UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Beatriz Ribeiro Borges

Análise comparativa de algoritmos de classificação de texto

Uberlândia, Brasil

2024

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Beatriz Ribeiro Borges

Análise comparativa de algoritmos de classificação de texto

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

Orientador: Prof. Dr. Paulo Henrique Ribeiro Gabriel

Universidade Federal de Uberlândia – UFU

Faculdade de Computação

Bacharelado em Sistemas de Informação

Uberlândia, Brasil

2024

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:	Sistemas de Informação: Bacharelado				
Defesa de:	FACOM31802 - Trabalho de Conclusão de Curso 2				
Data:	17/10/2024	Hora de início:	19:00	Hora de encerramento:	20:05
Matrícula do Discente:	12021BSI231				
Nome do Discente:	Beatriz Ribeiro Borges				
Título do Trabalho:	Análise comparativa de algoritmos de classificação de textos				
A carga horária curricular foi cumprida integralmente?	(X) Sim ( ) Não				

Reuniu-se no Anfiteatro/Sala 1B230, Campus Santa Mônica, da Universidade Federal de Uberlândia, a Banca Examinadora, designada pelo Colegiado do Curso de Graduação em Sistemas de Informação, assim composta: Professores: Dr. Bruno Augusto Nassif Travençolo - FACOM/UFU; Dr. Rodrigo Sanches Miani - FACOM/UFU; e Dr. Paulo Henrique Ribeiro Gabriel - FACOM/UFU, orientador da candidata.

Iniciando os trabalhos, o presidente da mesa, Dr. Paulo Henrique Ribeiro Gabriel, apresentou a Comissão Examinadora e a candidata, agradeceu a presença do público, e concedeu à 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 o senhor 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) Aprovado(a) Nota 100 (Somente números inteiros)

OU

( ) Aprovado(a) 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 **Paulo Henrique Ribeiro Gabriel**, **Professor(a) do Magistério Superior**, em 17/10/2024, às 20:08, 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 **Bruno Augusto Nassif Travençolo**, **Professor(a) do Magistério Superior**, em 17/10/2024, às 20:08, 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 **Rodrigo Sanches Miani**, **Professor(a) do Magistério Superior**, em 17/10/2024, às 20:08, 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 **5800872** e o código CRC **40784714**.

Referência: Processo nº 23117.071067/2024-01 SEI nº 5800872

Dedico este trabalho à minha avó Inês (in memoriam), que era tudo para mim.

Agradecimentos

Agradeço, primeiramente, à minha mãe, Sueli, que sempre me apoiou incondicionalmente e me deu a coragem e amparo para acreditar que posso ser o que eu quiser, mesmo que uma pessoa das Letras decida se aventurar pelo mundo da computação.

Ao meu namorado, Gustavo, que esteve ao meu lado em cada passo desta jornada, me incentivou a iniciar tudo isso, e ofereceu suporte, carinho e paciência, especialmente nos momentos mais difíceis.

Aos meus professores, que me guiaram e proporcionaram o conhecimento necessário para concluir este trabalho. Sou grata por cada orientação e aprendizado que me ajudaram a chegar até aqui. Em especial, agradeço a Elaine Ribeiro, Paulo Henrique Gabriel, Bruno Travençolo e Rodrigo Miani, que estiveram muito presentes na minha vida acadêmica. Obrigada pelas conversas e pelo apoio durante minha graduação.

"Des hommes raisonnables ? Des hommes détenteurs de la sagesse ? Des hommes inspirés par l'esprit ? Non, ce n'est pas possible." (BOULLE, 2011)

Resumo

A mineração de texto foca na extração de informações de dados não estruturados, como textos em linguagem natural. Diferente da mineração de dados tradicionais, que trabalha com dados estruturados em tabelas, a mineração de texto lida com informações que muitas vezes não seguem um padrão definido. A classificação de texto é uma importante tarefa que visa categorizar automaticamente grandes volumes de dados textuais em classes predefinidas. Embora algoritmos clássicos como *Naive Bayes* e Máquinas de Vetores de Suporte (SVM) sejam amplamente usados, eles têm limitações na captura de nuances semânticas complexas, pois analisam palavras isoladamente e não consideram o contexto completo. O BERT (*Bidirectional Encoder Representations from Transformers*) oferece uma alternativa poderosa ao Processamento de Linguagem Natural (PLN) ao capturar o contexto das palavras bidirecionalmente, o que melhora a qualidade das representações textuais. Essa capacidade avançada fez com que o BERT se tornasse uma escolha mais interessante para diversas tarefas de PLN. Com base nisso, este trabalho compara o desempenho do BERT com algoritmos clássicos em tarefas de classificação de texto, utilizando quatro bases de dados públicas (duas em inglês e duas em português) para análise em diferentes contextos linguísticos e de complexidade com classificações binárias e multiclasses. Os resultados mostram que o BERT supera os algoritmos tradicionais em acurácia e medida F1, embora os métodos clássicos ainda se saiam bem em tarefas simples com menos dados. Além dessas métricas, é relevante considerar o custo computacional do BERT, que é significativamente maior, com tempos de treinamento prolongados e maior demanda de recursos. Portanto, a escolha entre BERT e algoritmos clássicos deve equilibrar desempenho e recursos computacionais disponíveis.

Palavras-chave: BERT; classificação de texto; Naive Bayes; SVM.

Abstract

Text mining focuses on extracting information from unstructured data, such as natural language texts. Unlike traditional data mining, which works with structured data in tables, text mining deals with information that often does not follow a defined pattern. Text classification is an important task that aims to automatically categorize large volumes of textual data into predefined classes. Although classical algorithms such as Naive Bayes and Support Vector Machines (SVM) are widely used, they have limitations in capturing complex semantic nuances, as they analyze words in isolation and do not consider the full context. BERT (Bidirectional Encoder Representations from Transformers offers a powerful alternative to Natural Language Processing (NLP) by capturing the context of words bidirectionally, which improves the quality of textual representations. This advanced capability has made BERT a more interesting choice for several NLP tasks. Based on this, this work compares the performance of BERT with classical algorithms in text classification tasks, using four public databases (two in English and two in Portuguese) for analysis in different linguistic and complexity contexts with binary and multiclass classifications. The results show that BERT outperforms traditional algorithms in accuracy and F1 measure, although classical methods still perform well in simple tasks with less data. In addition to these metrics, it is relevant to consider the computational cost of BERT, which is significantly higher, with longer training times and greater resource demand. Therefore, the choice between BERT and classical algorithms must balance performance and available computational resources.

Keywords: BERT; text classification; Naive Bayes; SVM.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

A mineração de texto é um subprocesso da mineração de dados. Contudo, a representação de dados para mineração de dados clássica e mineração de texto são diferentes. Enquanto os métodos de mineração de dados analisam informações apresentadas em formatos estruturados, os métodos de mineração de texto trabalham com dados inicialmente não estruturados (WEISS; INDURKHYA; ZHANG, 2015). Uma tarefa importante da mineração de texto é a classificação. O principal objetivo da classificação é atribuir itens em uma coleção a categorias ou classes de destino utilizando algoritmos de aprendizado de máquina (KRISHNAIAH; NARSIMHA; CHANDRA, 2014). Nesse sentido, a classificação de texto representa uma etapa fundamental no processamento de linguagem natural (PLN) na qual os dados de texto são automaticamente classificados em um conjunto predefinido de classes (HASSAN; AHAMED; AHMAD, 2022).

Os usos da mineração de texto são vastos e variados, conforme evidenciado por uma série de estudos de caso apresentados por Weiss, Indurkhya e Zhang (2015). Esses estudos destacam aplicações práticas da mineração de texto, como a inteligência de mercado, onde notícias sobre empresas específicas são classificadas para análise. Outro exemplo é a classificação de documentos em bibliotecas digitais que demonstra a necessidade da eficácia da mineração de texto na organização e recuperação de informações. A classificação de problemas em centrais de atendimento é outra amostra de aplicação, facilitando a identificação e resolução rápida de questões.

Weiss, Indurkhya e Zhang (2015) continuam a exemplificação com a classificação de tópicos a artigos de notícias e a filtragem de e-mails, são dois exemplos adicionais de como a mineração de texto pode automatizar processos e melhorar a eficiência operacional. Por fim, a mineração de mídias sociais para análise de sentimentos destaca o papel crucial da classificação de texto na compreensão das opiniões e percepções do público em relação a produtos, serviços e marcas. Esses exemplos evidenciam a versatilidade e relevância da mineração de texto em uma variedade de contextos e setores.

Há diversos algoritmos que desempenham a tarefa de classificação de texto, alguns dos mais conhecidos e amplamente utilizados são *Naive Bayes*, k-Vizinhos Mais Próximos (K-NN), Árvore de Decisão, Floresta Aleatória, Máquina de Vetores de Suporte (SVM) e Regressão Logística. Esses algoritmos necessitam de um pré-processamento para reduzir o tamanho do texto, removendo palavras não relacionadas e inadequadas. Além disso, para que possam ser processados, cada texto é transformado em uma representação numérica na forma de uma representação vetorial (KRISHNAIAH; NARSIMHA; CHANDRA, 2014).

Uma alternativa aos algoritmos citados é o BERT (*Bidirectional Encoder Repre-*

sentations from Transformers, Representações Bidirecionais de Codificadores de Transformadores em português). A proposta do BERT é pré-treinar representações bidirecionais profundas a partir de texto não rotulado, capturando contextos tanto à esquerda quanto à direita em todas as camadas. Esse design permite que o modelo BERT pré-treinado seja ajustado com apenas uma camada de saída adicional, tornando-o versátil para uma variedade de tarefas envolvendo PLN (DEVLIN et al., 2019). Assim, ele se destaca não apenas como um extrator de características, mas também como um classificador. Sua capacidade de compreender o contexto das palavras em uma frase, capturando informações bidirecionalmente, o torna uma ferramenta relevante para tarefas de processamento de linguagem natural.

Conhecer e comparar experimentalmente algoritmos de classificação de texto em diferentes bases de dados é fundamental para o avanço da pesquisa nessa área. Segundo Magalhães, Matos e Souza (2019), é crucial realizar estudos na área de classificação de textos com o objetivo de desenvolver estratégias para classificar automaticamente informações importantes de forma inteligente. Compreender como diferentes algoritmos se comportam em diversas bases de dados permite uma avaliação mais abrangente de sua eficácia e robustez em diferentes contextos.

É preciso destacar que há uma escassez de trabalhos que realizaram comparações entre o uso do BERT e algoritmos clássicos de classificação de texto. Esta lacuna na literatura destaca a necessidade de pesquisas comparativas que abordem tanto o desempenho do BERT quanto de algoritmos tradicionais em diferentes conjuntos de dados e em diferentes idiomas, incluindo o português. Vale ressaltar ainda que, apesar da existência do BERTimbau, uma versão adaptada do BERT para a língua portuguesa, há uma carência de trabalhos comparativos que explorem seu desempenho em relação a outros métodos de classificação de texto, evidenciando uma área promissora para futuras investigações.

1.1 Objetivos

O objetivo geral deste estudo é realizar uma análise comparativa entre o desempenho de algoritmos de classificação de texto, incluindo a língua portuguesa. A pesquisa visa investigar a eficácia dos algoritmos ao lidar com vários conjuntos de dados. Os objetivos específicos deste trabalho são:

- Avaliar o desempenho do modelo BERTimbau em tarefas de classificação de texto em comparação com algoritmos clássicos, como *Naive Bayes* e Máquina de Vetores de Suporte.
- Investigar o impacto do pré-processamento de texto do modelo BERT nas diferentes abordagens de classificação, incluindo técnicas de redução de dimensionalidade e

representação vetorial.

- Analisar as vantagens e desvantagens de cada abordagem em termos de acurácia medida F1.
- Contribuir para preencher a lacuna na literatura em relação à comparação entre o BERT e algoritmos clássicos de classificação de texto, especialmente em língua portuguesa.

Para atender aos objetivos deste estudo, foi adotada a metodologia KDD (Knowledge Discovery in Databases) para a análise comparativa de algoritmos de classificação de texto. O processo KDD é estruturado em várias etapas essenciais: seleção dos dados, pré-processamento, extração de características, aplicação de algoritmos e avaliação dos resultados. Inicialmente, foram selecionadas quatro bases de dados distintas para garantir uma análise abrangente e comparativa. As bases foram submetidas a um pré-processamento que inclui a limpeza dos dados para caracteres especiais e tokenização, seguido pela extração de características usando embeddings gerados pelo BERT e BERTimbau. Em termos de algoritmos, foram empregados o BERT, SVM e Naive Bayes para a classificação dos textos. Por fim, para a avaliação dos modelos, utilizou-se as métricas de acurácia e F1-score.

Os resultados mostraram que o BERT superou os métodos clássicos em termos de acurácia e F1-score. Enquanto o BERT demonstrou superioridade em bases maiores e mais equilibradas, a diferença em relação aos algoritmos clássicos foi menos pronunciada em bases menores. Esses resultados sublinham a importância de considerar o tamanho e a qualidade das bases de dados ao escolher o algoritmo de classificação, além do impacto significativo do poder computacional necessário para treinar modelos avançados como o BERT.

1.2 Organização do Trabalho

Este trabalho está organizado em 6 capítulos, sendo primeiro a introdução. O Capítulo 2 apresenta os principais conceitos que serão utilizados ao longo do estudo, fornecendo uma base teórica para compreensão dos fundamentos da mineração de texto e classificação de documentos. No Capítulo 3, são discutidos os trabalhos relacionados a este estudo.

O Capítulo 4 aborda o desenvolvimento do trabalho, detalhando a metodologia adotada para realizar a análise comparativa entre os algoritmos de classificação de texto. No Capítulo 5, são apresentados os experimentos realizados e a análise dos resultados obtidos. Por fim, no Capítulo 6 estão as principais conclusões deste trabalho, destacando suas contribuições, limitações e sugerindo possíveis direções para pesquisas futuras na área de classificação de texto e processamento de linguagem natural.

2 Fundamentação Teórica

O objetivo deste capítulo é estabelecer uma base teórica para o entendimento e desenvolvimento do estudo, abordando os principais conceitos e técnicas relevantes. A Seção 2.1, descreve Mineração e Classificação de Texto. A Seção 2.2, Coleta de Dados, aborda as estratégias e procedimentos para obtenção de conjuntos de dados. A Seção 2.3, Pré-Processamento, explora as etapas essenciais para preparar os dados de texto para análise, incluindo técnicas como tokenização e normalização (remoção de *stopwords*, padronização do texto, radicalização e lematização). Em seguida, a Seção 2.4, Definição de Características, discute diferentes abordagens para representação de características de texto, como *Bag of Words* (BoW), n-gramas e TF-IDF. A Seção 2.5, Algoritmos de Classificação, apresenta os algoritmos utilizados para classificação de texto neste trabalho, como SVM, *Naive Bayes* e BERT. Por fim, a Seção 2.6, Métodos e Medidas de Avaliação, discute as métricas utilizadas para avaliar o desempenho dos algoritmos de classificação, incluindo matriz de confusão, acurácia, precisão, revocação, medida-F (F1), micromédia e macromédia.

2.1 Mineração e Classificação de Texto

A mineração de dados, conforme descrito por Aggarwal (2015), é um campo de estudo que envolve a coleta, limpeza, processamento, análise e extração de conhecimentos úteis a partir de dados. Esses métodos geralmente requerem um formato altamente estruturado.

Por outro lado, a mineração de texto lida principalmente com coleções de documentos em linguagem natural, em vez de uma série ordenada de números. Embora isso possa parecer uma distinção significativa, é importante notar que a mineração de dados e a mineração de texto estão interconectadas em muitos aspectos. Ambas baseiam-se em amostras de exemplos passados e compartilham métodos de aprendizado semelhantes. Após a aplicação de técnicas de Processamento de Linguagem Natural (PLN), os textos são convertidos em dados estruturados (WEISS; INDURKHYA; ZHANG, 2015).

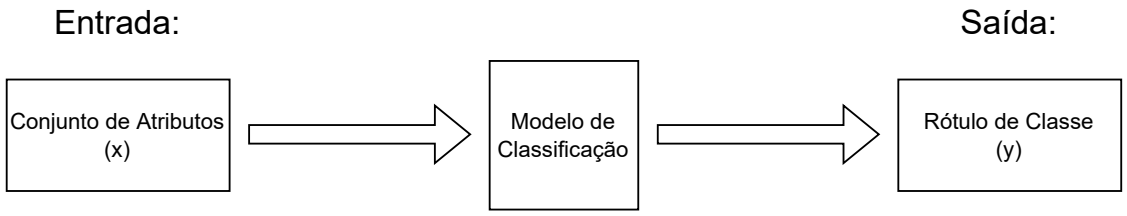
O PLN, conforme mencionado, desempenha um papel fundamental na mineração de texto, sendo subdividido em duas áreas principais: Interpretação (ou Compreensão) de Linguagem Natural e Geração de Linguagem Natural (CASELI; NUNES, 2023). A mineração de texto, em particular, concentra-se na Interpretação de Linguagem Natural e emprega extensivamente suas técnicas no pré-processamento dos dados para análise textual.

Na mineração de dados, uma das principais tarefas é a preditiva, que busca prever o valor de um atributo específico. Essas tarefas podem ser categorizadas em duas principais: classificação e regressão. A regressão é aplicada quando a variável alvo é contínua, enquanto a classificação é utilizada quando a variável alvo é discreta (TAN et al., 2019). Prever a quantidade de chuva em uma determinada localização é uma tarefa de regressão, já que a quantidade de chuva é uma variável contínua que pode assumir valores em um intervalo. Por outro lado, determinar se um e-mail é *spam* ou não é uma tarefa de classificação, pois as variáveis alvo são binárias, ou seja, elas podem assumir apenas um de dois possíveis valores.

Em uma tarefa de classificação, os dados geralmente consistem em uma coleção de instâncias. Cada instância é caracterizada por uma tupla (x, y), onde x representa o conjunto de valores de atributos que descrevem a instância, e y é o rótulo de classe correspondente à instância (TAN et al., 2019). Por exemplo, na classificação de texto, ao construir um modelo para prever se um e-mail é spam ou não, cada e-mail no conjunto de dados seria considerado uma instância. Os atributos podem incluir o assunto do e-mail, o remetente, a contagem de palavras, entre outros. O rótulo de classe indicaria se o e-mail é classificado como spam ou não.

