UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Estudo comparativo de abordagens para classificação de polaridade de análise de sentimentos em tweets sobre mudanças climáticas

Rosângela Pieroni

Trabalho de Conclusão de Curso
MBA em Inteligência Artificial e Big Data

ICMC USP SÃO CARLOS

UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Estudo comparativo de abordagens para classificação de polaridade em análise de sentimentos em tweets sobre mudanças climáticas

USP - São Carlos

2024

Rosângela Pieroni

Estudo comparativo de abordagens para classificação de polaridade em análise de sentimentos em tweets sobre mudanças climáticas

Trabalho de conclusão de curso apresentado ao Departamento de Ciências de Computação do Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo - ICMC/USP, como parte dos requisitos para obtenção do título de Especialista em Inteligência Artificial e Big Data.

Área de concentração: Inteligência Artificial

Orientador: Prof. Dr. Ricardo Rodrigues Ciferri

USP - São Carlos

2024

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

Pieroni, Rosângela
P619e Estudo comparativo de abordagens para classificação de polaridade de análise de sentimentos em tweets sobre mudanças climáticas / Rosângela Pieroni; orientador Ricardo Rodrigues Ciferri. -- São Carlos, 2024.
98 p.

Trabalho de conclusão de curso (MBA em Inteligência Artificial e Big Data) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2024.

1. PROCESSAMENTO DE LINGUAGEM NATURAL. 2. ANÁLISE DE SENTIMENTOS. 3. MUDANÇAS CLIMÁTICAS. I. Rodrigues Ciferri, Ricardo, orient. II. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

DEDICATÓRIA

A minha família pela compreensão e apoio incansável.

AGRADECIMENTOS

Ao Prof. Dr. Ricardo Rodrigues Ciferri, pela compreensão e ajuda que me proporcionou nessa jornada.

EPÍGRAFE

"O futuro da IA será moldado pelas escolhas que fazemos hoje sobre como desenvolvê-la e aplicá-la."

CRAWFORD, K.

RESUMO

PIERONI, R. **Estudo comparativo de abordagens para classificação de polaridade em análise de sentimentos em tweets sobre mudanças climáticas**. 2024. Trabalho de conclusão de curso (MBA em Inteligência Artificial e Big Data) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2024.

Diante da crescente importância das redes sociais na formação da opinião dos usuários sobre os mais abrangentes temas, que variam da compra de um produto, questões sociais até decisões políticas, é de suma importância direcionar os olhares para questões atuais que provocam impacto significativo na vida das pessoas e na sociedade, como é o caso das mudanças climáticas. Dessa forma, analisar os sentimentos expressos nas redes sociais para melhor compreender as percepções das pessoas com relação a esse assunto é fundamental para formulação de políticas, campanhas de conscientização e avaliação da eficácia de ações de mitigação. O objetivo desse trabalho foi aplicar abordagens de Processamento de Linguagem Natural (PLN) para realizar a análise de sentimentos em uma base de dados oriunda de redes sociais com textos acerca de mudanças climáticas e comparar os resultados obtidos. O conjunto de dados utilizado foi obtido a partir do arquivo nltk_split.csv disponível no Kaggle. Os dados foram pré-processados realizando a limpeza dos textos (remoção dos caracteres especiais, pontuação, tags, HTML, URLs), os textos foram convertidos para letras minúsculas (lowercasing), foram eliminadas as palavras irrelevantes (stopwords) e as palavras foram reduzidas à sua forma raiz (lemmatization). Os dados ainda foram filtrados de forma que ficaram balanceados entre as classificações de sentimento positivo e negativo (22.698 positivos e 17.704 negativos). Foram aplicados os métodos baseados em extração de recursos (Bag-of-Words, Term Frequency-Inverse Document Frequency e Word2Vec), em recursos léxicos (TextBlob e VADER), em aprendizado de máquina supervisionado (Regressão Logística), em aprendizado profundo (LSTM e GRU) e o Transformers (BERT). O desempenho dos modelos foi avaliado usando as métricas de precisão, revocação, F1-score e acurácia. Os resultados obtidos com as técnicas de extração de recursos se mostraram bem similares com relação as palavras que mais se destacaram na Bag-of-Words, com as palavras que apresentaram maior frequência no conjunto de dados quando aplicado o TF-IDF, e na proximidade dessas palavras quando aplicado o Word2Vec. Dentre as técnicas implementadas, o maior e o menor valor de acurácia foram identificados no método de recursos léxicos: VADER com 0,99 e TextBlob com 0,77. As demais técnicas implementadas (Regressão Logística, LSTM, GRU e BERT) apresentaram valores bem próximos ou iguais (0,95, 0,96, 0,98 e 0,98, respectivamente).

Palavras-chave: Processamento de Linguagem Natural; Análise de Sentimentos; Mudanças Climáticas.

ABSTRACT

PIERONI, R. **Comparative study of approaches for polarity classification in sentiment analysis in tweets about climate change**. 2024. Course completion work (MBA in Artificial Intelligence and Big Data) – Institute of Mathematical and Computer Sciences, University of São Paulo, São Carlos, 2024.

Given the growing importance of social networks in forming users' opinions on the most comprehensive topics, ranging from the purchase of a product, social issues to political decisions, it is extremely important to direct attention to current issues that have a significant impact on the lives of people and society, as is the case with climate change. Therefore, analyzing the feelings expressed on social networks to better understand people's perceptions regarding this issue is essential for formulating policies, awareness campaigns and evaluating the effectiveness of mitigation actions. The objective of this work was to apply Natural Language Processing (NLP) approaches to perform sentiment analysis on a database from social networks with texts about climate change and compare the results obtained. The dataset used was obtained from the nltk_split.csv file available on Kaggle. The data was pre-processed by cleaning the texts (removing special characters, punctuation, tags, HTML, URLs), texts were converted to lowercase letters, stopwords were eliminated and words were reduced to its root form (lemmatization). The data was also filtered so that it was balanced between positive and negative sentiment classifications (22,698 positive and 17,704 negative). Methods based on feature extraction (Bag-of-Words, Term Frequency-Inverse Document Frequency and Word2Vec), lexical resources (TextBlob and VADER), supervised machine learning (Logistic Regression), deep learning (LSTM and GRU) and Transformers (BERT)) were applied. The performance of the models was evaluated using precision, recall, F1-score and accuracy metrics. The results obtained with the feature extraction techniques were very similar in relation to the words that stood out most in Bag-of-Words, with the words that presented the highest frequency in the data set when TF-IDF was applied, and in proximity of these words when Word2Vec is applied. Among the techniques implemented, the highest and lowest accuracy values were identified in the lexical resources method: VADER with 0.99 and TextBlob with 0.77. The other implemented techniques (Logistic Regression, LSTM, GRU and BERT) presented very close or equal values (0.95, 0.96, 0.98 and 0.98, respectively).

Keywords: Natural Language Processing; Sentiment Analysis; Climate Change.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ANN	–	Redes Neurais Artificiais
BERT	–	Bidirectional Encoder Representations from Transformers
BoW	–	Bag of Words
CNNs	–	Redes Neurais Convulacionais
DNN	–	Redes Neurais Profundas
GPT	–	Generative Pre-trained Transformer
GPT-2	–	Generative Pre-trained Transformer 2
GRU	–	Gated Recurrent Units
LSTM	–	Long Short-Term Memory (Redes Neurais de Memória de Longo Prazo)
ML	–	Machine Learning
MSE	–	Erro Quadrático Médio
PLN	–	Processamento de Linguagem Natural
ReLU	–	Rectified Linear Unit
RNNs	–	Redes Neurais Recorrentes
SGD	–	Stochastic Gradient Descent
SVM	–	SVM (Máquinas de Vetores de Suporte)
TF-IDF	–	Term Frequency-Imersive Document Frequency
VADER	–	Valence Aware Dictionary and sEntiment Reasoner

SUMÁRIO

1 INTRODUÇÃO

1.1 Contexto

O comportamento do ser humano é influenciado por opiniões de outras pessoas, ou seja, as percepções sobre um assunto estão condicionadas à maneira como outras pessoas as fazem. Assim como as pessoas buscam opiniões de produtos, serviços ou acontecimentos com outras pessoas para tomar alguma decisão, as organizações também se utilizam desse mecanismo por meio de surveys, questionários, consultores para tomarem decisões.

Com a popularização de plataformas online, as pessoas passaram a compartilhar suas experiências e opiniões em reviews, fóruns, blogs, redes sociais. Dessa forma, a esfera de influenciadores de opiniões tomou uma proporção global. Além disso, as notícias e os relatórios sobre opiniões e tendências sobre diferentes tópicos também são publicados na web e tornam-se fatores que influenciam as tomadas de decisões que abrange do indivíduo até empresas.

Diversas empresas de pesquisas de tendências das mídias sociais apresentam um crescimento e uma dependência dos usuários das redes sociais para tomarem decisões que variam de compras de produtos até opiniões políticas. O Social Media Trends Report da Global Web Index (2021) mostra que 54% dos usuários de redes sociais as utilizam para pesquisar produtos antes de tomar decisões de compra. A pesquisa Global Millennial do Deloitte (2020) revelou que 47% dos consumidores foram influenciados por conteúdos em redes sociais antes de fazer uma compra. Além disso, mostrou a crescente importância das mídias sociais na formação de opinião sobre questões sociais, produtos e até decisões políticas. O relatório The Sprout Social Index: Edition XVII, Accelerate, (2021) relata que 91% das empresas aumentaram seus investimentos em redes sociais para influenciar decisões de negócios e 78% das empresas afirmam que as redes sociais fornecem insights que afetam diretamente suas decisões estratégicas. Segundo o relatório Social Media and Politics Report da Pew Research Center (2020), 55% dos americanos afirmam que a mídia social é uma das principais fontes de informação para decisões políticas e eleitorais. Esses dados mostram como as redes influenciam não só questões de consumo, mas também decisões de cunho social e político.

Porém, com essa enorme quantidade de dados torna-se impraticável uma análise manual, sendo necessária a criação de métodos automáticos para analisar os dados (LIU et al., 2010). Dessa forma, uma das áreas da inteligência artificial que auxilia nesse sentido é o Processamento de Linguagem Natural (PLN), que se dedica ao desenvolvimento de algoritmos

e métodos que permitem ao computador interpretar e interagir com a linguagem humana de forma útil e eficiente (Caseli et al, 2024). Alguns desses métodos mais comuns são: os métodos baseados em aprendizado de máquina que abrangem Regressão Logística, Máquinas de Vetores de Suporte (SVM), Árvore de Decisão e Redes Neurais.

Uma das aplicações da PLN é avaliar o sentimento expresso nos textos de redes sociais, o que é chamado de análise de sentimentos. A análise de sentimentos é um tipo de mineração de textos focado em identificar e extrair sentimentos subjetivos de um texto. O objetivo é determinar se o sentimento é positivo, negativo ou neutro. A mineração de textos é definida como um conjunto de técnicas e processos para descoberta de conhecimento inovador a partir de grandes coleções textuais (Rezende, 2003).

Nesse contexto, a análise de sentimentos pode ser aplicada para compreender as percepções das pessoas sobre vários assuntos. Os assuntos mais relevantes discutidos nas redes sociais variam conforme as tendências e eventos atuais, mas alguns temas tendem a aparecer com maior frequência devido ao impacto significativo na vida cotidiana e na sociedade, dentre eles estão as mudanças climáticas. As mudanças climáticas tem impacto na vida das pessoas em diversas dimensões: saúde, segurança alimentar, água e recursos hídricos, segurança e desastres naturais, economia e sustentabilidade, biodiversidade e ecossistemas, justiça social e equidade, qualidade de vida e bem-estar.

Mudanças climáticas são uma das questões mais críticas do nosso tempo. Ao explorar como diferentes abordagens classificam o sentimento sobre este tema, pode-se obter um entendimento mais rico e detalhado da relação entre as mídias sociais e as questões ambientais.

1.2 Motivação

A análise de sentimentos pode ser aplicada para compreender as percepções das pessoas sobre as mudanças climáticas e, consequentemente, proporcionar uma inteligência analítica que auxilie em formulação de políticas, campanhas de conscientização e na avaliação da eficácia de ações de mitigação.

Realizar esse estudo comparativo entre abordagens aplicadas para classificar o sentimento expresso em textos nas redes sociais sobre mudanças climáticas é relevante para a sociedade em vários contextos. Analisar sentimentos expressos nas redes sociais ajuda a entender como as pessoas percebem as mudanças climáticas, suas preocupações e atitudes. Essa compreensão é essencial para formuladores de políticas e organizações ambientais, permitindo que eles desenvolvam estratégias de comunicação mais eficazes e abordagens que ressoem com

o público. Compreender melhor os sentimentos e as opiniões predominantes ajudam a identificar barreiras e oportunidades para iniciativas comunitárias e políticas.

Além disso, cada abordagem para a classificação de sentimentos tem suas forças e limitações. Um estudo comparativo pode revelar quais métodos oferecem maior precisão e confiabilidade para diferentes tipos de dados e contextos, ajudando a aprimorar modelos existentes e a desenvolver novos. Ademais, a mineração de textos é desafiador e apresenta alto nível de complexidade semântica, pois se trata de subjetividade, emoções e opiniões.

