UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Ana Luísa Matias Corsi

# Estudo de Caso de um Sistema de Gerenciamento de Banco de Dados Vetorial para Manipulação de Dados de Redes Sociais Online

**Uberlândia, Brasil**

**2025**

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Ana Luísa Matias Corsi

# 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 apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção título de Bacharel em Ciência da Computação.

Orientador: Maria Camila Nardini Barioni

Universidade Federal de Uberlândia – UFU

Faculdade de Computação

Bacharelado em Ciência da Computação

Uberlândia, Brasil

2025

**UNIVERSIDADE FEDERAL DE UBERLÂNDIA**
Faculdade de Computação
Av. João Naves de Ávila, nº 2121, Bloco 1A - Bairro Santa Mônica, Uberlândia-MG, CEP 38400-902
Telefone: (34) 3239-4144 - http://www.portal.facom.ufu.br/ facom@ufu.br

# ATA DE DEFESA - GRADUAÇÃO

| | | | | | |
|---|---|---|---|---|---|
| Curso de Graduação em: | Bacharelado em Ciência da Computação | | | | |
| Defesa de: | Projeto de Graduação 2 - GBC082 | | | | |
| Data: | 09/10/2025 | Hora de início: | 10:30 | Hora de encerramento: | 11:20 |
| Matrícula do Discente: | 12121BCC004 | | | | |
| Nome do Discente: | Ana Luísa Matias Corsi | | | | |
| Título do Trabalho: | Estudo de caso de um Sistema de Gerenciamento de Banco de Dados Vetorial para manipulação de dados de Redes Sociais Online | | | | |
| A carga horária curricular foi cumprida integralmente? | ( X ) Sim ( ) Não | | | | |

Reuniu-se de forma online pela plataforma MS Teams, a Banca Examinadora, designada pelo Colegiado do Curso de Graduação em Bacharelado em Ciência da Computação, assim composta: Professores: Humberto Luiz Razente FACOM/UFU, Maria Adriana Vidigal de Lima FACOM/UFU e Maria Camila Nardini Barioni FACOM/UFU orientadora da candidata.

Iniciando os trabalhos, a presidente da mesa, Dra. Maria Camila Nardini Barioni, apresentou a Comissão Examinadora e a candidata, agradeceu a presença do público, e concedeu ao discente a palavra, para a exposição do seu trabalho. A duração da apresentação da 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 a candidata. Ultimada a arguição, que se desenvolveu dentro dos termos regimentais, a Banca, em sessão secreta, atribuiu o resultado final, considerando a candidata:

( X ) Aprovada Nota [100]

OU

( ) Aprovada 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 09/10/2025, às 11:22, 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 **Maria Adriana Vidigal de Lima**, **Professor(a) do Magistério Superior**, em 09/10/2025, às 11:22, 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 **Humberto Luiz Razente**, **Professor(a) do Magistério Superior**, em 09/10/2025, às 11:23, 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 **6749751** e o código CRC **6BDED966**.

# Agradecimentos

Dedico esta conquista, primeiramente, aos meus pais. Mais do que um porto seguro, vocês foram a inspiração que guiou meus passos. Reviver parte de suas histórias nesta mesma Universidade, sendo aluna de mestres que um dia foram os seus, conferiu a esta jornada um significado profundo e simbólico. Sou imensamente grata por todo o sacrifício, pelo incentivo incondicional e pela confiança inabalável em meu potencial.

À minha irmã, minha eterna gratidão por sua presença constante, seu apoio e por ser minha maior torcedora.

Aos meus avós, agradeço pelo carinho, pelo suporte e pelo interesse genuíno que sempre demonstraram em cada etapa dos meus estudos.

Aos amigos que a universidade me deu, companheiros de jornada, em especial à Laylla Royer, agradeço pela parceria nos desafios, pelas risadas que aliviaram a pressão e pelo conhecimento que construímos juntos.

Finalmente, meu reconhecimento a todos os professores que contribuíram para a minha formação. Em especial, à Profa. Dra. Maria Camila Nardini, agradeço pela orientação precisa, pela paciência e pelos ensinamentos.

# Resumo

O crescente volume de dados não estruturados, impulsionado por plataformas de redes sociais online, apresenta desafios para os Sistemas de Gerenciamento de Bancos de Dados relacionais tradicionais, limitados a buscas lexicais. SGBDs vetoriais surgem como solução, porém introduzem complexidade arquitetônica. Uma alternativa é a integração de capacidades vetoriais em SGBDs existentes. Este trabalho tem como objetivo avaliar a viabilidade da extensão PGVector do PostgreSQL como solução integrada para a análise de dados não estruturados. Para isso, foi desenvolvido um estudo de caso que utiliza um conjunto de dados de *tweets* da plataforma X (antigo Twitter). O método de trabalho abrange o pré-processamento dos textos e a geração de representações vetoriais (*embeddings*) por meio do modelo "paraphrase-multilingual-MiniLM-L12-v2" da biblioteca sentence-transformers. Os *embeddings* gerados foram armazenados e indexados no PostgreSQL utilizando o PGVector. Subsequentemente, foi aplicado o algoritmo de clusterização K-means para a descoberta de tópicos de forma não supervisionada, com o número ideal de clusters (K) sendo determinado pelo Método do Cotovelo. Os resultados demonstram a aplicabilidade da abordagem integrada, permitindo a execução de um *pipeline* completo de *Machine Learning* dentro do ecossistema relacional. O trabalho contribui ao apresentar um roteiro prático e validado, servindo como um recurso para futuras aplicações na área.

**Palavras-chave**: Sistema de Gerenciamento de Banco de Dados Vetorial, PGVector, *Embeddings*, K-Means, Dados de Redes Sociais Online.

# Abstract

The growing volume of unstructured data, driven by social media platforms, poses challenges for traditional relational Database Management Systems (DBMS), which are limited to lexical searches. Vector DBMSs emerge as a solution, yet they introduce architectural complexity. An alternative is the integration of vector capabilities into existing DBMSs. This work aims to evaluate the viability of the PostgreSQL extension PGVector as an integrated solution for the analysis of unstructured data. To this end, a case study was developed using a dataset of tweets from the X platform (formerly Twitter). The workflow covers text preprocessing and the generation of vector representations (embeddings) using the "paraphrase-multilingual-MiniLM-L12-v2" model from the sentence-transformers library. The generated embeddings were stored and indexed in PostgreSQL using PGVector. Subsequently, the K-means clustering algorithm was applied for unsupervised topic discovery, with the optimal number of clusters (K) determined by the Elbow Method. The results demonstrate the applicability of the integrated approach, enabling the execution of a complete Machine Learning pipeline within the relational ecosystem. This work contributes by presenting a practical and validated roadmap, serving as a resource for future applications in the field.

**Keywords**: Vector Database Management System, PGVector, Embeddings, K-Means, Online Social Media Data.

# Lista de ilustrações

# Lista de tabelas

# Lista de abreviaturas e siglas

| | |
|---|---|
| PLN | Processamento de Linguagem Natural |
| SGBD | Sistema de Gerenciamento de Banco de Dados |
| IoT | Internet das Coisas |
| WCSS | Soma dos Quadrados Intra-clusters |
| GPT | *Generative Pre-trained Transformer* |
| BERT | *Bidirectional Encoder Representations from Transformers* |
| TCC | Trabalho de Conclusão de Curso |

# Sumário

# 1 Introdução

Segundo a 12ª edição do relatório *Data Never Sleeps* (DOMO, 2024), a cada um minuto, 41 milhões de novas mensagens são trocadas pelo WhatsApp e 138.9 milhões de reels são produzidos no Instagram e Facebook, o que ressalta a quantidade elevada de dados gerados nas redes sociais online atualmente. Nesse cenário, o Processamento de Linguagem Natural tornou-se fundamental para diversas aplicações, incluindo busca de informações, tradução automática e análise de sentimentos (CAMBRIA; WHITE, 2014). Contudo, *"Dados são simplesmente números, e números por si mesmos não têm significado até que sejam organizados e interpretados"* (CLEVELAND, 1985). Diante disso, torna-se essencial, além de buscar a melhor forma de analisá-los, refletir sobre a melhor maneira de armazená-los, a fim de facilitar o seu processamento e interpretação.

Nesse sentido, os Sistemas de Gerenciamento de Bancos de Dados (SGBDs) vetoriais são considerados grandes aliados, uma vez que fornecem recursos tradicionais, como buscas otimizadas, transações, escalabilidade, tolerância a falhas, privacidade, e segurança, mas para dados não estruturados (PAN; WANG; LI, 2024). Contudo, a adoção de um sistema inteiramente novo representa uma mudança significativa para muitas organizações que já trabalham com ecossistemas consolidados em torno de uma estrutura relacional. Diante dessa problemática, surge a seguinte questão de pesquisa: É viável integrar capacidades de busca vetorial diretamente em um SGBD relacional, utilizando extensões como o **PGVector**, como alternativa à utilização de um SGBD vetorial dedicado?

A relevância deste estudo fundamenta-se na crescente demanda por arquiteturas de software que sejam ao mesmo tempo eficientes e otimizadas. Conforme a necessidade de analisar dados não estruturados aumenta, reforça-se a importância da utilização de ferramentas específicas para o armazenamento deles, uma vez que SGBDs tradicionais não são adequados para essa tarefa, o que é ressaltado no artigo de Mihai (2020). Embora a adoção de soluções vetoriais dedicadas seja uma tendência, ela introduz custos operacionais e complexidade na manutenção e sincronização de múltiplos sistemas, como é visto em Alake (2025). Portanto, este trabalho justifica-se ao investigar a viabilidade do **PGVector**, uma solução que busca integrar a computação vetorial ao ecossistema relacional já consolidado, representando uma alternativa com potencial para simplificar a infraestrutura tecnológica e reduzir custos para inúmeras organizações.

Em conformidade com o que foi exposto, considera-se o objetivo geral do trabalho: avaliar a viabilidade da extensão **PGVector** como ferramenta para computação vetorial no SGBD PostgreSQL por meio de um estudo de caso. Para alcançar este propósito, os seguintes objetivos específicos foram definidos:

1. Realizar uma revisão bibliográfica sobre os conceitos de SGBDs vetoriais, vetorização de dados, arquitetura da extensão PGVector, Processamento de Linguagem Natural e algoritmos de *Machine Learning*;

2. Desenvolver um protótipo para o pré-processamento, vetorização e persistência de um conjunto de dados textuais em um SGBD relacional;

3. Demonstrar a funcionalidade das buscas por similaridade utilizando os diferentes tipos de índices oferecidos pelo PGVector;

4. Implementar e executar o algoritmo de clusterização K-Means diretamente no SGBD como um estudo de caso para a descoberta de tópicos;

5. Analisar os resultados obtidos, discutindo as vantagens, limitações e implicações arquitetônicas do uso do PGVector em um ambiente relacional.

Para atingir os objetivos propostos, o método de trabalho adotado combina uma revisão bibliográfica com uma abordagem de pesquisa aplicada, materializada por meio de um estudo de caso. A fase prática consiste no desenvolvimento de um protótipo que utiliza a linguagem Python para o pré-processamento dos dados e a extensão PGVector no SGBD PostgreSQL para o armazenamento e a execução da análise vetorial. O objetivo desta implementação é validar a viabilidade de se conduzir um fluxo de *Machine Learning* diretamente no SGBD, com os resultados da clusterização sendo avaliados de forma qualitativa para verificar a coerência dos tópicos gerados.

O presente trabalho está estruturado em cinco capítulos, além desta introdução. O segundo capítulo apresenta a fundamentação teórica, abordando os conceitos de Processamento de Linguagem Natural, SGBDs vetoriais e a extensão PGVector. O terceiro capítulo dedica-se à análise dos trabalhos relacionados, contextualizando a pesquisa frente ao estado da arte. Em seguida, o quarto capítulo detalha o método e as ferramentas utilizadas na parte prática. O quinto capítulo contempla o desenvolvimento do estudo de caso em conjunto à apresentação e análise dos resultados obtidos. Por fim, no sexto capítulo, são apresentadas as considerações finais, as limitações do estudo e sugestões para trabalhos futuros.

# 2 Fundamentação Teórica

Este capítulo apresenta a fundamentação teórica que sustenta este trabalho. O objetivo é aprofundar os conceitos essenciais, analisar os avanços no estado da arte e identificar as lacunas na literatura que justificam esta pesquisa. Com base em fontes atuais e relevantes, busca-se construir um referencial sólido e abrangente que servirá de alicerce para a fase prática e experimental do estudo conduzido no Trabalho de Conclusão de Curso (TCC).

A exposição parte do desafio geral do armazenamento e análise de dados não estruturados, aprofundando-se nos conceitos de busca por similaridade e na tecnologia de *embeddings*. Em seguida, investiga-se o estado da arte das tecnologias de SGBDs para vetores, contrastando soluções dedicadas com a abordagem de extensões em SGBDs relacionais, como o PGVector. Por fim, são apresentados os fundamentos do algoritmo de clusterização K-means.

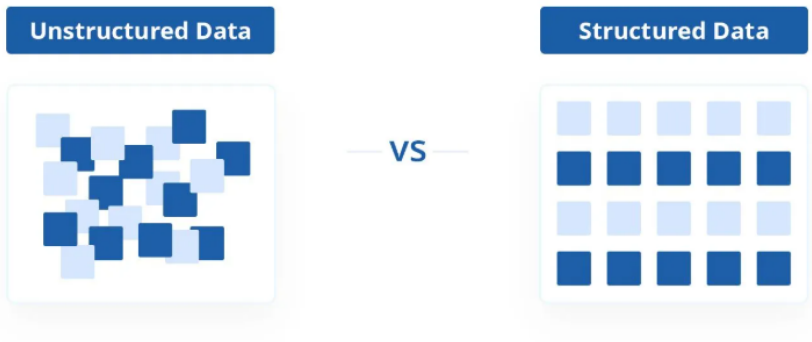
## 2.1 O Desafio dos Dados Não Estruturados

Para compreender a problemática abordada neste trabalho, é fundamental distinguir os dois principais tipos de dados: estruturados e não estruturados, o que é ilustrado na Figura 1. Os dados estruturados são caracterizados por possuírem um modelo de dados rígido e predefinido, sendo comumente organizados em formatos tabulares como linhas e colunas, o que facilita seu armazenamento e consulta. Em contrapartida, os dados não estruturados, foco deste TCC, carecem dessa organização formal (JONKER; GOMSTYN, 2025). Eles não seguem um esquema predefinido, resultando em uma natureza heterogênea e flexível. Alguns exemplos comuns incluem:

- E-mails;
- Postagens em redes sociais online;
- Imagens, vídeos e áudios;
- Mensagens de texto;
- Dados de sensores da Internet das Coisas (IoT).

Uma análise rápida revela uma demanda crescente por soluções capazes de armazenar e manipular essas informações. Tal cenário é impulsionado por dois fatores principais:

Figura 1 – Ilustração Dados Não Estruturados X Dados Estruturados




*Fonte:(KHAN, 2024)*

o aumento exponencial na geração de dados não estruturados e o rápido avanço de modelos de *Machine Learning* que dependem desses dados para oferecer soluções mais eficientes e precisas (WANG et al., 2021).

No primeiro caso, com o advento da Internet, potencializado pela utilização em massa de redes sociais online e *IoT*, ocorreu um aumento radical da produção e compartilhamento de mensagens em linguagem natural, fotos e vídeos. O crescimento é tão significativo que o mercado global de *Big Data*, que lida majoritariamente com esses dados, tem projeção de ultrapassar US$ 103 bilhões até 2027. Esse avanço reflete a crescente necessidade das organizações de extrair valor de fontes de informação diversas (Statista Research Department, 2023).

Já no segundo, a evolução acelerada e a popularização de ferramentas de inteligência artificial, as quais incorporam dados vetoriais para análise e processamento, demandam que diferentes tipos de informações, como vídeos, imagens e textos, sejam representados de maneira padronizada e processados com maior eficiência. Nesse sentido, os trabalhos Covington, Adams e Sargin (2016), Grbovic e Cheng (2018) demonstram essa aplicação ao modelarem, respectivamente, vídeos do YouTube e anúncios do Airbnb como *embeddings* para sistemas de recomendação.

