP ocorre quando um texto é `not_cyberbullying`, mas o algoritmo o classifica erroneamente como `cyberbullying`. O VN ocorre quando um texto é `not_cyberbullying` e o algoritmo, corretamente, o classifica como `not_cyberbullying`. O FN ocorre quando um texto é `cyberbullying`, mas o algoritmo não detecta isso e o classifica como `not_cyberbullying`. Juntando o VP, FP, VN e FN em uma tabela, é possível construir a matriz de confusão mostrada na tabela 1, que resumiria o desempenho de um algoritmo de classificação (OBI, 2023).

Tabela 1 – Estrutura da matriz de confusão

	Predito	
Real	`cyberbullying`	`not_cyberbullying`
`cyberbullying`	VP	FN
`not_cyberbullying`	FP	VN

Fonte: Elaborada pelo autor (2026).

A partir disso, é possível calcular as métricas de avaliação para o algoritmo. Abaixo, seguem as fórmulas para cada métrica.

- **Acurácia**

$$acurácia = \frac{VP + VN}{VP + VN + FP + FN} \tag{2.6}$$

- **Precisão**

$$precisão = \frac{VP}{VP + FP} \tag{2.7}$$

- ***Recall***

$$recall = \frac{VP}{VP + FN} \tag{2.8}$$

- ***F1-score***

$$F1\text{-}score = 2 \times \frac{precisão \times recall}{precisão + recall} \tag{2.9}$$

Quando o problema de classificação envolve mais de duas classes, calcula-se o *F1-score* individualmente para cada classe e, em seguida, agrega-se esses valores em uma única métrica global. Uma das formas mais utilizadas para essa agregação é a *macro F1-score*, que calcula a média aritmética simples do *F1-score* obtido em cada classe, atribuindo o mesmo peso a todas elas, independentemente da quantidade de exemplos que cada uma possui (OBI, 2023). A fórmula da *macro F1-score* é apresentada abaixo:

$$macro\ F1 = \frac{1}{N} \sum_{i=1}^{N} F1\text{-}score_i \tag{2.10}$$

onde N é o número total de classes e $F1\text{-}score_i$ é o *F1-score* calculado para a classe i.

Além das métricas numéricas, é possível utilizar técnicas de visualização para observar como os dados se distribuem no espaço de representação vetorial. Uma dessas técnicas é o *Uniform Manifold Approximation and Projection* (UMAP), um método que reduz a dimensionalidade dos dados, projetando-os em um espaço menor (geralmente em duas dimensões) para que possam ser visualizados em um gráfico. O UMAP busca preservar, nessa projeção, as relações de proximidade que os dados originalmente possuem: pontos que são semelhantes entre si tendem a permanecer próximos também na representação visual. Dessa forma, é possível observar de maneira gráfica se as classes de um conjunto de dados formam grupos bem definidos e separados entre si, ou se apresentam sobreposição (GHOJOGH et al., 2021).

3 Trabalhos Relacionados

Com o crescente acesso a dados não estruturados, motivado principalmente pelo comércio eletrônico e plataformas de recomendação, torna-se evidente a necessidade de SGBDV para a recuperação de informação. Este fornece recursos tradicionais de um SGBD (como otimização de consultas, transações, escalabilidade e tolerância a falhas), porém, para dados não estruturados. O trabalho de Pan, Wang e Li (2023) demonstrou como armazenar e organizar esse tipo de dado em um SGBDV de modo a obter buscas com eficiência e acurácia, além de exemplificar o uso de critérios em consultas e como executá-las. A pesquisa destes autores apresentou cinco principais obstáculos para o uso de dados vetoriais, sendo eles: (i) critérios vagos de pesquisa; (ii) comparações caras computacionalmente; (iii) dados de tamanho muito grande; (iv) falta de estrutura entre os dados; e (v) incompatibilidade com atributos.

Segundo Pan, Wang e Li (2023), para armazenar um dado não estruturado em um SGBDV, seria necessário primeiro codificá-lo em um vetor de características D-dimensional através de um modelo de *embedding*. Dessa forma, Reimers e Gurevych (2019) apresentaram o SBERT, um *framework* que gera representações vetoriais de sentenças que possuem significado semântico, isto é, gera um *embedding* capaz de ser comparado e recuperado semanticamente. O SBERT é uma modificação do BERT, sendo que este último não é capaz comparar sentenças em larga escala de forma eficiente. O SBERT foi avaliado tanto em similaridade textual semântica supervisionada quanto não supervisionada através da métrica de *Spearman*. Reimers e Gurevych (2019) concluíram que o *framework* teve uma melhora significativamente em relação a outros métodos de geração de *embeddings* e que ele é computacionalmente eficiente, permitindo a comparação de grandes volumes de dados, o que viabiliza o uso em SGBDVs. Conforme Pan, Wang e Li (2023), diversos tipos de dados não estruturados podem ser codificados em *embeddings*, como documentos textuais, imagens e vídeos, sendo textos o foco do SBERT. No contexto de *cyberbullying*, principalmente em redes sociais, dados textuais são utilizados para a detecção dessa prática. Sendo assim, o presente trabalho usou um *framework* capaz de gerar representações vetoriais de textos curtos que possuam significado semântico para uma melhor classificação.

Já em uma temática focada em *cyberbullying*, Teng e Varathan (2023) aplicaram um pré-processamento dos dados em língua inglesa com uma sequência de três etapas com técnicas que visam manter o máximo possível do texto original para não interferir na predição. Na primeira etapa, foi realizada a limpeza do texto através da remoção de *Uniform Resource Locators* (URLs), *e-mails*, menção de usuários, elementos HTML, múltiplos espaços, símbolos de novas linhas, símbolos inseridos como marcadores do processo

de anotação do *dataset* e espaço entre caracteres únicos consecutivos (por exemplo: "*W H A T*" em "*WHAT*"). Na segunda etapa, foi realizada uma normalização do texto usando redução de caracteres alongados (por exemplo: "*youuu*" em "*you*"), transformação de emojis e emoticons em texto, caracteres acentuados normalizados para o alfabeto inglês e em minúsculo, e conversão de gírias para palavras normais. Na última etapa, houve a remoção de números e pontuação do texto, e a lematização (exceto de pronomes e, no caso de retorno "*be*", manteve-se a palavra original). Os autores enfatizaram que não houve remoção de *stop-word*s, pois termos como negação e pronomes ajudam a identificar o *cyberbullying*. Essas técnicas de pré-processamento são relevantes para o presente trabalho, que também lidou com dados textuais de redes sociais.

O trabalho de Agrawal, Kumar e Tripathi (2024) apresentou o uso de *embeddings* para a detecção de *cyberbullying*, alegando que esse processo captura relações semânticas e complexidades contextuais presentes no conteúdo de *cyberbullying*. Isso resulta em uma representação mais detalhada do texto original, capaz de capturar sutilezas e contexto. Em seu trabalho, os autores utilizam o *framework Universal Sentence Embeddings* (USE) para a geração dos *embeddings*. De forma semelhante, este trabalho usou geração de *embeddings* no contexto de detecção de *cyberbullying*.

Na literatura, muitos trabalhos buscam detectar o *cyberbullying* através de modelos supervisionados, como Bharti et al. (2022), que realizaram uma comparação entre os algoritmos *random forest*, árvore de decisão, regressão logística, *naive Bayes*, máquina de vetores de suporte (SVM) e *Extreme Gradient Boosting*. A partir desses algoritmos, foi possível predizer se um dado era ou não *cyberbullying* de maneira eficiente, analisando resultados que os autores trouxeram a partir de métricas de avaliação.

Por outro lado, o trabalho de Li et al. (2024) trouxe um método de classificação híbrida: os autores mesclam um modelo supervisionado com o kNN, o que validou a busca por similaridade como pesos para classificadores na literatura. Os autores utilizaram o cálculo de distância euclidiana para encontrar os k-vizinhos mais próximos. Após isso, aplicaram técnicas de aumentação de texto sobre os vizinhos recuperados e converteram os valores encontrados em pesos de similaridade. Esses pesos foram normalizados, gerando uma distribuição de probabilidade que foi integrada aos classificadores por meio de interpolação linear. Tais classificadores foram: rede neural convolucional (CNN), memória de longo prazo (LSTM), B*idirectional Encoder Representations from Transformers* (BERT) e *Robustly Optimized BERT Approach* (RoBERTa).

Corsi (2025) realiza um estudo de caso de um SGBDV para manipulação de dados de redes sociais online. A autora destaca a problemática dos dados estruturados em SGBDs e propõe o uso de SGBDVs como solução, mais especificamente o PostgreSQL com extensão PGVector. São demonstradas técnicas para pré-processamento dos dados e geração de *embeddings*, bem como armazenamento e indexação destes e, posteriormente,

clusterização através do algoritmo *K-means*. Os resultados da pesquisa demonstraram que é plenamente viável o uso do PostgreSQL com extensão PGVector para a realização de tarefas de análise de dados não estruturados. Dessa maneira, o trabalho de Corsi (2025) serviu como base da presente pesquisa, uma vez que fundamenta a escolha do SGBDV e técnicas para PLN.

Por fim, o trabalho de Sathishkumar et al. (2025) abordou diferentes métricas de avaliação que devem ser usadas para verificar o desempenho da predição realizada. Os autores mencionaram o uso de: acurácia, precisão, *recall* e *F1-score*. A análise conjunta dessas métricas garante uma visão real da eficácia do sistema, uma vez que a análise individual pode gerar falsas conclusões devido ao fato de que cada métrica possui insuficiências. Por isso, as métricas utilizadas pelos autores serviram como referência para a avaliação do estudo de caso deste trabalho.

Tabela 2 – Comparativo dos trabalhos correlatos

Trabalho	**Relação com este trabalho**	**Cyberbullying**	**SGBDV**	**Embeddings**	**Classificação**
Pan, Wang e Li (2023)	Fundamenta os conceitos teóricos de SGBDV utilizados	Não	Sim	Não	Não realiza
Reimers e Gurevych (2019)	Propõe o SBERT, embedding utilizado neste trabalho	Não	Não	Sim	Não realiza
Teng e Varathan (2023)	Referência para as etapas de pré-processamento	Sim	Não	Sim	Externa
Agrawal, Kumar e Tripathi (2024)	Usa embeddings para detecção de cyberbullying	Sim	Não	Sim	Externa
Bharti et al. (2022)	Compara classificadores supervisionados para cyberbullying	Sim	Não	Sim	Externa
Li et al. (2024)	Valida a busca por similaridade como apoio à classificação	Não	Não	Sim	Externa
Corsi (2025)	Base para a escolha do PostgreSQL com PGVector	Não	Sim	Sim	Não realiza
Sathishkumar et al. (2025)	Referência para as métricas de avaliação	Sim	Não	Sim	Externa
Este trabalho	—	**Sim**	**Sim**	**Sim**	**Nativa no SGBD**

Fonte: Elaborada pelo autor (2026).