1.3 Objetivo

O objetivo é aplicar abordagens de PLN para realizar a análise de sentimentos em uma base de dados do Twitter com textos acerca de mudanças climáticas no período de 21/09/2017 até 19/05/2019 e comparar a eficácia da classificação dos sentimentos em positivo e negativo dessas abordagens.

1.4 Organização da Monografia

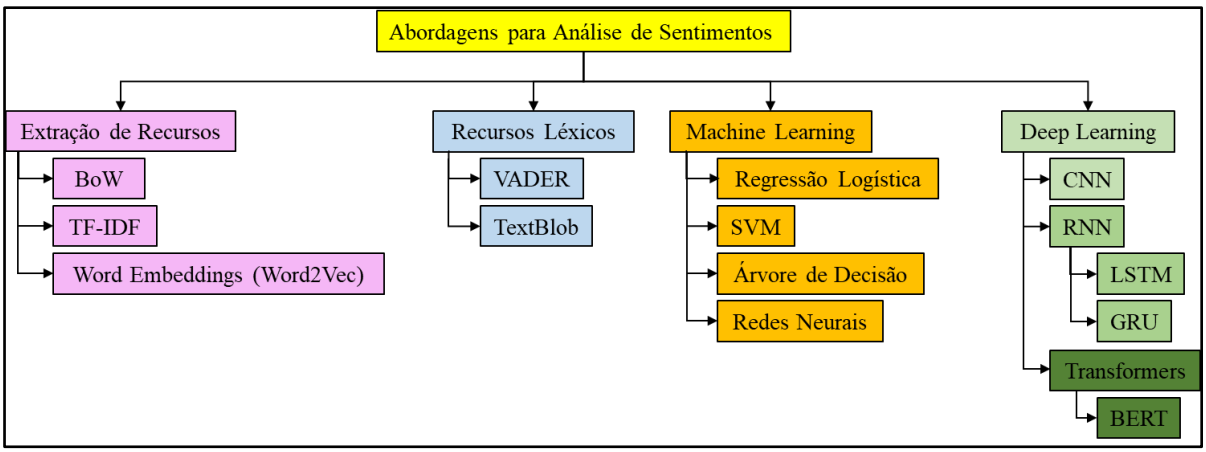
Esse trabalho está dividido em seis capítulos. No primeiro capítulo está a introdução com a definição do contexto, com a motivação e com o objetivo. No capítulo 2 está apresentada a fundamentação teórica de alguns métodos utilizados na análise de sentimentos. No capítulo 3 estão apresentados alguns trabalhos relacionados com seus objetivos principais e resultados obtidos. No capítulo 4 está descrita a proposta do trabalho. No capítulo 5 está descrita a avaliação experimental com a definição do dataset, resultados e discussão. No capítulo 6 estão apresentadas as conclusões e no capítulo 7 estão as referências.

2 FUNDAMENTAÇÃO TEÓRICA

Nesse capítulo é apresentado o referencial teórico considerado no escopo da elaboração deste trabalho. Primeiramente está a definição de PLN e seus marcos históricos representando seu desenvolvimento ao longo das décadas. Em seguida é apresentada a definição de análise de sentimentos e algumas abordagens para sua aplicação sobre um conjunto de textos. E, na sequência são apresentadas as definições das abordagens baseados em regras de recursos

léxicos, em aprendizado de máquina e em aprendizado profundo. Na Figura 1 é possível observar as abordagens para análise de sentimentos que estão no contexto desse trabalho.

Figura 1 - Abordagens para Análise de Sentimentos



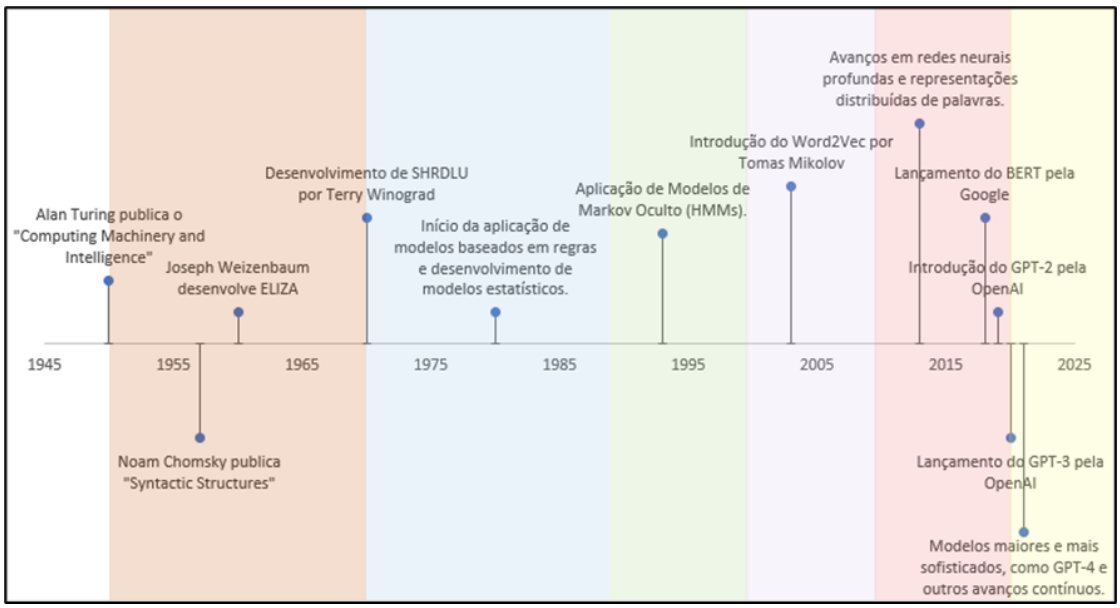
Fonte: Própria autora.

2.1 Processamento de Linguagem Natural (PLN)

O PLN é um campo de pesquisa que tem como objetivo investigar e propor métodos e sistemas de processamento computacional da linguagem humana (Caseli, 2024). PLN refere-se a um conjunto de técnicas e métodos usados para fazer com que os computadores entendam, interpretem e respondam a linguagem natural da maneira mais semelhante possível à forma como os humanos compreendem a linguagem. Ele envolve a aplicação de algoritmos e modelos computacionais para processar e analisar grandes volumes de dados textuais e de fala.

O desenvolvimento do PLN pode ser traçado por vários marcos históricos significativos (Hirschberg e Manning, 2015), que podem ser observados na Figura 2.

Figura 2 - Marcos históricos da PLN



Fonte: Própria autora.

Nas décadas de 1950 e 1960 ocorreram os primeiros passos:

- 1950: Alan Turing publica o artigo "*Computing Machinery and Intelligence*", introduzindo a ideia de que máquinas podem "pensar" e propondo o "Teste de Turing" para avaliar a inteligência das máquinas. Esse conceito inicial foi fundamental para a ideia de que computadores poderiam processar linguagem.
- 1957: Noam Chomsky publica "*Syntactic Structures*", que introduz a teoria da gramática gerativa. Suas ideias sobre a estrutura da linguagem e a gramática formal influenciaram profundamente o desenvolvimento de métodos de análise sintática no PLN.
- 1960: Joseph Weizenbaum desenvolve o ELIZA, um dos primeiros chatbots, que simulava uma conversa com um psicoterapeuta. Embora muito simples, ELIZA foi um dos primeiros sistemas a demonstrar a viabilidade da interação homem-máquina baseada em linguagem natural.

Nas décadas de 1970 e 1980 ocorreram desenvolvimento de algoritmos e modelos:

- 1970: na década de 1970 ocorre o desenvolvimento de modelos baseados em regras para análise sintática e semântica. Sistemas como o SHRDLU, desenvolvido por Terry Winograd, demonstram que é possível programar

computadores para entender e manipular a linguagem natural em um contexto limitado.

- 1980: o desenvolvimento de técnicas estatísticas começa a ganhar relevância. Frederick Jelinek e outros pesquisadores do IBM T.J. Watson Research Center começam a usar modelos baseados em probabilidade para o reconhecimento de fala e para a tradução automática.

Na década de 1990 ocorreu a revolução dos dados e modelos estatísticos:

- 1990: o avanço na computação e o crescimento da internet proporcionaram grandes volumes de dados textuais, permitindo a aplicação de técnicas de modelagem estatística. O uso de modelos de linguagem baseados em n-gramas se tornou popular para tarefas como correção ortográfica e para tradução automática.
- 1993: o Modelo de Markov Oculto (HMM) começou a ser amplamente utilizado para tarefas como o reconhecimento de fala e para rotulagem de partes do discurso (POS tagging).

Na década de 2000 ocorreram avanços em aprendizado de máquina e aprendizado profundo:

- 2000: Support Vector Machines (SVMs) e outras técnicas de aprendizado de máquina começaram a ser aplicadas ao PLN. A abordagem estatística começou a dominar as tarefas de processamento de linguagem.
- 2003: o conceito de Word Embeddings foi introduzido com o modelo Word2Vec desenvolvido por Tomas Mikolov e seus colegas. O Word2Vec representa palavras como vetores em um espaço contínuo, capturando suas relações semânticas e contextuais.

A década de 2010 foi a era dos modelos de transformers e aprendizado profundo:

- 2013: Yoshua Bengio e outros pesquisadores publicaram trabalhos fundamentais sobre redes neurais profundas e representações de palavras distribuídas, que influenciaram o desenvolvimento de modelos mais avançados de PLN.
- 2018: a publicação do modelo BERT (Bidirectional Encoder Representations from Transformers) pela Google revolucionou o campo. BERT é um modelo de

linguagem baseado em transformers que permite uma compreensão bidirecional do contexto das palavras em uma sentença.

- 2019: GPT-2 (Generative Pre-trained Transformer 2), desenvolvido pela OpenAI, demonstrou a capacidade dos modelos de linguagem para gerar texto coerente e realista, avançando significativamente na capacidade de geração e compreensão de linguagem.

Na década de 2020 está ocorrendo o desenvolvimento de Modelos Avançados e Aplicações Práticas:

- 2020: GPT-3, a terceira versão do modelo da OpenAI, foi lançada com 175 bilhões de parâmetros, estabelecendo novos padrões para a geração e compreensão de linguagem natural.
- 2021: a pesquisa e desenvolvimento continuam a avançar com modelos maiores e mais sofisticados, e a aplicação de técnicas de PLN se expande para áreas como atendimento ao cliente, análise de sentimentos, tradução automática e muito mais.

2.2 Análise de Sentimentos

Análise de sentimento é o processo para identificar e extrair informações subjetivas de textos usando PLN e mineração de texto (Rezende, 2003). O sentimento é classificado em positivo, quando o texto expressa uma opinião favorável ou otimista, em negativo, quando o texto expressa uma opinião desfavorável ou pessimista, em neutro, quando o texto não demonstra claramente uma opinião positiva ou negativa.

A análise de sentimento pode ser aplicada em vários contextos que podem se valer das opiniões e impressões das pessoas sobre determinado produto, assunto ou serviço para tomar decisões. Dessa forma, empresas podem usar análise de sentimento para obter a opinião dos clientes sobre produtos e serviços para identificar melhorias ou estratégias de marketing. Quando aplicadas em redes sociais a análise de sentimento pode auxiliar na percepção pública sobre eventos ou tópicos específicos. Também pode ser utilizada para avaliar a tonalidade de notícias sobre determinado assunto e auxiliar jornalistas, pesquisadores e analistas políticos (Zong at al, 2021).

Diversos métodos podem ser utilizados para realizar a análise de sentimentos em um texto. Os mais comuns incluem os baseados em regras, os modelos de aprendizado de máquina e os modelos de aprendizado profundo (Islam et al., 2024).

Os métodos baseados em regras podem utilizar dicionários que contém uma lista de palavras associadas a sentimentos positivos, negativos ou neutros. O texto é analisado para encontrar essas palavras e determinar o sentimento geral com base na contagem e no peso dessas palavras. Exemplos: SentiWordNet e AFINN. Além do VADER (Valence Aware Dictionary and sEntiment Reasoner) que inclui palavras e combinações de palavras com pontuações para sentimentos, levando em consideração emojis, gírias e outras nuances. Um outro método baseado em regras que também utiliza dicionários, mas permite analisar a polaridade (positivo ou negativo) e a subjetividade (opinião ou fato) do texto, é a ferramenta TextBlob.

Os métodos baseados em aprendizado de máquina com classificação supervisionada são aqueles que treinam algoritmos com grandes conjuntos de dados rotulados (textos que já foram classificados como positivos, negativos ou neutros) para reconhecer padrões e fazer previsões sobre sentimentos em novos conjuntos de dados. Esses algoritmos incluem: Regressão Logística, Máquinas de Vetores de Suporte (SVM), Árvore de Decisão e Redes Neurais. Antes da classificação pode ser realizada a extração de recursos utilizando o método Bag of Words (BoW), Term Frequency-Imersive Document Frequency (TF-IDF) ou as Word Embeddings, que são representações vetoriais de palavras (como Word2Vec). O método Bag of Words (BoW) representa o texto como um conjunto de palavras, desconsiderando a ordem em que elas aparecem. Cada palavra é convertida em um vetor e os algoritmos de aprendizado de máquina utilizam essas representações para classificar o sentimento. O método TF-IDF melhora o modelo Bag of Words ponderando a frequência das palavras com base em sua importância no texto. Os métodos baseados em aprendizado de máquina com modelos não supervisionados são os que agrupam os textos com base em características semelhantes e analisa a distribuição de sentimentos dentro de cada cluster.