A Figura 1 ilustra a ideia geral por trás da tarefa de classificação, apresentando um modelo abstrato do seu funcionamento. Nesse contexto, um modelo de classificação é uma representação abstrata da relação entre o conjunto de atributos e o rótulo de classe.

Figura 1 – Modelo abstrato do funcionamento da tarefa de classificação.



Fonte: Elaborado pela autora (2024)

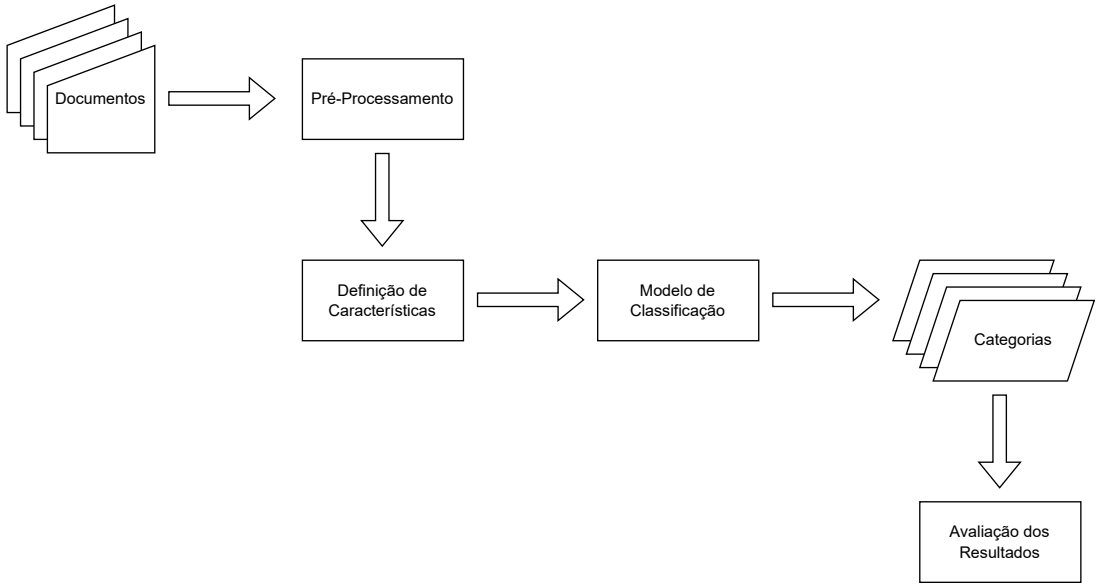
De acordo com Tan et al. (2019), um modelo de classificação pode ser expresso matematicamente como uma função alvo f que recebe como entrada o conjunto de atributos x e produz uma saída correspondente ao rótulo de classe previsto. O modelo classifica corretamente uma instância x se a previsão do modelo, ou seja, $f(x)$, é igual ao verdadeiro rótulo de classe y. Esse modelo é geralmente construído usando um algoritmo de aprendizado de máquina, que aprende essa relação a partir de um conjunto de dados de treinamento.

Quando se explora a mineração de texto, os dados processados são denominados

documentos e as classes podem ser chamadas de categorias. O processo de inserção dos documentos em classes, ou seja, a associação de um ou mais rótulos de classe em cada documento, é comumente chamado de classificação de textos (BAEZA-YATES; RIBEIRO-NETO, 2013). O objetivo da classificação de texto é dividir uma coleção de documentos em um conjunto de categorias predefinidas, como tecnologia, esportes ou entretenimento (ZONG; XIA; ZHANG, 2021).

A Figura 2 ilustra as etapas da classificação de texto. Devido à natureza não estruturada dos documentos, é necessário realizar um pré-processamento após a coleta dos dados antes de inseri-los no modelo de classificação. Em seguida, definem-se as características (conjunto de atributos), estruturando os dados. Finalmente, avança-se para a construção dos modelos de classificação, utilizando técnicas de aprendizado de máquina. Após a conclusão da classificação, os resultados são analisados aplicando métodos e medidas de avaliação adequados.

Figura 2 – Modelo abstrato do funcionamento da tarefa de classificação de texto.



Fonte: Elaborado pela autora (2024)

2.2 Coleta de Dados

O processo inicial na mineração de texto é a etapa de coleta de dados, que pode envolver a utilização de diferentes recursos e métodos. Isso inclui a utilização de *hardware* especializado, como redes de sensores, ou métodos manuais, como a coleta de pesquisas de usuários. Além disso, ferramentas de *software*, como mecanismos de rastreamento de

documentos da internet, são empregadas para coletar documentos relevantes (AGGARWAL, 2015).

As fontes e os métodos de aquisição de dados variam conforme a natureza das tarefas de mineração de texto. Existem basicamente duas situações em relação às fontes de dados: domínio aberto (público) e fechado (privado). No caso de dados de domínio aberto os dados são obtidos de redes sociais públicas disponíveis, abrangendo diversos tópicos e fontes. Por outro lado, os dados de domínio fechado estão restritos a redes privados internas e não podem ser acessados por usuários externos. Nesses casos, é necessário complementar os dados com informações obtidas de fontes públicas. Vale ressaltar que os dados provenientes de redes públicas, especialmente das mídias sociais, tendem a conter mais ruído e expressões mal formadas, o que demanda mais tempo para limpeza e pré-processamento (ZONG; XIA; ZHANG, 2021).

2.3 Pré-Processamento

Após a coleta inicial de dados, muitas vezes estes não estão em uma forma adequada para o processamento. Para tornar os dados adequados para o processamento, é essencial transformá-los em um formato que seja facilmente compreensível e utilizável pelos algoritmos de mineração (AGGARWAL, 2015). Para atingir esse objetivo, são empregadas diversas técnicas, entre as quais se destacam a tokenização, remoção de *stopwords*, padronização do texto e normalização.

2.3.1 Tokenização

A tokenização desempenha um papel crucial no Processamento de Linguagem Natural (PLN), pois representa o primeiro passo nesse processo e pode influenciar significativamente as etapas subsequentes (LANE; DYSHEL, 2021). Ao quebrar o texto não estruturado em linguagem natural em unidades discretas de informação, um tokenizador permite uma manipulação mais precisa e eficiente dos dados.

No contexto do PLN, essa separação em unidades linguísticas mínimas é denominada tokenização. No caso da língua portuguesa, a tokenização é realizada através da identificação dos limites das palavras utilizando delimitadores como espaços em branco ou símbolos de pontuação, como vírgulas, pontos e hifens (CASELI; NUNES, 2023).

No Exemplo 1.1, é apresentado o processo de tokenização utilizado na frase do Exemplo 1.

Exemplo 1: Eu serei rei, fiquem comigo e jamais sentirão fome outra vez! [1]

[1] Fonte: O Rei Leão. Direção de Rob Minkoff e Roger Allers. Estados Unidos da América: Walt Disney Pictures, 1994

Exemplo 1.1 - Após Tokenização: [‘Eu’, ‘serei’, ‘rei’, ‘,’, ‘fiquem’, ‘comigo’, ‘e’, ‘jamais’, ‘sentirão’, ‘fome’, ‘outra’, ‘vez’, ‘!’]

Caseli e Nunes (2023) ressaltam que “é necessário se atentar para casos específicos em que algumas pontuações não indicam separação”. A frase do Exemplo 2 exemplifica essa situação. No Exemplo 2.1, é possível observar a aplicação de sua tokenização.

Exemplo 2: Você receberá instruções sobre sua tarefa na sala de conferência às 2:00, ala D, sala A-113 [2]

Exemplo 2.1 - Após Tokenização: [‘Você’, ‘receberá’, ‘instruções’, ‘sobre’, ‘sua’, ‘tarefa’, ‘na’, ‘sala’, ‘de’, ‘conferência’, ‘às’, ‘2:00’, ‘,’, ‘ala’, ‘D’, ‘,’, ‘sala’, ‘A-113’, ‘.’]

O Exemplo 2.2 ilustra uma tokenização inadequada que considera o termo ‘2:00’ como três *tokens* separados, assim como ‘A-113’. Ambos devem ser tratados como um *token* único como mostrado no Exemplo 2.1.

Exemplo 2.2 - Após Tokenização Errônea: [‘Você’, ‘receberá’, ‘instruções’, ‘sobre’, ‘sua’, ‘tarefa’, ‘na’, ‘sala’, ‘de’, ‘conferência’, ‘às’, ‘2’, ‘:’, ‘00’, ‘,’, ‘ala’, ‘D’, ‘,’, ‘sala’, ‘A’, ‘-’, ‘113’, ‘.’]

2.3.2 Normalização

Normalização é um termo de aprendizado de máquina com alguns significados diferentes. No caso da mineração de texto, a normalização visa reduzir o número de características eliminando ou combinando diferentes *tokens* em uma única classe (BENGFORT; BILBRO; OJEDA, 2016). A normalização, então, visa lidar com *stopwords* (palavras estruturais de alta frequência), pontuação e a variação entre letras maiúsculas e minúsculas. Além disso, conforme mencionado em Bengfort, Bilbro e Ojeda (2016), as palavras com afixo ou formas morfológicas que indicam gênero, pluralidade, tempo, etc., podem ser agrupadas em uma única classe de palavras.

2.3.2.1 Remoção de *Stopwords*

As *stopwords* referem-se principalmente a palavras funcionais de alta frequência que aparecem em vários documentos com pouca informação de texto (ZONG; XIA; ZHANG, 2021). Na língua portuguesa, essas *stopwords* incluem preposições, pronomes, alguns advérbios, conjunções, artigos e, muitas vezes, verbos de ligação.

[2] Fonte: Os Incríveis. Direção de Brad Bird. Estados Unidos da América: Pixar Animation Studios/Walt Disney Pictures, 2004

O processo de remoção de *stopwords* é bastante simples e consiste em verificar a presença de cada palavra do texto em uma lista de *stopwords* previamente construída. Existem listas prontas que são passíveis de usar, como a do NLTK[3] para *Python* (CASELI; NUNES, 2023). A lista do NLTK contém 207 palavras, no entanto, não abrange todas as palavras das classes gramaticais mencionadas. Além disso, é importante considerar que pode haver palavras relevantes para o contexto da mineração de texto que não estejam presentes nesta lista.

No Exemplo 3, é apresentado o resultado da remoção de stopwords do Exemplo 1 com base na biblioteca NLTK.

Exemplo 3 - Remoção de *Stopwords*: [‘rei’, ‘,’, ‘fiquem’, ‘comigo’, ‘jamais’, ‘sentirão’, ‘fome’, ‘outra’, ‘vez’, ‘!’]

Caseli e Nunes (2023) ainda destacam que é preciso analisar a lista de *stopwords* para garantir sua adequação ao contexto da mineração de texto. A remoção indiscriminada pode resultar na perda de informações importantes. As autoras exemplificam a famosa expressão “ser ou não ser, eis a questão”. Se a remoção de *stopwords* for aplicada de forma cega, restaria apenas “eis” e “questão”, o que descaracterizaria completamente a frase e não representaria adequadamente o texto original.

2.3.2.2 Padronização do texto

Nesse processo, é comum descartar os caracteres de pontuação e manter apenas as palavras, também conhecidas como termos. Após a tokenização, também é feita a conversão dos caracteres para letras minúsculas. Caseli e Nunes (2023) explicam que essa prática visa garantir que a busca e análise não seja sensível a maiúsculas e minúsculas, facilitando a localização de palavras-chave independentemente de sua grafia.

A remoção de acentos, que também pode ser chamada de remoção de sinais diacríticos, é uma boa prática que pode ser realizada (CASELI; NUNES, 2023). O NLTK não possui uma função nativa para a remoção de acentuação. Entretanto, outras bibliotecas[4] Python podem lidar com essa tarefa. Uma biblioteca comum e útil para isso é a “unidecode”, que é capaz de transliterar caracteres Unicode com acentos para caracteres ASCII.

2.3.2.3 Radicalização e Lematização

A lematização, adaptada do termo em inglês *lemmatization*, e a radicalização (*stemming*, em inglês) são dois processos adicionais para simplificar e reduzir a variabilidade dos termos em um documento. Ambos consistem em análises léxicas para normalizar o texto e tornar a análise de dados textuais mais eficiente no campo do PLN.

[3] <https://www.nltk.org/howto/portuguese_en.html>
[4] <https://docs.python.org/3.12/library/index.html>

Radicalização

O algoritmo de radicalização é um método que reduz as palavras ao seu radical ou tema. Esse procedimento é realizado por meio de algoritmos de busca, que identificam a ocorrência de determinados radicais em cada palavra (SANTOS et al., 2022).

A biblioteca NLTK oferece módulos, com versões em português, como a nltk.stem[5], para a radicalização de palavras, utilizando o algoritmo desenvolvido por Martin Porter. Esse algoritmo é capaz de eliminar os sufixos (e, em alguns casos, os prefixos) das palavras, facilitando a análise de textos.

A Tabela 1 mostra a radicalização de algumas palavras:

Tabela 1 – Exemplo de tabela com radicalização de palavras.

Palavra original	**Radicalização**
correndo	corr
corrida	corr
corridas	corr
prato	prat
prata	prat
felizmente	feliz
infelizmente	infeliz
menina	menin
meninos	menin

Fonte: Elaborado pela autora (2024)

É importante ressaltar que a radicalização pode ser imprecisa e generalizadora. Como explicam Santos et al. (2022), essa imprecisão decorre, em parte, da falta de consideração pelo caráter dinâmico e evolutivo das línguas. Ao longo do tempo, algumas palavras que compartilham um radical podem adquirir significados próprios incompatíveis entre si. Isso é visível ao observar a Tabela 1. Percebe-se que as palavras "prato" e "prata" possuem o mesmo radical de acordo com o algoritmo de *stemmer* do NLTK. No entanto, essas palavras têm significados distintos na língua portuguesa. Além disso, é importante notar que as palavras "felizmente" e "infelizmente" possuem radicais diferentes. Embora gramaticalmente essas palavras compartilhem o mesmo radical, é importante manter essa distinção no algoritmo, especialmente na mineração de texto, cuja recuperação dessas palavras como opostas uma da outra é essencial para atribuir significados diferentes ao texto.

[5] <https://www.nltk.org/_modules/nltk/stem/rslp.html>

Lematização

A lematização consiste num processo para encontrar o lema de cada palavra, ou seja, sua forma morfológica semântica e sintaticamente canônica (SANTOS et al., 2022). O processo de lematização é a tarefa de determinar o lema de uma palavra em um determinado contexto ou na sua forma flexionada. Como explicam Santos et al. (2022):

> "A lematização segue a premissa de que, ao se reduzir todas as inflexões possíveis que as palavras podem assumir (como, por exemplo, em relação a tempos verbais, gênero, número ou grau), o vocabulário do corpus é simplificado, facilitando a capacidade preditiva dos modelos de aprendizado de máquina a serem aplicados. Através da lematização, todas as ocorrências de variações do verbo estar (como, por exemplo, 'estou', 'estaremos', 'estará'), seriam substituídas pela forma canônica do verbo".

A Tabela 2 apresenta quatro exemplos de palavras em sua forma original e após a lematização:

Tabela 2 – Tabela de Lematização.

Classe Gramatical	**Palavra Original**	**Lematização**
Verbo	correndo	correr
Substantivo	meninas	menina
Adjetivo	bonitas	bonito
Advérbio	rapidamente	rápido

Fonte: Elaborado pela autora (2024)

A tarefa de lematização envolve, frequentemente, a consulta a recursos linguísticos que possuem a definição de lemas e morfologia das palavras. O grande desafio dessa tarefa é a desambiguação sintática. Por exemplo, na sentença "Quem casa, quer casa", a palavra "casa" na sua primeira ocorrência terá como lema o verbo "casar", enquanto a segunda ocorrência terá como lema o substantivo "casa" (CASELI; NUNES, 2023).

2.4 Definição de Características

Definir as características que serão usadas para classificar os dados é crucial para criar um classificador eficaz. Essas características, chamadas de *features*, precisam ser atributos que possibilitem uma boa diferenciação entre os dados a serem classificados (BENEVENUTO; ARAÚJO; RIBEIRO, 2015). Por exemplo, ao classificar três animais, como tigres, cachorros e lhamas, pode-se considerar várias características, como filo, classe, família, tamanho, alimentação, tipo de vocalização e habitat. No entanto, algumas dessas características, como reino e classe, não os distinguiriam, pois todos os três pertencem ao filo dos cordados e à classe dos mamíferos. Portanto, é crucial selecionar características

relevantes para diferenciá-los dos demais. Para definir as características, são utilizados diversos métodos, tais como *Bag of Words*, N-gramas e TF-IDF (*Term Frequency-Inverse Document Frequency*).

2.4.1 *Bag of Words*

Bag of Words (saco de palavras) é uma representação vetorial binária que indica a presença ou ausência de termos de um documento em um vocabulário. Como explicado em Baeza-Yates e Ribeiro-Neto (2013), "o vocabulário é o conjunto de todos os termos de indexação distintos na coleção". Quando um termo ocorre em um documento, isso estabelece uma relação entre eles. Essas relações entre termos e documentos podem ser quantificadas, por exemplo, pela frequência do termo no documento (BAEZA-YATES; RIBEIRO-NETO, 2013).

A Tabela 3 ilustra uma *Bag of Words*. Na primeira coluna à esquerda estão as frases[6] que representam os documentos. Na primeira linha está o vocabulário, ou seja, todas as palavras contidas nos documentos sem repetição. Se uma palavra do vocabulário está presente no documento, é representada por 1; caso contrário, é representada por 0.

Tabela 3 – Exemplo de *Bag of Words*.