4 Método

Neste capítulo, é apresentado o método proposto para a classificação de *cyberbullying* em textos. O processo combina técnicas de PLN, representação vetorial semântica, armazenamento dos dados e classificação, integrando a busca por similaridade por meio do SGBDV e modelo supervisionado de aprendizado de máquina.

O método é estruturado em sete etapas sequenciais e dependentes. A primeira etapa consiste na coleta dos dados a partir de um conjunto de dados público. Em seguida, os dados passam por um pré-processamento textual, que os padroniza e os prepara para a etapa seguinte de representação vetorial. Os vetores gerados são divididos com amostragem e armazenados no SGBDV para serem utilizados na etapa de classificação. Por fim, os resultados passam por uma avaliação. Cada uma dessas etapas é detalhada nas subseções a seguir.

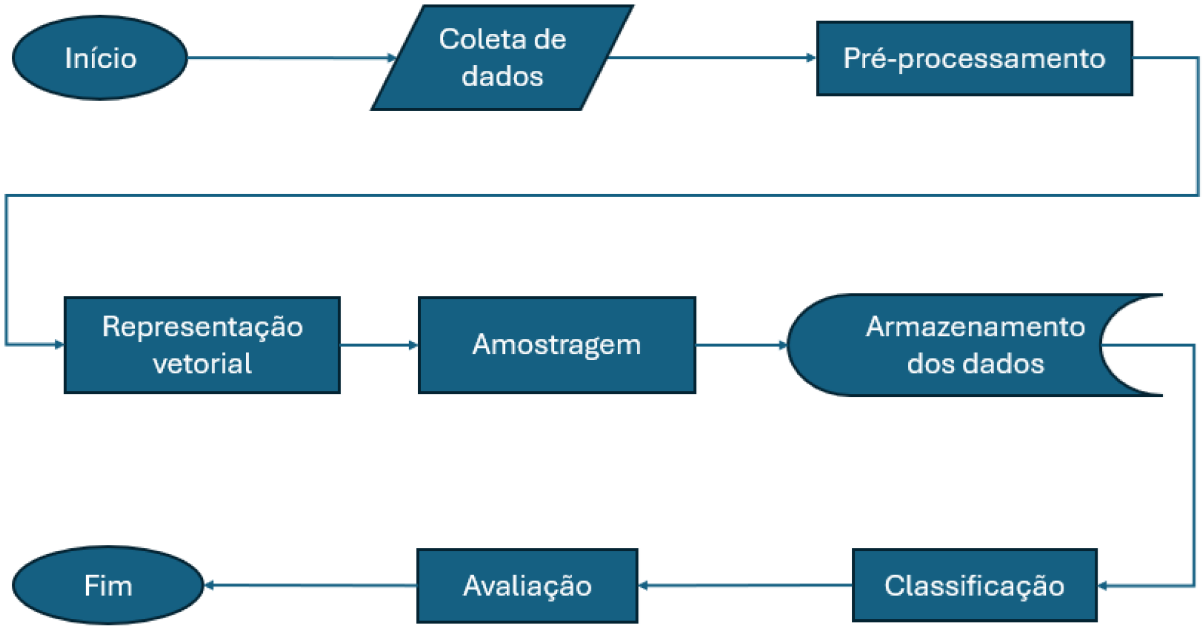

Figura 1 – Fluxograma do Método.

Fonte: Elaborada pelo autor (2026).

4.1 Coleta dos Dados

Na primeira etapa, os dados para a predição de *cyberbullying* foram coletados por meio da plataforma Kaggle[1]. Os dados foram extraídos do conjunto de dados de Larxel (2022) sobre classificação de *cyberbullying*. O autor deste conjunto de dados específico

[1] <https://www.kaggle.com/>

criou uma base de dados que classifica textos da plataforma X (antigo *Twitter*) em 6 rótulos, sendo eles: `not_cyberbullying`, caso não seja *cyberbullying*; `religion`, `age`, `gender` e `ethnicity`, caso seja identificado como *cyberbullying*, reconhecendo o seu tipo específico; e `other_cyberbullying`, caso seja identificado como *cyberbullying*, mas não se enquadre nos tipos anteriores. O conjunto de dados possui 47.000 textos de *tweets* na língua inglesa e está balanceado de forma a ter, aproximadamente, 8.000 *tweets* de cada classe. Este conjunto de dados foi escolhido para o trabalho atual por motivos como: o balanceamento que possui; a identificação do tipo de *cyberbullying*, possibilitando uma análise mais profunda; e a língua inglesa, uma vez que as etapas de pré-processamento foram planejadas para este idioma.

4.2 Pré-Processamento dos Dados

Na etapa de pré-processamento, o trabalho de Teng e Varathan (2023) serviu como referência. Esta foi uma etapa de extrema importância, pois padronizou e limpou os dados sem comprometer o seu conteúdo semântico, mantendo o máximo possível do texto original, a fim de melhorar a precisão da classificação.

Para isso, o processo foi dividido em quatro etapas principais discutidas a seguir. Para demonstrar o pré-processamento, oito *tweets* foram tomados como referência. Para cada etapa, foi mostrado o antes e o depois de cada *tweet*.

Tabela 3 – *Tweets* para Demonstração

ID	***Tweet***
0	mmmm #MKR Forget Deconstruction @lisamromano @garydlum @mattsparks88 THIS is Lemon Tart 🍋🍋🍋 mmmm http://t.co/iiLigXbbzw
1	@TeeYouVee if it's something that needs immediate attention, randi@randi.io works as well.
2	The amount of times the men in my family refer to women as bitches and make rape jokes and gay jokes is I N S A N E.
3	ITS BEAUTIFULLLLLLL :'(:'(:'(:'(
4	RT @Everestedup: Omg double elimination?! #MKR
5	@jaredchase i kept being the last person alive AND topping damage. it's like, what. this isn't hard.
6	RT @barbara_volkwyn: Islamic State Resupply Route Extended to 400 km After Peshmerga Take #Mosul– #Sinjar Road http://t.co/bvrcRNAn1w #ISI…
7	Film ini salah satu potret bullying dikalangan anak2 #mynameiskhan

Fonte: Elaborada pelo autor (2026).

4.2.1 Limpeza

A limpeza dos dados consistiu na remoção de URLs, *e-mails*, menções a usuários, espaços entre caracteres únicos consecutivos e múltiplos espaços. Tecnicamente, essas remoções foram implementadas por meio de expressões regulares específicas para cada padrão. Após a limpeza de todos os itens, foi feito um filtro dos idiomas presentes na amostra de dados, a fim de verificar se todos eram da língua inglesa — para a qual o pré-processamento foi elaborado. Analisando a base de dados, notou-se que existiam alguns idiomas indesejados, estes foram removidos.

Tabela 4 – Processo de Limpeza

ID	***Tweet* (Antigo)**	***Tweet* (Novo)**
0	mmmm #MKR Forget Deconstruction @lisamromano @garydlum @mattsparks88 THIS is Lemon Tart 🍋🍋🍋 mmmm http://t.co/iiLigXbbzw	mmmm #MKR Forget Deconstruction THIS is Lemon Tart 🍋🍋🍋 mmmm
1	@TeeYouVee if it's something that needs immediate attention, randi@randi.io works as well.	if it's something that needs immediate attention, works as well.
2	The amount of times the men in my family refer to women as bitches and make rape jokes and gay jokes is I N S A N E.	The amount of times the men in my family refer to women as bitches and make rape jokes and gay jokes is INSANE.
3	ITS BEAUTIFULLLLLLL :'(:'(:'(:'(	ITS BEAUTIFULLLLLLL :'(:'(:'(:'(
4	RT @Everestedup: Omg double elimination?! #MKR	RT: Omg double elimination?! #MKR
5	@jaredchase i kept being the last person alive AND topping damage. it's like, what. this isn't hard.	i kept being the last person alive AND topping damage. it's like, what. this isn't hard.
6	RT @barbara_volkwyn: Islamic State Resupply Route Extended to 400 km After Peshmerga Take #Mosul– #Sinjar Road http://t.co/bvrcRNAn1w #ISI…	RT: Islamic State Resupply Route Extended to 400 km After Peshmerga Take #Mosul– #Sinjar Road #ISI…
7	Film ini salah satu potret bullying dikalangan anak2 #mynameiskhan	*null*

Fonte: Elaborada pelo autor (2026).

4.2.2 Normalização

A segunda etapa consistiu na normalização do texto de caracteres alongados (por exemplo: "*youuu*" em "*you*"), transformação de emojis e emoticons em texto, caracteres acentuados normalizados para o alfabeto inglês, caracteres normalizados para minúsculo, expansão de contrações (por exemplo: "*You're*" em "*You are*") e conversão de gírias para palavras normais. Para esta última parte, foram utilizadas três bases de dados *online*[2,3,4] que contêm gírias e seus respectivos significados.

Do ponto de vista técnico, os caracteres alongados foram tratados por expressão regular combinada a uma verificação em um dicionário de palavras válidas do inglês, evitando alterar palavras legítimas com letras repetidas. Emojis e emoticons foram convertidos em texto por bibliotecas específicas de mapeamento. A remoção de acentos utilizou normalização Unicode, separando caracteres base de seus acentos gráficos. A conversão de gírias baseou-se em consulta direta a um dicionário construído a partir das três fontes citadas, e a expansão de contrações utilizou uma biblioteca especializada nesse tipo de tratamento para o inglês.