Os modelos baseados em aprendizado profundo são os que podem entender contextos mais complexos e sutilezas nas linguagens. As Redes Neurais Convulacionais (CNNs) são utilizadas para extrair características dos textos e identificar padrões complexos que indicam sentimentos. As Redes Neurais Recorrentes (RNNs), que incluem o Long Short-Term Memory (LSTM) e o Gated Recurrent Units (GRU), são eficazes para lidar com a dependência de longo prazo no texto, importante para capturar o contexto e o sentimento. Os modelos Transformers, como o Bidirectional Encoder Representations from Transformers (BERT) e o Generative Pre-

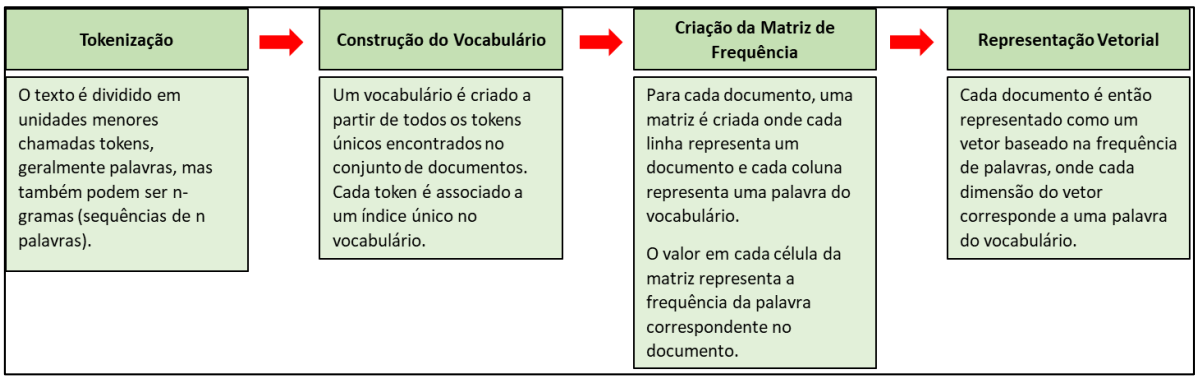
trained Transformer (GPT), têm se mostrado altamente eficazes em tarefas de análise de sentimentos, graças à sua capacidade de compreender contextos complexos e sutilezas da linguagem. No entanto, a análise de sentimentos enfrenta diversos desafios, como a ambiguidade linguística, onde palavras e frases podem ter diferentes significados dependendo do contexto. Além disso, ironia e sarcasmo podem confundir sistemas automatizados, e a diversidade linguística, como gírias e jargões, também dificulta a análise, comprometendo a precisão da interpretação e a determinação adequada da polaridade do sentimento (Islam et al., 2024).

2.3 Bag of Words (BoW)

Bag of Words (BoW) é uma técnica fundamental na análise de texto e Processamento de Linguagem Natural (PLN) que transforma texto em uma representação vetorial para facilitar a análise e a modelagem (Jurafsky e Martin, 2023). Embora seja uma abordagem relativamente simples, ela serve como base para muitas técnicas mais avançadas e é amplamente utilizada em tarefas de classificação de texto, recuperação de informações e análise de sentimentos.

Na Figura 3 é possível observar as etapas de como funciona o Bag of Words.

Figura 3 - Etapas do Bag of Words



Fonte: Própria autora.

Exemplo:

- **Dado o texto de entrada:**

  Document 1: "Eu adoro este lugar"

  Document 2: "Este lugar é maravilhoso"

  Document 3: "Eu não gosto deste lugar"

- **Tokenização**

  Tokens: ["Eu", "adoro", "este", "lugar", "é", "maravilhoso", "não", "gosto", "deste"]

- **Construção do Vocabulário**

  Vocabulário: ["Eu", "adoro", "este", "lugar", "é", "maravilhoso", "não", "gosto", "deste"]

- **Matriz de Frequência**

Documento	Eu	adoro	este	lugar	é	maravilhoso	não	gosto	deste
1	1	1	1	1	0	0	0	0	0
2	0	0	1	1	1	1	0	0	1
3	1	0	0	1	0	0	1	1	1

- **Representação Vetorial**

  Documento 1: [1, 1, 1, 1, 0, 0, 0, 0, 0]

  Documento 2: [0, 0, 1, 1, 1, 1, 0, 0, 1]

  Documento 3: [1, 0, 0, 1, 0, 0, 1, 1, 1]

As vantagens são:

- Fácil de entender e implementar.
- Rápido para criar e processar representações vetoriais de texto.
- Serve como base para técnicas mais complexas, como TF-IDF e modelos baseados em embeddings.

As limitações são:

- Ignora a ordem das palavras: Não leva em conta a ordem ou a estrutura gramatical das palavras, o que pode perder informações contextuais importantes.
- A matriz de frequências pode ser muito grande para textos longos ou grandes corpora, levando a problemas de dimensionalidade.
- Não captura relações semânticas entre palavras ou sinônimos.

2.4 Term Frequency-Inverse Document Frequency (TF-IDF)

O Term Frequency-Inverse Document Frequency (TF-IDF) é uma técnica utilizada para a representação e análise de texto, especialmente em tarefas de recuperação de informações e mineração de texto (Jurafsky e Martin, 2023). TF-IDF é uma métrica que avalia a importância de uma palavra em um documento dentro de um conjunto de documentos, ajustando pela frequência da palavra em todos os documentos. A ideia é que palavras frequentes em um documento específico e raras em outros documentos são mais importantes para caracterizar o conteúdo do documento.

TF-IDF é composto por duas partes principais:

- Term Frequency (TF)

  A frequência do termo (TF) mede a importância de uma palavra dentro de um documento. É calculada pela fórmula:

$$TF(t,d) = \frac{Número\ de\ vezes\ que\ o\ termo\ t\ aparece\ no\ documento\ d}{Número\ total\ de\ termos\ no\ documento\ d}$$

- Inverse Document Frequency (IDF)

  A frequência inversa de documentos (IDF) ajusta o TF, penalizando palavras que aparecem com muita frequência em muitos documentos do corpus. É calculada pela fórmula:

$$IDF(t,D) = \log\frac{Número\ total\ de\ documentos\ D}{Número\ de\ documentos\ que\ contêm\ o\ termo\ t}$$

O IDF ajuda a identificar termos que são mais exclusivos para certos documentos.

Se a palavra aparece em muitos documentos, IDF será baixo, indicando que a palavra não é muito distintiva.

Se a palavra aparece em poucos documentos, IDF será alto, indicando que a palavra é mais relevante para aqueles documentos.

- TF-IDF

A combinação de TF e IDF é dada por:

$$TF - IDF\ (t,d,D) = TF(t,d)X\ IDF(t,D)$$

Pontuação TF-IDF alta: indica que o termo é importante no documento, mas não é comum em todo o conjunto de documentos.

Pontuação TF-IDF baixa: Indica que o termo é muito comum em documentos (e, portanto, menos informativo) ou aparece com pouca frequência no documento específico.

Exemplo:

Considere um corpus com três documentos:

1. Doc 1: "O gato está em cima do tapete."
2. Doc 2: "O gato é um felino."
3. Doc 3: "Gatos são ótimos animais de estimação."

Vamos calcular o TF-IDF para o termo "gato" nesses documentos.

1. Cálculo da TF:
   - Doc 1: TF("gato", Doc 1) = 1/6
   - Doc 2: TF("gato", Doc 2) = 1/5
   - Doc 3: TF("gato", Doc 3) = 1/4
2. Cálculo da IDF:
   - "gato" aparece em todos os 3 documentos.
   - IDF("gato", Corpus) = log(3/3) = 0
3. Cálculo do TF-IDF:
   - Doc 1: TF-IDF("gato", Doc 1) = (1/6) * 0 = 0
   - Doc 2: TF-IDF("gato", Doc 2) = (1/5) * 0 = 0
   - Doc 3: TF-IDF("gato", Doc 3) = (1/4) * 0 = 0

Neste caso, "gato" tem uma pontuação TF-IDF de 0 em todos os documentos porque é muito comum em todo o conjunto de documentos. Isso demonstra que o TF-IDF é eficaz para destacar termos que são únicos e significativos em documentos específicos.

As vantagens são:

- A fórmula TF-IDF é relativamente simples de entender e implementar.
- A pontuação TF-IDF oferece uma maneira intuitiva de entender a importância de um termo em relação a um documento e ao conjunto de documentos.
- TF-IDF ajuda a identificar termos que são importantes para um documento específico, mas não comuns em todo o conjunto de documentos, melhorando a relevância dos resultados.

- Termos que aparecem frequentemente em muitos documentos (como "o", "a", "de") são penalizados pela IDF, ajudando a reduzir o impacto de palavras comuns e melhorar a relevância.
- TF-IDF é utilizado em várias tarefas de processamento de texto, como recuperação de informações, classificação de textos e extração de palavras-chave.

As limitações são:

- TF-IDF considera apenas a frequência de termos e não entende o significado ou o contexto dos termos, o que pode limitar a precisão em tarefas que requerem compreensão semântica.
- Em documentos muito longos, termos frequentes podem ter uma pontuação TF-IDF que não reflete bem sua importância, porque a TF será alta, mas a IDF pode ser baixa, resultando em pontuações menos discriminativas.
- TF-IDF não trata sinônimos ou variações de termos (por exemplo, "gato" e "felino" são tratados como termos distintos), o que pode limitar a capacidade de capturar a real importância semântica de um termo.
- A pontuação TF-IDF é fortemente dependente do conjunto de documentos. Mudanças nesse conjunto (adicionar ou remover documentos) podem alterar significativamente as pontuações TF-IDF, tornando o sistema menos robusto para novos dados.
- TF-IDF não lida bem com palavras que têm múltiplos significados (polissemia) ou com palavras que aparecem em contextos diferentes, o que pode levar a interpretações incorretas.

Alternativas e Melhorias

Para superar algumas das limitações do TF-IDF, técnicas mais avançadas podem ser usadas, como:

- Modelos de Word Embeddings: como o Word2Vec que capturam relações semânticas e contextuais entre palavras.
- Modelos Baseados em Transformers: como o BERT, o GPT e suas variantes, que entendem o contexto e a semântica das palavras em um texto.

- Essas abordagens oferecem uma compreensão mais rica e contextual dos textos, complementando ou até substituindo o TF-IDF em muitas aplicações avançadas.

2.5 Word2Vec

Word2Vec é uma técnica de modelagem de palavras desenvolvida por pesquisadores da Google que transforma palavras em vetores de alta dimensão, capturando relações semânticas e contextuais entre elas. É uma das abordagens mais influentes para o aprendizado de representações de palavras, e seu impacto se estende a muitas áreas do PLN (Jurafsky e Martin, 2023).

Os principais conceitos são os vetores de palavras e os modelos de treinamento. Word2Vec mapeia palavras para vetores numéricos em um espaço de alta dimensão. Esses vetores capturam semântica e contexto, permitindo que palavras com significados semelhantes estejam próximas no espaço vetorial.

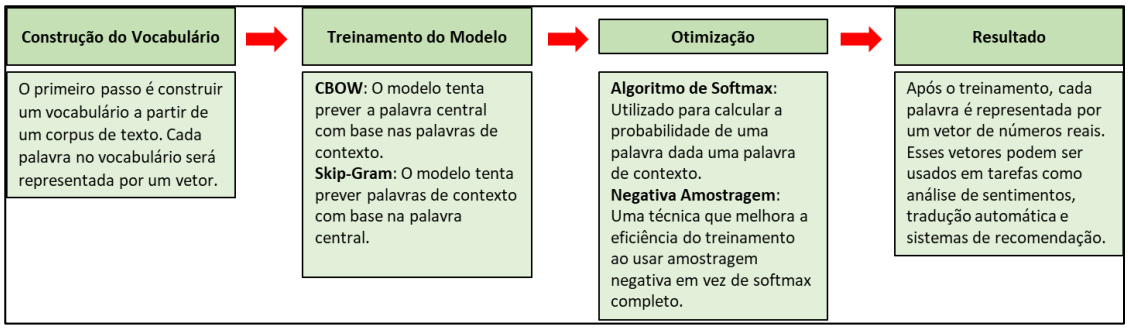
Os modelos de treinamento são:

- Continuous Bag of Words (CBOW): o modelo CBOW prevê uma palavra com base nas palavras ao seu redor. Por exemplo, dado o contexto "O gato está em cima do", o CBOW tentaria prever a palavra "tapete".
- Skip-Gram: o modelo Skip-Gram faz o inverso do CBOW. Ele usa uma palavra para prever o contexto em que a palavra aparece. Por exemplo, dado "tapete", o Skip-Gram tentaria prever palavras como "O", "gato", "está", "em", "cima", "do".

O objetivo do treinamento de Word2Vec é ajustar os vetores de palavras para que palavras que ocorrem em contextos semelhantes tenham vetores semelhantes. O modelo é treinado usando redes neurais de forma eficiente e escalável.

O funcionamento ocorre em quatro etapas, como é possível observar na Figura 4.

Figura 4 - Etapas do Word2Vec



Fonte: Própria autora.