O desafio central reside na incompatibilidade fundamental entre a natureza dos dados não estruturados e a arquitetura dos SGBDs relacionais. Estes sistemas são projetados sobre um paradigma de busca lexical, no qual as consultas operam com base em correspondências exatas ou padrões de texto. Contudo, a extração de valor de dados não estruturados depende de uma busca semântica, que visa recuperar informações com base na similaridade de significado e, frequentemente, envolve técnicas de *Machine Learning* e Processamento de Linguagem Natural (PLN) (JONKER; GOMSTYN, 2025).

Entretanto, antes de compreender o armazenamento dos dados não estruturados, é essencial compreender de que forma esses dados podem ser representados. Nesse con-

texto, destaca-se a importância do processo de transformação em representações vetoriais, técnica conhecida como *embedding* vetorial, que permite capturar o significado das informações. Esse procedimento é detalhado na seção a seguir.

## 2.2 Representação Semântica dos Dados: *Embeddings*

A solução para a limitação da busca lexical reside na capacidade de traduzir o significado semântico inerente aos dados não estruturados para um formato numérico que um computador possa processar. A abordagem mais eficaz para essa tarefa é a representação de palavras, frases ou até mesmo documentos inteiros como vetores em um espaço de alta dimensionalidade. Pode-se imaginar este espaço como um grande "mapa de conceitos", onde palavras com significados semelhantes, como "cão" e "gato", estão localizadas próximas uma da outra, enquanto palavras com significados distintos, como "cão" e "livro", estão distantes.

Formalmente, essa representação é conhecida como *embedding*: um vetor denso, de ponto flutuante e com um número fixo de dimensões, projetado para capturar as relações semânticas do dado original (ZILLIZ, 2025). A principal característica desses vetores é que eles preservam as relações entre os conceitos, permitindo a chamada "aritmética vetorial". A analogia clássica que demonstra essa capacidade é a operação vetor("*King*") - vetor("*Man*") + vetor("*Women*"), cujo resultado é um vetor muito próximo ao vetor("*Queen*") (MIKOLOV et al., 2013).

A geração de *embeddings* é realizada por meio do treinamento de redes neurais profundas, que resultam em um modelo capaz de converter dados brutos em vetores semânticos, conforme ilustra a Figura 2. Abordagens pioneiras, como o Word2Vec e o GloVe, estabeleceram as bases para essa técnica ao aprenderem representações vetoriais a partir do contexto em que as palavras coexistem em grandes volumes de texto.

Mais recentemente, a arquitetura *Transformer* refinou essa abordagem, resultando em *embeddings* significativamente mais ricos. Modelos como o BERT (*Bidirectional Encoder Representations from Transformers*) e a família GPT (*Generative Pre-trained Transformer*) alcançam essa melhoria por meio de dois mecanismos: a autoatenção (*self-attention*) e a codificação posicional (*positional encoding*). A primeira permite que o modelo entenda o contexto, ponderando a importância de cada palavra em relação às outras na mesma sentença. A segunda adiciona informações sobre a ordem dos termos, garantindo que a estrutura da frase seja considerada na representação final (GALLI; DONOS; CALCIOLARI, 2024).

Contudo, a utilização direta de modelos como o BERT para tarefas de busca por similaridade em larga escala é computacionalmente inviável. A arquitetura original exige que pares de sentenças sejam processados simultaneamente, um método com alto

Figura 2 – Ilustração do Fluxo de transformação de dados brutos em representações vetoriais (*embeddings*)


*Fonte: Adaptado de (TRIPATHI, 2023)*

custo de processamento que não escala para grandes conjuntos de dados. Para resolver essa limitação, foi desenvolvida a arquitetura Sentence-BERT (SBERT), que utiliza redes siamesas para gerar *embeddings* de sentenças individuais que podem ser comparados de forma extremamente eficiente com métricas como a distância do cosseno (REIMERS; GUREVYCH, 2019).

A implementação dessa abordagem é facilitada pela biblioteca Python sentence-transformers, que oferece acesso a milhares de modelos pré-treinados. Destaca-se o modelo "paraphrase-multilingual-MiniLM-L12-v2", por seu caráter multilíngue, ou seja, mapeia sentenças de diferentes idiomas para um único espaço vetorial, de modo que frases com o mesmo significado resultem em vetores próximos (REIMERS; GUREVYCH, 2020). Isso é possível através do treinamento em *corpus* com dezenas de idiomas e é ideal para dados de redes sociais online, onde a mistura de línguas e o uso de anglicismos são comuns.

## 2.3 Tecnologias para Armazenamento e Consulta Vetorial

Um Sistema Gerenciamento de Banco de Dados Vetorial é uma ferramenta projetada para o armazenamento, organização e indexação de informações em forma de vetores (IBM, 2025). Em outras palavras, ele oferece funcionalidades semelhantes às de SGBDs tradicionais, como buscas otimizadas, escalabilidade e tolerância a falhas, mas voltadas para dados não estruturados. Para entender como esses sistemas operam, é crucial conhecer os tipos de consulta que eles executam e as métricas que utilizam para definir a "proximidade" entre os vetores.

### 2.3.1 Tipos de Consulta e Métricas de Similaridade

É possível realizar diferentes tipos de consultas (*queries*), entretanto, não são todos os sistemas que implementam todas as variedades. O grande foco nesse cenário são as

consultas de busca (c,k), que têm como objetivo retornar os k vizinhos mais próximos de um vetor, sejam eles exatos (com grau de aproximação c=0) ou aproximados (onde c>0) (PAN; WANG; LI, 2024). Dentro dessas categorias, destacam-se as buscas *Nearest Neighbor Search* (NNS), com k=1, e *k-Nearest Neighbors Search* (k-NN), com k>1.

O que possibilita avaliar se um vetor X é mais próximo de um vetor Y são os *scores* de similaridade, definidos por funções de distância ou de similaridade. As principais métricas incluem a Distância de Hamming, a Distância de Minkowski, a Distância do Cosseno e o Produto Interno (*Inner Product*).

- **Distância de Hamming**

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

onde $a$ e $b$ são vetores e $\delta(a_i, b_i)$ é o delta de Kronecker.

- **Distância de Minkowski**

$$d_M(a,b) = \left( \sum_{i=1}^{n} |a_i - b_i|^p \right)^{\frac{1}{p}} \tag{2.2}$$

onde $p$ é um parâmetro que define a norma usada para calcular a distância;

Com $p$=1, tem-se a Distância de Manhattan (L1);

Com $p$=2, obtém-se a Distância Euclidiana (L2).

- **Distância do Cosseno**

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

equivalente ao ângulo entre a e b.

- **Produto Interno**

$$f(\mathbf{a}, \mathbf{b}) = \sum_{i=1}^{n} a_i b_i \tag{2.4}$$

No caso específico da extensão PGVector, do PostgreSQL, foi incorporado um conjunto de operadores customizados, para que fosse possível realizar consultas vetoriais de forma integrada. Os principais operadores são:

- `<->` – Distância L2 (Euclidiana);
- `<+>` – Distância L1 (Manhattan);
- `<#>` – Produto Interno Negativo;

- `<=>` – Distância do Cosseno;
- `<~>` – Distância de Hamming (para vetores binários);
- `<%>` – Distância de Jaccard (para vetores binários).

Esses novos operadores vetoriais são utilizados de forma integrada com cláusulas SQL padrão para construir consultas de k-vizinhos mais próximos (k-NN). A cláusula ORDER BY é fundamental, pois é nela que o operador de distância (como <=>) é aplicado para ordenar todos os registros com base em sua similaridade com o vetor de consulta. Em seguida, a cláusula LIMIT K é utilizada para restringir o resultado aos K itens mais bem classificados, ou seja, os vizinhos mais próximos desejados (PostgreSQL Global Development Group, 2024a).

### 2.3.2 Abordagens de Sistemas: Nativos X Estendidos

Atualmente, existem mais de 20 Sistemas de Gerenciamento de Bancos de Dados vetoriais desenvolvidos nos últimos 6 anos (PAN; WANG; LI, 2024). Esses sistemas podem ser classificados em duas categorias: sistemas nativos, que são projetados especificamente para o armazenamento e gerenciamento de dados vetoriais, e sistemas estendidos, que incorporam funcionalidades para lidar com dados vetoriais em SGBDs já existentes. Entre os avanços mais notáveis, destaca-se a capacidade desses sistemas de suportar uma ampla gama de consultas, incluindo vetoriais, tradicionais e textuais, além de oferecer suporte a múltiplos índices e otimização das buscas.

Na Figura 3, é possível observar uma lista de SGBDs que suportam dados no formato vetorial. Dentre eles tem-se os sistemas Nativos (Nat) e sistemas Estendidos (Ext), que adaptam tecnologias de banco de dados já existentes. No grupo dos nativos, observa-se uma especialização: alguns são puramente focados em vetores (subtipo Vec), enquanto outros adotam uma abordagem mista (subtipo Mix), gerenciando tanto vetores quanto metadados para permitir consultas mais ricas. A categoria de sistemas estendidos, por sua vez, demonstra a rápida adoção da busca vetorial por tecnologias consolidadas, abrangendo tanto bancos NoSQL (como Redis e MongoDB) quanto Relacionais (Rel).

Os sistemas nativos, como Pinecone, Milvus e Weaviate, são construídos do zero para máxima performance em operações com vetores. Sua principal vantagem é a baixa latência em buscas de alta escala. Contudo, essa abordagem exige a manutenção de uma arquitetura complexa com múltiplos sistemas (um para os dados relacionais, outro para os vetores), o que eleva custos e pode gerar desafios de sincronização de dados.

A abordagem estendida busca simplificar a arquitetura integrando funcionalidades vetoriais em sistemas consolidados. Um exemplo é o PGVector, uma ferramenta que

Figura 3 – Tabela de SGBDs Vetoriais

| Name | Type | Sub-type |
|---|---|---|
| EuclidesDB (2018) [12] | Nat | Vec |
| Vearch (2018) [15, 77] | Nat | Vec |
| Pinecone (2019) [2] | Nat | Vec |
| Vald (2020) [8] | Nat | Vec |
| Chroma (2022) [10] | Nat | Vec |
| Weaviate (2019) [1] | Nat | Mix |
| Milvus (2021) [16, 122] | Nat | Mix |
| NucliaDB (2021) [18] | Nat | Mix |
| Qdrant (2021) [9] | Nat | Mix |
| Manu (2022) [61] | Nat | Mix |
| Marqo (2022) [19] | Nat | Mix |
| Vespa (2020) [17] | Ext | NoSQL |
| Cosmos DB (2023) | Ext | NoSQL |
| MongoDB (2023) | Ext | NoSQL |
| Neo4j (2023) | Ext | NoSQL |
| Redis (2023) | Ext | NoSQL |
| AnalyticDB-V (2020) [130] | Ext | Rel |
| PASE+PG (2020) [136] | Ext | Rel |
| pgvector+PG (2021) [7] | Ext | Rel |
| SingleStoreDB (2022) [11, 99] | Ext | Rel |
| ClickHouse (2023) [20] | Ext | Rel |
| MyScale (2023) [21] | Ext | Rel |

*Fonte: Adaptado de (PAN; WANG; LI, 2024)*

aproveita a extensibilidade do PostgreSQL para suportar operações com vetores (PostgreSQL Global Development Group, 2024b). Ele adiciona um novo tipo de dado específico para armazená-los (VECTOR), juntamente com funções que permitem sua manipulação de maneira eficiente. As consultas são feitas diretamente em SQL e os resultados são frequentemente aproximados, gerados por meio de buscas por similaridade. Sendo assim, ocorre a comparação entre vetores, utilizando métricas de distância, para identificar itens mais semelhantes em relação a um vetor de consulta.

Especificamente, no PGVector, são implementadas buscas exatas e aproximadas, sendo a busca aproximada acelerada por meio do índice HNSW (*Hierarchical Navigable Small World*), que melhora o desempenho da recuperação de vetores em grandes volumes de dados. Além disso, a extensão se destaca pela variedade de *similatiry scores* implementadas, sendo elas: Distância L2 (*L2 distance*), Produto Interno (*Inner Product*), Distância do Cosseno (*Cosine distance*), Distância L1 (*L1 distance*), Distância de Hamming (*Hamming distance*) e Distância Jaccard (*Jaccard distance*) (PostgreSQL Global Development

Group, 2024a).

## 2.4 Análise Exploratória e Agrupamento de Dados

A extração de valor a partir da vasta quantidade de dados não estruturados é uma tarefa fundamental, porém complexa, pois estes dados, por natureza, não possuem rótulos ou classificações pré-existentes. Diante desse cenário, o aprendizado de máquina não supervisionado surge como a abordagem metodológica ideal. Diferentemente de métodos supervisionados, que requerem um conjunto de dados previamente classificado, as técnicas não supervisionadas são projetadas para descobrir estruturas e padrões latentes de forma autônoma, permitindo a organização e a análise de informações brutas (CHEN, 2024).

Uma das principais categorias do aprendizado não supervisionado é a clusterização, ou análise de agrupamento. O objetivo fundamental dessa técnica é organizar um conjunto de dados em grupos, chamados clusters, de modo que os itens dentro de um mesmo cluster possuam alta similaridade entre si, ao mesmo tempo em que sejam distintos dos itens de outros clusters. Formalmente, a clusterização busca uma partição dos dados que minimiza a distância intra-cluster (a distância entre os membros de um mesmo grupo) e maximiza a distância inter-cluster (a distância entre grupos diferentes) (TAN; STEINBACH; KUMAR, 2006).

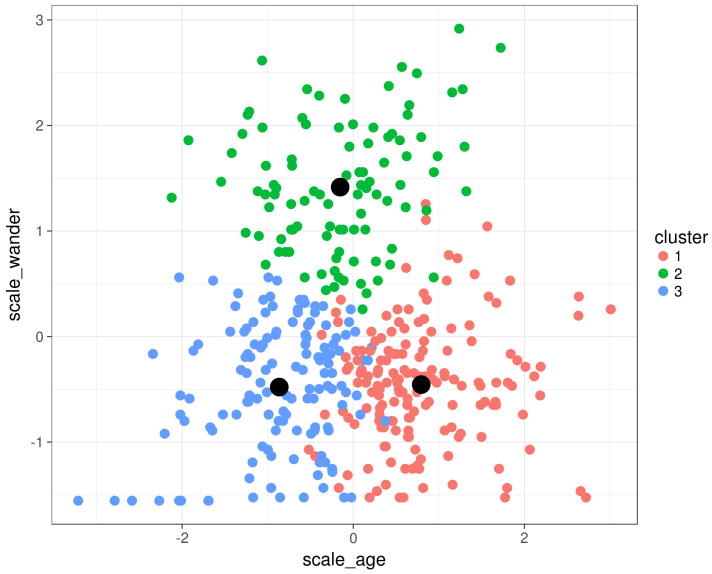
Os algoritmos de clusterização dividem-se em duas abordagens principais: particional e hierárquica. A abordagem particional, da qual o K-means é o principal exemplo, segmenta o conjunto de dados em um número K pré-definido de grupos distintos e não sobrepostos. Diferentemente, a abordagem hierárquica não exige a especificação prévia do número de clusters e cria uma estrutura aninhada de grupos, que pode ser visualizada como uma árvore (dendrograma).

O K-Means desfruta de notória popularidade, principalmente por sua simplicidade e fácil interpretabilidade. Além de sua eficiência computacional, que o torna escalável para grandes volumes de dados, ele se mostra particularmente adequado para tarefas de análise exploratória. No contexto deste trabalho, por exemplo, sua aplicação permite a descoberta de tópicos ou sentimentos em dados textuais representados como *embeddings*, justificando sua ampla adoção na área de Mineração de Dados.

A Figura 4 ilustra o resultado de um exemplo da aplicação do algoritmo K-Means, com o número de clusters (k) igual a 3. Nota-se os centroides em preto e cada clusters resultante com sua respectiva cor.