	Bag of Words											
	Se	**preparem**	**para**	**o**	**golpe**	**do**	**século**	**ter**	**nova**	**vida**	**uma**	**sensacional**
Se preparem para o golpe do século	1	1	1	1	1	1	1	0	0	0	0	0
Se preparem para ter nova vida	1	1	1	0	0	0	0	1	1	1	0	0
Uma vida sensacional	0	0	0	0	0	0	0	0	0	1	1	1

Fonte: Elaborado pela autora (2024)

Ao considerar "Se preparem para o golpe do século" como o documento 1 (D1), "Se preparem para ter nova vida" o documento 2 (D2) e "Uma vida sensacional" o documento 3 (D3), sua representação vetorial seria a seguinte:

D1 = {1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0}.

D2 = {1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0}.

D3 = {0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1}.

O tamanho do vetor será sempre o mesmo do tamanho do vocabulário.

2.4.2 N-gramas

Para formar a representação vetorial de um documento, o modelo *Bag of Words* corresponde e conta separadamente cada elemento no documento, negligenciando muitas

[6] Fonte: O Rei Leão. Direção de Rob Minkoff e Roger Allers. Estados Unidos da América: Walt Disney Pictures, 1994

informações de correlação entre as palavras (YAN et al., 2020). Por outro lado, o N-grama pode analisar as palavras em sequências de acordo com a quantidade definida.

A extração de pares (bigramas), tríplices (trigramas), quádruplas (quatro gramas) e até quíntuplas (cinco gramas) de *tokens* é chamada de N-gramas. Usar N-gramas permite que sua máquina saiba sobre os *tokens* individuais e compostos (LANE; DYSHEL, 2021). Um exemplo em português que ilustra bem esse conceito é a expressão "leite condensado". Sintaticamente, "leite" é um substantivo e "condensado" é um adjetivo cada palavra com seu significado individual. Juntas, essas palavras formam uma expressão composta. Portanto, "leite condensado" pode ser considerado uma locução substantiva, que é uma expressão formada por duas ou mais palavras que, em conjunto, tem a função de um substantivo, adquirindo um novo significado.

2.4.3 TF-IDF

Ao analisar um conjunto de termos para indexar um documento, observa-se que nem todos os termos são igualmente úteis para descrever seu conteúdo. A presença ou ausência de um termo em um documento por si só não é suficiente. Existem termos mais gerais e outros mais específicos. Uma palavra que aparece em todos os documentos de uma coleção não é útil como termo de indexação, enquanto uma palavra que aparece em apenas alguns documentos é mais útil, pois reduz o conjunto de documentos relevantes para o usuário (BAEZA-YATES; RIBEIRO-NETO, 2013).

A frequência do termo (TF, *Term Frequency*) e a frequência inversa de documento (IDF, *Inverse Document Frequency*) são os fundamentos da ponderação de termos TF-IDF. TF mede a importância de um termo em um documento, calculada com base na frequência de ocorrências (f) de um termo i em um documento j, ilustrada na Equação 2.1:

$$TF_{i,j} = f_{i,j} \tag{2.1}$$

Uma variante da ponderação TF utilizada na literatura é mostrada na Equação 2.2, cujo logaritmo utiliza base 2. A expressão com o logaritmo é a forma preferível para a ponderação TF, porque torna os pesos diretamente comparáveis aos pesos IDF (BAEZA-YATES; RIBEIRO-NETO, 2013).

$$TF_{i,j} = 1 + \log_2(f_{i,j}) \tag{2.2}$$

Essa medida é importante, pois as palavras que aparecem com mais frequência em um documento geralmente são mais relevantes para distinguir seu conteúdo. No entanto, a TF por si não é suficiente para identificar as palavras mais importantes, já que algumas

palavras podem ser muito frequentes em muitos documentos e, portanto, não contribuem para a distinção do conteúdo (CASELI; NUNES, 2023).

Já o IDF é fundamental para atribuir um peso maior às palavras que são frequentes, mas ocorrem em poucos documentos de uma coleção. O IDF é calculado dividindo o número total de documentos da coleção pela quantidade de documentos que contêm a palavra em questão e tomando o logaritmo desse resultado, conforme Equação 2.3:

$$IDF_{i,j} = \log\left(\frac{N}{n_i}\right), \tag{2.3}$$

onde N representa o número total de documentos na coleção e n_i é o número de documentos que contêm o termo i.

O IDF avalia a raridade de uma palavra em um conjunto de documentos, sendo importante, pois palavras que aparecem em poucos documentos têm um maior poder de distinção do conteúdo de um documento (CASELI; NUNES, 2023).

O TF-IDF de um termo t de um documento d é dado pelo produto entre o valor TF (definido pela Equação 1) e o valor de IDF (definido pela Equação 2), conforme a Equação 2.4 e 2.5.

$$\text{TF-IDF}_{i,j} = \text{TF}_{i,j} \times \text{IDF}_{i,j} \tag{2.4}$$

Substituindo as expressões:

$$\text{TF-IDF}_{i,j} = [1 + \log(f_{i,j})] \times \log\left(\frac{N}{n_i}\right) \tag{2.5}$$

O TF-IDF assume que os recursos mais discriminativos são aqueles que aparecem com frequência no documento atual e raramente em outros documentos (ZONG; XIA; ZHANG, 2021). Assim, um termo que ocorre muitas vezes em um documento, mas não em muitos documentos da coleção, terá um peso mais alto, enquanto um termo que ocorre em muitos documentos terá um peso mais baixo.

Considerando uma coleção de documentos que inclui os seguintes itens:

D1 = Se preparem para o golpe do século.

D2 = Se preparem para ter nova vida, se preparem!

D3 = Uma vida sensacional!

Com o seguinte vocabulário:

V = {Se, preparem, para, o, golpe, do, século, ter, nova, vida, uma, sensacional}.

A Tabela 4 mostra um exemplo da frequência dos termos e os valores de TF logarítmicos.

Tabela 4 – Exemplo de frequência e valores do TF para a coleção exemplo.

#	Termo	$F_{i,1}$	$F_{i,2}$	$F_{i,3}$	$TF_{i,1}$	$TF_{i,2}$	$TF_{i,3}$
1	**Se**	1	2	0	1	2	-
2	**preparem**	1	2	0	1	2	-
3	**para**	1	1	0	1	1	-
4	**o**	1	0	0	1	-	-
5	**golpe**	1	0	0	1	-	-
6	**do**	1	0	0	1	-	-
7	**século**	1	0	0	1	-	-
8	**ter**	0	1	0	-	1	-
9	**nova**	0	1	0	-	1	-
10	**vida**	0	1	1	-	1	1
11	**uma**	0	0	1	-	-	1
12	**sensacional**	0	0	1	-	-	1
Tamanho do documento		7	8	3			

Fonte: Elaborado pela autora (2024)

A Tabela 5 mostra o exemplo do IDF. Já a Tabela 6, mostra os valores do TF-IDF de cada termo em cada documento.

Tabela 5 – Valores de IDF para a coleção exemplo com N=3.

#	Termo	ni	IDFi
1	**Se**	2	0,584963
2	**preparem**	2	0,584963
3	**para**	2	0,584963
4	**o**	1	1,584963
5	**golpe**	1	1,584963
6	**do**	1	1,584963
7	**século**	1	1,584963
8	**ter**	1	1,584963
9	**nova**	1	1,584963
10	**vida**	2	0,584963
11	**uma**	1	1,584963
12	**sensacional**	1	1,584963
Total de documento (N)		3	

Fonte: Elaborado pela autora (2024)

Quanto maior o valor do TF-IDF para um termo em um documento específico, mais importante e exclusivo esse termo é considerado para esse documento em comparação com os outros documentos na coleção.

Tabela 6 – Valores de TF-IDF para a coleção exemplo.

		TF-IDF		
#	**Termo**	**D1**	**D2**	**D3**
1	**Se**	0,584963	1,169925	-
2	**preparem**	0,584963	1,169925	-
3	**para**	0,584963	0,584963	-
4	**o**	1,584963	-	-
5	**golpe**	1,584963	-	-
6	**do**	1,584963	-	-
7	**século**	1,584963	-	-
8	**ter**	-	1,584963	-
9	**nova**	-	1,584963	-
10	**vida**	-	0,584963	0,584963
11	**uma**	-	-	1,584963
12	**sensacional**	-	-	1,584963

Fonte: Elaborado pela autora (2024)

2.5 Algoritmos de Classificação

A classificação de textos é fortemente influenciada pelo campo do aprendizado de máquina, o qual se dedica a desenvolver e implementar algoritmos para identificar padrões nos dados de entrada. Esses algoritmos dependem de uma fase de aprendizado para construir modelos ou funções que capturem esses padrões. Existem três tipos principais de algoritmos de aprendizado de máquina: supervisionado, não supervisionado e semi-supervisionado (BAEZA-YATES; RIBEIRO-NETO, 2013).

No aprendizado supervisionado, o objetivo é adquirir o conhecimento de um mapeamento da entrada para a saída, com os valores corretos da saída fornecidos por um supervisor (ALPAYDIN, 2014). Por outro lado, no aprendizado não supervisionado, não há um supervisor para orientar o processo, e apenas os dados de entrada estão disponíveis. Nesse contexto, o objetivo principal é identificar padrões ou regularidades nos dados de entrada sem a orientação de um supervisor (ALPAYDIN, 2014). No aprendizado semi-supervisionado, são fornecidos apenas alguns exemplos rotulados, e o objetivo é extrair o máximo possível de informações de uma grande coleção de exemplos não rotulados (RUSSELL; NORVIG, 2013). Além dos três tipos mencionados, Russell e Norvig (2013) também abordam o aprendizado por reforço, na qual um agente aprende com base em uma série de recompensas ou punições recebidas.

A classificação automática de documentos de texto utiliza diferentes técnicas de aprendizado de máquina aplicadas a diferentes coleções de documentos (GUIMARÃES; MEIRELES; ALMEIDA, 2019). A escolha da técnica mais apropriada depende do contexto

específico do problema e das características dos dados textuais em questão.

2.5.1 Máquinas de Vetores de Suporte

Máquinas de Vetores de Suporte (SVM, do inglês *Support Vector Machines*) é um algoritmo de aprendizado supervisionado usado para classificação binária. Baseia-se em duas ideias fundamentais: a separabilidade linear dos dados e a possibilidade de separação não linear utilizando funções de kernel (ZONG; XIA; ZHANG, 2021). Os documentos são representados como pontos ou vetores em um espaço dimensional. A tarefa é encontrar um hiperplano de decisão que possa separar eficazmente os elementos em duas classes distintas, Ca e Cb. Quando os dados podem ser separados perfeitamente por um hiperplano, o conjunto de dados é chamado de linearmente separável. Os problemas de classificação são geralmente linearmente separáveis (BAEZA-YATES; RIBEIRO-NETO, 2013). A equação geral de um hiperplano separador é expressa como:

$$wx + b = 0, \tag{2.6}$$

onde x representa os atributos de um dado objeto. w é o vetor de pesos (ou vetores de coeficientes) que define a direção do hiperplano, o vetor de pesos controla a inclinação e a orientação do hiperplano no espaço de características. b é o termo de polarização que define a posição do hiperplano no espaço de características, controla a sua posição ao longo da direção definida pelo vetor de pesos. Dependendo do sinal da equação, uma instância de dados pode pertencer a qualquer lado do hiperplano (TAN et al., 2019). Os vetores de pesos w e o termo de polarização b são ajustados no treinamento para encontrar o hiperplano de separação ótimo que melhor separa as classes no espaço de características.

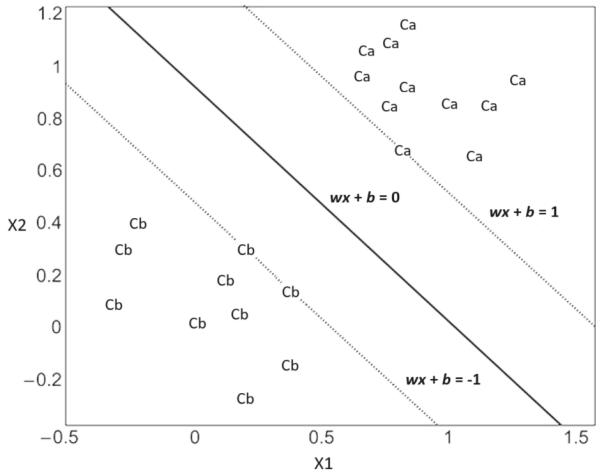
A Figura 3 exemplifica um conjunto de dados linearmente separáveis com duas classes, Ca e Cb. O hiperplano que zera a equação geral maximiza as distâncias para os documentos mais próximos de cada classe, servindo como o melhor hiperplano separador. O resultado positivo classifica o documento como Ca e o resultado negativo como Cb.

Na mesma Figura 3, as linhas pontilhadas paralelas delimitam a região onde se deve procurar uma solução, chamados de hiperplanos delimitadores. Um documento que pertence a um hiperplano delimitador é chamado de vetor de suporte. As linhas paralelas aos hiperplanos delimitadores são consideradas as melhores opções para hiperplanos de decisão (BAEZA-YATES; RIBEIRO-NETO, 2013).

2.5.2 *Naive Bayes*

Os classificadores probabilísticos, como *Naive Bayes*, constroem um modelo que quantifica a relação entre as variáveis de característica e a variável de destino (classe) como probabilidade (AGGARWAL, 2015). Baeza-Yates e Ribeiro-Neto (2013) desenvolvem que

Figura 3 – Vetores de suporte de um hiperplano com 2 classes Ca e Cb.



Fonte: Adaptado de Tan et al. (2019)

os classificadores probabilísticos trabalham de maneira análoga ao modelo probabilístico clássico. Dado um documento d_j e uma classe c_p, é atribuído a cada par de documento-classe $[d_j, c_p]$ a probabilidade de que d_j pertença à classe c_p. Uma vez que essas probabilidades condicionais foram calculadas para todas as classes, o classificador atribui aos documentos as classes com as maiores estimativas de probabilidade. Para computar a probabilidade, um classificador probabilístico aplica o teorema de Bayes:

$$Pr(c_p|d_j) = \frac{Pr(c_p) \times Pr(d_j|c_p)}{Pr(d_j)} \tag{2.7}$$

- **Probabilidade Condicional:** $Pr(c_p|d_j)$ é a probabilidade condicional de uma classe c_p dado um documento d_j, ou seja, é a probabilidade de um documento específico d_j ser classificado como pertencendo à classe c_p, com base nas características desse documento.
- **Probabilidade a Priori:** $Pr(c_p)$ é a probabilidade a priori da classe c_p, ou seja, a probabilidade de selecionar aleatoriamente um documento dessa classe dentre todas as classes.
- **Probabilidade Condicional do Documento:** $Pr(d_j|c_p)$ é a probabilidade condicional do documento d_j dado que ele pertence à classe c_p, ou seja, é a probabilidade de observar um documento específico d_j sabendo que ele é da classe c_p, considerando a distribuição dos documentos dentro da classe.
- **Probabilidade do Documento:** $Pr(d_j)$ é a probabilidade marginal de ocorrer um documento d_j no contexto do conjunto de todos os documentos, ou seja, é a probabilidade de se observar o documento independentemente de sua classe. É usada como fator de normalização na equação de *Bayes*.

Quando há duas classes, C_1 e C_2, $Pr(x)$ é prontamente computado como na Equação 2.8, onde C é o rótulo de classe trabalhada e x um vetor do documento que denota a presença ou ausência de palavras de um dicionário (WEISS; INDURKHYA; ZHANG, 2015).

$$Pr(x) = Pr(x|C_1)Pr(C_1) + Pr(x|C_2)Pr(C_2) \tag{2.8}$$

Weiss, Indurkhya e Zhang (2015) explica que a chave para usar essas equações é calcular $Pr(x|C)$. Assumindo-se que as palavras são independentes, em vez de procurar a probabilidade do vetor completo x, é possível procurar a probabilidade da presença ou ausência de cada palavra, $Pr(x_j|C)$, e multiplicá-las todas juntas:

$$Pr(x) = \sum_c Pr(C) \prod_j Pr(x_j|C) \tag{2.9}$$

- A probabilidade $\Pr(C)$ é determinada pela frequência ($freq$) de documentos rotulados como C dividida por n, o número total de documentos:

$$Pr(C) = \frac{freq(C)}{n} \tag{2.10}$$

- $Pr(x_j = 1|C)$ é a probabilidade de x_j ser 1 dado que o documento pertence à classe C. Esta probabilidade é calculada pela frequência de documentos rotulados como C que contêm a palavra w_j, dividida pela frequência total de documentos rotulados como C:

$$Pr(x_j = 1|C) = \frac{freq(x_j = 1, rótulo = C)}{freq(C)} \tag{2.11}$$

- $Pr(x_j = 0|C)$ é a probabilidade de x_j ser 0 dado que o documento pertence à classe C. Como só existem duas possibilidades para x_j (1 ou 0), se a probabilidade de x_j ser 1 é $Pr(x_j = 1|C)$, então a probabilidade de x_j ser 0 será simplesmente:

$$Pr(x_j = 0|C) = 1 - Pr(x_j = 1|C) \tag{2.12}$$

Então, dado um vetor de características binárias x, a pontuação de probabilidade da classe C é:

$$Pr(C|x) = \frac{\Pr(C) \prod_j Pr(x_j = 0|C)}{Pr(x)} \prod_j \left(\frac{Pr(x_j = 1|C)}{Pr(x_j = 0|C)} \right)^{x_j} \tag{2.13}$$

2.5.2.1 *Naive Bayes* Multinomial

Naive Bayes clássico apenas trabalha a presença e ausência de palavras em um documento e ignora sua frequência. Em comparação, sua distribuição multinomial é usada com mais frequência e geralmente tem melhor desempenho de classificação (ZONG; XIA; ZHANG, 2021). Baeza-Yates e Ribeiro-Neto (2013) propõem que uma vez que as frequências dos termos podem ser utilizadas para melhorar a qualidade dos resultados, é possível ajustar o classificador para considerá-las, tornando assim o *Naive Bayes* Multinomial a escolha mais indicada.