As vantagens são:

- Word2Vec captura relacionamentos semânticos entre palavras, como similaridade e analogias. Por exemplo, "rei" - "homem" + "mulher" resulta em um vetor próximo a "rainha".
- Word2Vec é eficiente em termos de escalabilidade (computação e memória), especialmente com grandes corpora.
- Pode ser usado para melhorar várias tarefas de PLN, incluindo tradução automática, análise de sentimentos e recuperação de informações.
- Reduz a dimensionalidade das palavras em comparação com representações one-hot, tornando os cálculos mais eficientes.

As limitações são:

- Word2Vec não captura o contexto de uma palavra em diferentes frases, o que pode ser uma limitação em tarefas que dependem do contexto mais amplo.
- A qualidade das representações de palavras é fortemente influenciada pela qualidade e pelo tamanho do conjunto de treinamento. Textos ruidosos ou de tamanho reduzido podem levar à geração de vetores menos precisos.
- Os vetores são estáticos e não mudam com base no contexto da frase, o que pode ser uma limitação para alguns casos de uso onde o contexto dinâmico é importante.
- Word2Vec trata diferentes sentidos de uma palavra como uma única entidade, o que pode levar a vetores que não capturam bem a ambiguidade lexical.

2.6 Valence Aware Dictionary and sEntiment Reasoner (VADER)

VADER é uma ferramenta específica para análise de sentimentos, projetada para lidar com textos de redes sociais, utiliza dicionário de polaridade. Desenvolvido para ser sensível a nuances de sentimentos expressos em texto, além de ser eficaz em detectar sentimentos em textos que incluem emojis, gírias, abreviações e outras características informais (Hutto e Gilbert, 2014).

VADER utiliza um dicionário de palavras e frases associadas a sentimentos, cada uma com uma pontuação de valência (polaridade). As palavras são associadas a valores de sentimento que vão de -1 (muito negativo) a +1 (muito positivo). O modelo inclui uma lista de emojis e gírias com suas respectivas polaridades, melhorando a precisão na detecção de sentimentos em textos informais. Inclui regras para lidar com modificadores e intensificadores que afetam a polaridade das palavras (por exemplo, "muito bom" ou "pouco ruim"). Identifica e ajusta a polaridade das palavras baseadas em negações (por exemplo, "não gostei" é tratado como negativo).

A pontuação de sentimento possui três métricas principais:

1. Polaridade: valor contínuo que vai de -1 a +1, indicando o sentimento geral.
2. Subjetividade: indica o grau de subjetividade do texto (como opiniões versus fatos).
3. Classificação: categorização geral do sentimento como positivo, negativo ou neutro.

As vantagens são:

- Implementação simples e rápida.
- Sensível a emojis e gírias, ou seja, capaz de lidar bem com texto informal e comunicação moderna.
- Boa performance em textos curtos e grandes volumes de dados.

As limitações são:

- Pode não capturar contextos muito sutis ou ironia complexa.
- A precisão depende da qualidade e abrangência do dicionário de sentimentos.

2.7 TextBlob

TextBlob é uma biblioteca Python para Processamento de Linguagem Natural (PLN) que facilita a análise de texto, oferecendo funcionalidades para tarefas comuns como análise de sentimentos, tradução e correção gramatical. É uma ferramenta popular devido à sua simplicidade e facilidade de uso, ideal para iniciantes e projetos que não exigem a complexidade dos métodos mais avançados (Jurafsky e Martin, 2023).

Mede a orientação do sentimento do texto, variando de -1 (muito negativo) a +1 (muito positivo) e avalia o grau de subjetividade do texto, de 0 (fato) a 1 (opinião). Permite traduzir texto para diferentes idiomas usando a API do Google Translate. Identifica o idioma em que o texto está escrito. Sugere correções para erros gramaticais e ortográficos em um texto. Identifica e rotula as partes do discurso (substantivo, verbo, adjetivo, etc.) em um texto. Extrai e identifica frases nominais (substantivos e suas modificações) do texto.

As vantagens são:

- API simples e direta, ideal para iniciantes e protótipos rápidos.
- Oferece várias funcionalidades úteis de PLN em uma única biblioteca.
- Usa a API do Google Translate para tradução e detecção de idioma.

As limitações são:

- Pode não ser tão preciso quanto métodos mais avançados baseados em aprendizado de máquina ou modelos de linguagem mais sofisticados.
- A tradução e a detecção de idioma dependem da API do Google, o que pode implicar em limitações de uso ou custos.

2.8 Regressão Logística

A Regressão Logística é um modelo estatístico utilizado para prever a probabilidade de ocorrência de um evento binário, isto é, um evento que pode ter apenas duas possibilidades, como "sim" ou "não", "1" ou "0". É amplamente usada em problemas de classificação onde o objetivo é separar dados em categorias distintas. Apesar do nome, a regressão logística é um modelo de classificação, não de regressão.

Os principais conceitos são:

- Função Logística (Sigmoid)

A função logística é uma função sigmoide que transforma uma entrada contínua em uma probabilidade entre 0 e 1. A fórmula da função sigmoide é:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Onde z é a combinação linear das variáveis independentes.

- Modelo de Regressão Logística

  É a combinação linear. O modelo assume que a probabilidade p de uma observação pertencer a uma classe (por exemplo, "1") pode ser expressa como:

$$p = \sigma(w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n)$$

  onde $w_0, w_1, \ldots, w_n$ são os coeficientes do modelo e $x_1, x_2, \ldots, x_n$ são as variáveis independentes.

- Função de Custo

  A função de custo usada na regressão logística é a entropia cruzada (ou log loss), que mede a diferença entre as probabilidades previstas pelo modelo e as classes reais. A fórmula para a função de custo é:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y_i \log\left(h_\theta(x_i)\right) + (1 - y_i) \log\left(1 - h_\theta(x_i)\right)\right]$$

  Onde $h_\Theta(x_i)$ é a probabilidade prevista pelo modelo, e y_i é a classe real.

- Treinamento:

  Os coeficientes w são ajustados usando algoritmos de otimização, como o Gradiente Descendente, para minimizar a função de custo e encontrar os melhores parâmetros do modelo.

As ferramentas e bibliotecas que podem ser usados para aplicar regressão logística são:

- Scikit-Learn: é uma biblioteca de aprendizado de máquina em Python que fornece implementações eficientes de regressão logística.
- NLTK (Natural Language Toolkit) é uma biblioteca completa para processamento de linguagem natural, que pode ser usada em combinação com Scikit-Learn para pré-processamento e análise de texto. É usada principalmente para tarefas de pré-processamento, como tokenização e remoção de stop words, que podem ser combinadas com modelos de regressão logística implementados em Scikit-Learn.

- TensorFlow é uma biblioteca de aprendizado de máquina desenvolvida pelo Google, e Keras é uma API para TensorFlow que simplifica a construção e o treinamento de modelos. Embora TensorFlow e Keras sejam mais frequentemente usados para redes neurais, a implementação de regressão logística é possível e pode ser útil em contextos mais avançados.
- FastText é uma biblioteca desenvolvida pelo Facebook para a representação de palavras e modelos de classificação. Pode ser usada para tarefas de classificação de texto, incluindo a classificação de sentimentos, e é conhecida por sua eficiência e precisão.
- SpaCy é uma biblioteca de Processamento de Linguagem Natural avançada em Python que pode ser usada para pré-processamento de texto e integração com modelos de classificação. Embora SpaCy não ofereça uma implementação direta de regressão logística, pode ser combinado com bibliotecas como Scikit-Learn para treinar e avaliar modelos de classificação após o pré-processamento de texto.

Algumas considerações:

- A eficácia da regressão logística para a classificação de sentimentos depende muito da qualidade do pré-processamento dos dados, como a tokenização, remoção de stop words e lematização.
- Técnicas de vetorização como Bag of Words (BoW) ou TF-IDF são comuns antes de aplicar a regressão logística. Modelos mais avançados podem usar Word Embeddings (Word2Vec) ou modelos baseados em transformers (BERT).
- É importante usar métricas de avaliação apropriadas, como acurácia, precisão, revocação e F1-score, para medir o desempenho do modelo de classificação de sentimentos.

As vantagens são:

- O modelo fornece probabilidades diretamente, o que facilita a interpretação dos resultados. A interpretação dos coeficientes é também direta; cada coeficiente representa a mudança na log-odds (ou logaritmo das razões de chances) da variável dependente para uma unidade de mudança na variável independente.

- É um modelo relativamente simples e computacionalmente eficiente, adequado para muitas tarefas de classificação.
- Técnicas de regularização, como L1 (Lasso) e L2 (Ridge), podem ser aplicadas para evitar overfitting e melhorar a generalização do modelo.
- Oferece a vantagem de prever não apenas a classe, mas também a probabilidade associada a cada classe.

As limitações são:

- Assume uma relação linear entre as variáveis independentes e a variável dependente na escala logit, o que pode não capturar relações complexas em dados não lineares.
- Pode não ser tão eficaz quanto modelos mais complexos (como árvores de decisão ou redes neurais) em capturar padrões não lineares e interações complexas entre variáveis.
- Se as variáveis independentes são altamente correlacionadas, pode haver problemas de multicolinearidade que afetam a estabilidade dos coeficientes do modelo.
- Originalmente projetada para problemas de classificação binária, embora possa ser estendida para múltiplas classes usando técnicas como regressão logística multinomial.

2.9 Máquinas de Vetores de Suporte (SVM)

As Máquinas de Vetores de Suporte (SVM) são técnicas de aprendizado de máquina utilizadas tanto para classificação quanto para regressão. O objetivo principal do SVM é identificar o hiperplano ideal que separa os dados em diferentes classes no espaço de características. Essa abordagem é particularmente eficaz em cenários de alta dimensionalidade e quando há uma margem clara entre as classes (Hsu, 2003).

O SVM é uma técnica de aprendizado de máquina que pode ser usada para classificação binária e multiclasses, o que o torna adequado para classificar o sentimento expresso em um texto como positivo, negativo, neutro, ou outras categorias de sentimentos.

Os principais conceitos são:

- Hiperplano: é uma superfície que separa o espaço de características em duas classes. Em dados bidimensionais, o hiperplano é uma linha, enquanto em

espaços de maior dimensão, ele é uma superfície de dimensão n−1, onde n representa o número de características.

- Margem Máxima: o SVM busca o hiperplano que maximize a margem entre as classes. A margem é a distância entre o hiperplano e os pontos de dados mais próximos de cada classe, conhecidos como vetores de suporte.
- Vetores de Suporte: são os pontos de dados mais próximos ao hiperplano, que determinam sua posição e orientação. Esses vetores são essenciais para o treinamento do modelo SVM.
- Transformação de Espaço: quando os dados não são linearmente separáveis no espaço original, o SVM pode utilizar uma função kernel para mapear os dados em um espaço de alta dimensão, onde eles se tornam linearmente separáveis. Funções kernel comuns incluem:
    - Kernel Linear: Sem transformação, usado quando os dados são linearmente separáveis.
    - Kernel Polinomial: Introduz características polinomiais.
    - Kernel RBF (Radial Basis Function): Baseado na função gaussiana, é eficaz para capturar relações complexas entre características.
    - Kernel Sigmoide: Baseado na função sigmoide.
- Parâmetro C: Regula o equilíbrio entre maximizar a margem e minimizar o erro de classificação. Um valor elevado de C prioriza a minimização dos erros de treinamento, enquanto um valor mais baixo permite uma margem maior, aceitando possivelmente mais erros de classificação.

O funcionamento do SVM é realizado em dois passos: treinamento e classificação. No treinamento o objetivo é encontrar o hiperplano que maximiza a margem entre as classes, o que é feito resolvendo um problema de otimização. Na Função de Custo é incluído um termo para a margem e um termo para o erro de classificação, ajustado pelo parâmetro de regularização C. Após o treinamento, o SVM usa o hiperplano encontrado para classificar novos dados, determinando de qual lado do hiperplano o novo ponto de dado está.

Para aplicar SVM na análise de sentimentos é necessário primeiramente realizar o pré-processamento dos dados. Isso implica em usar um conjunto de dados onde os textos já foram classificados quanto ao sentimento, remover as tags HTML, remover os caracteres especiais e normalizar o texto, por exemplo, converter o texto para minúsculo, realizar a tokenização, o stop words e a lematização. Após essa etapa é possível extrair as características utilizando Bag

of Words, TF-IDF, N-grams e Word Embeddings. Nessa etapa, é realizado o treinamento do modelo dividindo os dados em treinamento e testes para avaliar o desempenho do modelo. É possível usar diferentes funções kernel para transformar os dados e encontrar o melhor hiperplano e ajustar o parâmetro de regularização. E, por fim, realizar a avaliação do modelo com métricas de performance como acurácia, precisão, revocação e F1-score.

As vantagens são:

- SVM é eficaz em espaços de alta dimensão e em casos onde o número de características é maior do que o número de amostras.
- A maximização da margem ajuda a melhorar a generalização do modelo e a reduzir o overfitting.
- A capacidade de usar diferentes funções kernel permite que o SVM lide com problemas não linearmente separáveis.
- Mesmo em situações de alta dimensionalidade, SVM pode ser robusto contra overfitting, especialmente quando o parâmetro de regularização é bem ajustado.