Tabela 5 – Processo de Normalização

ID	***Tweet* (Antigo)**	***Tweet* (Novo)**
0	mmmm #MKR Forget Deconstruction THIS is Lemon Tart 🍋🍋🍋 mmmm	m #mkr forget deconstruction this is lemon tart :lemon::lemon::lemon: m
1	if it's something that needs immediate attention, works as well.	if it is something that needs immediate atention, works as wel.
2	The amount of times the men in my family refer to women as bitches and make rape jokes and gay jokes is INSANE.	the amount of times the men in my family refer to women as bitches and make rape jokes and gay jokes is insane.
3	ITS BEAUTIFULLLLLLL :'(:'(:'(:'(	its beautiful crying crying crying crying
4	RT: Omg double elimination?! #MKR	rt: oh my gosh double elimination?! #mkr
5	i kept being the last person alive AND topping damage. it's like, what. this isn't hard.	i kept being the last person alive and topping damage. it is like, what. this is not hard.
6	RT: Islamic State Resupply Route Extended to 400 km After Peshmerga Take #Mosul–#Sinjar Road #ISI…	rt: islamic state resupply route extended to 40 km after peshmerga take #mosul– #sinjar road #isi…
7	*null*	*null*

Fonte: Elaborada pelo autor (2026).

[2] <https://slang.net/terms/online_chat>
[3] <https://slang.net/terms/text_messaging>
[4] <https://slang.net/terms/social_media>

4.2.3 Remoção

A terceira etapa consistiu em remover números, pontuação do texto e múltiplos espaços, cada um tratado por meio de uma expressão regular específica. Foi necessário repetir o processo de remover múltiplos espaços — que já havia sido realizado na etapa de limpeza —, pois durante a realização da normalização, notou-se que alguns processos criaram novos múltiplos espaços.

Tabela 6 – Processo de Remoção

ID	***Tweet* (Antigo)**	***Tweet* (Novo)**
0	m #mkr forget deconstruction this is lemon tart :lemon::lemon::lemon: m	m mkr forget deconstruction this is lemon tart lemonlemonlemon m
1	if it is something that needs immediate atention, works as wel.	if it is something that needs immediate atention works as wel
2	the amount of times the men in my family refer to women as bitches and make rape jokes and gay jokes is insane.	the amount of times the men in my family refer to women as bitches and make rape jokes and gay jokes is insane
3	its beautiful crying crying crying crying	its beautiful crying crying crying crying
4	rt: oh my gosh double elimination?! #mkr	rt oh my gosh double elimination mkr
5	i kept being the last person alive and topping damage. it is like, what. this is not hard.	i kept being the last person alive and topping damage it is like what this is not hard
6	rt: islamic state resupply route extended to 40 km after peshmerga take #mosul– #sinjar road #isi…	rt islamic state resupply route extended to km after peshmerga take mosul sinjar road isi
7	*null*	*null*

Fonte: Elaborada pelo autor (2026).

4.2.4 Lematização

A quarta e última etapa do pré-processamento se deu na lematização do texto. Esta reduziu palavras flexionadas à sua forma base (por exemplo: "*crying*" e "*cried*" em "*cry*"). A lematização foi realizada com o lematizador do NLTK, utilizando a classe gramatical de cada palavra para identificar a forma base correta. Ressalta-se que não houve lematização de pronomes e em caso de retorno de lema "*be*".

Tabela 7 – Processo de Lematização

ID	***Tweet* (Antigo)**	***Tweet* (Novo)**
0	m mkr forget deconstruction this is lemon tart lemonlemonlemon m	m mkr forget deconstruction this lemon tart lemonlemonlemon m
1	if it is something that needs immediate atention works as wel	if something that need immediate atention work a wel
2	the amount of times the men in my family refer to women as bitches and make rape jokes and gay jokes is insane	the amount of time the men in family refer to woman a bitch and make rape joke and gay joke insane
3	its beautiful crying crying crying crying	beautiful cry cry cry cry
4	rt oh my gosh double elimination mkr	rt oh gosh double elimination mkr
5	i kept being the last person alive and topping damage it is like what this is not hard	i keep the last person alive and topping damage like what this not hard
6	rt islamic state resupply route extended to km after peshmerga take mosul sinjar road isi	rt islamic state resupply route extend to km after peshmerga take mosul sinjar road isi
7	*null*	*null*

Fonte: Elaborada pelo autor (2026).

Durante a execução do pré-processamento, foram utilizadas diversas bibliotecas da linguagem Python, cada uma responsável por uma tarefa específica de tratamento textual. A tabela 8 resume as bibliotecas empregadas e sua utilidade neste trabalho.

Tabela 8 – Bibliotecas utilizadas no pré-processamento

Biblioteca	**Utilidade**
re	Criação de expressões regulares para identificar e remover padrões textuais, como espaços entre caracteres únicos consecutivos, múltiplos espaços e caracteres alongados.
unicodedata	Remoção de acentuação gráfica, separando caracteres base de seus acentos e normalizando o texto para o alfabeto inglês.
urlextract	Identificação e remoção de URLs presentes nos *tweets*.
emot	Identificação de emoticons e conversão destes em sua descrição textual correspondente.
emoji	Conversão de emojis em sua descrição textual correspondente.
contractions	Expansão de contrações da língua inglesa.
langdetect	Detecção do idioma de cada *tweet*.
NLTK	Fornecimento do *corpus* de palavras válidas do inglês (utilizado na normalização de caracteres alongados), tokenização, classificação gramatical e lematização das palavras por meio da classe `WordNetLemmatizer`.

Fonte: Elaborada pelo autor (2026).

4.3 Representação Vetorial dos Dados

Após o pré-processamento dos dados, eles foram representados de maneira vetorial, a fim de armazená-los no PostgreSQL e utilizar a extensão PGVector, que requer dados vetoriais. Nesta etapa, foram utilizadas três ferramentas de geração de *embeddings*, com o intuito de gerar comparações entre seus respectivos resultados após a classificação. Cada ferramenta gerou 44.192 *embeddings*, totalizando 132.576 *embeddings*.

A primeira ferramenta a ser usada foi o SBERT[5], um *framework* que agrupa vários possíveis modelos para geração de *embeddings*. O modelo escolhido para o SBERT foi o *"sentence-transformers/all-MiniLM-L6-v2"*, a escolha do modelo se deu com base no idioma dos dados, uma vez que este é ideal para textos na língua inglesa. Este modelo foi carregado diretamente por meio de sua biblioteca própria, que já retorna o vetor da sentença como um todo, sem etapas adicionais de processamento, contendo este 384 dimensões.

O modelo BERT[6] também foi utilizado. Com ele, foi possível analisar se as implementações feitas no SBERT realmente melhoraram ou não a classificação final dos *embeddings*. Diferente do SBERT, este modelo não gera nativamente um único vetor para a frase inteira, produzindo em vez disso um vetor para cada palavra do texto; por isso, foi necessário calcular a média desses vetores para obter um único *embedding* por *tweet*, contendo este 768 dimensões.

Por fim, utilizou-se o modelo FastText[7]. Como explicado previamente, esse modelo é capaz de representar palavras raras ou desconhecidas, sendo, portanto, ideal para linguagem informal, comum em contextos de redes sociais, que contém gírias e abreviações das mais diversificadas. Este modelo foi carregado diretamente a partir de um arquivo pré-treinado local — disponibilizado pelos próprios autores do FastText para a língua inglesa[8] —, sem etapas adicionais de processamento, contendo este 300 dimensões.

A tabela 9 traz uma exposição de como seria o corpo de um *embedding*; ressalta-se que devido à alta dimensionalidade de *embeddings*, a representação apresentada foi abreviada para fins didáticos.

[5] <https://github.com/huggingface/sentence-transformers>
[6] <https://github.com/google-research/bert>
[7] <https://github.com/facebookresearch/fastText>
[8] <https://github.com/facebookresearch/fastText/blob/master/docs/crawl-vectors.md>

Tabela 9 – Processo de Representação Vetorial

Tweet	***Embedding***
disadvantage of way too nice and caring feel too bad to say no sometimes too emotional bully in high school make fun of for always in the friendzone by girl always feelings alone unappreciated and unimportant	[-0.09758734, -0.038671423, 0.006598736, -0.101795316, -0.056472674, -0.028993761, ..., 0.057263795, 0.0024695923, 0.042530145, 0.04253929, -0.021647785, 0.0013830045]

Fonte: Elaborada pelo autor (2026).

A escolha dos três modelos de *embedding* avaliados — SBERT, BERT e FastText — não foi arbitrária, mas reflete a evolução histórica das técnicas de representação vetorial de texto discutida na seção 2.4 (Representação Semântica dos Dados e *Embeddings*). O FastText gera vetores estáticos: uma mesma palavra recebe sempre a mesma representação, mesmo quando possui vários significados possíveis. O BERT avança ao gerar representações dinâmicas, sensíveis ao contexto da sentença, mas sua arquitetura o torna inadequado para comparação por similaridade entre vetores. O SBERT, por fim, combina as duas vantagens: gera representações dinâmicas e, ao mesmo tempo, foi projetado especificamente para permitir a comparação por similaridade de forma eficiente. Avaliar esses três modelos permite, portanto, observar de forma controlada o impacto de cada uma dessas evoluções no desempenho da classificação por similaridade realizada neste trabalho.

4.4 Amostragem

Para fins de teste e treino, foi feita uma amostragem na base de dados. O método escolhido foi o estratificado, que separa uma quantidade balanceada de cada classe na amostra, o que garante exemplos de treino suficientes para uma melhor classificação. Tecnicamente, essa amostragem foi implementada agrupando os dados por classe e sorteando 70% de cada grupo, com uma semente fixa ($random\_state = 42$) para garantir a reprodutibilidade do experimento.

Cada classe recebeu 70% dos dados para treino e o restante dos dados (30%) foi reservado para teste, estes tiveram seus rótulos alterados para *"null"*. Por fim, os dados para teste foram copiados para um novo dataframe que foi utilizado na etapa de avaliação para comparar o resultado do classificador com o rótulo real. Essa técnica de divisão do conjunto de dados para treino e teste seguiu a técnica proposta por Teng e Varathan (2023). A divisão proposta pode ser analisada nas tabelas 10 e 11.

Tabela 10 – Conjunto de dados rotulados - Treino

Classe	**Quantidade**
`religion`	5579
`age`	5558
`gender`	5335
`ethnicity`	5174
`other_cyberbullying`	4729
`not_cyberbullying`	4560
Total	**30935**

Fonte: Elaborada pelo autor (2026).