Apesar de suas vantagens, é crucial reconhecer as limitações inerentes ao algoritmo (AHMED; SERAJ; ISLAM, 2020; JAIN, 2010), que podem impactar a qualidade dos resultados:

Figura 4 – Exemplo de resultado de uma aplicação do algoritmo K-Means com número de clusters igual a 3




*Fonte: (YOBER, 2018)*

- Necessidade de pré-definir K: A principal desvantagem do algoritmo é a necessidade de especificar o número de clusters (K) antes de sua execução, o que nem sempre é uma informação conhecida a priori;
- Sensibilidade a *Outliers*: Por basear-se no cálculo de médias (centroides), o K-Means é altamente sensível a *outliers* (valores atípicos), que podem deslocar os centroides e distorcer o resultado dos agrupamentos;
- Pressuposto de Clusters Esféricos: O algoritmo assume que os clusters são de formato esférico e de tamanho similar, apresentando dificuldade em identificar corretamente agrupamentos com formatos mais complexos, alongados ou com densidades muito diferentes.

O funcionamento do algoritmo é um processo simples e iterativo que se inicia com a escolha de K centroides aleatórios. Na primeira etapa, chamada de atribuição, cada ponto de dado no conjunto é associado ao cluster do centroide mais próximo, com base em uma métrica de distância. Em seguida, na etapa de atualização, o centroide de cada cluster é recalculado como a média de todos os pontos que lhe foram atribuídos. O algoritmo repete os passos de atribuição e atualização até que a composição dos clusters se estabilize, ou seja, nenhum ponto mude de cluster, ou um número máximo de iterações seja atingido.

Para mitigar a limitação da escolha do número de clusters (K), uma heurística comum é o Método do Cotovelo (*Elbow Method*) (HAN; KAMBER; PEI, 2012). Essa técnica consiste em executar o algoritmo K-Means para um intervalo de valores de K e, para cada execução, calcular a Soma dos Quadrados Intra-clusters (WCSS), uma métrica

que representa a coesão do cluster. Os valores de WCSS são então plotados em um gráfico em função de K. Idealmente, o gráfico resultante se assemelha a um braço, e o "cotovelo" da curva, ponto onde a taxa de diminuição do WCSS se torna sutil, é considerado o indicativo do número ótimo de clusters para os dados, pois evidencia o ponto a partir do qual adicionar um novo cluster não resulta em uma redução significativa do WCSS (TOMAR, 2025).

Além do Método do Cotovelo, existe a Análise de Silhueta, uma técnica que mede o quão bem cada objeto se encaixa em seu próprio cluster em comparação com outros clusters. Para cada ponto de dado, calcula-se o coeficiente de silhueta, que varia de -1 a +1, com base em duas métricas: a distância média para os outros pontos no mesmo cluster (coesão) e a distância média para os pontos do cluster vizinho mais próximo (separação). Um coeficiente próximo de +1 indica que o ponto está bem alocado em seu cluster; um valor próximo de 0 indica que ele está na fronteira entre dois clusters; e um valor negativo sugere que ele pode ter sido atribuído ao cluster errado. Para determinar o número ótimo de grupos, executa-se o K-means para diferentes valores de K e escolhe-se o valor que maximiza o coeficiente de silhueta médio para todo o conjunto de dados (ROUSSEEUW, 1987).

# 3 Trabalhos Correlatos

Dado o caráter atual da temática e a crescente quantidade de dados gerados na Internet, observa-se um aumento significativo no número de trabalhos voltados à sua manipulação e interpretação. No artigo de Salgueiro e Lifschitz (2022), discute-se a representação e a análise de informações oriundas de redes sociais online em SGBDs, uma tarefa que, por sua vez, não é trivial, devido à natureza não estruturada e volume elevado desses elementos. O trabalho elucida a necessidade da utilização de um modelo que forneça o desempenho e funcionalidades de um SGBD relacional, mas que seja capaz de lidar com representações complexas de dados. Além disso, propõe uma categorização das informações extraídas em três grupos principais: análises centradas no usuário, nas conexões e nos conteúdos, evidenciando como essas categorias influenciam tanto o formato quanto a representação dos dados.

Nesse sentido, dando enfoque para a dificuldade de armazenamento comentada, explora-se a utilização de SGBDs vetoriais. Embora seja um tema recente e com poucos trabalhos que focam na utilização dessas ferramentas, é possível encontrar alguns exemplos. Um deles é o artigo de Winnicki et al. (2024), o qual enfatiza as características de um SGBD vetorial (VDB) e as diferenças para os SGBDS relacionais, como a forma de realização de consultas, melhor desempenho e menor latência. Além disso, apresenta o BioVDB, um SGBD vetorial desenvolvido com o Qdrant, voltado ao armazenamento e recuperação de dados de expressão gênica. Nesse sistema, cada amostra é representada como um vetor multidimensional, cuja dimensionalidade corresponde ao número de genes. A busca é realizada por similaridade vetorial, utilizando o algoritmo HNSW para indexação eficiente.

Já no contexto da complexidade na análise de dados provenientes de redes sociais online, o trabalho de Hou, Han e Cai (2020) apresenta um panorama abrangente sobre as técnicas de Mineração de Dados aplicadas nesse domínio. É proposto um *pipeline*, composto por quatro etapas principais: coleta, armazenamento, análise e visualização. A coleta de dados é viabilizada principalmente por APIs oficiais, como a Twitter Streaming API, que permitem a obtenção de dados em tempo real. Na etapa de armazenamento, o trabalho destaca soluções que combinam persistência e desempenho. A fase de análise é apontada como o núcleo do *pipeline*, reunindo técnicas como análise de tópicos, séries temporais e sentimentos, aplicadas em contextos como detecção de eventos, estudo de comportamento e difusão de informações. Por fim, a visualização é tratada como etapa fundamental para facilitar a interpretação e comunicação dos resultados.

Aprofundando a etapa de análise, especificamente no que tange ao tratamento de

textos de redes sociais online, o trabalho de Tiburcio (2021) propõe uma sequência de etapas de pré-processamento para os dados do X (antigo Twitter). O método de trabalho inicia-se com a tokenização, que segmenta o texto em unidades individuais (tokens). Em seguida, realiza-se a remoção de *stopwords* para eliminar palavras comuns e sem grande valor semântico (como artigos e preposições). Por fim, o vocabulário é normalizado por meio de lematização (redução da palavra à sua forma de dicionário) e *stemming* (redução ao seu radical). O objetivo desse conjunto de etapas é limpar e padronizar o conteúdo textual, reduzindo o ruído e a complexidade para as etapas subsequentes de análise, como a geração de *embeddings*.

No que tange à representação vetorial, o trabalho de Gomes, Attux e Cruz (2025) aplica o *framework* BERTopic para realizar modelagem de tópicos em redes sociais brasileiras. Esse *framework* opera em um *pipeline* que consiste, primeiramente, na conversão de textos em *embeddings* vetoriais; em seguida, no agrupamento (clusterização) desses vetores para identificar grupos de documentos semanticamente similares; e, por fim, na extração de palavras-chave que representam os tópicos formados. A etapa fundamental desse processo, e que o conecta diretamente com a presente pesquisa, é a geração de *embeddings*, uma tarefa para a qual o BERTopic frequentemente utiliza modelos da família SBERT. Alinhado a essa abordagem, este trabalho adota o modelo SBERT multilíngue (paraphrase-multilingual-MiniLM-L12-v2), cuja escolha se fundamenta na necessidade de analisar o ambiente linguístico heterogêneo das redes sociais online.

O trabalho de Sousa (2025) demonstra uma aplicação prática da extensão PGVector em um domínio diferente da análise textual, focando na manipulação e análise de vetores de áudio. O autor detalha a implementação de operações fundamentais, como a criação de tabelas com o tipo de dado VECTOR e a execução de buscas por similaridade utilizando métricas como a distância Lp e a distância do cosseno. A relevância deste estudo para a presente pesquisa reside na demonstração da versatilidade do PGVector como uma ferramenta de propósito geral para a análise vetorial, reforçando sua aplicabilidade em diferentes contextos de dados não estruturados, como áudio no estudo citado e textual no trabalho vigente.

Por fim, o trabalho de Souza et al. (2017) apresenta uma análise comparativa de algoritmos, incluindo o K-means, para a detecção de tópicos em redes sociais online. Em seu método de trabalho, os autores empregam a Análise de Silhueta para avaliar a qualidade e determinar o número ótimo de clusters. Este estudo se conecta à presente pesquisa pelo uso do K-means em um domínio de dados similar. Contudo, as abordagens divergem em pontos-chave: para a determinação do número de clusters, este trabalho utiliza o Método do Cotovelo (*Elbow Method*). A distinção mais fundamental, no entanto, reside na plataforma de execução: enquanto o estudo citado realiza a análise em um ambiente computacional externo, a pesquisa deste TCC foca na viabilidade de executar

a clusterização de forma integrada, diretamente no SGBD PostgreSQL com a extensão PGVector.

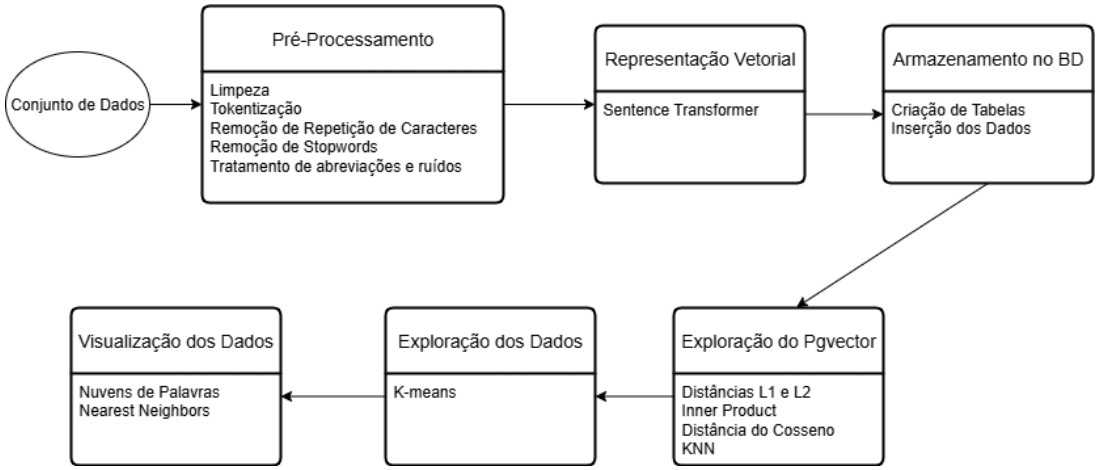
# 4 Método de Trabalho

Nesse capítulo foi apresentada uma sequência de etapas que visa englobar um exemplo completo, transitando por todas as partes do processo, desde o tratamento dos dados, até o eventual armazenamento deles e, por fim, sua aplicação em um algoritmo de agrupamento para demonstrar a viabilidade da extensão PGVector na execução de tarefas de *Machine Learning* diretamente no SGBD.

## 4.1 Etapas do Método

Para alcançar os objetivos desejados para este trabalho, foi adotado um conjunto de passos que orientaram sua construção de forma organizada e coerente. O método de trabalho escolhido busca garantir a consistência do processo e a validação dos resultados obtidos. A Figura 5 apresenta as principais etapas que compõem o método de pesquisa que foi adotado.

Figura 5 – Etapas do método proposto.




*Fonte: Elaborado pelo autor (2025).*

### 4.1.1 Definição dos Requisitos da Pesquisa

Para materializar a proposta deste trabalho — investigar o uso de SGBDs vetoriais na análise de dados de mídias sociais — a escolha de um conjunto de dados pertinente é fundamental. Para isso, foi feita uma busca em repositórios públicos por postagens extraídas da rede social online X, antigo *Twitter*, onde existe um foco maior em publicações no formato de texto, e por dados que fossem majoritariamente em português.

### 4.1.2 Pré-Processamento de Dados

Os dados oriundos de redes sociais online são caracterizados por sua natureza não estruturada, o que gera um desafio significativo para o Processamento de Linguagem Natural. Em razão disso, foi executada uma sequência de processos a fim de transformar cada *tweet* em um vetor padronizado, o qual posteriormente será armazenado no SGBD. Para isso, foi utilizado como referência o trabalho de Tiburcio (2021), o qual também manipula dados da rede social online X.

#### 4.1.2.1 Limpeza dos Dados

Nesta etapa, o objetivo é normalizar o *corpus* textual através da remoção de ruídos que poderiam comprometer a análise semântica. Contudo, ressalta-se que a abordagem não visa uma limpeza exaustiva de todos os padrões possíveis, mas sim o tratamento dos casos mais recorrentes.

Essa estratégia garante que a influência de ruídos seja minimizada, permitindo que as etapas subsequentes, como a geração de *embeddings*, sejam focadas no conteúdo semântico relevante do texto.

#### 4.1.2.2 Tokenização

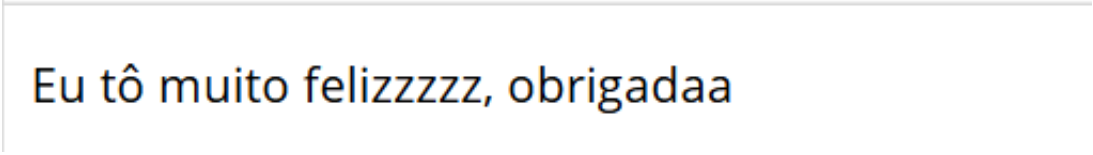
A tokenização é considerada uma das etapas mais simples e fundamentais do pré-processamento em tarefas de Processamento de Linguagem Natural (*Natural Language Processing* – NLP). Nessa fase, o *tweet* resultante da etapa anterior, ainda representado como uma *string*, é segmentado em unidades menores denominadas **tokens**. Para isso, ocorre a divisão nos espaços em branco, resultando em um vetor de palavras.

#### 4.1.2.3 Remoção de Repetição de Caracteres

É comum na linguagem informal, predominante nas redes sociais online, que usuários estendam palavras pela repetição de caracteres, como uma forma de adicionar ênfase ou expressar intensidade. Esse fenômeno, conhecido como alongamento lexical, pode ser observado na Figura 6, onde a palavra “feliz” é escrita com múltiplas ocorrências da letra “z” para transmitir emoção, assim como a palavra “obrigada” escrita com mais de um “a” no final. A alta frequência de casos como esse no conjunto de dados demanda um tratamento específico para padronizar o vocabulário, garantindo que variações como “felizzz” e “feliz” sejam tratadas como o mesmo token.

Dessa forma, foi implementada uma função para normalizar essas repetições. A estratégia adotada utiliza uma abordagem híbrida para evitar a correção excessiva de palavras legítimas.

Figura 6 – Exemplo de texto com alongamento lexical.




*Fonte: Elaborado pelo autor (2025).*

#### 4.1.2.4 Remoção de *Stopwords*

Mesmo após a limpeza de caracteres, a análise exploratória do *corpus* revela a persistência de *stopwords* — palavras de alta frequência que, embora essenciais para a coesão gramatical, não carregam significado semântico relevante para a análise de tópicos.

Para tratar essa questão, foi adotada uma estratégia de remoção customizada, que leva em conta tanto as *stopwords* tradicionais da língua portuguesa quanto as particularidades da linguagem informal das redes sociais online.

#### 4.1.2.5 Tratamento de Abreviações e Ruídos

Como etapa final do pré-processamento, foi realizada a normalização do vocabulário, visando padronizar abreviações e variações informais que são semanticamente equivalentes. A linguagem das redes sociais online é marcada pelo uso intensivo de formas contraídas (e.g., vc, tbm, sdds), que, se não tratadas, seriam interpretadas pelo modelo como tokens distintos de suas formas padrão (“você”,“também”, “saudades”), fragmentando o significado e aumentando desnecessariamente a complexidade do vocabulário.

Para solucionar essa questão, foi criado um dicionário de mapeamento customizado, contendo as abreviações mais recorrentes no *corpus* e suas respectivas formas expandidas.

### 4.1.3 Representação Vetorial

Com o conjunto de dados devidamente pré-processado, o próximo passo consiste na sua conversão para um formato vetorial, pré-requisito para seu armazenamento e manipulação com a extensão **PGVector**. Para esta tarefa, foi adotada a técnica de geração de *embeddings* semânticos que, ao contrário de técnicas baseadas em frequência de palavras, os *embeddings* representam cada texto como um vetor denso em um espaço multidimensional, onde a proximidade entre vetores reflete a similaridade semântica entre os textos originais.