As limitações são:

- O treinamento de SVM pode ser computacionalmente intensivo, especialmente com grandes conjuntos de dados e complexos kernels. O custo é quadrático em relação ao número de amostras, o que pode ser um problema com dados muito grandes.
- A escolha e a configuração adequada do kernel e dos parâmetros de regularização são cruciais para o desempenho do modelo e podem exigir experimentação e validação cuidadosa.
- SVM não é tão interpretável quanto alguns outros modelos de aprendizado de máquina, como a regressão logística. Isso pode ser uma limitação quando a interpretabilidade é importante.
- SVM pode ter dificuldades com classes desbalanceadas, onde uma classe é significativamente menos representada que a outra. Técnicas de balanceamento, como reamostragem ou ajuste de pesos, podem ser necessárias.

2.10 Árvore de Decisão

Árvore de Decisão é um modelo de aprendizado de máquina usado para tarefas de classificação e de regressão. Utiliza uma estrutura em forma de árvore para tomar decisões e prever valores ou classes com base em características dos dados. É uma ferramenta intuitiva e podem lidar com dados tanto categórico quanto numéricos (Loh, 2011).

A estrutura de árvore e composta por um nó raiz, nós internos, folhas e ramos. O nó raiz é o nó inicial e representa toda a amostra de dados. Cada nó interno representa uma condição sobre uma característica dos dados. As folhas ou os nós terminais representam a previsão ou a classe final após a aplicação de todas as condições. E os ramos são as divisões que representam os resultados das condições nos nós internos.

A árvore é construída dividindo repetidamente os dados em subgrupos baseados em características, de forma que os subgrupos se tornam cada vez mais homogêneos em relação à variável alvo. A construção da árvore é feita através da divisão recursiva dos dados em nós filhos até que um critério de parada seja atingido (como a profundidade máxima da árvore ou a quantidade mínima de amostras por nó). A divisão é feita de maneira a maximizar a homogeneidade dos grupos resultantes.

Os critérios mais comuns da divisão são:

- Índice de Gini: mede a impureza dos dados de um nó. Quanto menor o índice, melhor a separação.
- Entropia: mede a quantidade de desordem ou impureza dos dados baseada na teoria da informação. A menor entropia indica melhor separação.
- Chi-Squared: testa a independência entre variáveis.

Para regressão, pode-se usar a variância dos dados nos ramos.

A poda de árvore é o processo de remover partes da árvore que não contribuem significativamente para a precisão do modelo. Isso é feito para evitar o overfitting e melhorar a generalização do modelo.

As vantagens são:

- as árvores de decisão são altamente interpretáveis e fáceis de visualizar, pois você pode seguir o caminho de decisões que leva a uma previsão.
- não requer normalização das características, pois a árvore de decisão lida com características categóricas e numéricas diretamente.
- pode capturar relações não lineares entre características e a variável alvo.
- capaz de modelar interações complexas entre características de maneira natural.

As limitações são:

- as árvores de decisão têm uma tendência a se ajustar muito bem aos dados de treinamento, o que pode levar ao overfitting e reduzir a capacidade de generalização.
- pequenas variações nos dados podem levar a grandes alterações na estrutura da árvore.
- o critério de divisão local pode levar a uma árvore subótima globalmente.
- árvores complexas podem ser difíceis de interpretar e podem exigir poda para simplificação.

2.11 Redes Neurais

Redes Neurais são um dos principais métodos de aprendizado de máquina e são amplamente utilizadas para uma variedade de tarefas, desde reconhecimento de imagem e Processamento de Linguagem Natural (PLN) até jogos e diagnósticos médicos. Elas são inspiradas no funcionamento do cérebro humano e consistem em camadas de unidades de processamento chamadas neurônios (Haykin, 2009).

Cada unidade em uma rede neural é chamada de neurônio, e cada neurônio aplica uma função de ativação a uma combinação ponderada de entradas. As redes neurais são formadas por camadas de neurônios, e existem três tipos principais de camadas:

- Camada de entrada: responsável por receber os dados brutos, ou seja, as características de entrada do modelo.
- Camadas ocultas: camadas intermediárias que processam as informações recebidas da camada de entrada. Cada neurônio em uma camada oculta está conectado a todos os neurônios da camada anterior e da próxima.
- Camada de saída: gera a saída do modelo, como uma classificação ou predição.

Cada conexão entre os neurônios possui um peso, que é ajustado durante o treinamento para melhorar a precisão do modelo. Bias é um valor adicional que ajuda o modelo a se ajustar mais precisamente aos dados.

Após o cálculo dos pesos e bias, a saída do neurônio é passada por uma função de ativação (como ReLU, Sigmoid, Tanh), que decide se o neurônio deve "disparar" ou não.

As redes neurais são treinadas usando algoritmos como backpropagation que ajusta os pesos da rede com base nos erros cometidos durante as previsões. O erro é calculado usando a função de perda que mede a diferença entre a precisão da rede e a realidade. As funções de perda mais comuns são o Erro Quadrático Médio (MSE) usados para problemas de regressão e a Entropia Cruzada usada para problemas de classificação. E, por fim, os pesos são ajustados utilizando algoritmo de otimização como, por exemplo, o gradiente descendente, cujo objetivo é minimizar a função de perda. Variantes do gradiente descendente, como SGD (Stochastic Gradient Descent), Adam e RMSprop, podem ser usados para melhorar a eficiência do treinamento.

Os tipos de Redes Neurais:

- Perceptron: é o modelo de rede neural mais simples, com apenas uma camada de neurônios. Usado para tarefas de classificação binária.
- Redes Neurais Feedforward: são redes onde as conexões entre os neurônios seguem apenas uma direção, da entrada para a saída. Incluem perceptrons multicamadas (MLPs).
- Redes Neurais Artificiais (ANNs): são as mais básicas e consistem em camadas totalmente conectadas.
- Redes Neurais Convolucionais (CNNs): são utilizadas principalmente em processamento de dados com estrutura de grade, como imagens, pois utilizam operações convulacionais para capturar características espaciais de imagens.
- Redes Neurais Recorrentes (RNNs): são utilizadas para processar dados sequenciais, como séries temporais e processamento de linguagem natural. Mantêm um estado interno que pode capturar dependências temporais.
- Redes Neurais Profundas: são redes com múltiplas camadas ocultas, capazes de aprender representações de alto nível dos dados. Incluem CNNs e RNNs.
- Redes Neurais de Memória de Longo Prazo (LSTM) e Gated Recurrent Units (GRUs): são versões avançadas das RNNs que lidam com o problema do desvanecimento do gradiente e são eficazes para capturar dependências de longo prazo em dados sequenciais.

Redes neurais são extremamente eficazes na análise de sentimentos devido à sua capacidade de capturar relações complexas e contextos sutis em dados textuais. Elas são

amplamente utilizadas para classificar textos em categorias como positivo, negativo ou neutro, identificando nuances emocionais em diferentes contextos.

As abordagens básicas para realizar análise de sentimentos com redes neurais são:

- Rede Neural Feedforward Simples:
    - Para uma abordagem inicial pode usar uma rede neural feedforward (perceptron multicamada) onde o texto é convertido em vetores de características, como contagem de palavras ou TF-IDF.
    - Exemplo: Usar um vetor de características de um documento como entrada e uma camada oculta com uma função de ativação ReLU. A camada de saída usa uma função de ativação sigmoid para classificação binária (positivo/negativo).
- Word Embeddings:
    - Word2Vec é uma técnica para representar palavras como vetores densos. Esses embeddings podem ser usados como entrada para redes neurais para capturar o significado semântico das palavras.
    - Exemplo: Utilizar word embeddings para representar o texto e alimentá-los a uma rede neural feedforward.

Word Embeddings oferecem representações densas que capturam o significado semântico das palavras.

As abordagens avançadas para realizar análise de sentimentos com redes neurais são:

- Redes Neurais Convolucionais (CNNs):
    - CNNs são frequentemente usadas para processamento de texto, pois podem capturar padrões locais, como n-gramas, que são úteis para entender o contexto emocional de uma sentença.
    - Exemplo: Convolucionar as representações de palavras com filtros para extrair características relevantes e depois passar por camadas totalmente conectadas para classificação.
- Redes Neurais Recorrentes (RNNs):
    - RNNs e suas variantes, como LSTM (Long Short-Term Memory) e GRU (Gated Recurrent Unit), são projetadas para lidar com dados sequenciais e podem capturar dependências temporais no texto.
    - Exemplo: Usar LSTM para processar sequências de palavras e captar a dependência de longo prazo no texto para inferir sentimentos.

- Transformers:
    - Transformers, como BERT (Bidirectional Encoder Representations from Transformers) e GPT (Generative Pre-trained Transformer), são modelos de estado da arte para muitas tarefas de Processamento de Linguagem Natural (PLN). Eles podem capturar o contexto das palavras em ambas as direções (esquerda e direita).
    - Exemplo: Fine-tuning de um modelo BERT pré-treinado para a tarefa de análise de sentimentos.

Modelos como LSTM e Transformers são capazes de capturar dependências de longo prazo e contextos complexos.

As vantagens são:

- Redes neurais podem modelar relações complexas e não lineares.
- Capazes de aprender representações de alto nível dos dados.

As limitações são:

- Requer grandes quantidades de dados para treinamento eficaz.
- Pode ser intensivo em termos de tempo de processamento e memória.
- Redes neurais profundas podem ser difíceis de interpretar e entender.

2.12 Redes Neurais Feedforward

As Redes Neurais Feedforward são um tipo básico de rede neural amplamente utilizado em diversas aplicações de aprendizado de máquina. Elas são chamadas de "feedforward" porque a informação flui em apenas uma direção: da camada de entrada para a camada de saída, passando pelas camadas ocultas, sem ciclos ou retroalimentação (Haykin, 2009).

A camada de entrada recebe os dados brutos, representando as características de um conjunto de dados, com cada neurônio correspondendo a uma característica de entrada. As camadas ocultas, ou intermediárias, são responsáveis pelo processamento, onde cada neurônio aplica uma função de ativação a uma combinação ponderada das saídas da camada anterior. A camada de saída, por sua vez, gera a resposta final do modelo. Em problemas de classificação, funções de ativação como softmax ou sigmoid podem ser utilizadas.

Os neurônios são responsáveis por receber entradas, multiplicar essas entradas por pesos, adicionar um viés e aplicar uma função de ativação para gerar a saída.

As funções de ativação podem ser:

- ReLU (Rectified Linear Unit): utilizada para introduzir não linearidades.

$$ReLU(x) = max(0, x)$$

- Sigmoide: utilizada para problemas de classificação binária.

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

- Tanh: utilizada para normalizar a saída entre -1 e 1.

$$tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

Na progragação direta (Forward Propagation) os dados de entrada são passados pelas camadas da rede. Em cada camada, os dados são processados pelos neurônios que aplicam peses, viés e funções de ativação.

As redes Feedforward são geralmente treinadas usando o algoritmo backpropagation que ajusta os pesos da rede com base no erro da previsão. O erro é calculado pela função de perda e a atualização dos pesos é feita usando o Gradiente Descendente ou suas variantes. A função de perda usada para problemas de regressão é a Erro Quadrático Médio (MSE) e a utilizada para problemas de classificação é a Entropia Cruzada.

As vantagens são:

- Estrutura relativamente simples e fácil de implementar e entender.
- Pode ser aplicada a uma ampla gama de problemas de classificação e regressão.
- Serve como base para redes mais complexas, como redes neurais profundas (RNNs).

As limitações são:

- Não é adequada para dados sequenciais onde a ordem das entradas é importante.
- Pode necessitar de grandes quantidades de dados para evitar overfitting e treinar eficazmente.
- Modelos simples podem não capturar todas as complexidades dos dados, especialmente para tarefas mais desafiadoras.

2.13 Redes Neurais Profundas

Redes Neurais Profundas referem-se a uma subárea do aprendizado de máquina que utiliza redes neurais com múltiplas camadas para modelar e aprender representações de alto nível dos dados. Estas redes são chamadas de "profundas" devido ao grande número de camadas ocultas que possuem, o que permite que elas aprendam características complexas e hierárquicas dos dados (Schmidhuber, 2015**)**.

A camada de entrada recebe os dados brutos. Por exemplo, em uma rede para reconhecimento de imagem, a camada de entrada pode receber pixels da imagem. As camadas ocultas consistem em várias camadas de neurônios que aplicam funções de ativação a combinações ponderadas das entradas. Cada camada extrai características de nível superior dos dados. E a camada de saída gera o resultado final da rede, como uma classificação ou uma previsão.

As funções de ativação podem ser:

- ReLU (Rectified Linear Unit): utilizada para introduzir não linearidades.

$$ReLU(x) = max(0, x)$$

- Sigmoide: utilizada para problemas de classificação binária.

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

- Tanh: utilizada para normalizar a saída entre -1 e 1.

$$tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

As redes neurais profundas são geralmente utilizam o método backpropagation para calcular os gradientes da função de perda em relação aos pesos e atualizar os pesos da rede. O erro é calculado pela função de perda que mede a diferença entre a previsão da rede e o valor real. A função de perda usada para problemas de regressão é a Erro Quadrático Médio (MSE) e a utilizada para problemas de classificação é a Entropia Cruzada.

Algoritmos de otimização, como o Adam e o RMSprop, são usados para ajustar os pesos da rede para minimizar a função de perda. O Batch Normalization é utilizado para normalizar as saídas das camadas para melhorar a estabilidade da rede e acelerar o treinamento. O Dropout é usado para desativar aleatoriamente uma fração dos neurônios durante o treinamento para prevenir overfitting.

Tipos de Redes Neurais Profundas:

- Redes Neurais Feedforward (DNN):
    - Redes com múltiplas camadas totalmente conectadas. São usadas para uma variedade de tarefas, desde classificação até regressão.