Tabela 11 – Conjunto de dados não rotulados - Teste

Classe	**Quantidade**
`religion`	2391
`age`	2382
`gender`	2287
`ethnicity`	2217
`other_cyberbullying`	2026
`not_cyberbullying`	1954
Total	**13257**

Fonte: Elaborada pelo autor (2026).

4.5 Armazenamento dos Dados

Com os dados já tratados e em formato vetorial, foi realizado o armazenamento deles no SGBD PostgreSQL. Para isso, primeiramente, foi realizada a modelagem da estrutura do SGBDV, contendo tabelas e campos para armazenar todos os dados necessários, ilustrado na figura 2. A conexão com o banco e a manipulação dos vetores foram feitas por meio de uma biblioteca de *driver* com suporte nativo ao tipo vetorial do PGVector, permitindo a inserção direta dos *embeddings* gerados nas tabelas correspondentes.

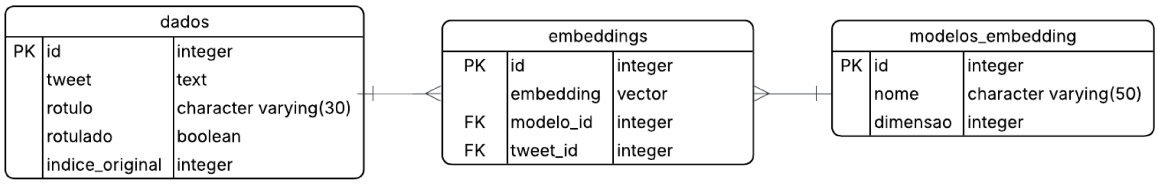

Figura 2 – Diagrama de Entidade e Relacionamento.

Fonte: Elaborada pelo autor (2026).

O DER apresentado na figura 2 é composto por três entidades: `dados`, `embeddings` e `modelos_embedding`. A entidade `dados` armazena o texto tratado de cada *tweet*, seu rótulo (classe de *cyberbullying*), um indicador booleano que sinaliza se o registro pertence ao conjunto de treino ou teste, e um índice que referencia a posição original do dado antes da amostragem. Esse índice foi mantido para permitir a rastreabilidade de cada registro amostrado até a base de dados original, viabilizando validações posteriores sem a necessidade de repetir o processo de amostragem. A entidade `modelos_embedding` registra os três modelos de geração de *embeddings* utilizados (SBERT, BERT e FastText), armazenando o nome e a dimensionalidade de cada um. Por fim, a entidade `embeddings` relaciona cada *tweet* ao seu respectivo vetor gerado, associando-o tanto ao modelo que o originou quanto ao dado textual correspondente por meio de chaves estrangeiras. Essa separação entre texto, *embedding* e modelo permitiu que um mesmo *tweet* possuísse múltiplas representações vetoriais simultâneas, uma para cada modelo avaliado, sem duplicação dos dados textuais.

Após a criação do DER, as tabelas puderam ser criadas via comandos SQL. No entanto, antes disso, foi necessário ativar a extensão PGVector que, por padrão, vem desativada no PostgreSQL. Para isso, executou-se o algoritmo da listagem 4.1.

Listagem 4.1 – Ativar a extensão PGVector dentro do PostgreSQL

```
1 CREATE EXTENSION IF NOT EXISTS vector;
```

Fonte: Elaborada pelo autor (2026).

De maneira análoga à extensão PGVector, o índice HNSW — utilizado para otimizar o algoritmo kNN —, também precisa ser ativado no SGBDV. Este, contudo, precisa ser criado para cada modelo de *embedding*, não bastando uma única criação global. Para isso, executou-se o algoritmo da listagem 4.2.

Listagem 4.2 – Criar Índice HNSW dinamicamente

```
1  CREATE OR REPLACE FUNCTION criar_indice_hnsw(modelo_nome VARCHAR(50))
2  RETURNS VOID AS $$
3  DECLARE
4      v_modelo_id INT;
5      v_dimensao  INT;
6  BEGIN
7      SELECT id, dimensao INTO v_modelo_id, v_dimensao
8      FROM modelos_embedding
9      WHERE nome = modelo_nome;
10
11     IF v_modelo_id IS NULL THEN
12         RAISE EXCEPTION 'Modelo % não encontrado', modelo_nome;
13     END IF;
```

```
14
15      EXECUTE format(
16          'CREATE INDEX IF NOT EXISTS hnsw_%s ON embeddings
17           USING hnsw ((embedding::vector(%s)) vector_cosine_ops)
18           WHERE modelo_id = %s',
19          modelo_nome, v_dimensao, v_modelo_id
20      );
21  END;
22  $$ LANGUAGE plpgsql;
```

Fonte: Elaborada pelo autor (2026).

Ao final da criação das tabelas e índices, os dados foram armazenados no PostgreSQL. A tabela `dados` armazenou 44.192 registros, correspondentes aos *tweets* resultantes do pré-processamento. A tabela `embeddings` totalizou 132.576 registros, resultado da geração de *embeddings* por cada um dos três modelos para todos os *tweets*. Por fim, a tabela `modelos_embedding` registrou os 3 modelos utilizados: SBERT, BERT e FastText.

4.6 Classificação dos Dados

Para a etapa de classificação, foi adotada a técnica de aprendizado de máquina kNN. Diferentemente do uso de bibliotecas prontas de aprendizado de máquina, o algoritmo de classificação kNN foi integralmente implementado pelo autor deste trabalho como uma função em PL/pgSQL, executada dentro do próprio PostgreSQL. Essa implementação utilizou ferramentas fornecidas pela extensão do PGVector, como a função para cálculo de distâncias e a criação de índices, mas a lógica de votação majoritária e classificação foi construída especificamente para este estudo de caso. Ao executar o algoritmo diretamente no SGBDV, evita-se a transferência dos *embeddings* para uma aplicação externa, aproveitando as operações vetoriais nativas do PGVector para tornar o processo de classificação exclusivo do SGBDV.

Para a execução do algoritmo do kNN, foi utilizada a similaridade do cosseno — métrica suportada pelo PGVector e popular em análise de texto (CUNNINGHAM; DELANY, 2021). Vale ressaltar que Pan, Wang e Li (2023) abordam a escolha de uma métrica adequada para uma aplicação específica, dizendo que, até então, não há princípios que guiem a escolha de uma medida ideal. Além do cálculo da distância, o kNN exige que seja declarado um valor para k, que é a quantidade de vizinhos mais próximos. Como foi explicado por Syriopoulos et al. (2025), não existe um valor para k perfeito universal, sendo assim, foram testados 9 diferentes valores para k (3, 5, 7, 9, 11, 15, 21, 31 e 51). Assim, foi possível analisar os diferentes resultados e definir o melhor valor para o caso específico deste trabalho.

Além disso, é importante ressaltar que o algoritmo de classificação foi executado para cada modelo de *embedding* (SBERT, BERT e FastText). Sendo assim, foram retornados 27 resultados (três modelos com nove variações de k cada um), os quais foram comparados para verificar o melhor modelo com o melhor valor de k possível. Essa execução foi realizada em Python, que percorreu cada combinação de modelo e valor de k, conectando-se ao PostgreSQL por meio da biblioteca psycopg2 com suporte ao PGVector e chamando a função de classificação diretamente no SGBDV a cada iteração.

O kNN utilizado pode ser visto no algoritmo 1 e na listagem 4.3. A função `classificar_knn` recebe dois parâmetros: `k` — que é a quantidade de vizinhos tomados para votação do rótulo do *tweet* que está sendo classificado — e `modelo_embedding` — que se refere ao modelo (SBERT, BERT ou FastText) usado naquela classificação.

São declaradas quatro variáveis na função (linhas 5–8), sendo elas e seus usos:

- `v_registro` — guarda temporariamente cada *tweet* não rotulado durante o *loop*
- `v_rotulo_atribuido` — guarda o rótulo vencedor da votação kNN para o *tweet* atual
- `v_modelo_id` — id do modelo de *embedding*
- `v_dimensao` — dimensão do vetor do modelo

A primeira etapa do algoritmo consiste em recuperar o id e a dimensão do modelo de *embedding* a ser usado e validar a sua existência (linhas 12–19). Esses valores foram usados para filtrar os dados e classificar apenas *tweets* do *embedding* escolhido. Após isso, é feita uma limpeza no campo `rotulo` da tabela `dados` para todos os dados registrados como "teste" na amostragem (linhas 22–23). Essa limpeza garante que cada execução produza um resultado limpo e sem qualquer contaminação de execuções anteriores com outros valores de k ou modelos diferentes.

Nas linhas 26–55 é onde o kNN — apresentado na seção 2.6 (Técnicas de Aprendizado de Máquina) — de fato é realizado. O algoritmo começa criando um *loop* que itera para cada *tweet* não rotulado que possui *embedding* gerado para o modelo solicitado (linhas 26–32). A subquery em 36–42 calcula a distância do cosseno entre o *tweet* da iteração atual e os k vizinhos mais próximos. A query externa em 35–46 agrupa esses k vizinhos por rótulo e os ordena de forma decrescente, retornando a classe com maior contagem. Por fim, os dados de teste foram atualizados com a classe retornada pela consulta.

Algoritmo 1: Pseudo-código do kNN

for *cada tweet não rotulado:* **do**

1. Pega seu *embedding* do modelo solicitado;
2. Encontra os k *tweets* rotulados mais próximos (cosseno);
3. Faz votação por maioria entre os rótulos dos k vizinhos;
4. Grava o rótulo vencedor na tabela dados;

Fonte: Elaborado pelo autor (2026).