Para o *Naive Bayes* Multinomial, a equação 2.13 pode ser reescrita da seguinte maneira (WEISS; INDURKHYA; ZHANG, 2015):

$$Pr(C|x) = \frac{1}{\Pr(x)} \cdot \exp\left(\sum_j w_j \cdot x_j + b\right) \tag{2.14}$$

- $Pr(C|x)$ é a probabilidade posterior da classe C dado x.
- $Pr(x)$ é a probabilidade marginal de x.
- w_j é o peso associado à presença da característica x_j. Valores maiores de w_j indicam que a presença de x_j torna mais provável a classe C, enquanto valores menores indicam o contrário.
- b é o termo de viés.

Nesse contexto, o valor do peso w_j é dado por:

$$w_j = \ln\left(\frac{\lambda + freq(x_j = 1, label = C)}{\lambda m + \sum_{j'=1}^{m} freq(x_{j'} = 1, label = C)}\right), \tag{2.15}$$

sendo que

- λ é um parâmetro de suavização, $\lambda > 0$.
- $freq(x_j = 1, label = C)$ é a frequência da característica x_j na classe C.
- m é o número de características, vocabulário global.

Já o termo de viés (b) é calculado da seguinte maneira:

$$b = \ln\left(\frac{freq(label = C)}{n}\right), \tag{2.16}$$

onde:

- $freq(label = C)$ representa a frequência da classe C.
- n é o número total de exemplos, documentos de treinamento.

O modelo multinomial é frequentemente usado em aplicações de categorização de texto. Ele normaliza o comprimento de um documento, o que geralmente leva a um desempenho ligeiramente melhor. O parâmetro λ é frequentemente definido como 1 na literatura. No entanto, um valor menor (como 0,01), às vezes pode ser mais eficaz (WEISS; INDURKHYA; ZHANG, 2015). Dessa forma, é possível perceber que existem vários métodos para treinar os pesos lineares diretamente.

2.5.3 BERT

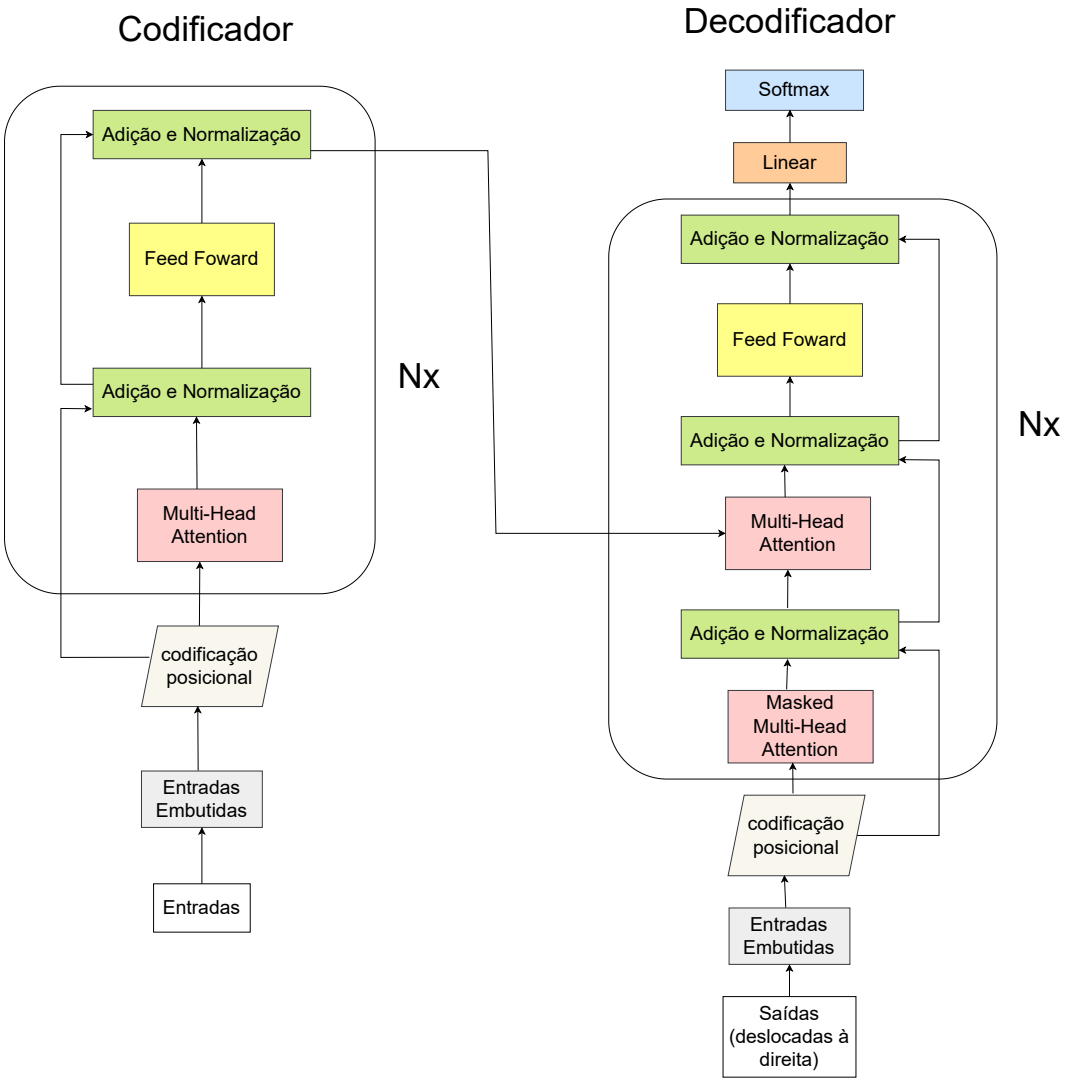
BERT, que significa *Bidirectional Encoder Representations from Transformers* (Representações de Codificadores Bidirecionais de Transformadores), é um modelo de representação de linguagem. Ele foi projetado para o pré-treinamento de modelos de linguagem e se baseia na arquitetura *Transformer* (DEVLIN et al., 2019). Para a compreensão do BERT, são explicadas as arquiteturas do modelo *Transformer* e do próprio BERT.

2.5.3.1 Arquitetura do modelo *Transformer*

A arquitetura *Transformer* é construída com base nos mecanismos de atenção. Conforme explicado por Vaswani et al. (2017), a arquitetura *Transformer* segue uma estrutura codificador-decodificador. O codificador recebe uma sequência de entrada composta por representações de símbolos $(x_1, ..., x_n)$, e produz uma sequência de representações contínuas $z = (z_1, ..., z_n)$. Utilizando z como entrada, o decodificador gera uma sequência de saída $(y_1, ..., y_m)$ de símbolos, um de cada vez. Em cada etapa, o modelo é autorregressivo, consumindo os símbolos previamente gerados como entrada adicional ao gerar o próximo. O *Transformer* segue essa estrutura geral empregando camadas de autoatenção e camadas *feedforward*. As camadas são totalmente conectadas ponto a ponto tanto no codificador quanto no decodificador. A Figura 4 representa a arquitetura *Transformer*, na qual observam-se dois componentes: o codificador e o decodificador.

Codificador (à esquerda): Começa com as Entradas (*Inputs*), que representam as palavras de entrada ou *tokens* da sequência. As entradas são convertidas em vetores de Entradas Embutidas (*Input Embedding*). Cada palavra também recebe uma codificação posicional (*Positional Encoding*) para indicar sua posição na sequência. Os vetores das Entradas Embutidas são alimentados em uma camada de *Multi-Head Attention*, onde a atenção é calculada entre todas as palavras da sequência. A saída da camada de *Multi-Head Attention* é combinada com a entrada original usando uma operação de adição e normalização (*Add e Norm*). A saída desta etapa é passada por uma camada de *FeedForward*

Figura 4 – Modelo da arquitetura *Transformer*.



Fonte: Adaptado de Vaswani et al. (2017)

e sua saída também passa por uma operação de adição e normalização. Este processo é repetido N vezes (onde N é o número de camadas), como indicado pelo rótulo "Nx".

Decodificador (à direita): Começa com as Saídas deslocadas à direita (*Outputs shifted right*), são as palavras de entrada, mas deslocadas para a direita em relação à sequência original. Isso garante que, durante a geração de saída, o modelo não tenha acesso às palavras futuras da sequência, evitando assim obter vantagem indevida durante o treinamento. O processo de entrada é o mesmo do codificador e, além das duas subcamadas em cada camada do codificador, o decodificador insere uma terceira subcamada: *Masked Multi-Head Attention* e adição e normalização. A saída desta etapa é passada por uma segunda camada de *Multi-Head Attention*, onde a atenção é calculada juntamente com a saída do codificador. O processo termina com a geração das probabilidades de saída (*Output*

Probabilities) após passar pela camada linear e a função *Softmax*. Essas probabilidades representam a chance de cada palavra ser a próxima na sequência de saída.

Para entender o funcionamento da Figura 4, que ilustra o modelo da arquitetura *Transformer*, é importante esclarecer alguns termos principais. A seguir, é detalhado os componentes presentes no codificador e decodificador.

- **Entradas Embutidas**: Também chamada de *Embedding*, é uma técnica de seleção de características que ocorre naturalmente como parte do algoritmo de mineração de dados (TAN et al., 2019). Da mesma forma que outros modelos de transdução de sequência, *Transformer* usa *embedding* para converter os *tokens* de entrada e os *tokens* de saída em vetores de dimensão d_{modelo}. O modelo compartilha a mesma matriz de peso entre as camadas de *embedding* e a transformação linear pré-*softmax*, o que significa que os mesmos pesos são usados tanto para incorporar os *tokens* de entrada quanto para projetar as saídas para o espaço de probabilidade. Isso é feito para garantir consistência e facilitar o treinamento. Além disso, na camada de *embedding*, os pesos são multiplicados por $\sqrt{d_{\text{modelo}}}$ (VASWANI et al., 2017).

- **Adição e Normalização**: Na adição, o resultado da subcamada é adicionado ao resultado da subcamada anterior. Na normalização, usa-se a função LayerNorm$(x +$ Sublayer$(x))$, onde Sublayer(x) é a função implementada pela própria subcamada. Todas as subcamadas no modelo, assim como as camadas de *embedding*, produzem saídas de dimensão $d_{\text{modelo}} = 512$ (VASWANI et al., 2017).

- **Atenção**: Uma função de atenção pode ser descrita como o mapeamento de uma consulta e um conjunto de pares chave-valor para uma saída, onde a consulta, chaves, valores e saída são todos vetores. A saída é calculada como uma soma ponderada dos valores, onde o peso atribuído a cada valor é calculado por uma função de compatibilidade entre a consulta e a chave correspondente (VASWANI et al., 2017). A atenção é usada para calcular uma ponderação de importância para cada palavra em uma frase. Isso permite que o modelo considere o contexto de cada palavra e como ela se relaciona com todas as outras na frase. Pode ser formalizado como:

$$\text{Atenção}(q, K, V) = \text{softmax}\left(\frac{qK^T}{\sqrt{d_k}}\right)V \tag{2.17}$$

  onde q, K e V representam a consulta, a lista de chaves e a lista de valores, respectivamente. d_k é o tamanho do vetor de chave. Para o mecanismo de autoatenção do codificador, as consultas, chaves e valores são da mesma camada. Por exemplo, calcula-se a saída da primeira camada no codificador na posição j-ésima. x_j é o vetor de soma da Entrada Embutida e da Codificação Posicional. A consulta é o vetor x_j.

As chaves e os valores são os mesmos ($K = V$), e ambos são a matriz *embedding* (matriz que contém todos os vetores de entradas embutidas) $x = [x_0 \dots x_n]$.

  - ***Softmax***: A função *Softmax* é usada para representar uma distribuição de probabilidade sobre uma variável discreta com n possíveis valores (GOODFELLOW; BENGIO; COURVILLE, 2016). Seu intervalo, então é dado entre 0 e 1, a soma de todos os valores resultantes da função *softmax* será igual a 1. Quanto mais próximo de 1 o valor *softmax* de um elemento, maior é a probabilidade atribuída pelo modelo àquela classe, categoria ou *token*. É dada pela equação 2.18 que calcula a função *softmax* para o i-ésimo elemento do vetor z, onde z_i é o valor do i-ésimo elemento, e^{z_i} é a exponencial desse elemento, e $\sum_{j=1}^{K} e^{z_j}$ é a soma de todas as exponenciais no vetor, onde K é o tamanho do vetor.

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{2.18}$$

- ***Multi-Head Attention***: Em vez de executar uma única função de atenção com chaves, valores e consultas dimensionais d_{model}, *Transformer* projeta linearmente as consultas, chaves e valores h vezes para dimensões d_k, d_v e d_q, respectivamente. Em cada uma dessas versões projetadas de consultas, chaves e valores, é executada a função de atenção em paralelo. Estes são concatenados e novamente projetados linearmente, resultando nos valores finais (VASWANI et al., 2017).

- ***Masked Multi-Head Attention***: Por ser um modelo bidirecional, não é possível simplesmente condicionar cada palavra às palavras anteriores e seguintes, pois isso permitiria que a palavra que está sendo prevista "se visse" indiretamente em um modelo multicamadas. Para tanto, usa-se a técnica de mascarar algumas das palavras na entrada e então condicionar cada palavra bidirecionalmente para prever as palavras mascaradas (VASWANI et al., 2017).

- ***FeedForward***: Nos modelos *feedforward* a informação flui através da função sendo avaliada a partir da entrada x, passando pelas computações intermediárias e finalmente para a saída y. Não há conexões de *feedback* (GOODFELLOW; BENGIO; COURVILLE, 2016). Os dados fluem em uma única direção, da entrada para a saída, sem formar ciclos ou retroalimentação. Vaswani et al. (2017) explicam que no modelo *Transformer*, cada uma das camadas no codificador e decodificador contém uma rede *feedforward* totalmente conectada, que é aplicada a cada posição separadamente e de maneira idêntica. Isso consiste em duas transformações lineares com uma ativação ReLU entre elas:

$$\text{FFN}(\mathbf{x}) = \max(0, \mathbf{x}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2 \tag{2.19}$$

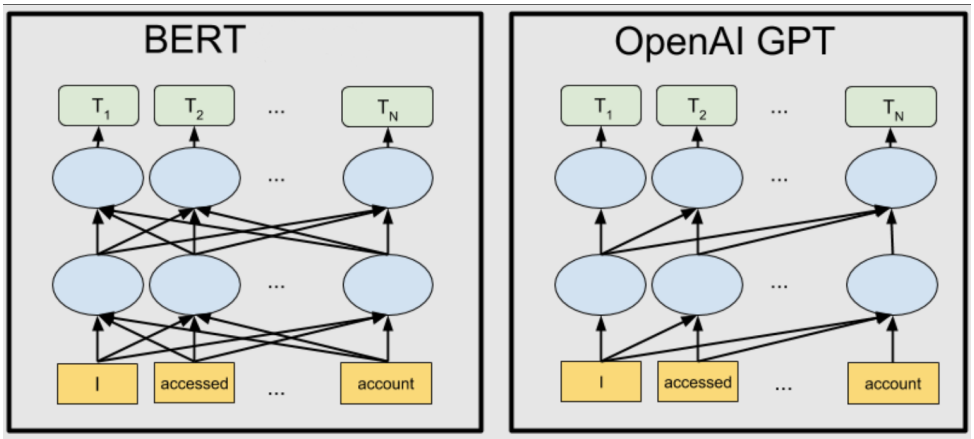
do qual x é a entrada, as operações lineares são representadas por $\mathbf{x}\mathbf{W}_1 + \mathbf{b}_1$ e $\mathbf{W}_2 + \mathbf{b}_2$, onde $\mathbf{W}_1$ e $\mathbf{W}_2$ são matrizes de pesos e $\mathbf{b}_1$ e $\mathbf{b}_2$ são vetores de constante bias. Essas operações são seguidas por uma função de ativação ReLU, representada por $\max(0, \cdot)$. A dimensionalidade da entrada e saída é $d_{\text{model}} = 512$, e a camada interna intermediária tem dimensionalidade $d_{\text{ff}} = 2048$.

2.5.3.2 Arquitetura do modelo BERT

Muitos modelos recentes de representação de linguagem, como o OpenAI GPT, são unidirecionais. Os modelos unidirecionais são treinados prevendo cada palavra condicionada às palavras anteriores da frase. BERT foi concebido para pré-treinar representações profundas bidirecionais a partir de texto não rotulado (DEVLIN et al., 2019). Isso significa que antes de ser usado em tarefas específicas de PLN, o BERT passa por um processo de pré-treinamento. Durante o pré-treinamento, o modelo é alimentado com grandes quantidades de texto não rotulado e aprende a representar o texto usando as redes neurais profundas. Sendo um modelo bidirecional, ele tem acesso às palavras anteriores e futuras. Devlin et al. (2019) utilizam o BooksCorpus (800 milhões de palavras) e a Wikipedia em inglês (2,5 bilhões de palavras). Para a Wikipedia, são extraídos apenas os trechos de texto e ignorando listas, tabelas e cabeçalhos.

A Figura 5 compara as arquiteturas dos modelos de linguagem BERT e OpenAI GPT. A frase usada para representação é "*I accessed ... account*" (Eu acessei... conta). No lado esquerdo, o modelo BERT é representado, onde cada *token* de entrada está conectado a todos os outros *tokens* para criar uma representação contextualizada, exemplificando o modelo bidirecional. Já no lado direito, o modelo OpenAI GPT é apresentado com conexões sequenciais entre os *tokens*, evidenciando o fluxo unidirecional.

Figura 5 – Comparação da arquitetura de rede neural entre BERT e OpenAI GPT.