### 4.1.4 Armazenamento no Banco de Dados

Finalizada a etapa de geração dos *embeddings*, o próximo passo consiste em persistir os dados processados no PostgreSQL[1]. Esta fase foi dividida em duas tarefas: primeiramente, a modelagem da estrutura relacional para acomodar os *tweets*, informações complementares e os vetores numéricos gerados, seguida, da inserção do conjunto de dados no SGBD.

### 4.1.5 Consultas Vetoriais

Uma vez que os *embeddings* estão armazenados e otimizados pelo PGVector, o próximo passo foi utilizá-los para extrair *insights* dos dados. Esta seção foca na execução de consultas vetoriais para demonstrar, na prática, como a extensão permite realizar análises complexas, como a busca por similaridade semântica, diretamente no ambiente do SGBD, eliminando a necessidade de transferir grandes volumes de dados para aplicações externas, resultando em um melhor desempenho.

### 4.1.6 Exploração dos Dados

O próximo passo da investigação consistiu em um estudo de caso prático, aplicando o PGVector para a **descoberta de tópicos** sobre o conjunto de *tweets* previamente persistido. A finalidade foi simular uma situação real de análise de dados, onde se busca extrair e agrupar automaticamente as publicações de uma rede social online com base na similaridade de seu conteúdo. Para tal, as mensagens foram clusterizadas de modo que textos com significado semelhante fossem agrupados, evidenciando a utilidade da busca vetorial para a geração de *insights*.

É fundamental ressaltar, contudo, que a relevância deste experimento transcende a simples análise de tópicos. O propósito primário dessa aplicação foi apresentar, através de um estudo de caso prático, a capacidade da extensão PGVector de suportar um fluxo de trabalho de *Machine Learning* de ponta a ponta. Ao realizar a clusterização diretamente no SGBD, evidencia-se a vantagem de manter a análise próxima aos dados, eliminando a latência de rede e a sobrecarga de transferência. Dessa forma, o sucesso da execução desse *pipeline* responde diretamente à questão central da pesquisa sobre a possibilidade de integrar tais funcionalidade a um ambiente relacional.

[1] https://www.postgresql.org/

# 5 Desenvolvimento e Análise dos Resultados

Este capítulo se dedica à aplicação prática do método de trabalho definido no Capítulo 4. Sua estrutura, portanto, integra o desenvolvimento e a análise, pois em um estudo de caso, cada etapa da implementação constitui um resultado relevante em si. Dessa forma, a descrição de cada fase do *pipeline* será imediatamente seguida pela discussão de suas observações e resultados.

## 5.1 Definição dos Requisitos de Pesquisa

O conjunto de dados escolhidos, disponível em DATA SET[1], reúne cerca de 780 mil *tweets*, já classificados como positivos ou negativos, extraídos via API da própria rede social online[2], entre 01/08/2018 a 20/10/2018. As instâncias não possuem tema específico e estão organizadas em uma tabela com as seguintes colunas:

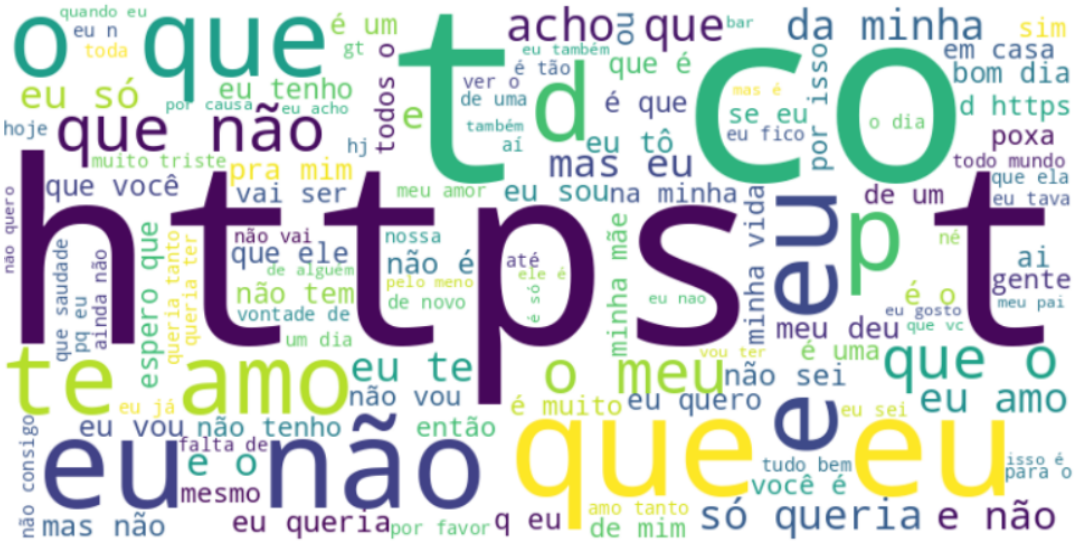
- **id**: identificador alfanumérico direto do *Twitter*;
- **tweet_text**: texto completo do *tweet*;
- **tweet_date**: data da criação do *tweet*;
- **sentiment**: *label* do sentimento;
- **query_used**: *query* usada para coletar o *tweet*.

A análise exploratória inicial do conjunto de dados revelou uma distribuição de sentimentos de 33% de *tweets* positivos e 67% negativos, conforme as métricas do repositório de origem. Para aprofundar a compreensão do conteúdo textual e identificar os termos mais recorrentes, foram geradas nuvens de palavras com e sem *stopwords*, Figuras 7 e 8, por meio da biblioteca WordCloud[3].

Observa-se, na Figura 7, uma grande quantidade de pronomes e artigos, característicos da língua portuguesa e essenciais para a construção das frases, mas que, isoladamente, possuem baixo valor semântico. Também é frequente a presença de gírias e abreviações, típicas da linguagem informal amplamente utilizada em mensagens nas redes sociais online. Nesse contexto, o tratamento dos dados torna-se fundamental para assegurar maior consistência e qualidade nas análises posteriores, conforme é detalhado na próxima seção.

[1] https://www.kaggle.com/datasets/augustop/portuguese-tweets-for-sentiment-analysis?select=TweetsWithTheme.csv
[2] API Twitter
[3] WordCloud

Figura 7 – Nuvem de Palavras com *StopWords*.




*Fonte: Elaborado pelo autor (2025).*

Figura 8 – Nuvem de Palavras sem *StopWords*.


*Fonte: Elaborado pelo autor (2025).*

Além disso, definiu-se a utilização da linguagem Python[4] e da execução do código em um *notebook* hospedado no próprio Kaggle[5], onde o conjunto de dados está disponibilizado. A escolha da linguagem deve-se à sua popularidade na área, aliada à ampla disponibilidade de bibliotecas, o que facilita o desenvolvimento e a integração de soluções. Já a opção pelo uso do *notebook* no *Kaggle* justifica-se pela praticidade: o ambiente permite integração direta com o conjunto de dados, execução interativa de código, visu-

[4] Linguagem de programação amplamente empregada em ciência de dados e *Machine Learning*, devido à sua sintaxe simples, vasta comunidade e bibliotecas especializadas como NumPy, Pandas, Scikit-learn, TensorFlow e PyTorch. (Awari, 2024)

[5] Plataforma online que disponibiliza *datasets*, recursos computacionais e ambiente interativo para análise e modelagem de dados. https://www.kaggle.com/

alização imediata dos resultados e utilização gratuita de recursos avançados, como GPU NVIDIA T4 (2×), essencial para a criação de *embeddings* e responsável por acelerar significativamente o tempo de processamento.

## 5.2 Pré-Processamento de Dados

Para ilustrar a aplicação desses passos, foi selecionado um subconjunto de oito *tweets*, Figura 9, do conjunto de dados, nos quais foram executadas cada uma das ações que foram detalhas nas subseções abaixo.

Figura 9 – Exemplos do conjunto de dados.

| | |
|---|---|
| 1 | vamo lá! 👏👏👏""não quero dinheiro eu quero amor sincero, é isso que eu espero"" :) https://t.co/RbSt66QY2P |
| 2 | @trashrocksite siiiiiiiiim! mereceu demais. :) |
| 3 | Hahaha, só vi e li agora isso aqui, @okjigu e @RiotPixie https://t.co/hCX83Ic68Q e lembrei do que escrevi há alguns anos :) https://t.co/ImUQ6GppNK Olha que bacana, haahahaha |
| 4 | @lenasuperpop_ aaaa se tem uma coisa que eu entendo essa coisa é dor muscular kkkkkk miga, toma um dorflex/anti-inflamatório leve, banho quente e dá uma massageada no local (ou locais kkk)! a dor talvez não suma 100%, mas dá uma aliviada boa. e continue mesmo :) |
| 5 | @veek_oficial Parabéns pela oferta de R$20 por 30 dias. Vocês se tornaram a melhor opção no mercado p/ quem quer manter um número que não usa mais como principal. E isso virará 100% de lucro, já que esse perfil de cliente não gera despesa de VU-M :D |
| 6 | @youngsnowblood @thegr8angelica a para vc q é❤️❤️ mt obg, eu me sinto até melhor agr :) |
| 7 | Até q deu p/ matar um pouco a saudade dos migos, mas eu, como a geleia emotiva q sou, fui me despedir deles na rodo, CHOREI MAIS Q SLA OQ, sdds de quando eu n chorava por quase nada :) |
| 8 | Qual a boa notícia dessa matéria? A pesquisa indica que houve um aumento nos investimentos das pessoas físicas no mercado financeiro. :D E a má notícia? A maioria -... https://t.co/SqzxZM4Egk |

*Fonte: Elaborado pelo autor (2025).*

### 5.2.1 Limpeza dos Dados

Para isso, foram empregadas expressões regulares, por meio da biblioteca re[6] do Python, para suprimir sistematicamente diversos padrões. O processo incluiu a eliminação de metadados da plataforma (menções e *hashtags*), conteúdo não verbal (*links* e *emojis*), além de características da linguagem informal, como risadas, dígitos e sinais de pontuação.

Além disso, antes do processo de limpeza, todos os *tweets* foram convertidos para letras minúsculas, a fim de evitar divergências decorrentes de variações de capitalização. A Figura 10 apresenta o resultado da aplicação das etapas de limpeza no subconjunto de dados selecionado anteriormente.

### 5.2.2 Tokenização

Para a realização dessa tarefa, foi utilizada a função word_tokenize da biblioteca Natural Language Toolkit (NLTK)[7], amplamente empregada em tarefas de Processamento de Linguagem Natural. Foi aplicada a função de tokenização a cada elemento da coluna

[6] re
[7] Natural Language Toolkit

Figura 10 – Exemplos do conjunto de dados após a etapa de limpeza

| | |
|---|---|
| 1 | vamo lá  não quero dinheiro eu quero amor sincero  é isso que eu espero |
| 2 | siiiiiiiiim  mereceu demais |
| 3 | só vi e li agora isso aqui e e lembrei do que escrevi há alguns anos olha que bacana |
| 4 | aaaa se tem uma coisa que eu entendo essa coisa é dor muscular miga toma um dorflex anti inflamatório leve banho quente e dá uma massageada no local ou locais a dor talvez não suma mas dá uma aliviada boa e continue mesmo |
| 5 | parabéns pela oferta de por dias vocês se tornaram a melhor opção no mercado p quem quer manter um número que não usa mais como principal e isso virará de lucro já que esse perfil de cliente não gera despesa de vu m d |
| 6 | a para vc q é mt obg  eu me sinto até melhor agr |
| 7 | até q deu p matar um pouco a saudade dos migos mas eu como a geleia emotiva q sou fui me despedir deles na rodo chorei mais q sla oq sdds de quando eu n chorava por quase nada |
| 8 | qual a boa notícia dessa matéria a pesquisa indica que houve um aumento nos investimentos das pessoas físicas no mercado financeiro d e a má notícia a maioria |

*Fonte: Elaborado pelo autor (2025).*

gerada após a limpeza, criando uma nova coluna chamada token. Dessa forma, cada *string* de texto, correspondente a um *tweet* previamente limpo, é segmentada em uma lista de tokens, permitindo análises mais granulares nas etapas subsequentes do pré-processamento.

O resultado dessa etapa aplicada no subconjunto de exemplo pode ser contemplado na Figura 11 .

Figura 11 – Exemplos do conjunto de dados após a etapa de tokenização

| | |
|---|---|
| 1 | ['vamo', 'lá', 'não', 'quero', 'dinheiro', 'eu', 'quero', 'amor', 'sincero', 'é', 'isso', 'que', 'eu', 'espero'] |
| 2 | ['siiiiiiiiim', 'mereceu', 'demais'] |
| 3 | ['só', 'vi', 'e', 'li', 'agora', 'isso', 'aqui', 'e', 'e', 'lembrei', 'do', 'que', 'escrevi', 'há', 'alguns', 'anos', 'olha', 'que', 'bacana'] |
| 4 | ['aaaa', 'se', 'tem', 'uma', 'coisa', 'que', 'eu', 'entendo', 'essa', 'coisa', 'é', 'dor', 'muscular', 'miga', 'toma', 'um', 'dorflex', 'anti', 'inflamatório', 'leve', 'banho', 'quente', 'e', 'dá', 'uma', 'massageada', 'no', 'local', 'ou', 'locais', 'a', 'dor', 'talvez', 'não', 'suma', 'mas', 'dá', 'uma', 'aliviada', 'boa', 'e', 'continue', 'mesmo'] |
| 5 | ['parabéns', 'pela', 'oferta', 'de', 'por', 'dias', 'vocês', 'se', 'tornaram', 'a', 'melhor', 'opção', 'no', 'mercado', 'p', 'quem', 'quer', 'manter', 'um', 'número', 'que', 'não', 'usa', 'mais', 'como', 'principal', 'e', 'isso', 'virará', 'de', 'lucro', 'já', 'que', 'esse', 'perfil', 'de', 'cliente', 'não', 'gera', 'despesa', 'de', 'vu', 'm', 'd']" |
| 6 | ['a', 'para', 'vc', 'q', 'é', 'mt', 'obg', 'eu', 'me', 'sinto', 'até', 'melhor', 'agr'] |
| 7 | ['até', 'q', 'deu', 'p', 'matar', 'um', 'pouco', 'a', 'saudade', 'dos', 'migos', 'mas', 'eu', 'como', 'a', 'geleia', 'emotiva', 'q', 'sou', 'fui', 'me', 'despedir', 'deles', 'na', 'rodo', 'chorei', 'mais', 'q', 'sla', 'oq', 'sdds', 'de', 'quando', 'eu', 'n', 'chorava', 'por', 'quase', 'nada'] |
| 8 | ['qual', 'a', 'boa', 'notícia', 'dessa', 'matéria', 'a', 'pesquisa', 'indica', 'que', 'houve', 'um', 'aumento', 'nos', 'investimentos', 'das', 'pessoas', 'físicas', 'no', 'mercado', 'financeiro', 'd', 'e', 'a', 'má', 'notícia', 'a', 'maioria'] |

*Fonte: Elaborado pelo autor (2025).*

### 5.2.3 Remoção de Repetição de Caracteres

O fenômeno, conhecido como alongamento lexical, pode ser observado no segundo *tweet* do resultado da etapa de tokenização (Figura 11), onde a palavra “sim” é escrita com múltiplas ocorrências da letra “i” para transmitir emoção.

O tratamento dessas ocorrências foi realizado por meio de um função. Primeiramente, ela verifica se o token pertence a uma lista de exceções predefinidas, que contém palavras em português e inglês onde a duplicação de letras é correta (e.g., "“reeleito”, “happy”, “pizza”). Caso o token esteja nessa lista, ele é mantido em sua forma original.

Para os demais casos, é aplicada uma expressão regular que identifica sequências de dois ou mais caracteres idênticos e as reduz a uma única ocorrência. Uma exceção foi adicionada para não alterar as letras “s” e “r”, preservando assim a grafia correta de dígrafos como “ss” e “rr”. Essa abordagem garante a padronização de palavras com ênfase, como “nãaaao” para "“não”, ao mesmo tempo em que protege a integridade de palavras válidas, resultando em um *corpus* mais padronizado para as próximas etapas.