Listagem 4.3 – Algoritmo do kNN

```
CREATE OR REPLACE FUNCTION classificar_knn(k INT, modelo_embedding
    VARCHAR(50))
RETURNS VOID AS $$

DECLARE
    v_registro RECORD;
    v_rotulo_atribuido VARCHAR(30);
    v_modelo_id INT;
    v_dimensao INT;

BEGIN
    -- Busca o id e dimensão do modelo informado
    SELECT id, dimensao INTO v_modelo_id, v_dimensao
    FROM modelos_embedding
    WHERE nome = modelo_embedding;

    -- Verifica se o modelo existe
    IF v_modelo_id IS NULL THEN
        RAISE EXCEPTION 'Modelo % não encontrado', modelo_embedding;
    END IF;

    -- Limpa classificações anteriores dos dados não rotulados
    UPDATE dados SET rotulo = NULL
    WHERE rotulado = FALSE;

    -- Classificação kNN
    FOR v_registro IN (
        SELECT d.id, e.embedding
        FROM dados d JOIN
             embeddings e ON e.tweet_id = d.id
        WHERE d.rotulado = FALSE AND
```

```
31              e.modelo_id = v_modelo_id
32      ) LOOP
33
34          EXECUTE format('
35              SELECT rotulo FROM (
36                  SELECT d.rotulo
37                  FROM dados d JOIN
38                       embeddings e ON e.tweet_id = d.id
39                  WHERE d.rotulado = TRUE AND
40                        e.modelo_id = $1
41                  ORDER BY e.embedding::vector(%s) <=> $2::vector(%s)
42                  LIMIT $3
43              ) vizinhos
44              GROUP BY rotulo
45              ORDER BY COUNT(*) DESC
46              LIMIT 1
47          ', v_dimensao, v_dimensao)
48          INTO v_rotulo_atribuido
49          USING v_modelo_id, v_registro.embedding, k;
50
51          UPDATE dados
52          SET rotulo = v_rotulo_atribuido
53          WHERE id = v_registro.id;
54
55      END LOOP;
56  END;
57  $$ LANGUAGE plpgsql;
```

Fonte: Elaborada pelo autor (2026).

Dentre os algoritmos apresentados na seção 2.6 (Técnicas de Aprendizado de Máquina), o kNN foi o escolhido para a etapa de classificação deste trabalho. Essa escolha se justifica pela sua compatibilidade natural com o funcionamento do PGVector: diferentemente da regressão logística e do random forest, que exigem uma fase de treinamento prévia para o ajuste de coeficientes ou para a construção das árvores de decisão, o kNN realiza a classificação diretamente a partir do cálculo de distância entre vetores — operação já nativamente suportada pela extensão PGVector por meio de seus operadores de similaridade. Dessa forma, foi possível implementar todo o processo de classificação diretamente no SGBDV, sem a necessidade de treinar e exportar um modelo para uma aplicação externa, o que está diretamente alinhado ao objetivo deste trabalho de avaliar o PGVector como um ambiente completo de classificação por similaridade.

5 Resultados

Após a rotulação pelo algoritmo kNN, técnicas como acurácia, precisão, *recall* e *F1-score* foram aplicadas para verificar o desempenho do classificador. Estas técnicas foram escolhidas com base no trabalho de Sathishkumar et al. (2025). As tabelas 12, 13 e 14 trazem os resultados obtidos com essas métricas de avaliação para os modelos SBERT, BERT e FastText, respectivamente. As classes `not_cyberbullying` e `other_cyberbullying` foram abreviadas para `not_cb` e `other_cb` nas tabelas devido ao tamanho destas.

Tabela 12 – Resultados do kNN por valor de k – SBERT

k	**F1-score por classe**						**Macro avg F1**	**Acurácia**
	age	**ethnicity**	**gender**	**religion**	**not_cb**	**other_cb**		
3	0,87	0,88	0,81	0,93	0,47	0,53	0,75	0,78
5	0,89	0,90	0,83	0,93	0,48	0,55	0,76	0,79
7	0,89	0,89	0,83	0,93	0,49	0,56	0,77	0,79
9	0,89	0,89	0,83	0,93	0,49	0,57	0,77	0,79
11	0,89	0,90	0,83	0,93	0,48	0,57	0,77	0,79
15	0,89	0,90	0,83	0,93	0,48	0,57	0,77	0,79
21	0,89	0,90	0,83	0,93	0,48	0,57	0,77	0,79
31	0,89	0,90	0,83	0,93	0,48	0,57	0,77	0,79
51	0,89	0,90	0,83	0,93	0,48	0,57	0,77	0,79

Fonte: Elaborada pelo autor (2026).

Tabela 13 – Resultados do kNN por valor de k – BERT

k	**F1-score por classe**						**Macro avg F1**	**Acurácia**
	age	**ethnicity**	**gender**	**religion**	**not_cb**	**other_cb**		
3	0,75	0,78	0,71	0,86	0,32	0,35	0,63	0,68
5	0,78	0,80	0,72	0,86	0,36	0,37	0,65	0,69
7	0,78	0,80	0,73	0,86	0,37	0,38	0,65	0,70
9	0,78	0,80	0,73	0,85	0,36	0,39	0,65	0,70
11	0,78	0,80	0,73	0,85	0,35	0,39	0,65	0,70
15	0,78	0,80	0,74	0,85	0,35	0,39	0,65	0,70
21	0,78	0,80	0,73	0,85	0,35	0,39	0,65	0,70
31	0,78	0,80	0,73	0,85	0,35	0,39	0,65	0,70
51	0,78	0,80	0,73	0,85	0,35	0,39	0,65	0,70

Fonte: Elaborada pelo autor (2026).

Tabela 14 – Resultados do kNN por valor de k – FastText

k	F1-score por classe						Macro avg F1	Acurácia
	age	ethnicity	gender	religion	not_cb	other_cb		
3	0,70	0,73	0,73	0,84	0,23	0,24	0,58	0,65
5	0,73	0,74	0,73	0,83	0,25	0,25	0,59	0,66
7	0,72	0,75	0,74	0,84	0,24	0,24	0,59	0,66
9	0,72	0,75	0,74	0,84	0,23	0,23	0,59	0,66
11	0,72	0,75	0,74	0,84	0,22	0,24	0,58	0,66
15	0,71	0,75	0,74	0,83	0,22	0,23	0,58	0,66
21	0,71	0,75	0,74	0,83	0,22	0,23	0,58	0,66
31	0,71	0,75	0,74	0,83	0,22	0,23	0,58	0,66
51	0,71	0,75	0,74	0,83	0,22	0,23	0,58	0,66

Fonte: Elaborada pelo autor (2026).

Para a comparação entre os diferentes valores de k e entre os três modelos de *embedding*, adotou-se a *macro F1-score* como principal critério de análise, uma vez que essa métrica pondera igualmente todas as seis classes do problema, evitando que o desempenho das classes majoritárias mascare um desempenho ruim nas classes minoritárias. A acurácia geral do classificador é apresentada de forma complementar para critérios de desempate em *macro F1-score* iguais, juntamente com o *F1-score* por classe.

Analisando os resultados, pode-se notar que o valor ótimo de k se estabiliza cedo e varia pouco em todos os modelos, sendo os melhores valores: $k = 9$ para SBERT, $k = 7$ para BERT e $k = 5$ para FastText. Verificando a *macro F1-score* de cada um desses k, é possível apontar o SBERT como o melhor modelo de geração de *embeddings* na classificação.

O desempenho superior do SBERT em relação aos demais modelos era esperado e pode ser explicado pela abordagem de cada modelo. O FastText, por ser um modelo baseado em subpalavras, gera representações estáticas que não capturam o contexto da sentença como um todo, o que limita a sua capacidade de representar semanticamente *tweets* com estruturas mais complexas ou ambíguas.

O BERT, embora supere o FastText ao gerar vetores dinâmicos sensíveis ao contexto, possui uma limitação arquitetural relevante para o cenário do kNN: os seus *embeddings* produzidos são inadequados para comparações por métricas de distância. Isso compromete diretamente a eficácia do algoritmo do kNN aplicado, uma vez que ele se baseia inteiramente na proximidade no espaço vetorial entre dois *embeddings*.

O SBERT, por sua vez, foi desenvolvido especificamente para resolver essa limitação do BERT. Sendo assim, ele gera *embeddings* que podem ser diretamente comparados pela distância do cosseno de maneira eficiente. Essa propriedade é exatamente o que o kNN vetorial implementado via PGVector explora, tornando assim o SBERT o modelo

ideal e com melhor desempenho.

Em uma perspectiva de *F1-score* específica para cada classe, é possível notar que os rótulos `not_cyberbullying` e `other_cyberbullying` possuem valores muito abaixo da média geral. Isso ocorre pois ambas são "classes de exclusão": a primeira engloba todo conteúdo que não caracteriza *cyberbullying*, enquanto a segunda reúne formas de *cyberbullying* não consideradas pelos demais rótulos. Por essa razão, seus dados tendem a ser semanticamente heterogêneos entre si. Essa heterogeneidade se reflete no espaço vetorial, onde os *embeddings* dessas classes se distribuem de forma dispersa e sem coesão, sem delimitar regiões bem definidas. Como consequência, os vizinhos mais próximos de um ponto dessas classes frequentemente pertencem a outros rótulos, prejudicando diretamente o desempenho do classificador kNN.

Analisando as matrizes de confusão nas figuras 3, 4 e 5, essa heterogeneidade se confirma de forma consistente nos três modelos. No SBERT ($k = 9$), apenas 42% dos exemplos de `not_cyberbullying` foram classificados corretamente, com 26% sendo confundidos com `other_cyberbullying`; este último, por sua vez, apresentou acerto de apenas 54%, com 18% de suas amostras classificadas como `not_cyberbullying`. O padrão se intensifica no BERT ($k = 7$) e no FastText ($k = 5$), nos quais as taxas de acerto dessas duas classes caem para 28% e 29%, e 18% e 18%, respectivamente, evidenciando uma confusão mútua crescente à medida que a qualidade dos *embeddings* diminui.

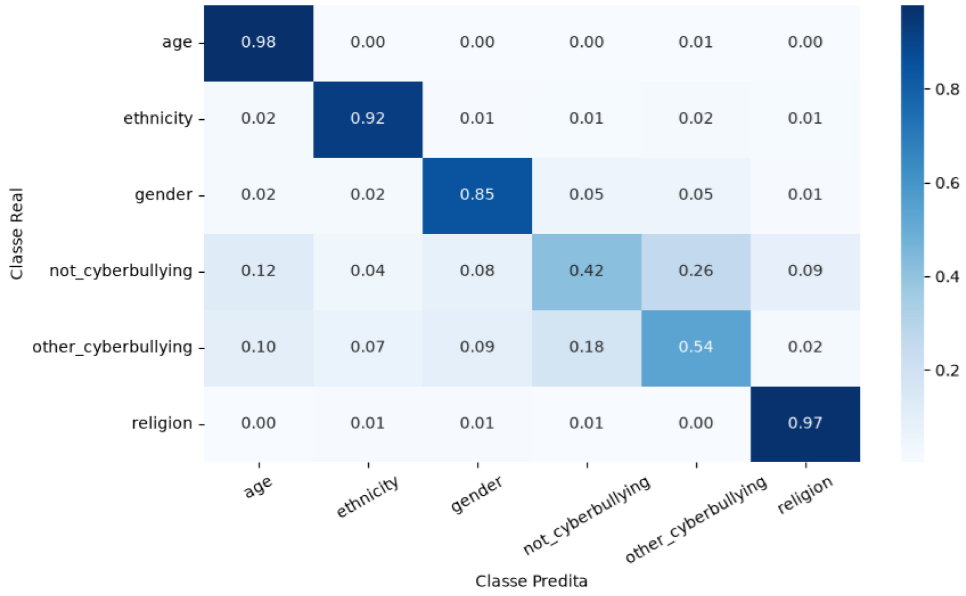

Figura 3 – Matriz de Confusão — SBERT ($k = 9$).