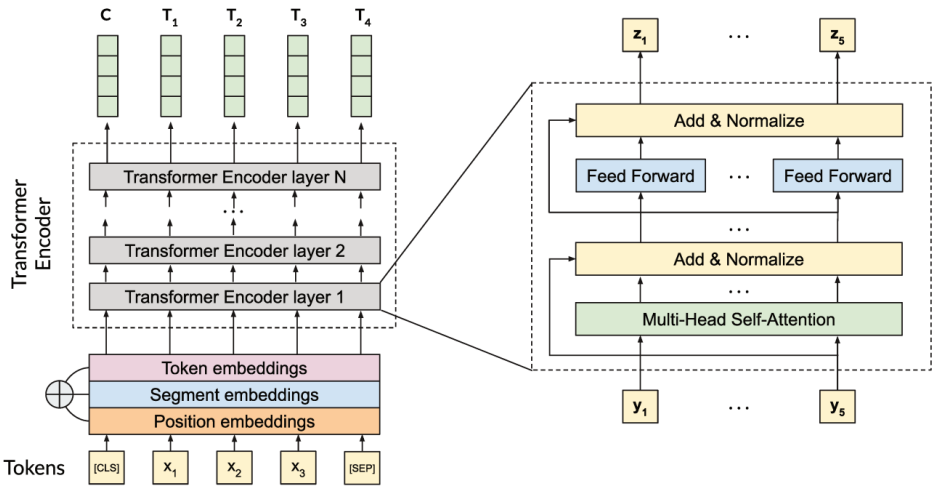


Fonte: Adaptado de <https://blog.research.google/2018/11/open-sourcing-BERT-state-of-art-pre.html>

O BERT é um codificador *Transformer* bidirecional composto por várias camadas, baseado na implementação original do *Transformer*. O uso de *Transformers* tornou-se

amplamente adotado, e a implementação do BERT é praticamente idêntica à versão original, com exceção que é usado apenas o codificador. Existem duas variantes principais do BERT: o BERTlarge, com 24 camadas de *Transformer*, tamanho oculto de 1024, 16 camadas de autoatenção e um total de 340 milhões de parâmetros; e o BERTbase, com 12 camadas de *Transformer*, tamanho oculto de 768, 12 camadas de autoatenção e um total de 110 milhões de parâmetros (DEVLIN et al., 2019).

Figura 6 – Arquitetura geral do BERT.



Fonte: Souza (2020)

- [CLS] é um *token* especial adicionado no início de cada sequência para fins de classificação e [SEP] é outro *token* especial, mas usado para separar duas sequências.
- *Token embeddings*: representam informações específicas associadas a cada *token* na sequência, sua representação semântica, por exemplo.
- *Embeddings* de segmento: são utilizados para distinguir diferentes segmentos de texto dentro da sequência. Por exemplo, se a entrada consistir em duas sentenças, esses *embeddings* seriam usados para indicar a qual sentença cada *token* pertence.
- *Embeddings* de posição: codificam a posição relativa dos *tokens* na sequência, a posição de cada palavra ou elemento na frase.

Existem dois passos distintos no processo do BERT: o pré-treinamento e o ajuste fino. Durante o pré-treinamento, o modelo é submetido a um treinamento em dados não rotulados, abordando diferentes tarefas específicas. Já no ajuste fino, o BERT é inicializado com os parâmetros pré-treinados e refinado utilizando conjuntos de dados rotulados provenientes de tarefas específicas de processamento de linguagem natural (tarefas *downstream*). Cada uma dessas tarefas *downstream* é associada a um modelo

finamente ajustado, mesmo que inicializado com os mesmos parâmetros pré-treinados (DEVLIN et al., 2019).

Pré-Treinamento: (DEVLIN et al., 2019) explica que o BERT não utiliza modelos de linguagem tradicionais de esquerda para a direita ou de direita para a esquerda para pré-treinamento. Em vez disso, usa-se duas tarefas não supervisionadas: MLM (Modelo de Linguagem Mascarada, do inglês, *Masked Language Model*) e NSP (Previsão da Próxima Frase, do inglês, *Next Sentence Prediction*).

MLM é usado para mascarar uma porcentagem dos *tokens* de entrada aleatoriamente e, em seguida, esses *tokens* mascarados serão previstos pelo modelo. Para isso, o gerador de dados de treinamento escolhe 15% das posições dos *tokens* aleatoriamente para previsão. Para tornar o processo mais eficaz, há três opções para substituir cada *token* mascarado: (1) com 80% de probabilidade, o *token* original é substituído por [MASK], (2) com 10% de probabilidade, o *token* original é substituído por um *token* aleatório selecionado do vocabulário ou (3) com 10% de probabilidade, o *token* original é mantido sem alteração. O *token* que foi mascarado durante o pré-treinamento será usado para prever o *token* original durante o treinamento, utilizando a perda de entropia cruzada como uma medida para avaliar o quão precisa foi a predição. Quanto menor a perda de entropia cruzada, melhor o desempenho do modelo na tarefa de previsão. Essa técnica de mascaramento aleatório durante o pré-treinamento permite que o modelo seja exposto a uma variedade de contextos e palavras, incentivando-o a aprender representações mais robustas e generalizadas da linguagem.

NSP é usado para treinar um modelo que entende as relações entre frases. É realizado o pré-treinamento para uma tarefa binarizada de previsão de próxima sentença que pode ser gerada facilmente a partir de qualquer corpus monolíngue. A tarefa de previsão de próxima sentença é transformada em um problema binário, onde há apenas duas opções: "*IsNext*" (Próxima) ou "*NotNext*" (Não Próxima). Especificamente, ao escolher as frases A e B para cada exemplo de pré-treinamento, 50% do tempo B é a próxima frase real que segue A (*IsNext*), e 50% do tempo é uma frase aleatória do corpus (*NotNext*).

Ajuste Fino: O ajuste fino do BERT é relativamente simples devido ao seu mecanismo de autoatenção no *Transformer*. Esse mecanismo permite que o BERT seja usado em uma variedade de tarefas secundárias, tanto em texto único quanto em pares de texto na classificação. Ao realizar o ajuste fino, todos os parâmetros do modelo são atualizados utilizando os dados específicos da tarefa de interesse. Essa atualização é feita de ponta a ponta, o que significa que o modelo inteiro é treinado simultaneamente. O BERT utiliza o mecanismo de autoatenção para processar os dados de entrada, e, por ser bidirecional, implica em considerar as relações entre as palavras em ambos os sentidos, possibilitando assim uma compreensão mais profunda e abrangente do contexto em que as palavras estão inseridas.

Após o processamento, o BERT gera representações para cada *token* na saída. Essas representações de *token* capturam informações sobre o significado e contexto de cada palavra no texto. Elas são utilizadas em diversas tarefas de processamento de linguagem natural, como identificação de entidades nomeadas em um texto ou perguntas e respostas. Em tarefas de classificação, uma representação especial [CLS] é gerada na saída, indicando o rótulo correspondente.

2.5.3.3 BERT Multilíngue

BERT apresenta diversas variações em várias línguas. Além do próprio BERT que foi treinado em algumas línguas, o *Multilingual* BERT (mBERT), treinado em 104 idiomas, foi projetado pelo Google para ser a versão multilíngue do BERT (WU; DREDZE, 2020). Outra variação multilíngue é o XLM-RoBERTa, uma versão do modelo RoBERTa (também baseado em BERT), desenvolvido pelo Facebook AI Research (FAIR). Esse modelo foi treinado em um conjunto de dados massivo de 2,5 TB de texto da CommonCrawl, que inclui 100 idiomas diferentes (VELANKAR; PATIL; JOSHI, 2023). Contudo, os modelos multilíngues baseados em BERT são inferiores aos modelos monolíngues (WU; DREDZE, 2020).

Há várias versões disponíveis em diversas línguas de modelos monolíngues baseados em BERT. Para o português brasileiro, há, por exemplo, o BERTimbau, que utiliza, para dados de pré-treinamento, o corpus brWaC (*Brazilian Web as Corpus)*. Esse corpus resulta de um rastreamento de páginas brasileiras que contém 2,68 bilhões de *tokens* de 3,53 milhões de documentos (SOUZA; NOGUEIRA; LOTUFO, 2020).

2.6 Métodos e Medidas de Avaliação

Para avaliar o desempenho de uma solução, treina-se em uma amostra e testa-se em outra. Normalmente, os dados podem ser divididos aleatoriamente em duas partes: uma para treinamento e outra para teste. Uma vez que os dados são divididos em amostras de treinamento e teste, o aprendizado ocorre no conjunto de treinamento. O desempenho pode ser estimado em termos de várias medidas (WEISS; INDURKHYA; ZHANG, 2015).

2.6.1 Matriz de Confusão

A avaliação do desempenho de um modelo de classificação baseia-se nas contagens de registros de teste preditas correta e incorretamente pelo modelo. Essas contagens são tabuladas em uma tabela conhecida como matriz de confusão (TAN; STEINBACH; KUMAR, 2014). Essa, possui um tamanho $n \times n$ e é associada a um classificador. Seu objetivo é mostrar a classificação prevista e real, onde n é o número de classes diferentes (VISA et al., 2011).

Zong, Xia e Zhang (2021) exemplificam que há M categorias em uma tarefa de classificação de texto, representadas por $C_1, ..., C_M$. Para cada uma das classes, calcula-se as estatísticas sobre o número de documentos relacionados aos seguintes quatro casos, onde uma das classes é considerada positiva e as demais são consideradas negativas:

1. Verdadeiro Positivo (VP): quando o sistema prevê corretamente um exemplo como positivo.
2. Verdadeiro Negativo (VN): quando o sistema prevê corretamente um exemplo como negativo.
3. Falso Positivo (FP): quando o sistema incorretamente prevê um exemplo como positivo.
4. Falso Negativo (FN): quando o sistema incorretamente prevê um exemplo como negativo.

Tabela 7 – Matriz Confusão.

		Previsto	
		Positivo	Negativo
Real	Positivo	VP	FN
	Negativo	FP	VN

Fonte: Elaborado pela autora (2024)

A Tabela 7 mostra como é uma Matriz Confusão. A Tabela 8 apresenta um exemplo da matriz utilizando a classificação de filmes com $n = 2$, onde um filme pode ser classificado como de ação ou não (pertencendo a qualquer outro gênero que não seja ação). As linhas representam as classes reais dos filmes, enquanto as colunas representam as classes previstas pelo modelo. Os números contidos na tabela indicam as contagens de casos para cada combinação de classe real e classe prevista. O valor 20 na célula onde a classe real é "Ação" e a classe prevista é "Ação" significa que houve 20 filmes corretamente classificados como "Ação". O valor 7 na célula onde tanto a classe real quanto a classe prevista são "Não Ação" indica que 7 filmes foram corretamente classificados como "Não Ação". No entanto, os valores fora da diagonal principal representam casos de classificação incorreta: o valor 5 na célula onde a classe real é "Ação" e a classe prevista é "Não Ação" indica que houve 5 filmes erroneamente classificados como "Não Ação", quando na verdade eram "Ação". O valor 3 na célula onde a classe real é "Não Ação" e a classe prevista é "Ação" indica que 3 filmes foram erroneamente classificados como "Ação", quando na verdade eram "Não Ação".

Tabela 8 – Matriz Confusão com $n = 2$.

		Previsto	
		Ação	Não Ação
Real	Ação	20	5
	Não Ação	3	7

Fonte: Elaborado pela autora (2024)

2.6.2 Acurácia e Erro

Embora uma matriz de confusão forneça as informações necessárias para determinar o desempenho de um modelo de classificação, resumir essas informações com um único número torna mais conveniente ao comparar o desempenho de diferentes modelos. Isso pode ser feito usando uma métrica de desempenho, como a acurácia, que é definida da seguinte maneira (TAN; STEINBACH; KUMAR, 2014):

$$\text{Acurácia} = \frac{\text{Número de previsões corretas}}{\text{Número total de previsões}} = \frac{VP + VN}{VP + FN + FP + VN} \tag{2.20}$$

Equivalentemente, o desempenho de um modelo pode ser expresso em termos de sua taxa de erro:

$$\text{Erro} = \frac{\text{Número de previsões erradas}}{\text{Número total de previsões}} = \frac{FP + FN}{VP + FN + FP + VN} \tag{2.21}$$

Como concluem Tan, Steinbach e Kumar (2014), a maioria dos algoritmos de classificação busca modelos que atinjam a maior precisão, ou analogamente, a menor taxa de erro quando aplicados ao conjunto de teste.

2.6.3 Precisão e Revocação

Embora as taxas de erro sejam úteis para estimar o desempenho das predições em geral, para a maioria das aplicações de texto, como a classificação, uma análise mais detalhada dos erros é desejável. Quando há um grande número de dados negativos, um classificador pode alcançar uma precisão muito alta, ou seja, uma taxa de erro muito baixa, simplesmente dizendo que todos os dados são negativos. Assim, é útil medir o desempenho da classificação ignorando os dados negativos corretamente previstos e, em seguida, examinando os tipos de erros cometidos pelo classificador (WEISS; INDURKHYA; ZHANG, 2015). A Precisão (Equação 2.22) e a Revocação (Equação 2.23) são duas medidas que focam nos dados positivos.

$$\text{Precisão} = \frac{\text{Número de previsões positivas corretas}}{\text{Número total de previsões positivas}} = \frac{VP}{VP + FP} \tag{2.22}$$

$$\text{Revocação} = \frac{\text{Número de previsões positivas corretas}}{\text{Número de documentos de classe positiva}} = \frac{VP}{VP + FN} \tag{2.23}$$

- A precisão é uma métrica que mede a proporção de exemplos positivos previstos corretamente em relação ao total de exemplos previstos como positivos pelo classificador.
- A revocação é uma métrica que mede a proporção de exemplos positivos previstos corretamente em relação ao total de exemplos positivos reais.

Weiss, Indurkhya e Zhang (2015) elucidam que quanto maior a precisão, menor o número de erros falso positivos cometidos pelo classificador, ou seja, a maioria dos exemplos previstos como positivos realmente são positivos. Os autores ainda afirmam que os classificadores com grande revocação têm pouquíssimos exemplos positivos erroneamente classificados como classe negativa, sendo o valor da revocação equivalente à verdadeira taxa positiva.

2.6.4 Medida-F e F1

Weiss, Indurkhya e Zhang (2015) definem a Medida-F como a média harmônica de precisão e revocação. É frequentemente usada para medir o desempenho quando um único número é preferido. Baeza-Yates e Ribeiro-Neto (2013) complementam que a medida-F combina valores de precisão e revocação e permite a atribuição de diferentes pesos para cada uma delas. Geralmente, a medida F_β pode ser usada para examinar a troca (*tradeoff*) entre revocação e precisão (TAN; STEINBACH; KUMAR, 2014). Ela é definida como mostrado na Equação 2.24:

$$F_\beta = \frac{(\beta^2 + 1) \cdot P \cdot R}{\beta^2 \cdot P + R} \tag{2.24}$$

- P indica a precisão.
- R indica a revocação.
- β define a importância relativa da precisão e da revocação.

  Quando se define $\beta = 0$, somente a precisão é considerada. Se $\beta = \infty$, apenas a revocação é considerada. Valores baixos de β tornam a medida-F_β mais próxima da precisão, enquanto valores altos a tornam mais próxima da revocação.

A forma mais comum da medida-F é obtida quando pesos iguais são atribuídos à precisão e revocação. Quando $\beta = 1$, a medida-F_β se torna a medida padrão F1, dada pela Equação 2.25 (BAEZA-YATES; RIBEIRO-NETO, 2013):

$$F1 = 2 \cdot \frac{P \cdot R}{P + R} \tag{2.25}$$

2.6.5 Macromédia e Micromédia

Como explicam Baeza-Yates e Ribeiro-Neto (2013), "também é comum derivar um valor único de F1 para um classificador computando a média dos valores de F1 entre todas as classes". Há duas equações de média: Macromédia F1 e Micromédia F1, ilustradas a seguir, a partir das demonstrações de Zong, Xia e Zhang (2021).

Na Macromédia, calcula-se o desempenho de cada classe individualmente. Em seguida, tira-se a média desses desempenhos para obter a métrica geral.

A Equação 2.26 refere-se a Macromédia F1:

$$\text{Macro\_F1} = \frac{2 \times \text{Macro\_}P \times \text{Macro\_}R}{\text{Macro\_}P + \text{Macro\_}R} \tag{2.26}$$

onde:

$$\text{Macro\_}P = \frac{1}{C} \sum_{j=1}^{C} \left(\frac{VP_i}{VP_i + FP_i} \right), \tag{2.27}$$

$$\text{Macro\_R} = \frac{1}{C} \sum_{j=1}^{C} \frac{VP_i}{VP_i + FN_i}, \tag{2.28}$$

sendo C o número total de classes.

Na Micromédia, agrega-se as contagens de verdadeiros positivos, falsos positivos e falsos negativos de todas as classes. Em seguida, calcula-se a precisão e a revocação globais com base nesses totais.

A Equação 2.29 refere-se a Micromédia F1:

$$\text{Micro\_F1} = \frac{2 \times \text{Micro\_P} \times \text{Micro\_R}}{\text{Micro\_P} + \text{Micro\_R}} \tag{2.29}$$

onde:

$$\text{Micro\_}P = \frac{\sum_{j=1}^{C} VP_i}{\sum_{j=1}^{C} (VP_i + FP_i)}, \tag{2.30}$$

$$\text{Micro\_}R = \frac{\sum_{j=1}^{C} VP_i}{\sum_{j=1}^{C} (VP_i + FN_i)} \tag{2.31}$$

Na Micromédia F1, cada documento tem o mesmo peso na computação da métrica. Já na Macromédia F1, cada categoria tem o mesmo peso. Isso significa que a Macromédia

F1 é mais eficaz em capturar a capacidade do classificador de funcionar bem para várias classes. Isso se torna crucial quando a distribuição das classes está muito desequilibrada. Nesse cenário, é importante considerar ambas as métricas de média (BAEZA-YATES; RIBEIRO-NETO, 2013).

3 Trabalhos Relacionados

Este capítulo tem como objetivo apresentar alguns dos principais trabalhos relacionados com o tema análise comparativa de algoritmos de classificação de texto. Houve uma atenção especial na busca por trabalhos que abordassem a língua portuguesa. Ademais, todas as buscas foram direcionadas para estudos mais recentes, dos últimos cinco anos, e priorizaram aqueles que utilizaram ao menos um algoritmo a ser empregado neste trabalho.

Embora a pesquisa sobre o BERT e o BERTimbau, sua versão para o português brasileiro, seja recente e ainda não conte com extensivas comparações, há uma preferência por estudos que adotem o estado da arte. A busca foi conduzida principalmente no site SBC-OpenLib (SOL), uma biblioteca digital mantida pela Sociedade Brasileira de Computação, utilizando *strings* de busca como "classificação de texto *and* português", "categorização *and* português" e "classificação *and* português". O Google Acadêmico também foi utilizado com a string "*text classification comparison BERTimbau*" e *text classification + BERT*, buscando garantir uma ampla abrangência de resultados. A inclusão do termo "BERTimbau" assegura a presença de trabalhos que, mesmo em inglês, utilizam um corpus em português, como exemplificado no estudo Moreira et al. (2023).