- Redes Neurais Convolucionais (CNNs):
    - Utilizadas principalmente para tarefas de visão computacional, como reconhecimento de imagem e detecção de objetos. Elas aplicam filtros convolucionais para capturar características espaciais.
- Redes Neurais Recorrentes (RNNs):
    - Projetadas para dados sequenciais e temporais, como texto e séries temporais. Incluem variantes como LSTM e GRU para capturar dependências de longo prazo.
- Transformers:
    - Arquitetura recente que usa mecanismos de atenção para capturar dependências contextuais em dados sequenciais. Amplamente utilizada em tarefas de processamento de linguagem natural, como tradução automática e modelagem de linguagem.

As vantagens são:

- Podem aprender e modelar características complexas dos dados devido à profundidade da rede e às múltiplas camadas.
- Aplicáveis a uma ampla gama de tarefas e tipos de dados, como imagens, texto e áudio.
- Geralmente oferecem melhor desempenho em tarefas complexas em comparação com modelos de aprendizado de máquina mais simples.

As limitações são:

- Treinamento de redes profundas pode exigir hardware potente e muitos recursos computacionais.
- Requer grandes quantidades de dados rotulados para treinar efetivamente e evitar overfitting.
- Redes profundas podem ser difíceis de interpretar e entender. A complexidade das arquiteturas e a falta de transparência podem ser desafiadoras para a análise e a depuração.

2.14 Redes Neurais Convulacionais (CNNs)

As Redes Neurais Convolucionais (CNNs) são um tipo específico de rede neural profunda desenvolvida para processar dados estruturados em grade, como imagens e vídeos. Elas são amplamente empregadas em tarefas de visão computacional, incluindo reconhecimento de imagens, detecção de objetos e segmentação. A principal característica das CNNs é sua capacidade de extrair automaticamente características (features) das entradas, como bordas, texturas e padrões, por meio de operações de convolução (Gu et al., 2015).

As camadas convulacionais aplicam filtros (kernels) sobre a entrada (por exemplo, uma imagem) para extrair características locais. Esses filtros são pequenos, como 3X3 ou 5X5, e se movem pela entrada (deslizando ou convoluindo) para gerar mapas de características (feature maps).

A operação de convolução envolve a multiplicação dos valores dos pixels da imagem pelos valores do filtro e o somatório desses produtos. O resultado é um mapa de características que destaca certos padrões, como bordas ou texturas.

Por exemplo, em uma imagem em tons de cinza de 28x28 pixels pode ser aplicado um filtro 3X3, e o resultado é um mapa de características menor que destaca padrões específicos, como bordas ou texturas.

As camadas de pooling, ou subamostragem, diminuem a dimensionalidade dos mapas de características, mantendo as informações mais relevantes. Isso contribui para reduzir a complexidade computacional e torna o modelo mais robusto a variações espaciais. A redução pode ser feita utilizando o Max Pooling selecionando o valor máximo em uma janela de pooling, geralamente 2X2. Ou pode utilizada o Average Pooling que calcula a média dos valores na janela de pooling.

Após cada operação de convolução, uma função de ativação (como a ReLU) é aplicada para introduzir não-linearidade no modelo. Isso possibilita que a rede aprenda representações mais complexas.

As camadas de normalização, como Batch Normalization, são frequentemente usadas para acelerar o treinamento e melhorar a estabilidade da rede.

As camadas de regularização, como Dropout, são usadas para regularizar a rede ao desativar aleatoriamente uma fração dos neurônios durante o treinamento para prevenir o overffiting.

No final da rede, após várias camadas convolucionais e de pooling, os mapas de características extraídos são "achatados" em um vetor e passados para camadas totalmente conectadas. Estas camadas realizam a classificação ou regressão com base nas características extraídas.

A camada de saída pode usar uma função de ativação como Softmax (para classificação) ou uma função linear (para regressão).

As vantagens são:

- são muito eficazes em identificar padrões espaciais e texturas em imagens.
- o pooling reduz as dimensões dos dados, mantendo características importantes e reduzindo a complexidade.
- Camadas convolucionais extraem características de baixo nível (como bordas) nas primeiras camadas e características de alto nível (como formas ou objetos) nas camadas mais profundas.

As limitações são:

- Treinamento de CNNs pode ser intensivo em termos de recursos computacionais e memória, especialmente para redes profundas e grandes conjuntos de dados.
- Requer grandes quantidades de dados rotulados para alcançar um bom desempenho e evitar overfitting.
- A arquitetura e o ajuste de hiperparâmetros podem ser complexos e requerer experimentação.

2.15 Redes Neurais Recorrentes (RNNs)

As Redes Neurais Recorrentes (RNNs) são um tipo de rede neural projetadas para lidar com dados sequenciais ou temporais. Diferente das redes neurais feedforward, as RNNs têm conexões que permitem que informações sejam passadas de uma etapa para a outra ao longo da sequência, o que permite que elas mantenham um "estado interno" e capturem dependências temporais (Mienye et al, 2024).

As RNNs têm conexões recorrentes, o que significa que cada unidade na RNN tem uma conexão para si mesma. Dessa forma, a saída de um neurônio em um momento específico é usada como entrada para o mesmo neurônio no próximo passo temporal, o que permite que a rede mantenha informações de estados anteriores. A saída é combinada com a entrada atual para calcular o novo estado.

A camada de entrada recebe os dados sequenciais, como uma sequência de palavras ou uma série temporal. A camada oculta recorrente calcula a saída e o novo estado interno com

base na entrada atual e no estado anterior. E a camada de saída gera a saída final da rede, que pode ser uma previsão ou uma classificação.

As RNNs usam funções de ativação, como tanh ou ReLU, para processar os estados ocultos. Essas funções introduzem não-linearidade na rede, permitindo que ela capture padrões complexos.

Os tipos de RNNs são:

- RNNs Simples: definida como estrutura básica de uma RNN com unidades recorrentes. Pode ter dificuldade em aprender dependências de longo prazo devido ao problema de desvanecimento ou explosão do gradiente.
- Long Short-Term Memory (LSTM): tipo avançado das RNNs que inclui células de memória e portas (input, forget e output) para controlar o fluxo de informações, permitindo que a rede mantenha ou descarte informações conforme necessário. Resolve o problema de desvanecimento de gradiente e pode capturar dependências de longo prazo mais efetivamente. Cada célula LSTM tem três portas que regulam a adição e remoção de informações da célula de memória.
- Gated Recurrent Unit (GRU): tipo de variante das LSTM com uma estrutura mais simples. Usa portas de atualização e portas de reinicialização para controlar o fluxo de informações. Possui menos parâmetros do que LSTM, o que pode levar a um treinamento mais eficiente.
- Bidirectional RNNs é uma extensão das RNNs que processa a sequência de entrada em ambas as direções (para frente e para trás) para capturar dependências contextuais em ambas as direções.
- Attention Mechanisms adiciona um mecanismo de atenção às RNNs para permitir que a rede foque em partes diferentes da entrada ao gerar a saída. Muito usado em tradução automática e processamento de linguagem natural.

As vantagens são:

- Capaz de aprender e manter informações sobre estados anteriores em uma sequência, o que é útil para dados sequenciais.
- Pode ser aplicada a uma variedade de tarefas envolvendo dados sequenciais.

As limitações são:

- Dificuldade em aprender dependências de longo prazo devido ao desvanecimento ou explosão do gradiente durante o treinamento.
- O processamento sequencial dos dados pode limitar a eficiência de treinamento e inferência.
- Pode ser mais lento para treinar comparado a outras arquiteturas, especialmente em longas sequências.

Quando a rede é muito profunda ou a sequência é extensa, os gradientes podem se tornar excessivamente pequenos (desvanecimento) ou grandes demais (explosão), dificultando o processo de treinamento. Para mitigar esses problemas, foram desenvolvidas variantes como LSTM e GRU. Embora as RNNs sejam teoricamente capazes de capturar dependências de longo prazo, na prática, isso pode ser desafiador devido aos problemas relacionados aos gradientes. As LSTMs e GRUs são mais eficazes nessa tarefa, mas ainda enfrentam dificuldades ao lidar com dependências extremamente longas.

As RNNs, especialmente em suas variantes LSTM e GRU, permanecem como ferramentas eficazes para lidar com dados sequenciais, embora tenham sido parcialmente substituídas por arquiteturas mais avançadas, como os Transformers. Ainda assim, continuam a ser amplamente utilizadas em diversas aplicações que envolvem a análise de sequências e séries temporais.

2.16 Long Short-Term Memory (LSTM)

A Long Short-Term Memory (LSTM) é uma variante das Redes Neurais Recorrentes (RNNs) criada para resolver o problema de aprendizado de dependências de longo prazo em sequências de dados (Zhang et al., 2018). Proposta por Hochreiter e Schmidhuber em 1997, a arquitetura LSTM foi projetada para superar as limitações das RNNs tradicionais, como o desvanecimento e explosão dos gradientes, permitindo à rede aprender e reter informações por períodos prolongados.

As redes neurais recorrentes tradicionais têm dificuldade em capturar dependências de longo prazo devido ao problema do desvanecimento e explosão do gradiente. Isso ocorre durante o treinamento das RNNs, onde o gradiente (o valor que informa como os pesos da rede devem ser ajustados) pode se tornar muito pequeno (desvanecimento) ou muito grande (explosão), dificultando a aprendizagem de padrões em sequências mais longas.

As LSTMs superam esses problemas usando um mecanismo de memória mais sofisticado, que controla o fluxo de informações ao longo do tempo. A estrutura básica de uma célula LSTM consiste em três componentes principais: célula de estado (cell state), portas (gates) e saída oculta (hidden state) (Sundermeyer et al, 2012).

Os LSTMs introduzem uma estrutura de célula de memória e portas para controlar o fluxo de informações, o que permite a retenção de informações relevantes por mais tempo.

A célula de estado é responsável por transportar informações ao longo de toda a sequência, funcionando como uma memória de longo prazo. Ela permite que informações relevantes sejam preservadas e transportadas de um passo temporal para o próximo, enquanto outras informações irrelevantes podem ser descartadas.

As LSTMs utilizam três portas principais para controlar o fluxo de informações: porta de esquecimento, porta de entrada e porta de saída.

A porta de entrada decide quais novas informações serão armazenadas na célula de estado. Também utiliza uma função sigmoide para decidir quais valores serão atualizados e uma função tanh para regular a escala das novas informações.

A porta de esquecimento determina quais informações devem ser descartadas da célula de estado. Essa porta usa uma função sigmoide para gerar valores entre 0 e 1, onde 0 significa "esquecer completamente" e 1 significa "manter completamente".

A porta de saída controla a saída da célula LSTM. A saída é uma versão filtrada da célula de estado, controlada pela porta de saída, e é passada para o próximo passo temporal e para as camadas superiores da rede.

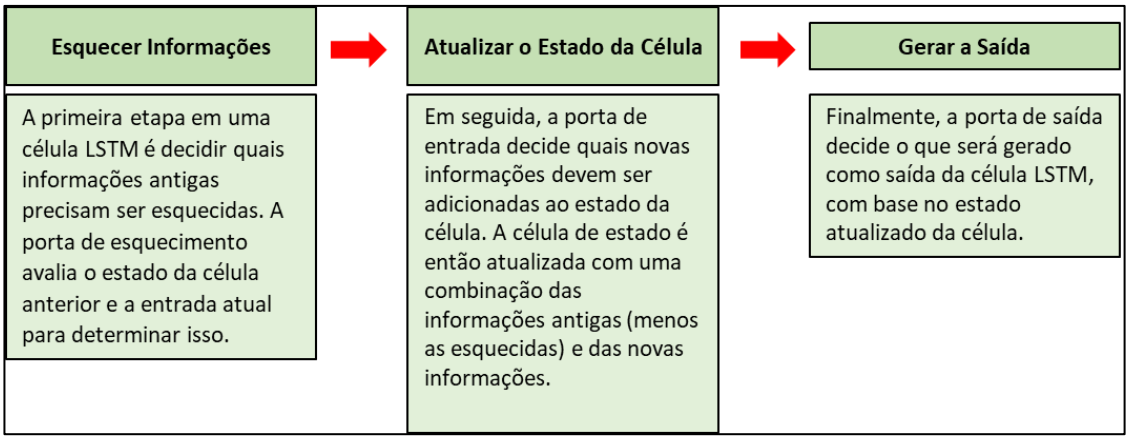
A saída oculta é uma versão filtrada do estado da célula, regulada pela porta de saída, e é utilizada para fazer previsões ou como entrada para a próxima célula LSTM no tempo.

A célula de memória é atualizada combinando as informações da porta de entrada e a célula de memória anterior. A saída da rede é calculada combinando o estado da célula de memória e a porta de saída.

O funcionamento de uma célula LSTM pode ser observado na

Figura 5.

Figura 5 - Funcionamento de uma célula LSTM



Fonte: Própria autora.

As vantagens são:

- As portas e a célula de memória permitem que a rede aprenda e retenha informações ao longo de longas sequências.
- Muito eficaz para tarefas que envolvem dados sequenciais, como texto e séries temporais.
- Reduz o problema de desvanecimento e explosão do gradiente encontrado em RNNs tradicionais.