A Figura 12 demonstra o sucesso desta etapa de normalização. Os exemplos 2 e 3 foram corrigidos, e a preservação de dígrafos como ’ss’ e ’rr’ confirma que a função operou conforme o esperado, padronizando o vocabulário sem introduzir erros.

Figura 12 – Exemplos do conjunto de dados após a etapa de remoção dos caracteres duplicados

| | |
|---|---|
| 1 | ['vamo', 'lá', 'não', 'quero', 'dinheiro', 'eu', 'quero', 'amor', 'sincero', 'é', 'isso', 'que', 'eu', 'espero'] |
| 2 | ['sim', 'mereceu', 'demais'] |
| 3 | ['só', 'vi', 'e', 'li', 'agora', 'isso', 'aqui', 'e', 'e', 'lembrei', 'do', 'que', 'escrevi', 'há', 'alguns', 'anos', 'olha', 'que', 'bacana'] |
| 4 | ['a', 'se', 'tem', 'uma', 'coisa', 'que', 'eu', 'entendo', 'essa', 'coisa', 'é', 'dor', 'muscular', 'miga', 'toma', 'um', 'dorflex', 'anti', 'inflamatório', 'leve', 'banho', 'quente', 'e', 'dá', 'uma', 'massageada', 'no', 'local', 'ou', 'locais', 'a', 'dor', 'talvez', 'não', 'suma', 'mas', 'dá', 'uma', 'aliviada', 'boa', 'e', 'continue', 'mesmo'] |
| 5 | ['parabéns', 'pela', 'oferta', 'de', 'por', 'dias', 'vocês', 'se', 'tornaram', 'a', 'melhor', 'opção', 'no', 'mercado', 'p', 'quem', 'quer', 'manter', 'um', 'número', 'que', 'não', 'usa', 'mais', 'como', 'principal', 'e', 'isso', 'virará', 'de', 'lucro', 'já', 'que', 'esse', 'perfil', 'de', 'cliente', 'não', 'gera', 'despesa', 'de', 'vu', 'm', 'd'] |
| 6 | ['a', 'para', 'vc', 'q', 'é', 'mt', 'obg', 'eu', 'me', 'sinto', 'até', 'melhor', 'agr'] |
| 7 | ['até', 'q', 'deu', 'p', 'matar', 'um', 'pouco', 'a', 'saudade', 'dos', 'migos', 'mas', 'eu', 'como', 'a', 'geleia', 'emotiva', 'q', 'sou', 'fui', 'me', 'despedir', 'deles', 'na', 'rodo', 'chorei', 'mais', 'q', 'sla', 'oq', 'sds', 'de', 'quando', 'eu', 'n', 'chorava', 'por', 'quase', 'nada'] |
| 8 | ['qual', 'a', 'boa', 'notícia', 'dessa', 'matéria', 'a', 'pesquisa', 'indica', 'que', 'houve', 'um', 'aumento', 'nos', 'investimentos', 'das', 'pessoas', 'físicas', 'no', 'mercado', 'financeiro', 'd', 'e', 'a', 'má', 'notícia', 'a', 'maioria'] |

*Fonte: Elaborado pelo autor (2025).*

### 5.2.4 Remoção de *Stopwords*

Ao analisar os *tweets* da Figura 12, por exemplo, nota-se que ainda são dominados por termos que servem para conectar os tokens, mas que não fornecem *insights* sobre o seu conteúdo.

Para removê-las,primeiramente, foi utilizada a lista padrão fornecida pela biblioteca NLTK[8]. A essa base, foi adicionado um conjunto de *stopwords* informais, criado especificamente para este trabalho, contendo abreviações, gírias e variações comuns no ambiente digital (e.g., vc, tá, pq, mano).

Uma exceção importante foi a remoção explícita da palavra “não” do conjunto final de *stopwords*. Essa decisão é crucial, pois a presença de uma negação é um indicador fundamental de sentimento, e sua remoção poderia inverter completamente o sentido de uma frase. O conjunto consolidado foi então aplicado para filtrar os tokens de cada *tweet*. A Figura 13 ilustra o resultado desse processo, resultando em listas de tokens compostas apenas por termos de maior relevância semântica.

[8] Natural Language Toolkit

Figura 13 – Exemplos do conjunto de dados após a etapa de remoção de *Stopwords*

| | |
|---|---|
| 1 | ['vamo', 'não', 'quero', 'dinheiro', 'quero', 'amor', 'sincero', 'espero'] |
| 2 | ['sim', 'mereceu', 'demais'] |
| 3 | ['vi', 'li', 'lembrei', 'escrevi', 'anos', 'olha', 'bacana'] |
| 4 | ['entendo', 'dor', 'muscular', 'miga', 'toma', 'dorflex', 'anti', 'inflamatório', 'leve', 'banho', 'quente', 'dá', 'massageada', 'local', 'locais', 'dor', 'talvez', 'não', 'suma', 'dá', 'aliviada', 'boa', 'continue'] |
| 5 | ['parabéns', 'oferta', 'dias', 'tornaram', 'melhor', 'opção', 'mercado', 'quer', 'manter', 'número', 'não', 'usa', 'principal', 'virará', 'lucro', 'perfil', 'cliente', 'não', 'gera', 'despesa', 'vu', 'm'] |
| 6 | ['mt', 'obg', 'sinto', 'melhor'] |
| 7 | ['deu', 'matar', 'pouco', 'saudade', 'migos', 'geleia', 'emotiva', 'despedir', 'rodo', 'chorei', 'sds', 'n', 'chorava', 'quase', 'nada'] |
| 8 | ['boa', 'notícia', 'matéria', 'pesquisa', 'indica', 'aumento', 'investimentos', 'pessoas', 'físicas', 'mercado', 'financeiro', 'má', 'notícia', 'maioria'] |

*Fonte: Elaborado pelo autor (2025).*

### 5.2.5 Tratamento de Abreviações e Ruídos

Uma função foi aplicada para percorrer a lista de tokens de cada *tweet* e substituir as abreviações encontradas, com base no dicionário de mapeamento customizado, contendo as abreviações mais recorrentes no *corpus* e suas respectivas formas expandidas. Em seguida, foi realizada uma última filtragem para remover tokens compostos por uma única letra, que frequentemente são resquícios de ruído do processo de limpeza e não possuem carga semântica.

A Figura 14 exibe o resultado da aplicação desta etapa final sobre o conjunto de dados de exemplo, que representa a conclusão de todo o processo de pré-processamento. Os tokens resultantes estão agora normalizados e devidamente tratados.

Figura 14 – Exemplos do conjunto de dados após a etapa de tratamento de abreviações e ruídos

| | |
|---|---|
| 1 | ['vamo', 'não', 'quero', 'dinheiro', 'quero', 'amor', 'sincero', 'espero'] |
| 2 | ['sim', 'mereceu', 'demais'] |
| 3 | ['vi', 'li', 'lembrei', 'escrevi', 'anos', 'olha', 'bacana'] |
| 4 | ['entendo', 'dor', 'muscular', 'amiga', 'toma', 'dorflex', 'anti', 'inflamatório', 'leve', 'banho', 'quente', 'dá', 'massageada', 'local', 'locais', 'dor', 'talvez', 'não', 'suma', 'dá', 'aliviada', 'boa', 'continue'] |
| 5 | ['parabéns', 'oferta', 'dias', 'tornaram', 'melhor', 'opção', 'mercado', 'quer', 'manter', 'número', 'não', 'usa', 'principal', 'virará', 'lucro', 'perfil', 'cliente', 'não', 'gera', 'despesa', 'vu'] |
| 6 | ['muito', 'obrigado', 'sinto', 'melhor'] |
| 7 | ['deu', 'matar', 'pouco', 'saudade', 'amigos', 'geleia', 'emotiva', 'despedir', 'rodo', 'chorei', 'saudades', 'não', 'chorava', 'quase', 'nada'] |
| 8 | ['boa', 'notícia', 'matéria', 'pesquisa', 'indica', 'aumento', 'investimentos', 'pessoas', 'físicas', 'mercado', 'financeiro', 'má', 'notícia', 'maioria'] |

*Fonte: Elaborado pelo autor (2025).*

Ao final do *pipeline* de pré-processamento, o conjunto de dados foi significativamente refinado. Partindo de *tweets* brutos, repletos de ruídos e informalidades, o processo sequencial de limpeza, normalização, filtragem de *stopwords* e padronização de vocabulário resultou em um conjunto de dados coeso e semanticamente mais rico. Cada etapa foi crucial para reduzir a complexidade e focar a análise nos termos que efetivamente

carregam significado, garantindo que a geração de *embeddings* opere sobre uma base mais padronizada. A eficácia dessa etapa é comprovada visualmente na Figura 15, que compara as nuvens de palavras geradas antes e depois do pré-processamento e demonstra a maior relevância semântica do vocabulário final.

Figura 15 – Comparativo da nuvem de palavras antes (esquerda) e após (direita) o pré-processamento.


*Fonte: Elaborado pelo autor (2025).*

## 5.3 Representação Vetorial

Com o conjunto de dados pré-processado, foi realizada a transformação do *tweet* resultante em um vetor. Para isso, foi utilizada a biblioteca **sentence-transformers**[9], que é baseada na arquitetura *Transformer*. Especificamente, foi selecionado o modelo pré-treinado 'paraphrase-multilingual-MiniLM-L12-v2'. A escolha deste modelo é justificada por duas características cruciais para este trabalho:

- Sua natureza **multilíngue**, que o torna robusto para o português e suas variações informais encontradas nas redes sociais online;
- Seu treinamento em tarefas de identificação de paráfrases, o que o torna excepcionalmente eficaz em mapear sentenças com significados semelhantes para vetores próximos no espaço vetorial, uma propriedade essencial para a utilização das ferramentas de busca da extensão do **PGVector**.

O processo de implementação consistiu em, primeiramente, converter as listas de tokens de volta para o formato de texto (*string*). Em seguida, o método **encode()** do modelo foi invocado para realizar a vetorização. Para otimizar o uso de memória e processamento em um conjunto de dados de grande volume, a conversão foi realizada em lotes (*batches*), com um tamanho de lote definido para 64, utilizando aceleração por GPU disponível no ambiente Kaggle. Ao final, cada *tweet* foi representado por um vetor de 384 dimensões.

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

A Figura 16 ilustra o *embedding* gerado a partir desse modelo.

Figura 16 – Exemplo de geração do *embedding* de um *tweet* com sentence-transformers.


*Fonte: Elaborado pelo autor (2025).*

## 5.4 Armazenamento no Banco de Dados

Concluída a geração dos *embeddings*, o próximo passo consiste em persistir os dados processados no PostgreSQL[10].

### 5.4.1 Modelagem dos Dados

Ao final do pré-processamento, o conjunto de dados foi enriquecido com colunas intermediárias que registram cada etapa da transformação, além da coluna final de *embeddings*. No entanto, para a persistência no banco de dados, optou-se por uma abordagem mais focada, mantendo apenas os dados essenciais para a análise.

Visto que o objetivo principal é a utilização das operações vetoriais do **PGVector**, decidiu-se não armazenar as colunas intermediárias do pré-processamento. A modelagem final da tabela preserva as colunas originais de identificação e metadados (**id, tweet_text, tweet_date e sentiment**), a lista final de tokens — útil para futuras análises de tópicos — e a representação vetorial (*embeddings*). Dessa forma, a estrutura da tabela **dados_twitter**, criada para este trabalho, segue a modelagem apresentada na Tabela 1.

### 5.4.2 Inserção dos Dados

O processo de inserção dos dados na tabela modelada foi orquestrado por um *script* em Python, utilizando a biblioteca **psycopg2**[11] como conector para o SGBD. Um passo essencial nesta configuração é a chamada da função `register_vector()`, disponibilizada

[10] https://www.postgresql.org/
[11] https://www.psycopg.org/docs

Tabela 1 – Estrutura da tabela `dados_twitter`

| **Coluna** | **Tipo de Dado (PostgreSQL)** | **Descrição** |
|---|---|---|
| id | BIGINT (PRIMARY KEY) | Identificador único do *tweet*. |
| tweet_text | TEXT | O texto original do *tweet*. |
| tweet_date | TIMESTAMPTZ | Data e hora da publicação. |
| sentimento | TEXT | Sentimento associado (positivo/negativo). |
| tokens | TEXT | Representação textual da lista de tokens. |
| embeddings | VECTOR(384) | Vetor de *embedding* semântico do texto. |

pela extensão **PGVector**[12]. Esta função registra o tipo de dado **VECTOR** no conector, permitindo que listas do Python sejam automaticamente convertidas no momento da inserção e que os vetores retornados pelo banco sejam corretamente interpretados pela aplicação. Essa compatibilidade é crucial para viabilizar a persistência da coluna *embeddings* de maneira eficiente.

Para garantir a consistência a cada execução, a tabela `dados_twitter` foi inicialmente esvaziada com o comando **TRUNCATE**. Em seguida, o *script* iterou sobre cada registro do conjunto de dados, executando instruções **INSERT**. O destaque deste procedimento está no tratamento do campo de *embeddings*: os vetores foram encapsulados na estrutura `Vector`, fornecida pelo **PGVector**. Esta etapa é decisiva, pois instrui o PostgreSQL a armazenar os *embeddings* em seu formato vetorial nativo e otimizado, habilitando assim os cálculos e as consultas por similaridade de forma performática.

## 5.5 Consultas Vetoriais

Para validar experimentalmente as métricas de distância implementadas no PGVector discutidas no Capítulo 2, foi selecionado um *tweet* âncora, apresentado na Figura 17, e, para cada uma delas (com exceção das específicas para vetores binários), foi executada uma consulta que retorna os cinco *tweets* mais próximos (top-5) de todo o conjunto de dados. O objetivo é analisar e comparar o *ranking* de resultados gerado por cada métrica, revelando como a escolha do operador de distância impacta o resultado de uma busca semântica na prática para diferentes interpretações de similaridade.

Ao analisar os resultados apresentados na Tabela 2, é possível observar que a escolha da métrica de distância impacta diretamente o resultado da busca por similaridade. A **Distância do Cosseno**, ideal para análise semântica, focou em *tweets* curtos e diretos que expressam gratidão. As **Distâncias L1 e L2** retornaram resultados semelhantes

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

Tabela 2 – Comparativo de ranking dos 5 vizinhos mais próximos por métrica de distância

| Rank | Tweet Retornado |
|---|---|
| | **Distância L1 (<+>)** |
| 1 | "@nathetas nat, obrigado por essas palavras... agradeço muito por isso, é importante pra mim pra caramba!..." |
| 2 | "@pedrosolidus @anticast grato pelas palavras, meu caro! :)" |
| 3 | "@uttwonini eu sou horrível com palavras também está tudo certo! muito obrigada por tudo meu anjinho..." |
| 4 | "tudo que importa na boa mesmo voces nao tem nem noção do quanto sao importantes pra mim..." |
| 5 | "@tamarapadilha_ imaginaaa, não precisa agradecer não! você é merecedora de todo apoio..." |
| | **Distância L2 (<->)** |
| 1 | "@nathetas nat, obrigado por essas palavras... agradeço muito por isso, é importante pra mim pra caramba!..." |
| 2 | "tudo que importa na boa mesmo voces nao tem nem noção do quanto sao importantes pra mim..." |
| 3 | "ei, não te conheço mas espero de vdd q vc supere o que ta passando... sem palavras :( obrigado" |
| 4 | "@_rafamaral ô, rafa!! que bom te encontrar aqui! <3 poxa, muito obrigada pelas palavras de incentivo." |
| 5 | "quando me elogiam eu fico mt boba jjkkkkkk deixou meu dia bem melhor : ) ..." |
| | **Distância do Cosseno (<=>)** |
| 1 | "@pedrosolidus @anticast grato pelas palavras, meu caro! :)" |
| 2 | "tudo que importa na boa mesmo voces nao tem nem noção do quanto sao importantes pra mim..." |
| 3 | "oh gente obrigado pelas palavras bonitas :)" |
| 4 | "@nathetas nat, obrigado por essas palavras... agradeço muito por isso, é importante pra mim pra caramba!..." |
| 5 | "@vinimzo sábias palavras, mestre. obrigado :)" |
| | **Produto Interno (<#>)** |
| 1 | "transbordando em gratidão :)" |
| 2 | "gratidão demais pelas coisas que tem acontecido :)" |
| 3 | "@bmantonieto :) gratidão" |
| 4 | "gratidão!!! :d <https://t.co/azjapzdpuc>" |
| 5 | "gratidão :)" |

entre si, priorizando um *tweet* mais longo e contextual, mas com variações no *ranking* intermediário. Notavelmente, o **Produto Interno** divergiu completamente, focando em textos que continham a palavra "gratidão", provavelmente por esses vetores possuírem uma combinação de direção e magnitude que maximiza o resultado dessa métrica. Isso demonstra que a escolha do operador de distância é uma decisão de modelagem crítica

Figura 17 – *Tweet* âncora utilizado nos experimentos de busca por similaridade.

| id | tweet_text |
|---|---|
| 1031742457262860000 | "obrigada por fazer parte dos meus dias, você não tem noção do quão grata sou por te ter. você é 10/10 e merece o mundo. :) — aaaaaaaa que lindaaaa❤❤ suas palavras melhoraram mto o meu dia!!! 💕 quem é??? https://t.co/lqdinlsf00" |

*Fonte: Elaborado pelo autor (2025).*

que define o próprio conceito de "similaridade" na busca vetorial.