O trabalho de Plath et al. (2022) desenvolve a base de dados MINA-BR para identificar discursos de ódio contra mulheres em textos em português brasileiro e cria um modelo de classificação automatizado. A pesquisa envolveu a coleta de comentários do Twitter e YouTube, rotulados por voluntários. O pré-processamento incluiu a padronização para minúsculas, remoção de *stopwords* com a biblioteca NLTK, eliminação de *hashtags* e espaços duplicados, e segmentação em *tokens*. O tratamento de emojis e abreviações foi realizado em duas etapas: uma com conversão de emojis para descrições e expansão de abreviações, e outra com remoção de emojis e manutenção das abreviações. As representações textuais foram feitas com *Bag of Words* e TF-IDF. A detecção de discurso de ódio utilizou SVM, *Naive Bayes* (Multinomial, Bernoulli e Gaussiano), Floresta Aleatória e BERT. Os resultados indicaram que o SVM com TF-IDF, sem emojis, tratamento de abreviações e *stemming*, apresentou a melhor performance, com uma Medida-F1 de 0.57.

O trabalho de Sousa et al. (2022) tem como objetivo rotular e classificar um conjunto de dados de textos clínicos em português utilizando técnicas de aprendizado de máquina. O banco de dados, de origem privada, contém 11.219 documentos médicos. O pré-processamento envolveu a conversão de documentos *rich text* para strings Python, remoção de caracteres específicos e pontuações. Os textos foram classificados em categorias binária (receitas e outros) e multiclasse (receitas, exames, atestados/declarações e outros). A vetorização dos textos utilizou n-gramas e TF-IDF. Para a classificação, foram aplicados

SVM, Floresta Aleatória, XGBoost e Perceptron Multicamadas (MLP). As métricas de avaliação incluíram acurácia, Kappa, medida-F1 e área sob a curva ROC (AUC). SVM, Floresta Aleatória e MLP obtiveram acurácia máxima de 0,999 na classificação binária, enquanto o MLP apresentou o melhor resultado na classificação multiclasse com 0,994. Apesar das pequenas diferenças entre os algoritmos, todos mostraram alta efetividade na identificação dos padrões nos dados.

Barbosa et al. (2022) investigam a eficácia de algoritmos de aprendizado de máquina na classificação de textos em tipos específicos, utilizando um corpus de textos em português. O estudo focou em textos dissertativos e narrativos, criando o corpus denominado *Corpus* de Tipos Textuais Brasileiros (*Corpus* TTBR), composto por textos da base de dados "uol-redacoes" e do corpus "OBras". Após a extração de 75 características textuais usando a ferramenta CohMetrix PT-BR, três algoritmos de classificação foram avaliados: Floresta Aleatória, SVM e *Stochastic Gradient Descent* (SGD). Os modelos foram avaliados com as métricas de precisão, revocação e medida-F1. O índice SYNSTRUTt, que mede a complexidade sintática, mostrou ser altamente relevante, com SVM e SGD apresentando melhores resultados quando treinados apenas com este índice. A Floresta Aleatória também teve bom desempenho quando foram utilizados 20 índices. Todos os modelos apresentaram métricas de precisão e revocação acima de 95%, indicando alta eficiência.

Moreira et al. (2023) analisam a eficácia do algoritmo BERTimbau na detecção de notícias falsas, comparando-o com o BERT e métodos tradicionais de classificação de texto. Utilizaram a base de dados Fake.br, composta por 7.200 notícias em português, rotuladas como verdadeiras ou falsas e divididas em seis categorias: política, TV e celebridades, sociedade e notícias do dia, ciência e tecnologia, economia e religião. O conjunto já estava pré-processado, incluindo remoção de stopwords, acentos e caracteres especiais. Foram aplicados os métodos `CountVectorizer` e `TfidfVectorizer` para vetorização, e a arquitetura *Transformers* para processamento com BERT. A tradução da base para o inglês permitiu comparar o desempenho do BERTimbau, treinado em português, com o BERT original. Os resultados mostraram que o BERTimbau superou significativamente os algoritmos tradicionais em precisão, revocação e medida-F1, além de superar a versão em inglês do BERT, destacando sua eficácia em tarefas de processamento de linguagem natural.

Braz Junior e Fileto (2021) investigam a utilização do BERTimbau e BERT multilíngue para analisar e classificar a textos em coerente e incoerente. As bases de dados utilizadas foram: OnlineEduc 1.0 (privada) e CSTNews3 (pública). O CSTNews foi criado para avaliar a sumarização de documentos jornalísticos de diferentes fontes (Jornal do Brasil, Folha de São Paulo e O Estado de São Paulo) e contém um total de 9.960 notícias, divididas igualmente em 4.980 coerentes e 4.980 incoerentes. O OnlineEduc 1.0 contém postagens de fóruns de dúvidas do Moodle ao longo de 4 semestres de disciplinas de um

curso superior a distância de uma universidade brasileira, entre 2017 e 2019, totalizando 1.080 postagens. Para o pré-processamento foi feita a tokenização, remoção de *stopwords* e lematização. O estudo demonstrou que os modelos do BERT podem fornecer medidas de coerência consistentes, com melhores resultados em textos de fórum (99% BERT multilíngue) em comparação com os textos públicos (97% BERTimbau).

Também utilizando classificação binária, Sun, Huang e Qiu (2019) abordam a análise de sentimentos positivos e negativos sobre aspectos de moradia em um bairro urbano. Os autores utilizaram dois conjuntos de dados: SentiHood e SemEval-2014 Task 4, com cerca de 5.000 sentenças cada. Para a comparação foram escolhidos os algoritmos BERT, Regressão Logística, Redes de Memória Neural e *Long Short-Term Memory*, um tipo de rede neural recorrente. Ajustando o modelo BERT pré-treinado utilizando os *embeddings* como entrada, o estudo alcançou uma acurácia de 85,5% no SentiHood e 83,7% no SemEval-2014 Task 4, obtendo resultados de ponta.

Os estudos revisados neste capítulo destacam a importância de investigar uma variedade de algoritmos e abordagens de processamento de linguagem natural para a classificação de texto em português, reconhecendo suas particularidades e aplicabilidades em diferentes contextos. Além disso, enfatizam a relevância do emprego de modelos avançados, como o BERT e o BERTimbau, para tarefas específicas de análise de texto. A Tabela 9 apresenta uma comparação entre os trabalhos analisados e suas principais características.

Tabela 9 – Comparação dos trabalhos relacionados.

Trabalhos	**Objetivo**	**Base de Dados**	**Pré-Processamento**	**Classificadores Utilizados**	**Medidas de Avaliação**	**Resultado**
Plath et al. (2022)	Criar uma base de dados focada em discursos de ódio contra mulheres e comparar modelos de classificação	Comentários do Twitter e YouTube	Padronização para minúsculas, remoção de stopwords, eliminação de hashtags, tratamento de emojis e abreviações, segmentação em tokens	SVM, *Naive Bayes* (Multinomial, Bernoulli, Gaussiano), Floresta Aleatória, BERT	Medida-F1	SVM com TF-IDF, sem emojis, com tratamento para abreviações, alcançou uma Medida-F1 de 0.57
Sousa et al. (2022)	Rotular e classificar um conjunto de dados de textos clínicos em português	Conjunto de dados privado com 11.219 amostras	Conversão de documentos rtf para strings Python, remoção de caracteres específicos, eliminação de pontuações	SVM, Floresta Aleatória, XGBoost, Perceptron Multicamadas (MLP)	Acurácia, Kappa, Medida-F1, Área sob a curva ROC (AUC)	SVM, Floresta Aleatória e MLP alcançaram os melhores resultados: classificação binária 0.999 e multiclasse MLP 0.994
Barbosa et al. (2022)	Comparar algoritmos de classificação de textos em tipos textuais específicos em língua portuguesa	Corpus do uol-redacoes e OBras	Extração de características textuais utilizando a ferramenta Coh-Metrix PT-BR, seleção de 75 características	Floresta Aleatória, SVM, Stochastic Gradient Descent (SGD)	Precisão, Revocação, Medida-F1	SVM e SGD obtiveram os melhores resultados com base apenas no índice SYNSTRUTt
Moreira et al. (2023)	Analisar a eficácia do algoritmo BERT na detecção de notícias falsas em comparação	Conjunto de dados Fake.br com 7200 notícias	Remoção de stopwords, remoção de acentos e caracteres especiais	BERT, BERTimbau, SGD, Regressão Logística, MLP, SVM, AdaBoost, *Naive Bayes*	Precisão, Revocação, Medida-F1	BERTimbau superou os outros algoritmos em termos de precisão, revocação e medida-F1
Braz Junior e Fileto (2021)	Classificar a coerência textual de notícias em português	OnlineEduc 1.0 (privado) e CSTNews3 (público)	Tokeninzação, remoção de *stopwords* e lematização	BERT multilíngue e BERTimbau	Acurácia	OnlineEduc 1.0 obteve 99% com o BERT multilígue e CSTNews3 97% com o BERTimbau
Sun, Huang e Qiu (2019)	Analisar sentimentos em relação a um bairro urbano	SentiHood e SemEval-2014 Task 4	Embeddings do BERT	BERT, Regressão Logística, Redes de memória neural e LSTM	Acurácia	BERT superou os outros algoritmos e alcançou uma acurácia de 85,5% no SentiHood e 83,7% no SemEval-2014 Task 4

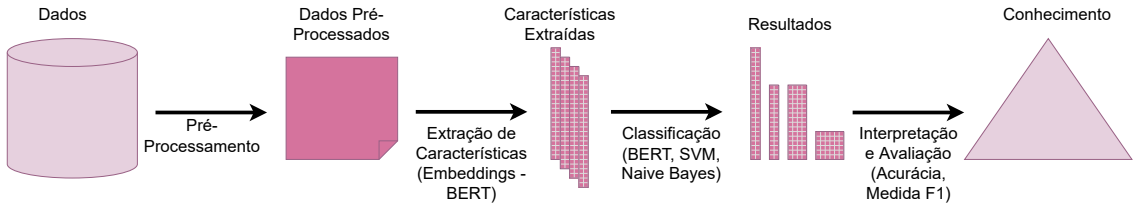
Fonte: Elaborado pela autora (2024)

4 Método Para Análise Comparativa de Algoritmos de Classificação de Texto

Este capítulo apresenta a metodologia adotada para conduzir a análise comparativa, seguindo o processo de Descoberta de Conhecimento em Bases de Dados (*Knowledge Discovery in Databases* – KDD). Como explica Goldschmidt (2015), a análise de grandes quantidades de dados é inviável sem o auxílio de ferramentas computacionais apropriadas. Para atender a esse contexto, existe uma área denominada KDD.

O processo KDD envolve diversas etapas. É preciso selecionar um conjunto de dados em que a descoberta será realizada. Em seguida, faz-se a limpeza e pré-processamento de dados, a extração das características (extração de *features*) que encontra as características úteis para representar os dados ou a transformação dos dados. A partir desse ponto, procede-se à escolha de um método e seus algoritmos específicos de mineração de dados (resumo, classificação, regressão, agrupamento, etc). Chega-se, então, à mineração de dados: busca por padrões ou a visualização dos resultados. Por fim, tem-se a avaliação dos resultados obtidos e o uso do conhecimento, documentando e relatando às partes interessadas (FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996). A Figura 7 ilustra os passos que foram seguidos para a tarefa de classificação utilizando o processo KDD.

Figura 7 – Uma visão geral das etapas que compõem o processo KDD.



Fonte: Adaptado de Fayyad, Piatetsky-Shapiro e Smyth (1996)

4.1 Bases de Dados

Para garantir comparabilidade, foram utilizadas quatro bases de dados. A primeira base de dados consiste em textos em português brasileiro classificados em duas classes distintas (CSTNews), o que permite a avaliação do desempenho do modelo em um contexto binário na língua portuguesa. A segunda base de dados também é composta por textos em português brasileiro, mas apresenta múltiplas classes (Fake.br), proporcionando um

cenário mais complexo e desafiador para tarefa de classificação. A terceira base de dados é constituída por textos em inglês separados em duas classes (Sentihood), possibilitando a comparação direta entre os desempenhos dos modelos em diferentes idiomas dentro de um contexto binário. Por fim, a quarta base de dados contém textos em inglês distribuídos em múltiplas classes (NewsKaggle), oferecendo um panorama abrangente para avaliar a eficácia do modelo em um contexto multiclasses em língua inglesa. A Tabela 10 sumariza as quatro bases de dados.

Tabela 10 – Resumo das Bases de Dados Utilizadas.

	Trabalho de Referência	**Língua**	**#Textos**	**#Classes**
CSTNews	Braz Junior e Fileto (2021)	Português	9.960	2
Fake.br	-	Português	3.600	6
SentiHood	Sun, Huang e Qiu (2019)	Inglês	5.215	2
NewsKaggle	-	Inglês	10.000	5

Fonte: Elaborado pela autora (2024)

CSTNews

CSTNews[1] é uma coleção pública de resumos de notícias provenientes de diversas fontes jornalísticas brasileiras, como "Jornal do Brasil", "Folha de São Paulo" e "O Estado de São Paulo". Esta coleção foi utilizada originalmente no trabalho de Braz Junior e Fileto (2021) que investiga o uso do BERTimbau para a classificação e a medição da coerência textual em português. Contém um total de 9960 notícias, divididas igualmente em 4980 coerentes e 4980 incoerentes. O corpus consiste em uma pasta contendo arquivos CSV (valores separados por vírgula), cada um dos quais inclui a identificação de cada documento, o próprio documento e a classe correspondente (1 para coerente e 0 para incoerente).

Fake.br

O Fake.br[2] foi desenvolvido pelo Núcleo Interinstitucional de Linguística Computacional (NILC-USP) para detectar notícias falsas, conforme citado em Moreira et al. (2023). O corpus utilizado contém notícias previamente classificadas como verdadeiras ou falsas. No total, o conjunto de dados possui 7.200 notícias, sendo 3.600 verdadeiras e 3.600 falsas. Diferentemente do trabalho de referência, para este estudo, será feito uso apenas das 3.600 notícias verdadeiras já rotuladas para a tarefa de classificação. O corpus é composto por arquivos de texto simples (TXT) divididos em duas pastas principais: *true*, que contém as notícias verdadeiras coletadas, e *true-meta-information*, que contém as informações de metadados de cada notícia (como autor, *URL* para a matéria, data de

[1] <https://github.com/osmarbraz/coheBERT_v1/tree/main/conjuntodedados/cstnews>>
[2] <https://github.com/roneysco/Fake.br-Corpus/tree/master>

publicação, categoria, entre outros). As notícias são categorizadas em seis diferentes temas: política, TV e celebridades, sociedade e cotidiano, ciência e tecnologia, economia e religião. A quantidade de notícias por categoria está indicada na Tabela 11.

Tabela 11 – Quantidade de notícias por categoria no corpus Fake.Br.

Categoria	**Quantidade**
Política	2090
TV e Celebridades	772
Sociedade e Cotidiano	638
Ciência e Tecnologia	56
Economia	22
Religião	22

Fonte: Elaborado pela autora (2024)

Os arquivos TXT foram unificados e transformados em um único arquivo CSV, simplificando o processo de manipulação dos dados. Esse arquivo contém os textos completos juntamente com suas categorias correspondentes.

SentiHood

SentiHood[3] é um conjunto de dados criado para ajudar a identificar a polaridade de opiniões sobre aspectos como segurança, preço e localização em bairros urbanos. Este conjunto de dados foi utilizado no artigo de Sun, Huang e Qiu (2019), cujo objetivo é a análise de sentimentos utilizando o modelo BERT pré-treinado. A classificação é dividida em positiva e negativa, com um total de 5.215 sentenças. Os dados estão separados em dois arquivos JSON: sentihood-train.json, para a fase de treinamento do modelo, e sentihood-test.json, para a fase de testes.

NewsKaggle

NewsKaggle[4] é um conjunto de dados que oferece uma coleção abrangente de artigos de notícias abrangendo vários domínios, incluindo negócios, tecnologia, esportes, educação e entretenimento. Os dados são extraídos da revista de notícias "The Indian Express" na língua inglesa. Este conjunto de dados foi criado para aplicações em Processamento de Linguagem Natural e Aprendizado de Máquina, como classificação de texto, agrupamento, previsão e reconhecimento de entidade nomeada. Contém um total de 10.000 notícias, distribuídas igualmente em 2.000 notícias para cada categoria. É composto por cinco arquivos CSV que contêm os seguintes campos: *Headlines* (título), *Description* (breve resumo da notícia), *Content* (a notícia completa), URL e *Category* (categoria). Para este trabalho, serão utilizados os campos *Content* e *Category*.

[3] <https://github.com/uclnlp/jack/tree/master/data/sentihood>
[4] <https://www.kaggle.com/datasets/banuprakashv/news-articles-classification-dataset-for-nlp-and-ml>

4.2 Pré-Processamento

O pré-processamento dos dados é realizado para lidar com o ruído e as possíveis expressões malformadas comuns em dados provenientes de fontes públicas. O modelo BERTimbau[5] e seu original BERT[6] são utilizados para a etapa de tokenização e transformação dos textos em português e inglês respectivamente. Isso inclui a divisão dos textos em *tokens*, a conversão desses *tokens* em IDs e a adição de *tokens* especiais para a entrada no modelo. Esses *tokens* especiais, como [CLS] e [SEP], são adicionados ao início e ao final das sequências. O *token* [CLS] é usado para tarefas de classificação e o *token* [SEP] é usado para separar diferentes partes da entrada (SOUZA, 2020).

Ao utilizar o BERT, tanto a remoção de *stopwords* quanto a radicalização não são necessárias. Isso se deve ao fato de que os modelos baseados em *transformers* são treinados para lidar com textos completos e considerar o contexto em que cada *token* está inserido (VASWANI et al., 2017).