Fonte: Elaborada pelo autor (2026).

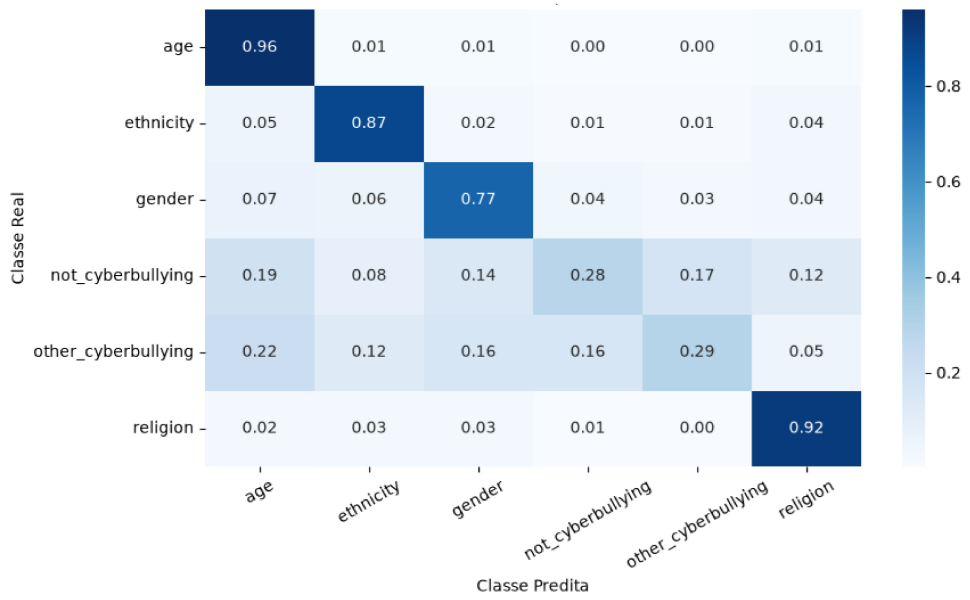

Figura 4 – Matriz de Confusão — BERT ($k = 7$).

Fonte: Elaborada pelo autor (2026).

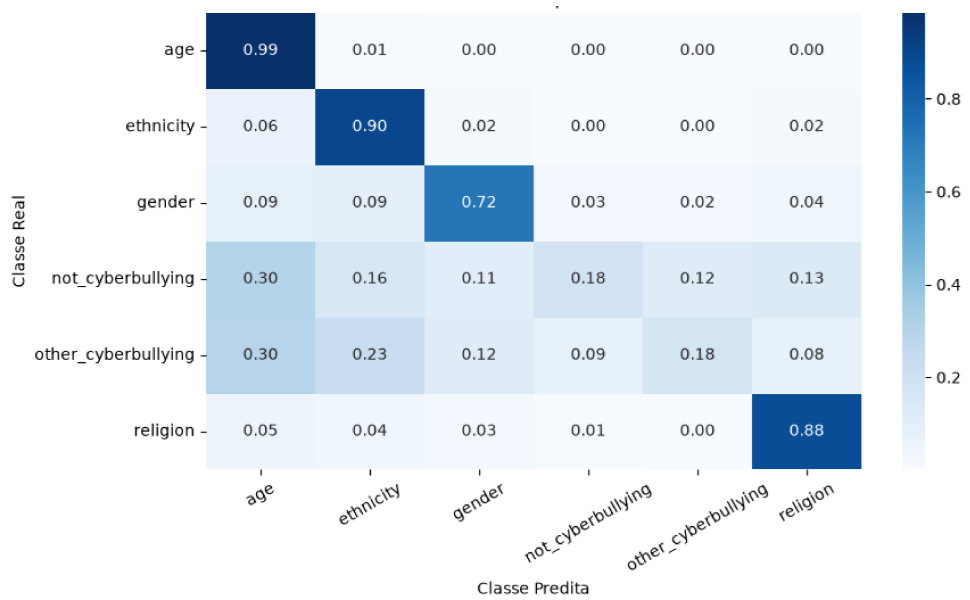

Figura 5 – Matriz de Confusão — FastText ($k = 5$).

Fonte: Elaborada pelo autor (2026).

Essa dispersão semântica é visualmente confirmada pela projeção UMAP nas figuras 6, 7 e 8: enquanto classes como `age`, `religion` e `gender` formam agrupamentos relativamente coesos no espaço vetorial — especialmente nos *embeddings* SBERT —, os pontos de `not_cyberbullying` e `other_cyberbullying` se distribuem de forma difusa na região central do espaço, sem separação clara entre elas. Tal comportamento sugere que essas duas categorias não possuem representações semânticas suficientemente distin-

tas nos *embeddings* gerados, o que representa uma limitação para o classificador kNN, independentemente do valor adotado para k.

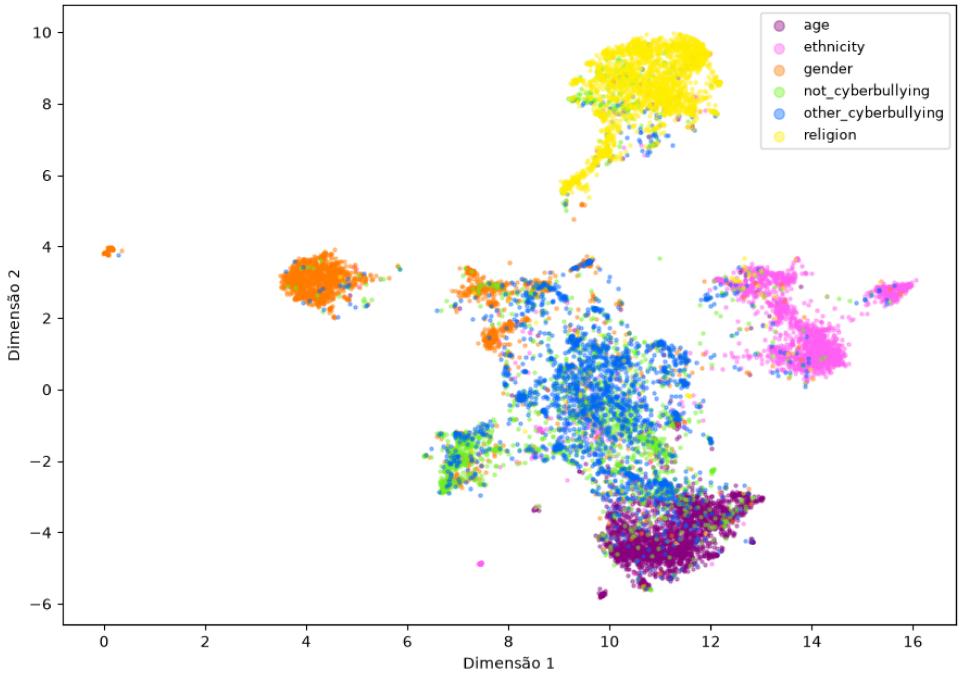

Figura 6 – UMAP — SBERT ($k = 9$).

Fonte: Elaborada pelo autor (2026).

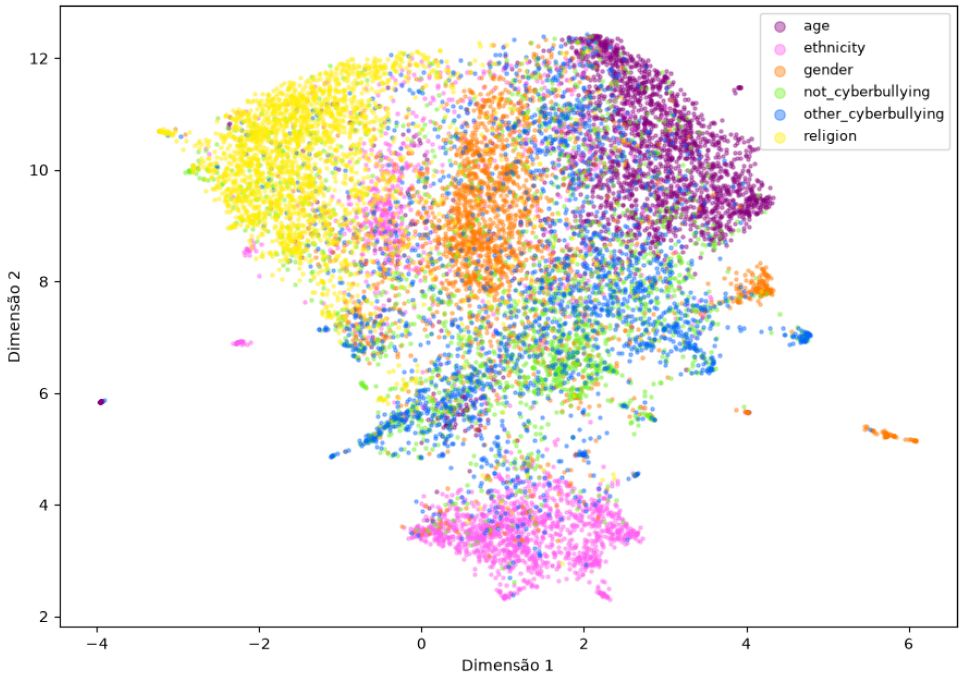

Figura 7 – UMAP — BERT ($k = 7$).

Fonte: Elaborada pelo autor (2026).