Além da análise comparativa, a funcionalidade de busca por vizinhos mais próximos é a base para a implementação de algoritmos de classificação, como o K-Nearest Neighbors (KNN). A lógica do KNN consiste em classificar um novo dado com base no rótulo da maioria de seus 'K' vizinhos mais próximos. Neste contexto, é possível classificar o sentimento de um *tweet* desconhecido encontrando seus vizinhos semânticos e realizando uma "votação" com base na coluna **sentiment** dos resultados retornados. Para demonstrar essa aplicação, foi selecionado um novo *tweet* âncora (Figura 18) e executada uma busca por seus 7 vizinhos mais próximos (K=7) utilizando a Distância do Cosseno.

Figura 18 – *Tweet* âncora para o experimento de classificação KNN.

| id<br>[PK] bigint | tweet_text<br>text | sentimento<br>text |
|---|---|---|
| 1031722781640220673 | tá começando aquela chuvinha marota :) | Positivo |

*Fonte: Elaborado pelo autor (2025).*

Tabela 3 – Resultado da busca de vizinhos para o KNN (K=7)

| **Rank** | **Vizinho Retornado** | **Sentimento** |
|---|---|---|
| 1 | "que começasse a chover :) https://t.co/lpsa3aviy7" | Positivo |
| 2 | "pq começou a chover? :(" | Negativo |
| 3 | "começou a chover :) https://t.co/y9ko4xkafv" | Positivo |
| 4 | "começou a chover :)" | Positivo |
| 5 | "começo a chuva de novo :(" | Negativo |
| 6 | "@gui_conti aff começou a chover :(" | Negativo |
| 7 | "será se essa chuva vai adiar minha provinha :)" | Positivo |

Com base nos resultados da Tabela 3, quatro dos sete vizinhos mais próximos são classificados como 'Positivo', enquanto três são 'Negativo'. A votação da maioria para determinar a classe final pode ser implementada de forma eficiente com uma única consulta SQL, que agrupa os vizinhos por sentimento e retorna a classe com a maior contagem, conforme o código apresentado na Listagem 5.1.

```sql
WITH kNN AS (
    SELECT sentimento
    FROM dados_twitter
    ORDER BY embeddings <=> (SELECT embeddings FROM dados_twitter
    WHERE id = [ID_DO_TWEET_ANCORA])
    LIMIT 7
),
classFreq AS (
    SELECT sentimento, COUNT(*) AS freq
    FROM kNN
    GROUP BY sentimento
)
SELECT sentimento
FROM classFreq
WHERE freq = (SELECT max(freq) FROM classFreq);
```

Listing 5.1 – Query SQL para votação da maioria no KNN

A execução desta consulta retorna o resultado exibido na Figura 19. Pelo critério da maioria, o *tweet* âncora seria classificado como **Positivo**, o que condiz com o que foi retornado.

Figura 19 – Resultado da consulta que simula o KNN.

| | sentimento text |
|---|---|
| 1 | Positivo |

*Fonte: Elaborado pelo autor (2025).*

## 5.6 Exploração dos Dados

Para realizar a clusterização de tópicos diretamente no SGBD, foi desenvolvida uma função em PL/pgSQL, a linguagem procedural do PostgreSQL, que foi denominada de Kmeans. Ela engloba toda a lógica do algoritmo, desde a inicialização dos centroides até a convergência. Dessa forma, todo o processamento computacionalmente intensivo do **K-Means** ocorre dentro do próprio SGBD. Na Listagem 5.2 é apresentado o código-fonte completo da implementação.

```sql
CREATE OR REPLACE FUNCTION KMeans(k INT, max_iteracoes INT)
RETURNS TABLE(ponto_id BIGINT, cluster_atribuido_id INT, distancia FLOAT
    ) AS $$
DECLARE
    iteracao INT := 0;
```

```
5       houve_mudanca BOOLEAN := TRUE;
6   BEGIN
7       TRUNCATE TABLE Centroides;
8
9       INSERT INTO Centroides (cluster_id, embedding)
10      SELECT
11          ROW_NUMBER() OVER () AS cluster_id,
12          embeddings
13      FROM (
14          SELECT embeddings
15          FROM teste
16          ORDER BY RANDOM()
17          LIMIT k
18      ) AS LinhasAleatorias;
19
20      RAISE NOTICE "K-Means: Inicialização com % centroides concluída.", k
        ;
21
22      WHILE iteracao < max_iteracoes AND houve_mudanca LOOP
23          iteracao := iteracao + 1;
24          RAISE NOTICE 'Iniciando iteração %...', iteracao;
25
26          -- a) Salva o estado anterior dos clusters para comparação
27          TRUNCATE TABLE Clusters_old;
28          INSERT INTO Clusters_old SELECT * FROM Clusters;
29
30          -- b) Passo de Atribuição: Calcula as novas distâncias e atribui
            os clusters
31          TRUNCATE TABLE Clusters;
32          INSERT INTO Clusters (id, centroide_id, dist)
33          SELECT DISTINCT ON (t.id)
34              t.id,
35              c.cluster_id,
36              t.embeddings <-> c.embedding
37          FROM
38              teste AS t
39          CROSS JOIN
40              Centroides AS c
41          ORDER BY
42              t.id,
43              t.embeddings <-> c.embedding ASC;
44
45          RAISE NOTICE 'Iteração %: Passo de Atribuição concluído.',
        iteracao;
46
47          -- c) Critério de Parada: Verifica se houve alguma mudança nas
        atribuições
```

```
48          -- Se a contagem de diferenças for 0, o loop irá parar na pró
    xima verificação.
49          SELECT EXISTS (
50              SELECT 1
51              FROM (
52                  (SELECT id, centroide_id FROM Clusters EXCEPT SELECT id,
     centroide_id FROM Clusters_old)
53                  UNION ALL
54                  (SELECT id, centroide_id FROM Clusters_old EXCEPT SELECT
     id, centroide_id FROM Clusters)
55              ) AS diferencas
56          ) INTO houve_mudanca;
57
58          IF NOT houve_mudanca THEN
59              RAISE NOTICE 'Iteração %: Algoritmo convergiu. Não houve
    mudanças.', iteracao;
60              EXIT; -- Sai do loop imediatamente
61          END IF;
62
63          -- d) Passo de Atualização: Recalcula a posição dos centroides
64          UPDATE Centroides centroide_set
65          SET
66              embedding = novos.novo_embedding
67          FROM (
68              SELECT
69                  cl.centroide_id,
70                  AVG(t.embeddings) AS novo_embedding
71              FROM
72                  teste AS t
73              JOIN
74                  Clusters AS cl ON t.id = cl.id
75              GROUP BY
76                  cl.centroide_id
77          ) AS novos
78          WHERE
79              centroide_set.cluster_id = novos.centroide_id;
80
81          RAISE NOTICE 'Iteração %: Passo de Atualização dos centroides
    concluído.', iteracao;
82
83      END LOOP;
84
85      IF iteracao >= max_iteracoes THEN
86          RAISE NOTICE 'K-Means: Atingido o número máximo de iterações (%)
    .', max_iteracoes;
87      END IF;
88  RETURN QUERY SELECT * FROM Clusters;
```

```
89
90      RAISE NOTICE 'K-Means: Processo finalizado.';
91
92 END;
93 $$ LANGUAGE plpgsql;
```

Listing 5.2 – Função K-Means em PL/pgSQL para clusterização de vetores.

Vale ressaltar que a construção desta função, implementando um algoritmo de *Machine Learning* inteiramente em SQL, foi inspirada nos princípios apresentados por Antonio Badia em sua obra *SQL for Data Science* (BADIA, 2020). O autor defende a capacidade dos SGBDs relacionais modernos de executarem tarefas analíticas complexas, tradicionalmente delegadas a ambientes externos, promovendo uma análise mais próxima e eficiente dos dados.

A lógica da função `K-Means`, apresentada na Listagem 5.2, pode ser decomposta em três etapas fundamentais que seguem o fluxo do algoritmo, conforme detalhado a seguir:

1. **Inicialização dos Centroides:** O processo inicia-se com a seleção de `k` pontos de dados aleatórios do conjunto de *tweets* para servirem como as posições iniciais dos centroides. Esta etapa é realizada através da seleção aleatoria dos k centroides iniciais, utilizando `ORDER BY RANDOM()` e `LIMIT k`, que são então inseridos na tabela `Centroides`, juntamente com um *cluster_id*, que é correspondente ao valor da linha, ou seja, o primeiro centroide selecionado tem *cluster_id=1*.

2. **Processo Iterativo de Convergência:** O núcleo da função é um laço `WHILE` que executa repetidamente as duas fases principais do K-Means até que a convergência seja atingida ou um número máximo de iterações seja alcançado. Este ciclo é composto por:

    - *Passo de Atribuição:* Esta é a operação mais crítica para a avaliação de performance, pois envolve encontrar o centroide mais próximo para cada um dos milhares de pontos de dados a cada iteração. A abordagem utilizada para essa tarefa foi: primeiramente, um `CROSS JOIN` executado entre a tabela de *tweets* e a de centroides, gerando um produto cartesiano com todos os pares possíveis de (*tweet*, centroide). Em seguida, para cada par, a distância é calculada utilizando o operador de distância Euclidiana L2 (`<->`) do `PGVector`. Por fim, o comando `ORDER BY t.id, distancia ASC` ordena o resultado de forma que, para cada *tweet*, a linha com a menor distância apareça primeiro. A construção `SELECT DISTINCT ON (t.id)` é então utilizada para selecionar apenas esta primeira linha de cada grupo, atribuindo eficientemente cada ponto ao seu cluster de menor distância em uma única consulta.

- *Passo de Atualização:* Após a atribuição de todos os pontos, a posição de cada centroide é recalculada. A extensão `PGVector` sobrecarrega a função de agregação `AVG()`, permitindo que ela calcule a média vetorial (o centroide geométrico) de todos os *embeddings* pertencentes a um mesmo cluster, atualizando a tabela `Centroides` com as novas posições.
- *Verificação de Convergência:* Ao final de cada iteração, o algoritmo verifica se houve alguma mudança nas atribuições dos clusters em comparação com a iteração anterior. Isso é feito de maneira declarativa em SQL através do operador `EXCEPT`. Se não houver diferenças, o processo é considerado convergente e o laço é encerrado.

3. **Finalização e Retorno dos Resultados:** Ao final do laço, seja por convergência ou por atingir o limite de iterações, a função retorna a tabela `Clusters` com a atribuição final de cada ponto de dado ao seu respectivo cluster, juntamente com a distância final até o centroide.

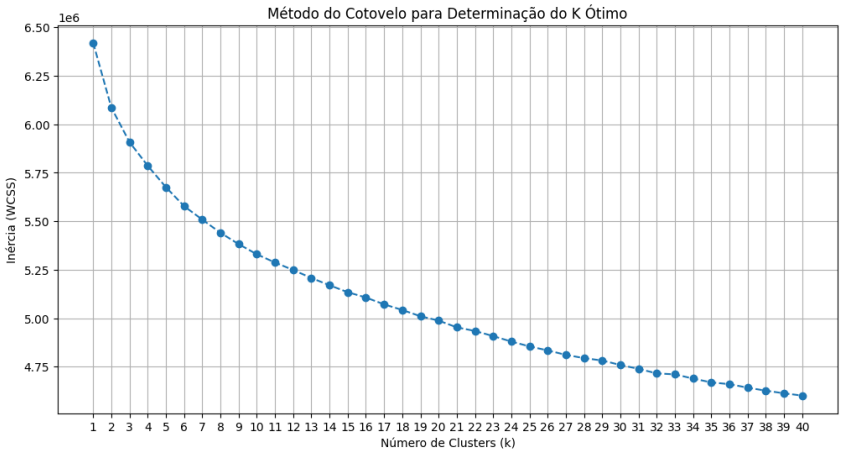
A correta parametrização do algoritmo K-Means é um passo fundamental para a obtenção de resultados significativos. O principal hiperparâmetro é o número de clusters (k), uma vez que é ele que determina a quantidade de grupos em que os dados serão "forçados" a se organizar. Para essa definição, foi empregado o Método do Cotovelo (*Elbow Method*), que se baseia na Soma dos Quadrados Intra-clusters (WCSS). Esta métrica representa a soma das distâncias quadráticas entre cada ponto e o centroide de seu respectivo cluster, medindo, assim, o quão compactos e coesos são os agrupamentos. O método consiste em calcular a WCSS para um intervalo de valores de k e plotar os resultados em um gráfico. O valor de k ideal é identificado no ponto que visualmente forma um "cotovelo", representando o ponto de equilíbrio onde a adição de um novo cluster não proporciona uma redução significativa na WCSS.

A Figura 20 representa o gráfico do cotovelo gerado para o conjunto de dados do experimento. Foi utilizado um intervalo de k entre 1 e 40. Notou-se que o cotovelo encontra-se entre os valores de k=2 a k=6.

Além do número de clusters, o limite máximo de iterações foi outro parâmetro fundamental definido para o experimento. Este valor atua como um critério de parada, garantindo que o algoritmo não entre em um laço de execução infinito caso não atinja a convergência e evitando, assim, o consumo excessivo de recursos computacionais. Para este estudo, foi estabelecido um limite de 300 iterações, um limiar definido empiricamente que se mostrou suficiente para a convergência nos testes preliminares.

Com os hiperparâmetros devidamente configurados, o algoritmo K-Means foi executado para o intervalo de k entre 2 e 6, sugerido pela análise do Método do Cotovelo. Para avaliar a qualidade semântica de cada resultado, foram geradas nuvens de palavras

Figura 20 – Gráfico do Cotovelo para o intervalo de k entre 1 e 40.




*Fonte: Elaborado pelo autor (2025).*

(*word clouds*) para cada cluster obtido. A análise qualitativa dessas visualizações indicou que a configuração com k=5 proporcionou a separação mais nítida e coerente dos tópicos, sendo, portanto, o valor escolhido para a análise final. O resultado pode ser observado na Figura 21.

A análise qualitativa das nuvens de palavras revelou que, embora a segmentação dos tópicos não seja perfeitamente distinta, com alguma sobreposição semântica entre os grupos, é possível identificar padrões temáticos emergentes em cada cluster. Isso indica que, para uma primeira análise exploratória, a clusterização foi capaz de agrupar os dados com certa coerência. A seguir, são descritos os principais padrões observados para cada um dos cinco clusters:

- **Cluster 1 - Sentimentos Afetivos e Gratidão:** Este agrupamento concentrou palavras de conotação majoritariamente positiva, expressando sentimentos pessoais como amor, gratidão e saudade. A presença de termos como *[ex: "obrigado", "amo", "saudade"]* sugere um tema central de afeto e apreciação.