Além disso, é feita a limpeza dos textos, removendo caracteres especiais e normalizando espaços em branco, utilizando funções e bibliotecas da linguagem Python. As categorias nominais das bases de dados são transformadas em valores numéricos. Isso é realizado para as bases SentiHood, Fake.br e NewsKaggle. A base CSTNews, no entanto, já contém valores binários prontos.

4.3 Extração de Características

Para a extração de características, foi utilizado o BERT em sua fase de pré-treinamento para criar os *embeddings* dos textos. Os *embeddings* são representações vetoriais que transformam palavras ou frases em números reais, capturando seu significado e semântica em um espaço vetorial. Em modelos baseados em *transformers* como o BERT, esses *embeddings* são contextuais, ou seja, sua representação pode variar conforme o contexto em que as palavras aparecem (DEVLIN et al., 2019).

O processo de geração de *embeddings* com BERT inicia-se com a tokenização dos textos, dividindo-os em *tokens* e convertendo-os em IDs numéricos. Esses *tokens* são então processados pelo modelo BERT, que gera *embeddings* para cada *token* com base no contexto completo da frase (SOUZA, 2020).

Para essa tokenização, foi utilizado o `BertTokenizer`[7] da biblioteca `Transformer`[8] do Python, que converte o texto em *tokens* e IDs compatíveis com o modelo BERT. Para

[5] <https://github.com/neuralmind-ai/portuguese-BERT>
[6] <https://huggingface.co/docs/transformers/model_doc/bert>
[7] <https://huggingface.co/docs/transformers/v4.44.2/en/model_doc/bert#transformers.BertTokenizer>
[8] <https://huggingface.co/docs/transformers/pt/index>

os textos em inglês foi usado o modelo `bert-base-cased`[9], que possui 12 camadas, 768 unidades ocultas e 12 camadas de atenção. Para os textos em português, foi utilizado o `neuralmind/bert-base-portuguese-cased`[10], que possui as mesmas quantidades de camadas que a versão original.

4.4 Algoritmos de Classificação

Para a classificação, foram empregados três algoritmos distintos: o BERT, SVM (*Support Vector Machine*) e *Naive Bayes* Multinomial. Utilizou-se o pacote Scikit-learn[11,12] para implementar o SVM e o *Naive Bayes*. Além disso, para o Ajuste Fino do BERT, ou seja, a classificação dos textos pelo BERT e BERTimbau, foi usada a classe `BertForSequence Classification`[13] da biblioteca `Transformer`. A implementação desses algoritmos está disponível no repositório do projeto no GitHub[14].

O Ajuste Fino do BERT é uma abordagem avançada que utiliza um modelo pré-treinado de *transformer* para aprimorar a precisão na classificação de textos. O desenvolvimento de modelos linguísticos pré-treinados, como o BERT, permitiu uma representação poderosa da linguagem que pode ser aplicada a tarefas linguísticas diferentes (LAI, 2024). O BERT, com sua capacidade de entender o contexto das palavras de forma bidirecional, é eficaz na captura de nuances semânticas e sintáticas.

O algoritmo SVM busca encontrar a melhor margem de separação entre classes em um espaço de características. No contexto da classificação de texto, o SVM é eficaz na criação de hiperplanos que maximizam a separação entre diferentes categorias (BAEZA-YATES; RIBEIRO-NETO, 2013). Este método é adequado para problemas de alta dimensionalidade, como aqueles encontrados na classificação de texto com *embeddings*.

O *Naive Bayes* Multinomial é um algoritmo probabilístico baseado no Teorema de Bayes, que assume que as características são independentes entre si. Essa suposição simplifica tanto a modelagem quanto a computação, tornando o método conhecido por sua simplicidade e eficiência. É frequentemente utilizado em tarefas de classificação de texto (WEISS; INDURKHYA; ZHANG, 2015). Dada sua ampla aplicação em classificações e considerando que os *embeddings* do BERT capturam informações contextuais complexas e interdependentes entre palavras, é interessante analisar como o *Naive Bayes* se comporta ao lidar com essas representações vetoriais avançadas.

[9] <https://huggingface.co/google-bert/bert-base-cased>
[10] <https://huggingface.co/neuralmind/bert-base-portuguese-cased>
[11] <https://scikit-learn.org/stable/modules/svm.html>
[12] <https://scikit-learn.org/stable/modules/naive_bayes.html>
[13] <https://huggingface.co/docs/transformers/model_doc/bert#transformers.BertForSequenceClassification>
[14] <https://github.com/biarborges/ClassificacaoSVMNaiveBayesBERT>

O *Naive Bayes* Multinomial é projetado para lidar com inteiros, mas o algoritmo aceita valores contínuos, desde que sejam não negativos. Por esse motivo, foi necessário empregar a função `MinMaxScaler` para normalização dos valores, uma vez que os *embeddings* gerados pelo BERT podem conter valores negativos. Além disso, foi testado o *Naive Bayes* Gaussiano, que lida naturalmente com valores contínuos e negativos, porém seus resultados se mostraram inferiores aos do *Naive Bayes* Multinomial.

4.5 Métodos e Medidas de Avaliação

Para a avaliação dos algoritmos, são utilizadas as medidas de acurácia e F1. A acurácia, conforme a Equação 2.20, avalia a capacidade do modelo ou algoritmo em classificar corretamente os dados de teste, representando a proporção de resultados corretos em relação ao total de resultados. A medida F1, descrita na Equação 2.25, é a média harmônica entre precisão e revocação, e avalia a capacidade do modelo de realizar previsões precisas (precisão) enquanto garante que todas as instâncias relevantes sejam detectadas (revocação) (WEISS; INDURKHYA; ZHANG, 2015).

Para implementar essas medidas de avaliação, é utilizado o pacote scikit-learn[15,16] em Python, que oferece funções para calcular a acurácia e a medida F1 a partir da matriz de confusão. Para as bases de dados binárias foi usado o parâmetro `average='binary'` do qual a métrica é calculada apenas para a classe positiva. Já para as duas bases multiclasse foi usado o parâmetro `average='weighted'`. Esse parâmetro é particularmente útil para bases desbalanceadas, como a Fake.br, pois calcula a medida F1 para cada classe e a combina em uma média ponderada com base no número de instâncias em cada classe. Em bases multiclasse balanceadas, o uso desse parâmetro produz uma média ponderada semelhante à média macro, que calcula a medida F1 para cada classe e depois obtém a média aritmética.

Para a divisão entre treinamento e teste, a base SentiHood foi fornecida em duas pastas separadas: a primeira contém 10% dos textos para teste, enquanto a segunda inclui 90% dos textos para treinamento. Esta divisão preexistente foi seguida, utilizando diretamente esses conjuntos para treinar e testar o modelo. Para as demais bases de dados, foi empregada a técnica de *holdout*, que consiste em dividir aleatoriamente os dados em dois conjuntos: um para teste e outro para treinamento (WITTEN; FRANK; HALL, 2011). Foi usada a função `train_test_split`[17] do Python para realizar a divisão, alocando 90% dos dados para treinamento e 10% para teste.

[15] <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html>
[16] <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html>
[17] <https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html>

5 Resultados

As bases CSTNews e SentiHood possuem trabalhos originais que permitem a comparação dos resultados obtidos com o ajuste fino do BERT a partir da acurácia. Para a implementação do BERT nessas bases, os modelos foram treinados por 4 épocas em cada conjunto de dados. Assegurando uma comparação consistente, as bases Fake.br e NewsKaggle também foram treinadas com o BERT por 4 épocas.

Em aprendizado de máquina, uma época refere-se a uma iteração completa através do conjunto de dados de treinamento. Durante cada época, o modelo ajusta seus pesos com base no erro observado nas previsões em relação aos dados reais. Quanto maior a quantidade de época, mais tempo levará para ser processado. Isso não significa, contudo, que um número maior de épocas resultará em um valor de precisão mais alto (HASTOMO et al., 2021).

5.1 Resultados para a base CSTNews

A base CSTNews é uma coleção pública de resumos de notícias em português brasileiro. Ela provém do trabalho de Braz Junior e Fileto (2021) que realiza a classificação e a medição da coerência textual em português. Esta base é utilizada para classificação binária em língua portuguesa, e os resultados obtidos serão comparados com aqueles do trabalho de referência. As Tabelas 12 e 13 mostram os resultados para comparação.

Tabela 12 – Resultados de Acurácia e F1 para os Algoritmos Avaliados na Base CSTNews.

CSTNews		
	Acurácia	F1
Naive Bayes	64,16%	64,15%
SVM	92,26%	92,23%
BERTimbau	97,99%	97,99%

Fonte: Elaborado pela autora (2024)

Tabela 13 – Resultado de Acurácia para BERT Avaliado na Base CSTNews do trabalho de Braz Junior e Fileto (2021).

CSTNews - Trabalho de referência	
	Acurácia
BERTimbau	97,44%

Fonte: Elaborado pela autora (2024)

Os resultados obtidos para a base CSTNews em português mostram um alto desempenho na classificação binária utilizando BERTimbau. O modelo brasileiro do BERT alcançou uma acurácia e F1 de 97,99% para a base CSTNews, enquanto o resultado obtido por Braz Junior e Fileto (2021) foi uma acurácia de 97,44%. Apesar dessa superação, os resultados são bastante semelhantes, evidenciando a consistência e eficácia do modelo BERT na tarefa de classificação. O *Naive Bayes* apresentou uma acurácia de 64,16% e uma pontuação F1 de 64,15%, enquanto o SVM obteve uma acurácia de 92,26% e uma pontuação F1 de 92,23%. Esses resultados reforçam que, entre os algoritmos testados, o BERT é o modelo mais eficaz para a base CSTNews, com desempenho muito próximo ao reportado no trabalho de referência.

A diferença dos resultados pode ser atribuída a vários fatores. Primeiramente, a semente de aleatoriedade desempenha um papel importante. A inicialização aleatória e a semente utilizada para treinamento e validação podem afetar os resultados, pois variações na inicialização dos pesos do modelo e na divisão dos dados podem levar a resultados ligeiramente diferentes. Além disso, o ambiente de treinamento também pode influenciar os resultados. Diferenças no hardware, otimizações de treinamento e capacidade computacional podem impactar o desempenho do modelo. Apesar dessas possíveis variações, os resultados obtidos estão alinhados com os resultados do trabalho de referência, indicando que a implementação do BERT foi eficaz para a base CSTNews.

5.2 Resultados para a base Fake.br

A base Fake.br é composta por 3.600 notícias verdadeiras em língua portuguesa, rotuladas em categorias como política, TV e celebridades, sociedade e cotidiano, ciência e tecnologia, economia, e religião. A seguir, são apresentados os resultados.

Tabela 14 – Resultados de Acurácia e F1 para os Algoritmos Avaliados na Base Fake.br.

Fake.br		
	Acurácia	F1
Naive Bayes	62,50%	60,39%
SVM	62,22%	60,87%
BERTimbau	66,66%	65,93%

Fonte: Elaborado pela autora (2024)

Os resultados mostram que o modelo brasileiro do BERT, BERTimbau, obteve a melhor performance, com uma acurácia de 66,66% e uma medida F1 de 65,93%, seguido pelo SVM, com acurácia de 62,22% e F1 de 60,87%, e pelo *Naive Bayes*, com acurácia de 62,50% e F1 de 60,39%. Por se tratar de uma tarefa de classificação multiclasse, a complexidade do problema é maior do que em uma tarefa de classificação binária, como a realizada com a base CSTNews. Embora os resultados não sejam tão altos quanto os

obtidos na classificação binária da base CSTNews, BERTimbau ainda é uma melhor opção para a tarefa de classificação.

5.3 Resultados para a base SentiHood

A base SentiHood é um conjunto de dados criado para ajudar a identificar a polaridade de opiniões sobre bairros urbanos. Este conjunto de dados é em inglês, possui uma classificação binária (positivo e negativo) e foi utilizado no artigo de Sun, Huang e Qiu (2019) para a análise de sentimentos utilizando o modelo BERT pré-treinado. As Tabelas 15 e 16 mostram os resultados para comparação.

Tabela 15 – Resultados de Acurácia e F1 para os Algoritmos Avaliados na Base SentiHood.

SentiHood		
	Acurácia	F1
Naive Bayes	76,59%	72,64%
SVM	76,95%	76,74%
BERT	86,42%	86,41%

Fonte: Elaborado pela autora (2024)

Tabela 16 – Resultado de Acurácia para BERT Avaliado na Base SentiHood do trabalho de Sun, Huang e Qiu (2019).

SentiHood - Trabalho de referência	
	Acurácia
BERT	85,50%

Fonte: Elaborado pela autora (2024)

Os resultados obtidos com os diferentes algoritmos mostram que o modelo BERT atingiu a melhor performance, com uma acurácia de 86,42% e uma medida F1 de 86,41%. O SVM apresentou uma acurácia de 76,95% e F1 de 76,74%, enquanto o *Naive Bayes* teve uma acurácia de 76,59% e F1 de 72,64%. Comparando com os resultados do trabalho de referência, do qual o BERT alcançou uma acurácia de 85,50%, pode-se observar que os resultados obtidos com o BERT nesta implementação superaram ligeiramente aqueles do trabalho de referência, mas ainda são similares. Assim como na base CSTNews, há alguns fatores para a variação dos resultados como a aleatoriedade na inicialização e a semente usada durante o treinamento e a validação, até mesmo as diferenças no ambiente de treinamento.

Em comparação com a base CSTNews de textos em português, que também é binária, a base SentiHood apresentou um desempenho menor em relação à acurácia e à medida F1. Contudo, é relevante observar que CSTNews tem quase o dobro de dados, o que interfere no treinamento e, portanto, nos testes de cada algoritmo.

5.4 Resultados para a base NewsKaggle

A base NewsKaggle é um conjunto de dados que oferece uma coleção de artigos de notícias classificados em negócios, tecnologia, esportes, educação e entretenimento. A seguir, ilustrados na Tabela 17, são apresentados os resultados.

Tabela 17 – Resultados de Acurácia e F1 para os Algoritmos Avaliados na Base NewsKaggle.

NewsKaggle		
	Acurácia	F1
Naive Bayes	87,50%	87,47%
SVM	97,10%	97,20%
BERT	98,60%	98,50%

Fonte: Elaborado pela autora (2024)

Para a base NewsKaggle, o algoritmo *Naive Bayes* apresentou uma acurácia de 87,50% e uma medida F1 de 87,47%. Por outro lado, o SVM alcançou uma acurácia de 97,10% e uma medida F1 de 97,20%. O modelo BERT se destacou ainda mais, com uma acurácia de 98,60% e uma medida F1 de 98,50%. Esses resultados indicam que o BERT foi altamente eficaz na tarefa de classificação multiclasse na língua inglesa, superando consideravelmente tanto o *Naive Bayes* quanto o SVM. Em comparação com a base que classifica em multiclasses em português, Fake.br, percebe-se um considerável aumento de acurácia e medida F1, mostrando que o BERT original parece ser melhor treinado que sua versão brasileira. Contudo, é importante lembrar que Fake.br tem quase 3 vezes menos dados que NewsKaggle para usar no treinamento e teste.

5.5 Discussão

Como mostrado na Tabela 18, que sumariza as caracteristicas das bases e os resultados obtidos, o BERT e o BERTimbau apresentaram resultados superiores aos métodos clássicos, como *Naive Bayes* e SVM, embora as diferenças nem sempre tenham sido significativas. Na base CSTNews, a diferença entre o BERTimbau e o SVM foi de apenas cerca de 5% tanto na acurácia quanto na medida F1. Na base Fake.br, o BERTimbau, o *Naive Bayes* e o SVM alcançaram resultados semelhantes, na faixa dos 60%. Por outro lado, na base Sentihood, o BERT em inglês mostrou uma diferença de até 10% em relação ao SVM e ao *Naive Bayes*, indicando uma superioridade mais clara, mas ainda na faixa de 70% a 80%, enquanto outras bases chegaram aos 90%. Na base NewsKaggle, o BERT superou o SVM por menos de 2%, enquanto em relação ao *Naive Bayes* a diferença foi de 10%. Esses números demonstram que, embora o BERT e o BERTimbau tenham desempenho superior, as diferenças em relação aos algoritmos clássicos nem sempre são acentuadas.

Tabela 18 – Sumarização das características de cada base e seus resultados.

	Língua	**#Texto**	**#Classes**	**Algoritmo**	**Resultado do Trabalho de Referência**	**Resultados obtidos**	
					Acurácia	**Acurácia**	**F1**
CSTNews	Português	9.960	2	Naive Bayes	-	64,16%	64,15%
				SVM	-	92,26%	92,23%
				BERTimbau	97,44%	97,99%	97,99%
Fake.br	Português	3.600	6	Naive Bayes	-	62,50%	60,39%
				SVM	-	62,22%	60,87%
				BERTimbau	-	66,66%	65,93%
SentiHood	Inglês	5.215	2	Naive Bayes	-	76,59%	72,64%
				SVM	-	76,95%	76,74%
				BERT	85,50%	86,42%	86,41%
NewsKaggle	Inglês	10.000	5	Naive Bayes	-	87,50%	87,47%
				SVM	-	97,10%	97,20%
				BERT	-	98,60%	98,50%

Fonte: Elaborado pela autora (2024)

A base em que o BERT obteve o melhor resultado foi a NewsKaggle, uma base em inglês e multiclasse desenvolvida para o site Kaggle, que reúne bases de dados para serem testadas. Apesar de se tratar de notícias reais e multiclasse, há uma preocupação na seleção das notícias e na quantidade de textos em cada classe, garantindo um equilíbrio na base. Além disso, é importante notar que essa base possui a maior quantidade de textos entre as analisadas, proporcionando mais dados para treinar o algoritmo e melhorar a precisão na classificação.

Por outro lado, o pior desempenho do BERT, no caso do BERTimbau, ocorreu na base Fake.br, um conjunto de dados em português brasileiro e multiclasse. O BERTimbau, por ter um vocabulário pré-treinado menor que o seu equivalente em inglês, pode não reconhecer certos vocabulários, impactando o desempenho. Também é relevante notar que, enquanto o melhor resultado do BERT foi na base com mais textos, o Fake.br possui a menor quantidade de textos entre as quatro bases selecionadas, resultando em menos exemplos para o treinamento do algoritmo.