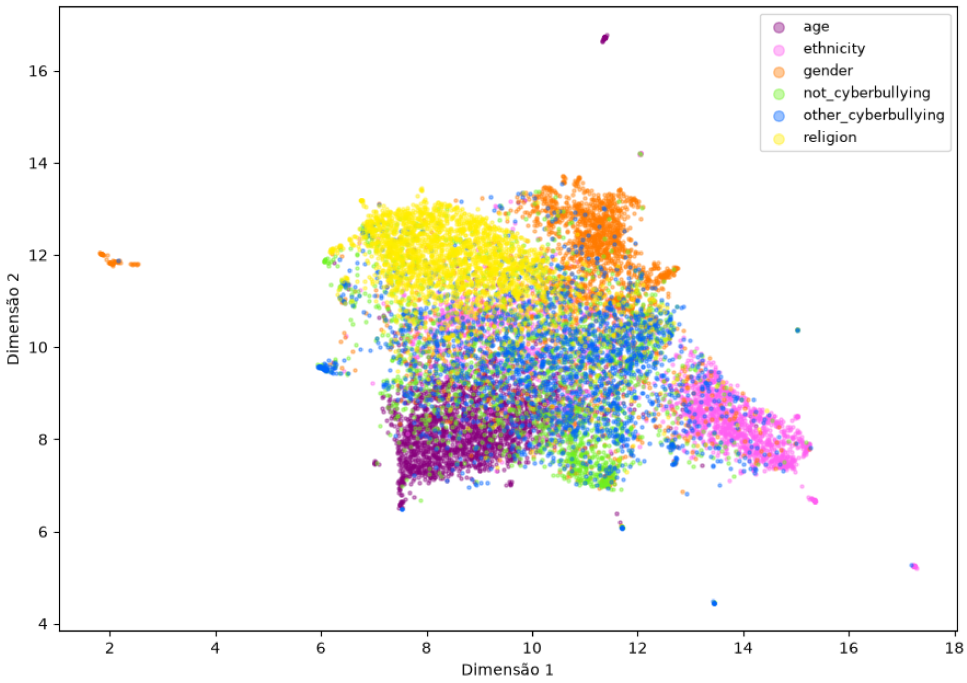

Figura 8 – UMAP — FastText ($k = 5$).

Fonte: Elaborada pelo autor (2026).

6 Conclusão

Os resultados obtidos demonstraram que é viável utilizar o PostgreSQL com a extensão PGVector para a detecção de *cyberbullying* por meio de busca por similaridade vetorial, validando a proposta deste trabalho. Dentre os três modelos de representação vetorial avaliados, o SBERT apresentou o melhor desempenho ($k = 9$, *f1-score* de 0,77), superando o BERT ($k = 7$, *f1-score* de 0,65) e o FastText ($k = 5$, *f1-score* de 0,59). Esse resultado confirma o que é discutido na literatura: *embeddings* projetados especificamente para comparação por similaridade — como é o caso do SBERT — são mais adequados a algoritmos baseados em distância vetorial, como o kNN, do que *embeddings* de "propósito geral", como o BERT, cuja arquitetura não foi originalmente projetada para esse tipo de comparação.

Por outro lado, a análise das matrizes de confusão e dos UMAPs revelou uma limitação consistente nos três modelos de *embedding*: as classes `not_cyberbullying` e `other_cyberbullying`, por serem definidas por exclusão, apresentaram alta heterogeneidade semântica, o que dispersou seus *embeddings* no espaço vetorial e prejudicou o desempenho do classificador especificamente para esses rótulos. Esse achado indica que a qualidade da classificação por similaridade depende diretamente da coesão semântica das classes envolvidas, e não apenas da qualidade do modelo de *embedding* utilizado.

Nesse sentido, a principal contribuição deste trabalho[1] foi a demonstração prática de que o PostgreSQL, por meio da extensão PGVector, é capaz de atuar não apenas como um repositório de *embeddings*, mas também como um ambiente completo de classificação por similaridade. Isso dispensa a necessidade de treinamento prévio de um modelo supervisionado tradicional, já que o próprio kNN realiza a classificação diretamente a partir dos dados armazenados. Além disso, a comparação entre os três modelos de *embedding* oferece um panorama prático de suas particularidades no contexto específico de textos curtos e informais, como os encontrados em redes sociais.

Destaca-se como sugestão para trabalhos futuros investigar estratégias para lidar com a ausência de padrões semânticos das classes `not_cyberbullying` e também `other_cyberbullying`, uma vez que essa limitação se mostrou consistente nos três modelos de *embedding* avaliados. Um melhor tratamento dessas duas classes poderia contribuir para classificações mais precisas, capazes de distinguir melhor o que de fato caracteriza *cyberbullying* e o que não se enquadra nessa definição.

Sugere-se também investigar o impacto da dimensionalidade dos vetores de *embedding* no desempenho da classificação. Os três modelos avaliados neste trabalho geram

[1] Código-fonte deste trabalho: https://github.com/mmesqigor/deteccao-cyberbullying-pgvector

vetores de dimensões distintas — 384 para o SBERT, 768 para o BERT e 300 para o FastText —, e uma análise controlada dessa variável poderia esclarecer melhor sua relação com o *F1-score* na tarefa de classificação.

Outra sugestão seria adotar uma divisão dos dados em três conjuntos — treino, validação e teste —, em vez da divisão binária utilizada neste trabalho. Um conjunto de validação distinto permitiria ajustar o valor de k de forma mais criteriosa, sem o risco de essa escolha ser influenciada pelos próprios dados de teste, tornando a avaliação final do classificador mais robusta.

Sugere-se ainda repetir o experimento com diferentes valores de semente para a amostragem estratificada. Como a semente utilizada neste trabalho foi fixada em um único valor ($random\_state = 42$), não é possível afirmar se os resultados obtidos se mantêm estáveis independentemente da composição específica da amostra sorteada; variar a semente permitiria estimar a robustez do método proposto frente a diferentes divisões dos dados.

Por fim, sugere-se a realização de uma análise estatística formal sobre as diferenças de desempenho observadas entre os modelos de embedding e entre os valores de k testados. Neste trabalho, a comparação entre SBERT, BERT e FastText baseou-se apenas na observação direta dos valores de *macro F1-score* e acurácia, sem testes que verifiquem se essas diferenças são estatisticamente significativas ou atribuíveis ao acaso — a aplicação desses testes forneceria maior rigor metodológico às conclusões apresentadas.

Este trabalho contou com contribuições de diversas disciplinas do curso de Sistemas de Informação. Lógica para Computação; Algoritmos e Programação I e II; e Estrutura de Dados I forneceram a base algorítmica e de raciocínio formal. Álgebra Linear; e Matemática para Ciência da Computação sustentaram os fundamentos matemáticos da representação vetorial e da distância cosseno. Banco de Dados I e II; Organização e Recuperação da Informação; e Ciência de Dados I e II embasaram diretamente a metodologia empregada, do PLN à avaliação final.

Referências

AGRAWAL, P.; KUMAR, A.; TRIPATHI, A. K. ENHANCING CYBERBULLYING DETECTION USING ENSEMBLE LEARNING AND EMBEDDINGS. **ShodhKosh: Journal of Visual and Performing Arts**, v. 5, p. 1322–1331, 2024. ISSN 2582-7472. Disponível em: <https://doi.org/10.29121/shodhkosh.v5.i1.2024.3194>. Citado na página 27.

BENGFORT, B.; BILBRO, R.; OJEDA, T. **Applied text analysis with Python: enabling language–aware data products with machine learning**. Sebastopol: O'Reilly, 2018. Disponível em: <https://books.google.com.br/books?hl=pt-BR&lr=&id=lK1fDwAAQBAJ&oi=fnd&pg=PT15&dq=BENGFORT,+B.%3B+BILBRO,+R.%3B+OJEDA,+T.+Applied+text+analysis+with+Python:+enabling+language--aware+data+products+with+machine+learning&ots=g5K9imLfws&sig=nEbp7_F5CjplVUehHvINMMLctxo#v=onepage&q&f=false>. Acesso em: 2026-03-16. Citado na página 19.

BHARTI, S.; YADAV, A. K.; KUMAR, M.; YADAV, D. Cyberbullying detection from tweets using deep learning. **Emerald Publishing Limited**, v. 51, p. 2695–2711., 2022. Disponível em: <https://doi.org/10.1108/K-01-2021-0061>. Citado na página 27.

CORSI, A. L. M. **Estudo de Caso de um Sistema de Gerenciamento de Banco de Dados Vetorial para Manipulação de Dados de Redes Sociais Online**. Trabalho de Conclusão de Curso, 2025. Disponível em: <https://repositorio.ufu.br/handle/123456789/47477>. Acesso em: 2026-03-17. Citado 2 vezes nas páginas 27 e 28.

CUNNINGHAM, P.; DELANY, S. J. k-Nearest Neighbour Classifiers - A Tutorial. **ACM Computing Surveys**, v. 54, p. 1–25, 2021. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/3459665>. Citado na página 39.

DEVLIN, J.; CHANG, M.-W.; LEE, K.; TOUTANOVA, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. **arXiv**, v. 2, 2019. ISSN 2331-8422. Disponível em: <https://doi.org/10.48550/arXiv.1810.04805>. Citado na página 19.

GHOJOGH, B.; GHODSI, A.; KARRAY, F.; CROWLEY, M. Uniform Manifold Approximation and Projection (UMAP) and its Variants: Tutorial and Survey. **arXiv preprint arXiv:2109.02508**, p. 1–11, 2021. Disponível em: <https://arxiv.org/abs/2109.02508>. Citado na página 25.

GOLDBERG, Y.; LEVY, O. word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method. **ArXiv**, v. 1, 2014. ISSN 2331-8422. Disponível em: <https://doi.org/10.48550/arXiv.1402.3722>. Citado na página 19.

GROOVER, S.; RAJU, V. V. Cyberbullying: A narrative review. **Journal of Mental Health and Human Behaviour**, v. 28, p. 17–26, 2023. ISSN 2543-1897. Disponível em: <https://doi.org/10.4103/jmhhb.jmhhb_47_22>. Citado na página 17.