As limitações são:

- Modelos LSTM podem ser mais lentos para treinar e menos eficientes em termos de recursos computacionais devido ao número de parâmetros e operações de porta.
- Embora eficazes para dependências temporais, LSTMs podem ser menos eficientes para modelar relações complexas em dados não temporais.
- Pode ser propenso ao sobre ajuste se não houver dados suficientes ou se a regularização não for bem aplicada.

2.17 Gated Recurrent Units (GRU)

A Gated Recurrent Units (GRU) é um tipo de Redes Neurais Recorrentes (RNNs) desenvolvida para simplificar e melhorar a eficiência do treinamento das RNNs. Proposta por Cho et al. em 2014, a arquitetura GRU é similar aos Long Short-Term Memory (LSTM), mas com uma estrutura mais simples e menos parâmetros. As GRUs são projetadas para lidar com

o problema de desvanecimento e explosão do gradiente e aprender dependências de longo prazo em sequências de dados (Nosouhian et al, 2021).

As GRUs, como as LSTMs, utilizam portas para controlar o fluxo de informações através do tempo, mas diferem em sua arquitetura ao simplificar o design das células.

Ao contrário das LSTMs, que têm um estado de célula separado e um estado oculto, as GRUs utilizam apenas o estado oculto para transportar informações ao longo do tempo.

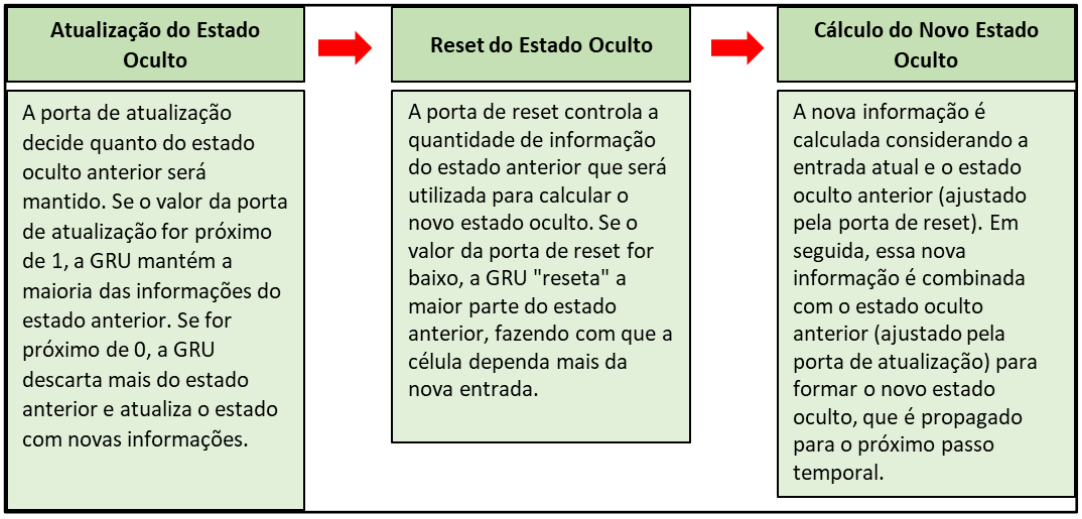
As GRUs utilizam duas portas principais: a porta de atualização e a porta de reset.

A porta de atualização controla a quantidade de informação do estado anterior que deve ser transportada para o próximo estado. Funciona de maneira semelhante à combinação das portas de esquecimento e de entrada nas LSTMs. Quando o valor da porta de atualização é alto, a célula mantém mais das informações anteriores; quando é baixo, atualiza mais com novas informações.

A porta de reset controla a quantidade de informação do estado anterior que deve ser "esquecida" ou ignorada ao calcular o novo estado oculto. Quando ativada, a porta de reset faz com que a rede ignore parte das informações anteriores, sendo útil para capturar dependências de curto prazo.

O funcionamento de uma célula GRU pode ser observado na Figura 6.

Figura 6 - Funcionamento de uma célula GRU



Fonte: Própria autora.

As vantagens são:

- As GRUs têm uma arquitetura mais simples do que as LSTMs, utilizando menos portas e não diferenciando entre o estado de célula e o estado oculto. Isso resulta

em menos parâmetros e, potencialmente, em um treinamento mais rápido e menor uso de memória.

- Embora sejam mais simples, as GRUs muitas vezes alcançam desempenho similar às LSTMs em muitas tarefas, especialmente quando se trata de capturar dependências de longo prazo em sequências de dados.
- A simplicidade das GRUs pode ajudar a reduzir o risco de overfitting, especialmente em conjuntos de dados menores ou em problemas menos complexos, onde uma arquitetura mais complexa, como as LSTMs, pode ser excessiva.
- Devido à sua simplicidade, as GRUs podem convergir mais rapidamente durante o treinamento, tornando-as uma escolha atraente para problemas onde o tempo de treinamento é uma preocupação.

As limitações são:

- Embora sejam mais simples que as LSTMs, as GRUs ainda são mais complexas do que RNNs simples. Em tarefas onde a dependência de longo prazo não é crítica, uma RNN simples pode ser uma solução mais eficiente.
- Em algumas situações, as GRUs podem não capturar dependências de longo prazo tão bem quanto as LSTMs, especialmente em sequências extremamente longas. Neste caso, LSTMs ou modelos baseados em atenção, como os Transformers, podem ser mais eficazes.
- Em alguns cenários de grande escala ou de ponta, onde os recursos computacionais são abundantes, as LSTMs ou outras arquiteturas mais avançadas podem ser preferidas devido à sua capacidade de capturar mais nuances e complexidade nos dados.

2.18 Transformers

Os Transformers são uma arquitetura de redes neurais desenvolvida para lidar com dados sequenciais, especialmente eficazes em tarefas de Processamento de Linguagem Natural (PLN). Introduzida por Vaswani et al. em 2017 no artigo "Attention is All You Need", a arquitetura Transformer revolucionou o campo da PLN devido à sua capacidade de capturar

relações contextuais de longo alcance sem a necessidade de processamento sequencial, como nas Redes Neurais Recorrentes (RNNs).

A arquitetura do Transformer é baseada em um mecanismo de atenção auto-regressivo, que permite que o modelo capture dependências entre diferentes partes de uma sequência sem precisar processá-las em ordem. Isso torna os Transformers extremamente eficientes e poderosos.

O componente central dos Transformers é o mecanismo de atenção, especificamente a atenção multi-cabeças (Multi-Head Attention). A atenção permite que o modelo se concentre em diferentes partes da sequência ao mesmo tempo, ponderando a importância de cada parte da sequência em relação às outras.

O cálculo da atenção envolve três vetores:

- Query (Q): a consulta que estamos tentando correlacionar com outros elementos da sequência.
- Key (K): chave que permite identificar a relevância de outros elementos em relação à consulta.
- Value (V): valor que é passado adiante como resultado da atenção.

O produto entre Q e K determina a "pontuação" da atenção, que é então usada para ponderar os valores V.

O Transformer original é dividido em duas partes: codificador e decodificador. O codificador processa a sequência de entrada (por exemplo, uma frase) e gera uma representação codificada. Consiste em várias camadas de atenção e feedforward. E o decodificador gera a sequência de saída (por exemplo, uma tradução de frase). Ele recebe as informações do codificador e usa camadas de atenção adicionais para prever a próxima palavra ou token na sequência de saída.

A normalização por camada é usada após cada operação de atenção e feedforward para estabilizar o treinamento e melhorar a eficiência. Entre as camadas de atenção, existem camadas feedforward totalmente conectadas que processam as representações intermediárias dos dados.

Como o Transformer não processa a sequência de forma ordenada, o positional encoding é adicionado às entradas para fornecer ao modelo informações sobre a posição relativa dos tokens na sequência. Isso é necessário para que o modelo entenda a ordem dos dados.

Self-Attention é uma versão especial de atenção usada nos Transformers, onde cada elemento da sequência de entrada se relaciona consigo mesmo e com todos os outros elementos da sequência. Isso permite que o modelo capture dependências em diferentes escalas, sem depender da ordem sequencial.

As vantagens são:

- Diferente das RNNs, que processam sequências de forma serial (uma etapa de cada vez), os Transformers permitem processamento paralelo, o que os torna muito mais rápidos para treinamento em grandes conjuntos de dados.
- Graças ao mecanismo de atenção, os Transformers podem capturar relações de longo alcance em sequências, mesmo quando os elementos relevantes estão distantes na sequência. Isso é especialmente útil em tarefas como tradução automática e resumo de texto.
- Os Transformers são altamente flexíveis e podem ser aplicados em diversas tarefas além do processamento de linguagem, como visão computacional e geração de música.

As limitações são:

- Apesar de sua eficiência no processamento paralelo, os Transformers podem ser computacionalmente caros, especialmente para tarefas que envolvem grandes sequências de dados. O treinamento de modelos grandes, como GPT-3, requer uma quantidade substancial de recursos computacionais.
- Os Transformers tendem a exigir grandes quantidades de dados para treinamento eficaz. Modelos como BERT e GPT são treinados em grandes quantidades de texto, o que pode não ser viável para todas as aplicações.
- A atenção quadrática em relação ao número de tokens faz com que o uso de memória cresça rapidamente em sequências longas.

O BERT (Bidirectional Encoder Representations from Transformers) é uma das variantes de Transformers amplamente utilizado em tarefas de compreensão de linguagem, como resposta a perguntas e análise de sentimentos. Realiza treinamento bidirecional para melhor compreensão contextual.

2.19 Bidirectional Encoder Representations from Transformers (BERT)

O Bidirectional Encoder Representations from Transformers (BERT) é um modelo de linguagem baseado na arquitetura Transformer desenvolvido pelo Google e introduzido no

artigo *"BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding"* por Devlin et al. em 2019. O BERT foi projetado para melhorar o entendimento do contexto em Processamento de Linguagem Natural (PLN) por meio de uma abordagem bidirecional, contrastando com os modelos unidimensionais de contexto que eram predominantes na época.

O BERT é baseado na arquitetura Transformer, mais especificamente a parte do codificador do Transformer. A arquitetura Transformer é baseada em mecanismos de atenção, e o codificador é responsável por processar as entradas e gerar representações contextuais.

Diferentemente de modelos anteriores, como os baseados em LSTM ou GRU, que processam o texto em uma única direção (da esquerda para a direita ou vice-versa), o BERT processa o texto em ambas as direções. Isso significa que o modelo considera o contexto das palavras antes e depois de uma palavra-alvo, proporcionando uma compreensão mais rica do significado.

Durante o treinamento, o BERT usa uma técnica chamada *Masked Language Model*. Isso envolve ocultar uma porcentagem aleatória das palavras de uma sequência e treinar o modelo para prever essas palavras ocultas com base no contexto das palavras restantes. Isso permite que o modelo aprenda representações contextuais profundas.

O BERT também é treinado para prever se uma sentença segue logicamente uma outra. Esse objetivo ajuda o modelo a entender relações entre pares de sentenças, o que é útil em tarefas como resposta a perguntas e compreensão de leitura.

As vantagens são:

- A bidirecionalidade permite uma compreensão mais profunda do contexto, levando a melhores representações das palavras.
- O treinamento em grandes corpora de texto permite que BERT adquira um conhecimento abrangente da linguagem.
- Pode ser ajustado para uma ampla gama de tarefas de PLN, incluindo classificação de texto, resposta a perguntas e muito mais.

As limitações são:

- Os modelos BERT são grandes e requerem significativos recursos computacionais para treinamento e inferência.
- A complexidade do modelo pode levar a desafios em termos de tempo de treinamento e necessidade de hardware especializado, como GPUs.

- Modelos baseados em Transformer, incluindo BERT, são frequentemente considerados como "caixas pretas", tornando difícil a interpretação e a explicação de suas decisões.

2.20 Interpretação dos resultados dos modelos

A interpretação dos resultados dos modelos pode ser realizada utilizando os resultados de F1-score, acurácia, revocação e precisão. A seguir é apresentado o que mede cada métrica e em que contextos elas são úteis e como usá-las juntas (Jiang e Zhang, 2019).

2.20.1 Acurária

A acurácia mede a proporção de previsões corretas em relação ao total de previsões. É uma boa métrica quando as classes estão bem balanceadas no conjunto dos dados. Caso o conjunto de dados não esteja balanceado, esse resultado pode ser enganoso. Por exemplo, se 90% dos dados são negativos e o modelo sempre prevê a classe negativa terá 90% de acuraria, porém não garante que o modelo identificou corretamente as classes minoritárias.

2.20.2 Precisão

A precisão mede a quantidade dos dados do conjunto de dados foram classificados como positivos são realmente positivos. Ou seja, mede a exatidão das previsões positivas do modelo. A precisão é importante quando o custo de falsos positivos é alto. Por exemplo, no diagnóstico de uma doença, se prever que uma pessoa está doente quando não está e gerar tratamentos desnecessários. A precisão não informa a quantidade de dados do conjunto de dados da classe positiva o modelo não conseguiu identificar (falsos negativos).

Se a precisão é de 80%, isso significa que 80% dos dados do conjunto de dados foram classificados como positivos realmente são positivos, enquanto 20% foram classificados incorretamente.

2.20.3 Revocação

A revocação mede quanto dados do conjunto de dados realmente pertencem à classe positiva foram corretamente classificados pelo modelo. O revocação é importante quando o

custo de falsos negativos é alto. Por exemplo, na detecção de fraudes, é preferível detectar o maior número possível de fraudes (mesmo que isso gere falsos positivos). A revocação não leva em consideração os falsos positivos, então pode ser alta, mas o modelo pode estar classificando muitos exemplos incorretamente como positivos.