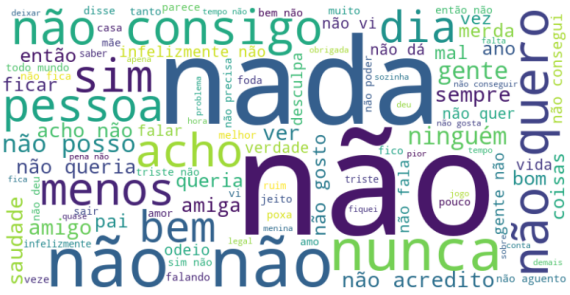
- **Cluster 2 - Negação e Ceticismo:** Em contraste com o anterior, este cluster reuniu termos predominantemente negativos. As palavras indicam um padrão de descrença, impossibilidade e negação, como pode ser visto em vocábulos como *[ex: "não", "nunca", "nada"]*.

- **Cluster 3 - Tristeza e Arrependimento:** Este grupo também apresentou um viés negativo, mas focado em emoções relacionadas à tristeza e ao lamento. Termos ligados a desculpas, choro e desapontamento, como *[ex: "desculpa", "triste", "chorando"]*, formam o núcleo semântico deste cluster.

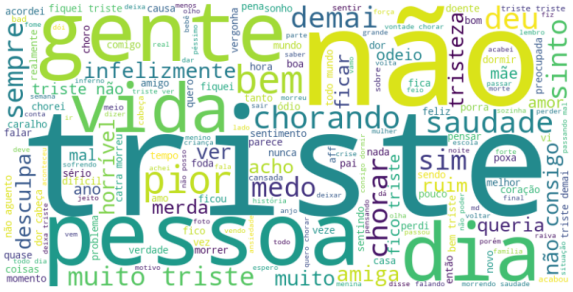
- **Cluster 4 - Opiniões e Afirmações do Cotidiano:** Este cluster se mostrou mais heterogêneo, agrupando termos comuns do discurso cotidiano. Palavras como *"sim"*,

Figura 21 – Nuvem de palavras dos clusters resultantes do K-Means, com k=5.


(a) Cluster 1




(b) Cluster 2




(c) Cluster 3


(d) Cluster 4


(e) Cluster 5

*Fonte: Elaborado pelo autor (2025).*

*"não"*, *"acho"* e *"quero"* indicam um padrão de expressões de opinião, afirmação e vontade. A presença de substantivos genéricos como *"gente"*, *"pessoa"*, *"dia"* e *"foto"* reforça a natureza conversacional e generalista deste agrupamento.

- **Cluster 5 - Interações Sociais Positivas:** Por fim, este agrupamento focou em termos positivos ligados à socialização. Palavras que remetem a elogios, encontros e saudade de interações sociais, como *[ex: "amo", "lindo", "amiga"]*, indicam um tema de apreciação social e conexão com outras pessoas.

A fim de proporcionar uma visualização mais granular e concreta dos resultados, a análise foi complementada pela seleção dos 10 vizinhos mais próximos (*Nearest Neighbors*) do centroide de cada cluster. A extração desses *tweets* permite uma inspeção direta das

mensagens que são mais centrais e representativas para cada um dos tópicos identificados.

Tabela 4 – Os 10 vizinhos mais próximos do centroide do Cluster 1 (Sentimentos Afetivos e Gratidão).

| **ID** | **Tweet** | **Tokens** | **Distância** |
|---|---|---|---|
| 1041514330850766858 | como eu te amo, que saudade yang :(( https://t.co/xdomcqzeku | ['amo', 'saudade', 'yang'] | 0.701 |
| 1040539198036602881 | @tiyopaulo haberdei bb boi :) | ['haberdei', 'boi'] | 0.738 |
| 1033382930851262471 | aaaaaa que saudade do pajé :( eu tenho que ir em goiânia logo | ['saudade', 'pajé', 'ir', 'goiânia'] | 0.750 |
| 1032060296163205120 | ja to com saudadea da zaira :( | ['saudadea', 'zaira'] | 0.777 |
| 1030643135976755200 | @camilalcantarx achei que vc me amava :( | ['achei', 'amava'] | 0.780 |
| 1030991489843847169 | achei que me amava :( https://t.co/g63iwgohw6 | ['achei', 'amava'] | 0.780 |
| 1033518530493378561 | @joaope_ornelas @muriloandrade77 :( achei que vc ainda me amava | ['achei', 'amava'] | 0.780 |
| 1032011195392372736 | queria tá assim com a lauren que eu gosto :( https://t.co/xfh5h91yiu | ['queria', 'lauren', 'gosto'] | 0.782 |
| 1038572701387055107 | @sincelouistan gosto, mas queria que fosse cacheado :( | ['gosto', 'queria', 'cacheado'] | 0.784 |
| 1038846309804007424 | @duranteoctavio achei q vc me amasse e tivesse comovido com a minha bad :( | ['achei', 'amasse', 'comovido', 'bad'] | 0.784 |

A Tabela 4 oferece uma análise concreta dos resultados, exibindo os *tweets* mais representativos do **Cluster 1**. A observação dos exemplos valida a interpretação temática de "Sentimentos Afetivos e Gratidão", com a notável recorrência de tokens como "amo", "saudade", "gosto" e "amava", que ancoram o cluster neste domínio semântico. Um ponto de particular interesse é que, embora os termos centrais sejam afetivos, o contexto de muitas mensagens é de lamento ou tristeza, frequentemente indicado pelo *emoticon* :(. Isso demonstra a capacidade do modelo de *embeddings* de agrupar as mensagens com base na similaridade do campo temático (o tópico de amor e saudade), independentemente da

polaridade do sentimento final.

Tabela 5 – Os 10 vizinhos mais próximos do centroide do Cluster 2 (Negação e Ceticismo).

| ID | Tweet | Tokens | Distância |
|---|---|---|---|
| 1046181954926907397 | @lilgoldenneedle poxa n da pra dar rt :( | ['poxa', 'não', 'dar', 'rt'] | 0.799 |
| 1038814300117377025 | @davyjonesrj amanhã nn :(( puuts, tava ansiosaaasso aqui dava | ['não', 'puts', 'ansiosasso', 'dava'] | 0.801 |
| 1041074706080772102 | @kimyugstan amas eu n trato :( | ['amas', 'não', 'trato'] | 0.804 |
| 1030997380303388672 | @yellowstarling beo teeeos :( ai mas pq? tu não curte? | ['beo', 'teos', 'não', 'curte'] | 0.817 |
| 1037868996966723585 | @guthierryt n aceito esse 1kg de açai aí :( | ['não', 'aceito', 'açai'] | 0.817 |
| 1032132456995020800 | @95johnnievb fica assim não, neném :(// | ['fica', 'não', 'neném'] | 0.819 |
| 1042606505311305729 | @ok_sween fica não neném :(( | ['fica', 'não', 'neném'] | 0.819 |
| 1030903517370884097 | @bbazemlol quem ainda n passou foste tu :) | ['não', 'passou', 'foste'] | 0.822 |
| 1030517573752942594 | sextou,porém não sextarei :( | ['sextou', 'porém', 'não', 'sextarei'] | 0.830 |
| 1044071908789153792 | @feitosinha97 nao teho 1 sossego >:-( | ['não', 'teho', 'sossego'] | 0.859 |

A análise dos exemplos da Tabela 5 reforça a interpretação do **Cluster 2** como um agrupamento temático de "Negação e Ceticismo". A presença do token "não" (e suas variações "n" e "nn") é uma constante em quase todos os *tweets* mais representativos, servindo como a âncora semântica do grupo. As mensagens frequentemente expressam frustração ("poxa"), negação de uma ação ("não sextarei", "não aceito") ou desapontamento. A coerência do cluster demonstra a capacidade do modelo em agrupar frases com base em uma função gramatical e semântica similar, mesmo que os contextos específicos de cada *tweet* variem.

A inspeção dos *tweets* mais próximos do centroide do **Cluster 3**, apresentados na Tabela 6, valida de forma contundente a interpretação temática de "Tristeza e Arrependimento". A recorrência de palavras como "triste", "tristinha" e "tristona" é a evidência mais explícita. Além disso, o cluster agrupa de forma coerente outras emoções do mesmo campo semântico, como "dor", "mágoas", "pena" e arrependimento ("choranu", "arrepende"). É

Tabela 6 – Os 10 vizinhos mais próximos do centroide do Cluster 3 (Tristeza e Arrependimento).

| ID | Tweet | Tokens | Distância |
|---|---|---|---|
| 1033375545248174080 | dor e mágoas saiam de mim :( | ['dor', 'mágoas', 'saiam'] | 0.863 |
| 1040985369225900033 | fica falando que tá triste tomara que morra — :( https://t.co/f91qjgrocf | ['fica', 'falando', 'triste', 'tomara', 'morra'] | 0.884 |
| 1043333118592274432 | @leehdlima @d3senheira @_renatchenha @cbs_everton :( foi triste demais quando raissa me contou isso | ['triste', 'demais', 'raissa', 'contou'] | 0.915 |
| 1031644967045873664 | deu tanta pena de anne hoje :(( | ['deu', 'tanta', 'pena', 'ane'] | 0.933 |
| 1031160333635936256 | to com saudade de fica doida :( | ['saudade', 'fica', 'doida'] | 0.953 |
| 1033520894814154753 | eu muito triste, tomara que eu morra :( | ['triste', 'tomara', 'morra'] | 0.962 |
| 1039301155178905600 | se arrepende muito de alguma coisa? — de algumas rsrsrs choranu :( https://t.co/xawsemf3zx | ['arrepende', 'choranu'] | 0.965 |
| 1034932813823438848 | eu to muito triste tirei não é recíproco :(! https://t.co/t3spukghfi | ['triste', 'tirei', 'não', 'recíproco'] | 0.989 |
| 1042221319985741824 | a susy rosnou pra mim tô tristona :( | ['susy', 'rosnou', 'tristona'] | 0.989 |
| 1041588283657084928 | quem foi que colocou essa au na minha tl agora eu to toda tristinha :( | ['colocou', 'au', 'tl', 'tristinha'] | 0.992 |

interessante notar a presença de desejos negativos extremos ("tomara que morra"), o que reforça a intensidade emocional do agrupamento. A consistência dos exemplos demonstra que o algoritmo foi eficaz em separar um conjunto de mensagens com um forte viés emocional negativo e pessimista.

Os exemplos do **Cluster 4**, conforme Tabela 7, confirmam sua natureza hetero-

Tabela 7 – Os 10 vizinhos mais próximos do centroide do Cluster 4 (Opiniões e Afirmações do Cotidiano).

| ID | Tweet | Tokens | Distância |
|---|---|---|---|
| 1038189776883527680 | @muthiaabcd ia qq aq ms esema :( | ['aq', 'ms', 'esema'] | 0.785 |
| 1042010156966535168 | qe eu sou um bosta ? todos sabem :) | ['qe', 'bosta'] | 0.816 |
| 1038383378217689088 | @lvdoyoung96 aq esmosi :( | ['aq', 'esmosi'] | 0.828 |
| 1033399970186121217 | @ln_cezar eii já to com sdds do vg :( | ['ei', 'saudades', 'vg'] | 0.848 |
| 1031977634966724608 | @boasarah_ quando eu dei vc preferiu o igor :( | ['dei', 'preferiu', 'igor'] | 0.860 |
| 1037643730227023872 | e agora, ao som de nysmaw :( | ['som', 'nysmaw'] | 0.881 |
| 1045459131883286528 | esta net pode ir pro caralhinho :) | ['net', 'ir', 'caralhinho'] | 0.919 |
| 1038171126856331264 | puta que pariu a bixinha :( :( :( https://t.co/ghfzpq3ibg | ['puta', 'pariu', 'bixinha'] | 0.929 |
| 1036569936452304896 | tenho tantas saudades do miguel, do artur do tomás da rita e da bia ferreira juro ahhhhhhh :(( | ['tantas', 'saudades', 'miguel', 'artur', 'tomás', 'rita', 'bia', 'ferreira', 'juro'] | 0.934 |
| 1038904468333576192 | po gente o catra :( | ['po', 'gente', 'catra'] | 0.936 |

gênea e conversacional. As mensagens agrupadas não compartilham um único tema, mas sim um estilo de linguagem informal e cotidiano, frequentemente utilizado para expressar opiniões ou relatar eventos pessoais. A presença de interjeições, gírias e palavrões reforça que o critério de agrupamento foi a similaridade na forma da expressão, e não necessariamente o sentimento por trás dela. Este cluster, portanto, captura com sucesso um padrão de discurso generalista e coloquial, distinto dos agrupamentos mais focados em emoções específicas.

A análise do Cluster 5, apresentado na Tabela 8, revela, em um primeiro nível, a eficácia do processo de clusterização implementado. O algoritmo agrupou com sucesso um conjunto de mensagens textualmente quase idênticas, cujo token principal e recorrente é *umarim*. A capacidade de agrupar itens de alta similaridade em um mesmo cluster valida o funcionamento fundamental de todo o fluxo de processamento, desde a geração dos

Tabela 8 – Os vizinhos mais próximos do centroide do Cluster 5 (Interações Sociais Positivas).

| ID | Tweet | Tokens | Distância |
|---|---|---|---|
| 1036664084677238785 | @aytackara umarim :) | ['umarim'] | 0.5638179 |
| 1044985153884499968 | @hassasmal umarim :)))) | ['umarim'] | 0.5638179 |
| 1029596111328161793 | @alchemien umarim. :) | ['umarim'] | 0.5638179 |
| 1045437470781902849 | umarim :) https://t.co/chr4mzm58s | ['umarim'] | 0.5638179 |
| 1046711237067853824 | @eeenapiyimki umarimm :( | ['umarim'] | 0.5638179 |
| 1044670645320273921 | @nevsinmengu umarim :) | ['umarim'] | 0.5638179 |
| 1049292948414910465 | @busralock umarim :)) | ['umarim'] | 0.5638182 |
| 1042921553313386497 | @ay3uke umarimm :( | ['umarim'] | 0.5638182 |
| 1040258917509603328 | @metubikbix umarim :) | ['umarim'] | 0.5638182 |
| 1037938242841137152 | @zainapova umarim :( | ['umarim'] | 0.5638182 |

*embeddings* até a execução do K-Means.

Aprofundando a análise, o termo *umarim* pertence à língua turca e significa "espero que" (REVERSO, 2025). Este fato torna-se particularmente interessante quando contrastado com a análise inicial da nuvem de palavras do cluster, que indicava uma predominância de termos relacionados a "Interações Sociais Positivas". Essa aparente discrepância é um forte indício da capacidade multilíngue do modelo de *embedding*. O resultado sugere que o 'paraphrase-multilingual-MiniLM-L12-v2' mapeou o conceito de "esperança" da língua turca para uma região do espaço vetorial próxima a conceitos de interações positivas em português, validando a robustez da abordagem para lidar com a diversidade linguística de redes sociais online.

Para validar essa hipótese foi realizado um teste quantitativo de distância do cosseno. O *embedding* do termo *umarim* (turco) foi gerado e comparado com os *embeddings* de sua tradução em português (espero que) e em inglês (*I hope*), além de termos de controle semanticamente distantes (*book*, livro e gato). Os resultados, apresentados na Tabela 9, demonstram uma alta similaridade (*scores* $> 0.8$) entre *umarim* e suas traduções, e baixa similaridade (*scores* $< 0.5$) com os termos de controle. Isso comprova que o modelo mapeou o conceito de "esperança" para a mesma região do espaço vetorial, independentemente do idioma, validando seu caráter multilíngue e a robustez do modelo escolhido para este trabalho.

Tabela 9 – Distância do Cosseno entre o termo *umarim* e outros termos em português e inglês.

| **Termo/Frase** | **Idioma** | ***Score* de Similaridade** |
|---|---|---|
| 'espero que' | Português | 0.9766 |
| 'I hope' | Inglês | 0.9487 |
| 'livro' | Português | 0.3732 |
| 'gato' | Português | 0.2799 |
| 'book' | Inglês | 0.2727 |

*Fonte: Elaborado pelo autor (2025).*

Em suma, este capítulo detalhou a implementação de um estudo de caso completo, demonstrando o fluxo de trabalho desde o pré-processamento e vetorização dos *tweets* até a execução de um algoritmo de clusterização K-Means diretamente no PostgreSQL com a extensão `PGVector`. A execução bem-sucedida deste protótipo valida a hipótese central da pesquisa, confirmando a viabilidade de se utilizar um SGBD relacional para realizar tarefas de análise vetorial que, tradicionalmente, exigiriam sistemas especializados. A análise dos clusters, embora exploratória, revelou a capacidade da abordagem em identificar padrões semânticos nos dados. As implicações destes achados, as limitações do estudo e as sugestões para trabalhos futuros são discutidas no capítulo de conclusão que se segue.