GUTIÉRREZ-BATISTA, K.; GÓMEZ-SÁNCHEZ, J.; FERNANDEZ-BASSO, C. Improving automatic cyberbullying detection in social network environments by fine-tuning a pre-trained sentence transformer language model. **Social Network Analysis and Mining**, v. 14, p. 136, 2024. Disponível em: <https://link-springer-com.ez34.periodicos.capes.gov.br/article/10.1007/s13278-024-01291-0#citeas>. Citado na página 15.

HOFF, D. L.; MITCHELL, S. N. Cyberbullying: causes, effects, and remedies. **Journal of Educational Administration**, v. 47, p. 652–665, 2009. Disponível em: <https://www-emerald-com.ez34.periodicos.capes.gov.br/jea/article/47/5/652/203351/Cyberbullying-causes-effects-and-remedies>. Citado na página 15.

JIANG, T.; GRADUS, J. L.; ROSELLINI, A. J. Supervised Machine Learning: A Brief Primer. **Behavior Therapy**, v. 51, p. 675–687, 2020. ISSN 0005-7894. Disponível em: <https://doi.org/10.1016/j.beth.2020.05.002>. Citado na página 22.

JURAFSKY, D.; MARTIN, J. H. **Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition**. Upper Saddle River: Prentice Hall, 2009. Disponível em: <https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf>. Acesso em: 2026-03-16. Citado 2 vezes nas páginas 18 e 19.

KANE, A. **pgvector**. 2021. Disponível em: <https://github.com/pgvector/pgvector>. Acesso em: 2026-07-04. Citado na página 22.

KOWALSKI, R. M.; GIUMETTI, G. W.; SCHROEDER, A. N.; LATTANNER, M. R. Bullying in the Digital Age: A Critical Review and Meta-Analysis of Cyberbullying Research Among Youth. **American Psychological Association**, v. 140, p. 1073–1137, 2014. Disponível em: <https://psycnet-apa-org.ez34.periodicos.capes.gov.br/fulltext/2014-04307-001.html>. Citado 2 vezes nas páginas 15 e 17.

LARXEL. **Cyberbullying Classification**. 2022. Disponível em: <https://www.kaggle.com/datasets/andrewmvd/cyberbullying-classification>. Acesso em: 2026-03-08. Citado na página 29.

LEWIS, P.; PEREZ, E.; PIKTUS, A.; PETRONI, F.; KARPUKHIN, V.; GOYAL, N.; KüTTLER, H.; LEWIS, M.; YIH, W.; ROCKTäSCHEL, T.; RIEDEL, S.; KIELA, D. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. **Curran Associates, Inc.**, v. 33, p. 9459–9474, 2020. Disponível em: <https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf>. Citado na página 15.

LI, J.; TANG, C.; LEI, Z.; ZHANG, Y.; LI, X.; YU, Y.; PI, R.; HU, L. KRA: K-Nearest Neighbor Retrieval Augmented Model for Text Classification. **eletronics**, v. 13, p. 3237, 2024. ISSN 2079-9292. Disponível em: <https://doi.org/10.3390/electronics13163237>. Citado na página 27.

MAHMUD, M. S.; HUANG, J. Z.; SALLOUM, S.; EMARA, T. Z.; SADATDIYNOV, K. A Survey of Data Partitioning and Sampling Methods to Support Big Data Analysis. **Big Data Mining and Analytics**, v. 3, p. 85–101, 2020. ISSN 2096-0654. Disponível em: <https://doi.org/10.26599/BDMA.2019.9020015>. Citado na página 23.

MARTINS, J. S.; LENZ, M. L.; SILVA, M. B. F. D. **Processamentos de Linguagem Natural**. SAGAH, 2020. Disponível em: <https://integrada.minhabiblioteca.com.br/reader/books/9786556900575>. Citado 3 vezes nas páginas 18, 19 e 22.

MAURYA, C.; MUHAMMAD, T.; DHILLON, P.; MAURYA, P. The effects of cyberbullying victimization on depression and suicidal ideation among adolescents and young adults: a three year cohort study from India. **BMC Psychiatry**, v. 22, p. 599, 2022. ISSN 1471-244X. Disponível em: <https://bmcpsychiatry-biomedcentral-com.ez34.periodicos.capes.gov.br/articles/10.1186/s12888-022-04238-x#citeas>. Citado na página 15.

MIKOLOV, T.; CHEN, K.; CORRADO, G.; DEAN, J. Efficient Estimation of Word Representations in Vector Space. v. 3, 2013a. Disponível em: <https://arxiv.org/abs/1301.3781v3>. Citado na página 15.

MIKOLOV, T.; SUTSKEVER, I.; CHEN, K.; CORRADO, G.; DEAN, J. Distributed Representations of Words and Phrases and their Compositionality. **ArXiv**, v. 1, 2013b. ISSN 2331-8422. Disponível em: <https://doi.org/10.48550/arXiv.1310.4546>. Citado na página 19.

Núcleo de Informação e Coordenação do Ponto BR. **Pesquisa sobre o uso da Internet por crianças e adolescentes no Brasil : TIC Kids Online Brasil 2024**. Comitê Gestor da Internet no Brasil, 2025. ISBN 978-65-85417-80-8. Disponível em: <https://cetic.br/media/docs/publicacoes/2/20250512154312/tic_kids_online_2024_livro_eletronico.pdf>. Citado na página 17.

OBI, J. C. A comparative study of several classification metrics and their performances on data. **World Journal of Advanced Engineering Technology and Sciences**, v. 08, p. 308–314, 2023. ISSN 2582-8266. Disponível em: <https://doi.org/10.30574/wjaets.2023.8.1.0054>. Citado na página 24.

PAN, J. J.; WANG, J.; LI, G. Survey of Vector Database Management Systems. **ArXiv**, v. 1, 2023. ISSN 2331-8422. Disponível em: <https://doi.org/10.48550/arXiv.2310.14021>. Citado 6 vezes nas páginas 18, 20, 21, 22, 26 e 39.

REIMERS, N.; GUREVYCH, I. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. **arXiv**, v. 1, 2019. ISSN 2331-8422. Disponível em: <https://doi.org/10.48550/arXiv.1908.10084>. Citado 2 vezes nas páginas 20 e 26.

RIBEIRO, P.; PIRES J. BASTOS, P.; RIGO, R.; REIS, H.; HUBNER, K.; M., C.; MARAFIGA, B.; SIMBA, D.; TSUNODA, D. Vector Databases and Embedding Models: Comparative evaluation of performance in semantic retrieval in Portuguese. **Research, Society and Development**, v. 14, p. 10, 2025. Disponível em: <https://rsdjournal.org/rsd/article/view/49768>. Citado na página 15.

SALMAN, H. A.; KALAKECH, A.; STEITI, A. Random Forest Algorithm Overview. **Babylonian Journal of Machine Learning**, v. 2024, p. 69–79, 2024. ISSN 3006–5429. Disponível em: <https://doi.org/10.58496/BJML/2024/007>. Citado na página 23.

SATHISHKUMAR, J.; VASUNDARA, S.; POOJA, S. S.; CHINNAMANI, R. Cyberbullying Detection System Using Natural Language Processing and Machine Learning. **INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN**

ENGINEERING AND MANAGEMENT (IJSREM), v. 9, 2025. ISSN 2582-3930. Disponível em: <https://doi.org/10.55041/IJSREM52831>. Citado 2 vezes nas páginas 28 e 43.

SPERANDEI, S. Understanding logistic regression analysis. **Biochem Medica**, v. 24, p. 12–18, 2014. ISSN 1846-7482. Disponível em: <http://dx.doi.org/10.11613/BM.2014.003>. Citado na página 23.

SUBRAMANIAN, D. K. R. Influence of Social Media in Interpersonal Communication. **INTERNATIONAL JOURNAL OF SCIENTIFIC PROGRESS AND RESEARCH**, v. 38, p. 70–75, 2017. ISSN 2349-4689. Disponível em: <https://www.researchgate.net/profile/Kalpathy-Subramanian/publication/319422885_Influence_of_Social_Media_in_Interpersonal_Communication/links/59a96d950f7e9b2790120fea/Influence-of-Social-Media-in-Interpersonal-Communication.pdf>. Citado na página 15.

SYRIOPOULOS, P. K.; KALAMPALIKIS, N. G.; KOTSIANTIS, S. B.; VRAHATIS, M. N. kNN Classification: a review. **Annals of Mathematics and Artificial Intelligence**, v. 93, p. 43–75, 2025. ISSN 1573-7470. Disponível em: <https://doi.org/10.1007/s10472-023-09882-x>. Citado 2 vezes nas páginas 23 e 39.

TENG, T. H.; VARATHAN, K. D. Cyberbullying Detection in Social Networks: A Comparison Between Machine Learning and Transfer Learning Approaches. **IEEE Access**, v. 11, p. 55533–55560, 2023. ISSN 2169-3536. Disponível em: <https://doi.org/10.1109/ACCESS.2023.3275130>. Citado 3 vezes nas páginas 26, 30 e 36.

WANG, B.; WANG, A.; CHEN, F.; WANG, Y.; KUO, C. C. J. Evaluating word embedding models: methods and experimental results. **APSIPA Transactions on Signal and Information Processing**, v. 8, 2019. ISSN 2048-7703. Disponível em: <https://doi.org/10.1017/ATSIP.2019.12>. Citado na página 19.

WANG, Z.; YAO, D.; LI, B.; MA, D.; LI, B.; LI, Z.; XIONG, F.; CUI, B.; TANG, L.; ZHANG, W. Text2VectorSQL: Towards a Unified Interface for Vector Search and SQL Queries. **Cornell University**, v. 2, 2025. Disponível em: <https://arxiv.org/abs/2506.23071>. Citado na página 15.

WARD, J. MemoriesDB: A Temporal-Semantic-Relational Database for Long-Term Agent Memory. **Cornell University**, 2025. Disponível em: <https://arxiv.org/abs/2511.06179>. Citado na página 15.

WAZLAWICK, R. S. **Metodologia de Pesquisa para Ciência da Computação. 3. ed.** Rio de Janeiro: GEN LTC, 2020. v. 3. Citado na página 16.

ZHAO, R.; MAO, K. Cyberbullying Detection Based on Semantic-Enhanced Marginalized Denoising Auto-Encoder. **IEEE Transactions on Affective Computing**, v. 8, p. 328–339, 2017. Disponível em: <https://ieeexplore-ieee-org.ez34.periodicos.capes.gov.br/document/7412690>. Citado na página 15.