Dessa forma, é fundamental conhecer bem a base de dados para escolher o algoritmo mais adequado. Como o BERT exige um poder computacional significativamente maior do que o SVM e o *Naive Bayes*, essa necessidade pode influenciar a escolha dependendo das limitações de recursos e do contexto da aplicação. Em bases maiores, o BERT demonstrou uma acurácia superior, mas em bases menores, a diferença entre o BERT e os algoritmos clássicos não foi tão expressiva.

6 Conclusão

Neste trabalho, foi realizada uma avaliação abrangente do desempenho do modelo BERT em tarefas de classificação de texto, comparando-o com algoritmos clássicos como *Naive Bayes* e Máquinas de Vetores de Suporte (SVM). A análise incluiu a investigação do impacto do pré-processamento de texto do BERT nas diferentes abordagens de classificação, além de uma avaliação detalhada das vantagens e desvantagens de cada técnica em termos de acurácia e medida F1.

O desempenho do BERT em tarefas de classificação de texto foi comparado com os algoritmos *Naive Bayes* e SVM em diferentes bases de dados: CSTNews, Fake.br, SentiHood e NewsKaggle. Os resultados demonstraram que o BERT, seja em sua versão brasileira, BERTimbau, ou na sua versão original em inglês, superou os algoritmos clássicos em termos de acurácia e medida F1.

O pré-processamento de texto demonstrou ter um impacto significativo na performance dos modelos utilizados. Técnicas de pré-processamento, como a tokenização, juntamente com as estratégias de redução de dimensionalidade e a representação vetorial oferecida pelo BERT, influenciam diretamente a qualidade dos *embeddings* gerados e, consequentemente, o desempenho dos modelos. A utilização do pré-treinamento do BERT para realizar o pré-processamento contribuiu para obter bons resultados de acurácia e medida F1 para os três algoritmos analisados.

O BERTimbau, adaptado para o português, demonstrou um bom desempenho nas bases de dados em língua portuguesa, alcançando índices interessantes de acurácia e medida F1 nas tarefas de classificação de texto. No entanto, em comparação com o BERT original, treinado em inglês, o BERTimbau apresenta oportunidades de aprimoramento. Isso pode ser atribuído ao fato de que o vocabulário do BERT original, sendo maior e mais diversificado devido ao maior volume de dados em inglês, proporciona um contexto mais rico e abrangente para a análise e geração dos *embeddings*.

Outro fator que pode influenciar os resultados inferiores do BERTimbau em relação ao BERT é a quantidade total de textos em cada base de dados. A base com a maior quantidade de exemplos, NewsKaggle em inglês, apresentou os melhores resultados, enquanto a base com a menor quantidade de textos, Fake.br em português, teve o desempenho mais baixo. Essa diferença na quantidade de dados disponíveis influencia diretamente o processo de treinamento e aprendizado do algoritmo, afetando sua capacidade de classificar corretamente durante a fase de testes.

A escolha entre as abordagens analisadas deve levar em conta o equilíbrio entre a precisão desejada e os recursos computacionais disponíveis. O BERT se destaca como uma

alternativa promissora para futuras tarefas de classificação, apresentando consistentemente os melhores resultados em termos de acurácia e medida F1 em diversas bases de dados. Contudo, o BERT demanda um maior poder computacional e um tempo de treinamento mais prolongado. Por outro lado, métodos como *Naive Bayes* e SVM, apesar de exigirem menos recursos computacionais e um tempo de treinamento reduzido, demonstraram uma performance inferior ao BERT, ficando aquém em termos de acurácia e medida F1 em todas as bases analisadas.

6.1 Trabalhos Futuros

A maior contribuição deste trabalho reside na avaliação comparativa do desempenho do modelo BERT em tarefas de classificação de texto, em relação a algoritmos clássicos como *Naive Bayes* e Máquinas de Vetores de Suporte (SVM) além do uso da fase de pré-treinamento do BERT para o pré-processamento. A utilização do pré-treinamento do BERT com sua tokenização e *embeddings* demonstrou bons resultados para os algoritmos clássicos. Uma abordagem promissora para futuros trabalhos é a comparação direta entre o BERT e métodos tradicionais de pré-processamento, como TF-IDF e n-gramas. Os métodos tradicionais são menos exigentes em termos de recursos computacionais e frequentemente produzem resultados eficazes. Realizar uma comparação direta pode esclarecer se o pré-treinamento do BERT realmente proporciona melhorias significativas em relação a essas técnicas estabelecidas. Essa análise pode validar se o investimento em BERT justifica seus custos mais elevados.

Outro viés a ser seguido para trabalhos futuros é exploração de outros algoritmos de classificação e técnicas de aprendizado de máquina, incluindo métodos baseados em aprendizado profundo. Tais abordagens podem oferecer novas perspectivas e potencialmente melhorar a acurácia. A avaliação e comparação de novos métodos podem fornecer informações importantes sobre como aprimorar a classificação de textos e fortalecer a eficácia dos sistemas de processamento de linguagem natural.

Além disso, para aprofundar a pesquisa, pode-se explorar a incorporação de dados adicionais provenientes de diversas fontes. Essa ampliação da base de dados pode proporcionar uma visão mais detalhada para a análise dos algoritmos em contextos distintos, especialmente em língua portuguesa, onde a quantidade de dados disponíveis é menor em comparação ao inglês. Outra abordagem possível é escolher bases de dados semelhantes, tanto em tipo de texto quanto em quantidade de textos. Textos semelhantes podem trazer resultados mais claros nas comparações dos algoritmos entre diferentes línguas. Um aspecto adicional a ser considerado em pesquisas futuras é o uso de testes estatísticos para avaliar os resultados, garantindo uma análise mais robusta das diferenças entre os métodos.

Referências

AGGARWAL, C. C. **Data Mining: The Textbook**. Cham: Springer International Publishing, 2015. 734 p. Citado 3 vezes nas páginas 16, 19 e 29.

ALPAYDIN, E. **Introduction to Machine Learning**. 3. ed. Massachusetts: The MIT Press, 2014. 640 p. Citado na página 28.

BAEZA-YATES, R.; RIBEIRO-NETO, B. **Recuperação de informação: Conceitos e tecnologia das máquinas de busca**. 2. ed. Porto Alegre: Bookman, 2013. 612 p. Citado 10 vezes nas páginas 18, 24, 25, 28, 29, 32, 43, 44, 45 e 54.

BARBOSA, G.; BATISTA, H.; MIRANDA, P.; SANTOS, J.; ISOTANI, S.; CORDEIRO, T.; BITTENCOURT, I. I.; MELLO, R. F. Aprendizagem de máquina para classificação de tipos textuais: Estudo de caso em textos escritos em português brasileiro. In: **Anais do XXXIII Simpósio Brasileiro de Informática na Educação**. Porto Alegre, RS, Brasil: SBC, 2022. p. 920–931. Disponível em: <https://sol.sbc.org.br/index.php/sbie/article/view/22470>. Citado 2 vezes nas páginas 47 e 49.

BENEVENUTO, F.; ARAÚJO, M.; RIBEIRO, F. N. Métodos para análise de sentimentos em mídias sociais. In: CRISTO, M. A. P. de; OLIVEIRA, D. F. de; PEREIRA, A. C. M.; BENEVENUTO, F.; MORENO, M. F.; ROESLER, V. (Ed.). **Proceedings of the 21st Brazilian Symposium on Multimedia and the Web**. ACM, 2015. p. 11. Disponível em: <https://doi.org/10.1145/2820426.2820642>. Citado na página 23.

BENGFORT, B.; BILBRO, R.; OJEDA, T. **Applied Text Analysis with Python: Enabling Language Aware Data Products with Machine Learning**. 1. ed. Sebastopol, California, USA: O'Reilly, 2016. 332 p. Citado na página 20.

BOULLE, P. **La Planète des Singes**. França: Julliard, 2011. 256 p. EBook (ePub). Citado na página 6.

BRAZ JUNIOR, O.; FILETO, R. Investigando coerência em postagens de um fórum de dúvidas em ambiente virtual de aprendizagem com o bert. In: **Anais do XXXII Simpósio Brasileiro de Informática na Educação**. Porto Alegre, RS, Brasil: SBC, 2021. p. 749–759. Disponível em: <https://sol.sbc.org.br/index.php/sbie/article/view/18103>. Citado 6 vezes nas páginas 10, 47, 49, 51, 56 e 57.

CASELI, H. de M.; NUNES, M. das G. V. **Processamento de Linguagem Natural: Conceitos, Técnicas e Aplicações em Português**. 1. ed. São Paulo: Brasileiras em PLN, 2023. 563 p. Citado 6 vezes nas páginas 16, 19, 20, 21, 23 e 26.

DEVLIN, J.; CHANG, M.-W.; LEE, K.; TOUTANOVA, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In: BURSTEIN, J.; DORAN, C.; SOLORIO, T. (Ed.). **Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies**. Minneapolis, Minnesota: Association for Computational Linguistics, 2019. v. 1, p. 4171–4186. Disponível em: <https://aclanthology.org/N19-1423>. Citado 6 vezes nas páginas 14, 33, 37, 38, 39 e 53.

FAYYAD, U.; PIATETSKY-SHAPIRO, G.; SMYTH, P. From data mining to knowledge discovery in databases. **AI Magazine**, v. 17, n. 3, p. 37, mar. 1996. Disponível em: <https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/1230>. Citado na página 50.

GOLDSCHMIDT, R. **Data Mining**. Rio de Janeiro: Grupo GEN, 2015. 296 p. ISBN 9788595156395. Acesso em: 08 abr. 2024. Citado na página 50.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. Massachusetts: MIT Press, 2016. 800 p. Citado na página 36.

GUIMARÃES, L. M. S.; MEIRELES, M. R. G.; ALMEIDA, P. E. M. de. Avaliação das etapas de pré-processamento e de treinamento em algoritmos de classificação de textos no contexto da recuperação da informação. **Perspectivas em Ciência da Informação**, v. 24, n. 1, 2019. Disponível em: <https://doi.org/10.1590/1981-5344/3505>. Citado na página 28.

HASSAN, S. U.; AHAMED, J.; AHMAD, K. Analytics of machine learning-based algorithms for text classification. **Sustainable Operations and Computers**, v. 3, p. 238–248, 2022. Disponível em: <https://doi.org/10.1016/j.susoc.2022.03.001>. Citado na página 13.

HASTOMO, W.; KARNO, A. S. B.; KALBUANA, N.; MEIRIKI, A.; SUTARNO. Characteristic parameters of epoch deep learning to predict Covid-19 data in Indonesia. **Journal of Physics: Conference Series**, IOP Publishing, v. 1933, n. 1, p. 012050, jun 2021. Disponível em: <https://dx.doi.org/10.1088/1742-6596/1933/1/012050>. Citado na página 56.

KRISHNAIAH, V.; NARSIMHA, G.; CHANDRA, N. S. Survey of classification techniques in data mining. **International Journal of Computer Sciences and Engineering**, v. 2, p. 65–74, 2014. Citado na página 13.

LAI, S. **Investigation of BERT Fine-tuning Strategies**. 2024. Stanford CS224N Default Project. Disponível em: <https://web.stanford.edu/class/cs224n/final-reports/256988931.pdf>. Acesso em: 02 ago. 2024. Citado na página 54.

LANE, H.; DYSHEL, M. **Natural Language Processing in Action**. 2. ed. USA: Manning, 2021. 675 p. Citado 2 vezes nas páginas 19 e 25.

MAGALHÃES, L. H. de; MATOS, F. F.; SOUZA, R. R. Comparação entre algoritmos de classificação aplicados na predição de notícias de jornais on-line. In: ENANCIB. **A Ciência da Informação e a era da Ciência de Dados**. Florianópolis, 2019. ISSN 2177-3688. Citado na página 14.

MOREIRA, L. S.; LUNARDI, G. M.; RIBEIRO, M. de O.; SILVA, W.; BASSO, F. P. A study of algorithm-based detection of fake news in brazilian election: Is BERT the best. **IEEE Latin America Transactions**, v. 21, n. 8, p. 897–903, 2023. Citado 4 vezes nas páginas 46, 47, 49 e 51.

PLATH, H. O.; PAIVA, M. E. O.; PINTO, D. L.; COSTA, P. D. P. Detecção de discurso de Ódio contra mulheres em textos em português brasileiro: Construção da base mina-br e modelo de classificação. **Revista Eletrônica de Iniciação Científica em Computação**, v. 20, n. 3, jul. 2022. Disponível em: <https:

//sol.sbc.org.br/journals/index.php/reic/article/view/2696>. Citado 2 vezes nas páginas 46 e 49.

RUSSELL, S.; NORVIG, P. **Inteligência artificial**. 3. ed. Rio de Janeiro: Elsevier, 2013. 1152 p. Citado na página 28.

SANTOS, F. A.; KOBELLARZ, J. K.; SOUZA, F. R. de; VILLAS, L. A.; SILVA, T. H. Processamento de linguagem natural em textos de mídias sociais: Fundamentos, ferramentas e aplicações. In: SAADE, D. C. M.; MINETTO, R.; WILLRICH, R.; SILVA, T. H.; DORINI, L. B. (Ed.). **Minicursos do XXVIII Simpósio Brasileiro de Sistemas Multimídia e Web**. [S.l.]: Sociedade Brasileira de Computação, 2022. p. 51–100. Citado 2 vezes nas páginas 22 e 23.

SOUSA, O. L. V. d.; MAGALHÃES, D. M. V.; CAMPELO, V. E. S.; SILVA, R. R. V. e. An automatic method to medical documents labeling and categorization. **iSys - Brazilian Journal of Information Systems**, v. 15, n. 1, p. 13:1–13:13, out. 2022. Disponível em: <https://sol.sbc.org.br/journals/index.php/isys/article/view/2260>. Citado 2 vezes nas páginas 46 e 49.

SOUZA, F.; NOGUEIRA, R.; LOTUFO, R. BERTimbau: Pretrained BERT models for Brazilian Portuguese. In: CERRI, R.; PRATI, R. C. (Ed.). **Intelligent Systems**. Cham: Springer International Publishing, 2020. p. 403–417. ISBN 978-3-030-61377-8. Citado na página 40.

SOUZA, F. C. de. **BERTimbau: pretrained BERT models for Brazilian Portuguese**. 61 p. Dissertação (Mestrado) — Universidade Estadual de Campinas, 2020. Disponível em: <https://hdl.handle.net/20.500.12733/1640809>. Acesso em: 22 mar. 2024. Citado 2 vezes nas páginas 38 e 53.

SUN, C.; HUANG, L.; QIU, X. Utilizing BERT for aspect-based sentiment analysis via constructing auxiliary sentence. In: BURSTEIN, J.; DORAN, C.; SOLORIO, T. (Ed.). **Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies**. Minneapolis, Minnesota: Association for Computational Linguistics, 2019. v. 1, p. 380–385. Disponível em: <https://aclanthology.org/N19-1035>. Citado 6 vezes nas páginas 10, 48, 49, 51, 52 e 58.

TAN, P.-N.; STEINBACH, M.; KARPATNE, A.; KUMAR, V. **Introduction To Data Mining**. 2. ed. New York: Pearson, 2019. 864 p. Citado 4 vezes nas páginas 17, 29, 30 e 35.

TAN, P.-N.; STEINBACH, M.; KUMAR, V. **Introduction To Data Mining**. New York: Pearson, 2014. 769 p. Citado 3 vezes nas páginas 40, 42 e 43.

VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, L.; POLOSUKHIN, I. Attention is all you need. In: **Advances in Neural Information Processing Systems**. Cambridge, MA: MIT Press, 2017. p. 6000–6010. Citado 5 vezes nas páginas 33, 34, 35, 36 e 53.

VELANKAR, A.; PATIL, H.; JOSHI, R. Mono vs multilingual bert for hate speech detection and text classification: A case study in marathi. In: GAYAR, N. E.; TRENTIN, E.; RAVANELLI, M.; ABBAS, H. (Ed.). **Artificial Neural Networks in Pattern**

Recognition. Cham: Springer International Publishing, 2023. p. 121–128. ISBN 978-3-031-20650-4. Citado na página 40.

VISA, S.; RAMSAY, B.; RALESCU, A.; VANDERKNAAP, E. Confusion matrix-based feature selection. **Proceedings of The 22nd Midwest Artificial Intelligence and Cognitive Science Conference 2011**, p. 120–127, 2011. Citado na página 40.

WEISS, S. M.; INDURKHYA, N.; ZHANG, T. **Fundamentals of Predictive Text Mining**. 2. ed. London: Springer International Publishing, 2015. 239 p. Citado 10 vezes nas páginas 13, 16, 31, 32, 33, 40, 42, 43, 54 e 55.

WITTEN, I. H.; FRANK, E.; HALL, M. A. **Data Mining: Practical Machine Learning Tools and Techniques**. Third. Burlington, MA, USA: Elsevier, 2011. 664 p. ISBN 978-0-12-374856-0. Citado na página 55.

WU, S.; DREDZE, M. Are all languages created equal in multilingual BERT? In: GELLA, S.; WELBL, J.; REI, M.; PETRONI, F.; LEWIS, P.; STRUBELL, E.; SEO, M.; HAJISHIRZI, H. (Ed.). **Proceedings of the 5th Workshop on Representation Learning for NLP**. Online: Association for Computational Linguistics, 2020. p. 120–130. Disponível em: <https://aclanthology.org/2020.repl4nlp-1.16>. Citado na página 40.

YAN, D.; LI, K.; GU, S.; YANG, L. Network-based bag-of-words model for text classification. **IEEE Access**, v. 8, 2020. Disponível em: <https://doi.org/10.1109/ACCESS.2020.2991074>. Citado na página 25.

ZONG, C.; XIA, R.; ZHANG, J. **Text Data Mining**. 2. ed. Singapore: Springer International Publishing and Tsinghua University Press, 2021. 351 p. Citado 8 vezes nas páginas 18, 19, 20, 26, 29, 32, 41 e 44.