# 6 Conclusão e Recomendações para Trabalhos Futuros

O presente trabalho foi motivado pela necessidade crescente de gerenciar e analisar dados não estruturados, um resultado direto do advento das redes sociais online e da consequente propagação massiva de informações. Diante da consolidada utilização dos SGBDs relacionais e da complexidade associada à adoção de sistemas específicos para dados não estruturados, esta pesquisa se propôs a investigar a viabilidade de uma abordagem integrada. O objetivo foi explorar uma forma de aproveitar a estrutura já conhecida desses sistemas para armazenar e processar essa nova demanda.

Para atender a essa proposta, o estudo concentrou-se em uma solução específica do PostgreSQL: a extensão PGVector. O objetivo central foi validar esta ferramenta como uma alternativa funcional para o armazenamento e processamento de dados de redes sociais online. Para tanto, o trabalho consistiu na construção de um estudo de caso completo, seguindo um *pipeline* de Mineração de Dados que abrangeu desde o tratamento dos dados e seu armazenamento até a execução de um algoritmo de *Machine Learning* diretamente no SGBD.

Conclui-se, a partir do experimento prático desenvolvido, que é **plenamente viável** utilizar o PostgreSQL com a extensão PGVector para realizar tarefas de análise de dados não estruturados, representados na forma vetorial. A execução bem-sucedida do algoritmo K-Means diretamente no ambiente de banco de dados funciona como uma validação funcional sólida, demonstrando que a integração do processamento desses dados em um ambiente relacional não é apenas teórica, mas sim concreta e aplicável.

A coerência qualitativa dos resultados, observada tanto nas buscas por similaridade quanto nos agrupamentos gerados pelo K-Means, comprova a eficácia de todo o *pipeline* de análise proposto. A capacidade de executar estas operações e obter agrupamentos temáticos discerníveis valida a extensão PGVector como uma solução que vai além do simples armazenamento, mostrando que também funciona como uma ferramenta de processamento analítico diretamente no ambiente relacional.

Nesse sentido, a principal contribuição deste estudo reside na demonstração de uma solução factível para manipulação e processamento de dados de redes sociais online dentro de um SGBD reacional. Ao detalhar cada etapa do processo, este trabalho oferece um roteiro validado para a integração de análises de similaridade em infraestruturas desse tipo de sistema, servindo como um recurso prático para futuros projetos na área. O código-

fonte está publicamente disponível[1].

É importante apontar que o foco da pesquisa foi estritamente a viabilidade funcional da solução, não incluindo, portanto, uma análise quantitativa de performance. Desta forma, não foram realizadas comparações de métricas de desempenho, como tempo de execução, consumo de CPU e utilização de memória, entre a abordagem com PGVector e SGBDs vetoriais dedicados.

Adicionalmente, o pré-processamento dos dados textuais (*tweets*) foi conduzido de forma exploratória; a limpeza e tokenização focaram-se na remoção dos ruídos mais recorrentes, sem o aprofundamento em técnicas mais exaustivas de normalização. Por fim, a escolha do algoritmo K-Means e dos seus hiperparâmetros serviu ao propósito deste estudo de caso, mas não foi acompanhada de uma validação rigorosa que atestasse a sua otimalidade para o conjunto de dados em questão.

Sendo assim, fica como sugestões para trabalhos futuros:

- **a partir da limitação de desempenho**: Uma extensão natural desse trabalho consistiria em uma análise de desempenho comparativa, confrontando o PGVector com um ou mais SGBDs vetoriais dedicados (como Pinecone) e NoSQL (como MongoDB) em diferentes cargas de trabalho e volumes de dados.
- **a partir da limitação de pré-processamento**: Os resultados podem ter sido impactados pela simplicidade das etapas de preparação textual. Futuras pesquisas podem explorar estratégias mais avançadas, como lematização, remoção contextualizada de *stopwords* ou ponderação de termos por relevância. Outra possibilidade é comparar o desempenho de diferentes representações vetoriais, ampliando a análise para além do modelo utilizado neste estudo.
- **a partir da limitação do algoritmo**: A escolha pelo K-Means trouxe vantagens de simplicidade e integração, mas também limitações conhecidas, como a necessidade de fixar o número de clusters. Caminhos futuros incluem a adoção de algoritmos que lidem melhor com densidade ou formatos não esféricos de dados, como o DBSCAN ou o HDBSCAN, bem como a incorporação de métricas de avaliação interna e externa para uma validação mais abrangente da qualidade dos agrupamentos.

[1] O código-fonte completo desenvolvido para este estudo de caso está disponível no repositório GitHub: https://github.com/anacorsi/tcc-pgvector-estudo-de-caso

# Referências

AHMED, M.; SERAJ, R.; ISLAM, S. M. S. The k-means algorithm: A comprehensive survey and performance evaluation. **Electronics**, v. 9, n. 8, 2020. ISSN 2079-9292. Disponível em: <https://doi.org/10.3390/electronics9081295>. Citado na página 20.

ALAKE, R. Your AI leader needs this: Strategic AI database architecture for AI. **Medium (MongoDB)**, jun. 2025. Acesso em: 10 mar. 2025. Disponível em: <https://medium.com/mongodb/your-ai-leader-needs-this-strategic-ai-database-architecture-for-ai-c80956e5813f>. Citado na página 11.

Awari. **Linguagens para Machine Learning: Quais as Mais Usadas**. 2024. Blog da Awari. Disponível em: <https://awari.com.br/linguagens-para-machine-learning-quais-as-mais-usadas/>. Acesso em: 26 set. 2025. Citado na página 31.

BADIA, A. **SQL for Data Science: Data Cleaning, Wrangling and Analytics with Relational Databases**. Burlington, MA: Morgan Kaufmann, 2020. ISBN 978-0128183350. Disponível em: <https://doi.org/10.1007/978-3-030-57592-2>. Citado na página 44.

CAMBRIA, E.; WHITE, B. Jumping nlp curves: A review of natural language processing research [review article]. **IEEE Computational Intelligence Magazine**, v. 9, n. 2, p. 48–57, 2014. Disponível em: <https://doi.org/10.1109/MCI.2014.2307227>. Citado na página 11.

CHEN, M. **O que é aprendizado não supervisionado?** 2024. Oracle. Disponível em: <https://www.oracle.com/br/artificial-intelligence/machine-learning/unsupervised-learning>. Acesso em: 21 set. 2025. Citado na página 20.

CLEVELAND, W. S. **The Elements of Graphing Data**. Belmont, CA: Wadsworth Publishing, 1985. Citado na página 11.

COVINGTON, P.; ADAMS, J.; SARGIN, E. Deep neural networks for YouTube recommendations. In: **Proceedings of the 10th ACM Conference on Recommender Systems**. New York, NY, USA: Association for Computing Machinery, 2016. (RecSys '16), p. 191–198. ISBN 9781450340359. Disponível em: <https://doi.org/10.1145/2959100.2959190>. Citado na página 14.

DOMO. **Data Never Sleeps 12.0**. 2024. Disponível em: <https://www.domo.com/learn/infographic/data-never-sleeps-12>. Acesso em: 10 mar. 2025. Citado na página 11.

GALLI, C.; DONOS, N.; CALCIOLARI, E. Performance of 4 pre-trained sentence transformer models in the semantic query of a systematic review dataset on peri-implantitis. **Information**, v. 15, n. 2, 2024. ISSN 2078-2489. Disponível em: <https://doi.org/10.3390/info15020068>. Citado na página 15.

GOMES, G. B. B.; ATTUX, R.; CRUZ, C. C. Enhancing contributions to brazilian social media analysis based on topic modeling with native bert models. **Journal of Information and Data Management**, v. 16, n. 1, p. 181–191, Jun. 2025. Disponível em: <https://journals-sol.sbc.org.br/index.php/jidm/article/view/4670>. Citado na página 24.

GRBOVIC, M.; CHENG, H. Real-time personalization using embeddings for search ranking at airbnb. In: **Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining**. New York, NY, USA: Association for Computing Machinery, 2018. (KDD '18), p. 311–320. ISBN 9781450355520. Disponível em: <https://doi.org/10.1145/3219819.3219885>. Citado na página 14.

HAN, J.; KAMBER, M.; PEI, J. **Data Mining: Concepts and Techniques**. 3rd. ed. Waltham, MA: Morgan Kaufmann, 2012. ISBN 978-0123814791. Citado na página 21.

HOU, Q.; HAN, M.; CAI, Z. Survey on data analysis in social media: A practical application aspect. **Big Data Mining and Analytics**, v. 3, n. 4, p. 259–279, 2020. Disponível em: <https://doi.org/10.26599/BDMA.2020.9020006>. Citado na página 23.

IBM. **Banco de Dados Vetorial**. 2025. Acessado em: 23 mar. 2025. Disponível em: <https://www.ibm.com/br-pt/think/topics/vector-database>. Citado na página 16.

JAIN, A. K. Data clustering: 50 years beyond k-means. **Pattern Recognit. Lett.**, v. 31, n. 8, p. 651–666, 2010. Disponível em: <https://doi.org/10.1016/j.patrec.2009.09.011>. Citado na página 20.

JONKER, A.; GOMSTYN, A. **Dados estruturados vs dados não estruturados: qual é a diferença?** 2025. IBM. Disponível em: <https://www.ibm.com/br-pt/think/topics/structured-vs-unstructured-data>. Acesso em: 20 set. 2025. Citado 2 vezes nas páginas 13 e 14.

KHAN, U. H. **Unstructured Data Processing: What It Is and How It Works**. 2024. Astera. Disponível em: <https://www.astera.com/pt/type/blog/unstructured-data-processing>. Acesso em: 21 set. 2025. Citado na página 14.

MIHAI, G. Comparison between relational and nosql databases. **Annals of Dunarea de Jos University of Galati Fascicle I Economics and Applied Informatics**, XXVI, p. 38–42, 12 2020. Disponível em: <https://doi.org/10.35219/eai15840409134>. Citado na página 11.

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

PAN, J. J.; WANG, J.; LI, G. Survey of vector database management systems. **The VLDB Journal**, v. 33, n. 46, may 2024. Disponível em: <https://link.springer.com/article/10.1007/s00778-024-00864-x>. Acesso em: 26 set. 2025. Citado 4 vezes nas páginas 11, 17, 18 e 19.

PostgreSQL Global Development Group. **pgvector 0.5.0 Released**. 2024. Acesso em: 13 mar. 2025. Disponível em: <https://www.postgresql.org/about/news/pgvector-050-released-2700/>. Citado 2 vezes nas páginas 18 e 20.

_____. **PostgreSQL Documentation**. [S.l.], 2024. Acesso em: 12 mar. 2025. Disponível em: <https://www.postgresql.org/docs/>. Citado na página 19.

REIMERS, N.; GUREVYCH, I. **Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks**. 2019. Disponível em: <https://doi.org/10.18653/v1/D19-1410>. Citado na página 16.

_____. Making monolingual sentence embeddings multilingual using knowledge distillation. In: **Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing**. Association for Computational Linguistics, 2020. Disponível em: <https://doi.org/10.18653/v1/2020.emnlp-main.365>. Citado na página 16.

REVERSO. **Tradução de 'umarım' em português**. 2025. Reverso Context. Disponível em: <https://context.reverso.net/traducao/turco-portugues/umarim>. Acesso em: 29 set. 2025. Citado na página 52.

ROUSSEEUW, P. J. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. **Journal of Computational and Applied Mathematics**, v. 20, p. 53–65, 1987. Disponível em: <https://doi.org/10.1016/0377-0427(87)90125-7>. Citado na página 22.

SALGUEIRO, M.; LIFSCHITZ, S. Um estudo de modelos e sistemas de bancos de dados para redes sociais. In: **Anais Estendidos do XXXVII Simpósio Brasileiro de Bancos de Dados**. Porto Alegre, RS, Brasil: SBC, 2022. p. 133–139. ISSN 0000-0000. Disponível em: <https://doi.org/10.5753/sbbd_estendido.2022.21855>. Citado na página 23.

SOUSA, V. G. **Estudo de caso do uso de um sistema de gerenciamento de bancos de dados vetoriais para a manipulação, análise e recuperação de áudio**. Dissertação (Trabalho de Conclusão de Curso (Graduação em Ciência da Computação)) — Universidade Federal de Uberlândia, Uberlândia, 2025. 54 f. Disponível em: <https://repositorio.ufu.br/handle/123456789/46301>. Acesso em: 23 set. 2025. Citado na página 24.

SOUZA, B. ; ALMEIDA, T.; MENEZES, A. A.; FIGUEIREDO, C.; NAKAMURA, F.; NAKAMURA, E. Uma abordagem para detecção de tópicos relevantes em redes sociais online. In: **Anais do VI Brazilian Workshop on Social Network Analysis and Mining**. Porto Alegre, RS, Brasil: SBC, 2017. p. 555–566. ISSN 2595-6094. Disponível em: <https://doi.org/10.5753/brasnam.2017.3264>. Citado na página 24.

Statista Research Department. **Big data market size revenue forecast worldwide from 2011 to 2027**. 2023. Statista. Disponível em: <https://www.statista.com/statistics/254266/global-big-data-market-forecast/>. Acesso em: 21 set. 2025. Citado na página 14.

TAN, P.-N.; STEINBACH, M.; KUMAR, V. **Introduction to Data Mining**. Boston, MA: Pearson Education, 2006. Citado na página 20.

TIBURCIO, G. V. **Avaliação Experimental de Classificadores para Análise de Sentimentos em Dados de Redes Sociais**. Dissertação (Trabalho de Conclusão de Curso (Graduação em Estatística)) — Universidade Federal de Uberlândia, Uberlândia, 2021. 61 f. Disponível em: <https://repositorio.ufu.br/handle/123456789/33142>. Acesso em: 23 set. 2025. Citado 2 vezes nas páginas 24 e 27.

TOMAR, A. Elbow method in k-means clustering: Definition, drawbacks, vs. silhouette score. **Built In**, March 13 2025. Acessado em: 23 de setembro de 2025. Disponível em: <https://builtin.com/data-science/elbow-method>. Citado na página 22.

TRIPATHI, R. **What are Vector Embeddings**. 2023. Pinecone. Disponível em: <https://www.pinecone.io/learn/vector-embeddings/>. Acesso em: 21 set. 2025. Citado na página 16.

WANG, J.; YI, X.; GUO, R.; JIN, H.; XU, P.; LI, S.; WANG, X.; GUO, X.; LI, C.; XU, X.; YU, K.; YUAN, Y.; ZOU, Y.; LONG, J.; CAI, Y.; LI, Z.; ZHANG, Z.; MO, Y.; GU, J.; JIANG, R.; WEI, Y.; XIE, C. Milvus: A purpose-built vector data management system. In: **Proceedings of the 2021 International Conference on Management of Data**. New York, NY, USA: Association for Computing Machinery, 2021. (SIGMOD '21), p. 2614–2627. ISBN 9781450383431. Disponível em: <https://doi.org/10.1145/3448016.3457550>. Citado na página 14.

WINNICKI, M. J.; BROWN, C. A.; PORTER, H. L.; GILES, C. B.; WREN, J. D. Biovdb: biological vector database for high-throughput gene expression meta-analysis. **Frontiers in Artificial Intelligence**, v. 7, 2024. ISSN 2624-8212. Disponível em: <https://doi.org/10.3389/frai.2024.1366273>. Citado na página 23.

YOBER, C. **K-Means Clustering Tutorial**. 2018. Disponível em: <https://rpubs.com/cyobero/k-means>. Acesso em: 21 set. 2025. Citado na página 21.

ZILLIZ. **Milvus: um banco de dados vetorial de alto desempenho para aplicações de IA**. 2025. Acessado em: 21 set. 2025. Disponível em: <https://milvus.io/pt/intro>. Citado na página 15.