Se a revocação for de 70%, isso significa que o modelo conseguiu identificar 70% dos dados do conjunto de dados da classe positiva, mas 30% passaram despercebidos.

2.20.4 F1-Score

O F1-score é a média harmônica entre a precisão e a revocação. Ele equilibra essas duas métricas, sendo útil quando há um trade-off entre maximizar a precisão e a revocação. Um F1-score alto indica que o modelo tem um bom equilíbrio entre falsos positivos e falsos negativos. O F1-score é útil quando há um desequilíbrio entre classes e tanto os falsos positivos quanto os falsos negativos são importantes. Se o F1-score for 0,85, isso significa que o modelo está equilibrando bem precisão e revocação. Um valor alto de F1-score indica que o modelo está funcionando bem para ambas as métricas, mas a interpretação exata depende da importância relativa de falsos positivos e falsos negativos.

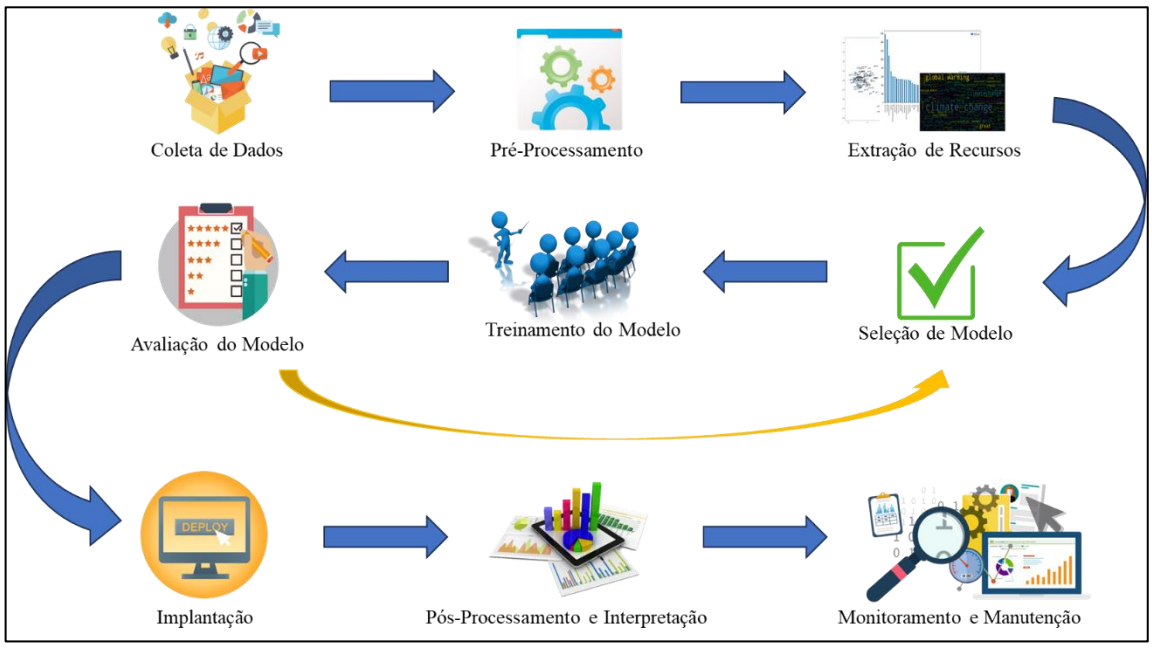
2.20.5 Interpretação das métricas

- Acurácia alta: o modelo está acertando a maioria das previsões, mas se as classes forem desbalanceadas, a acurácia pode esconder problemas em prever a classe minoritária.
- Precisão alta, revocação baixa: o modelo é conservador em suas previsões positivas, ou seja, evita falsos positivos, mas pode não estar identificando todas as instâncias da classe positiva (muitos falsos negativos).
- Precisão baixa, revocação alta: o modelo está identificando a maioria das instâncias positivas, mas comete muitos erros ao classificar instâncias negativas como positivas (muitos falsos positivos).
- F1-score alto: indica um bom equilíbrio entre precisão e revocação, o que é crucial em casos de desbalanceamento de classes.

2.21 Principais passos para aplicar a Análise de Sentimentos

Análise de sentimento é o processo para identificar e extrair informações subjetivas de textos usando PLN e mineração de texto. O sentimento é classificado em positivo, quando o texto expressa uma opinião favorável ou otimista, em negativo, quando o texto expressa uma opinião desfavorável ou pessimista. Os principais passos para aplicar a análise de sentimentos podem ser observados na Figura 7.

Figura 7: Principais passos para aplicar análise de sentimentos



Fonte: Própria autora.

1. Coleta de dados: nesse primeiro passo deve ser obtido os dados de fontes relevantes para a tarefa de análise de sentimentos.

2. Pré-processamento de dados: nesse passo várias ações podem ser realizadas
   - Limpeza de texto: remover o ruído dos dados de texto removendo caracteres especiais, pontuação, tags HTML, URLs e outros símbolos irrelevantes.
   - Tokenization (Tokenização): processo de dividir o texto em palavras ou tokens individuais.
   - Lowercasing (Letras minúsculas): conversão de todo o texto para letras minúsculas, garantindo consistência na representação das palavras.

- Stopword: remover palavras irrelevantes. Eliminar palavras comuns como "o", "é", "e", etc., que não possuem significado significativo para a análise de sentimento.
- Stemming or Lemmatization (Lematização): reduzir as palavras à sua forma base ou raiz para normalizar os dados do texto.
- Handling Negations (Lidando com Negações): abordar negações (por exemplo, "não é bom") para garantir uma análise de sentimento precisa.

3. Extração de recursos:
   - Bag-of-Words (BoW): representa os dados textuais como um vetor de características numéricas com base nas frequências das palavras.
   - TF-IDF (Term Frequency-Inverse Document Frequency): atribui pesos às palavras com base em sua frequência no documento e na frequência inversa em todo o corpus.
   - Word Embeddings: utiliza incorporações de palavras pré-treinadas, como Word2Vec, para representar palavras como vetores densos.

4. Seleção de modelo:

   Nesse passo devem ser escolhidos os modelos de análise de sentimento apropriados com base na natureza dos dados e na complexidade da tarefa de análise de sentimento, para serem aplicados ao conjunto de dados.

   Os modelos comuns são:

   - Modelos baseados em regras
   - Modelos baseados em léxico
   - Modelos de aprendizado de máquina
   - Modelos de aprendizado profundo

5. Treinando o modelo:

   Em modelos de aprendizado supervisionado, os dados são divididos em dois conjuntos: treinamento e validação. O modelo é treinado utilizando os dados de treinamento, enquanto seus parâmetros são otimizados para melhorar o desempenho.

6. Avaliação do modelo:

   O desempenho do modelo de análise de sentimento treinado é avaliado por meio de métricas como acurácia, precisão, revocação e F1-score.

7. Ajuste:

   Se o desempenho do modelo não for satisfatório, os parâmetros podem ser ajustados, podem ser utilizados mais dados, ou outra técnica de NLP pode ser utilizada.

8. Implantação:

   O modelo de análise de sentimento treinado pode ser aplicado em novos textos.

9. Pós-processamento e interpretação:

   As previsões de sentimento podem ser analisadas e os resultados podem ser interpretados para obter insights sobre o sentimento expresso nos dados de texto. As técnicas de pós-processamento podem incluir a agregação de pontuações de sentimento, a visualização de tendências de sentimento ou a extração de insights importantes.

10. Ajuste fino do modelo:

    O modelo de análise de sentimento pode ser ajustado com base no feedback, na alteração da distribuição de dados ou na evolução dos requisitos para melhorar o desempenho ao longo do tempo.

11. Monitoramento e Manutenção:

    Monitorar continuamente o desempenho do modelo de análise de sentimento em produção e atualizá-lo conforme necessário para manter sua eficácia.

Estas etapas fornecem uma estrutura estruturada para conduzir tarefas de análise de sentimento de forma eficaz e precisa. A implementação específica pode variar dependendo da natureza dos dados de texto e dos requisitos do projeto de análise de sentimento (ISLAM et al, 2024).

3 TRABALHOS RELACIONADOS

Perante o objetivo deste trabalho, pesquisou-se na literatura por trabalhos que utilizaram diferentes abordagens de PLN para realizar a análise de sentimentos em textos oriundos das redes sociais.

No artigo de Araújo et al. (2013) é apresentada uma comparação entre 8 (LIWC, Happiness Index, SentiWordNet, SASA, PANAS-t, Emoticons, SenticNet e SentiStrength) métodos de análise de sentimentos em termos de cobertura e em termos de identificação correta do sentimento utilizando duas bases de dados diferentes provenientes das redes sociais. Diversas métricas foram adotadas para avaliar a eficácia de um método: abrangência (que mede a fração de mensagens capturadas por um método), concordância (que avalia a correspondência de polaridade entre os métodos com base em uma base de dados rotulada), taxa de true positive (percentual de mensagens positivas corretamente identificadas), taxa de true negative (percentual de mensagens negativas corretamente previstas), acurácia (proporção de sentimentos corretamente identificados por um método), precisão (que avalia quão próximos estão os valores medidos entre si) e F-measure, que relaciona precisão e revocação. Como resultado, foi observado que os 8 métodos apresentam diferentes graus de abrangência e acurácia, sem que um único método tenha se destacado com os melhores resultados.

O artigo de Islam et al. (2024) oferece uma análise abrangente sobre a análise de sentimento baseada em aprendizagem profunda, abordando métodos, representações de texto, modelos de aprendizagem e suas aplicações. Os autores avaliam os avanços recentes em arquiteturas de aprendizado profundo, destacando prós e contras, além de discutir os desafios e as limitações enfrentados na área. O estudo apresenta um modelo inovador chamado CRDC, que combina cápsulas, CNN e RNN bidirecional, alcançando uma precisão superior em vários bancos de dados, como IMDB (88,15%) e Tóxico (98,28%).

O artigo também sugere que o aprendizado profundo supera outros métodos em termos de precisão, manipulação de dados e contexto. Embora métodos tradicionais de aprendizado supervisionado sejam mais rápidos, eles apresentam limitações no tratamento de aspectos como negação e intensificadores em frases. Já os métodos baseados em cápsulas e RNN, especialmente com atenção, mostraram-se mais eficazes.

Adicionalmente, os autores discutem os desafios da análise de sentimento, como a falta de coerência e a manipulação de significado semântico, além de identificarem a necessidade de

novas técnicas para lidar com dados multimodais e em tempo real. O estudo conclui que há espaço para a evolução e aprimoramento de métodos atuais para superar essas barreiras.

4 PROPOSTA

A proposta para esse trabalho é aplicar abordagens de PLN para realizar a análise de sentimentos em uma base de dados do Twitter com textos acerca de mudanças climáticas no período de 21/09/2017 até 19/05/2019 e comparar a eficácia da classificação dos sentimentos em positivo e negativo dessas abordagens.

Com base nos principais passos para aplicar a análise de sentimentos, para essa proposta os seguintes passos foram realizados:

1. Obtenção de uma base de dados sobre mudanças climáticas, que contém informações sobre sentimentos e textos, oriundos de redes sociais.

2. Os dados foram pré-processados realizando:
   - Limpeza de texto: removendo caracteres especiais, pontuação, tags, HTML, URLs e outros símbolos irrelevantes.
   - Tokenization.
   - Lowercasing: convertendo o texto em letras minúsculas para garantir consistência na representação das palavras.
   - Stopword: remoção de palavras irrelevantes que não possuem significado significativo para a análise de sentimento.
   - Lemmatization: reduzindo as palavras à sua forma base ou raiz para normalizar os dados do texto.

3. Foram realizadas as extrações de recursos utilizando:
   - Bag-of-Words (BoW), TF-IDF (Term Frequency-Inverse Document Frequency) e Word Embeddings utilizando Word2Vec.

4. Foram selecionados os modelos de recursos léxicos: VADER e TextBlob; o modelo de aprendizado de máquina supervisionado: Regressão Logística; e os modelos de aprendizado profundo: LSTM e GRU e o modelo transformers BERT.

5. Para modelo de aprendizado de máquina supervisionado, os dados foram divididos em dois conjuntos: os de treinamento e os de validação. Dessa forma, o modelo foi treinado usando os dados de treinamento.

6. O desempenho dos modelos de análise de sentimento treinados foi avaliado usando métricas de avaliação precisão, revocação, F1-score e acurácia.

5 AVALIAÇÃO EXPERIMENTAL

5.1 Dataset

A base de dados com postagens sobre mudanças climáticas, que contém informações de análise de sentimentos, foi obtida no Kaggle (https://www.kaggle.com/datasets/thedevastator/social-media-sentiment-and-climate-change). Essa base de dados é procedente do Twitter e pode ser utilizada para verificar a frequência com que determinadas palavras aparecem nos comentários sobre mudanças climáticas e para analisar o sentimento dessas postagens com relação aos sentimentos positivos e negativos. Foi utilizado o arquivo nltk_split.csv que possui dados do período de 21/09/2017 até 19/05/2019.

Esse arquivo tem um conjunto de 163669 linhas e 17 colunas, descritas na Tabela 1.

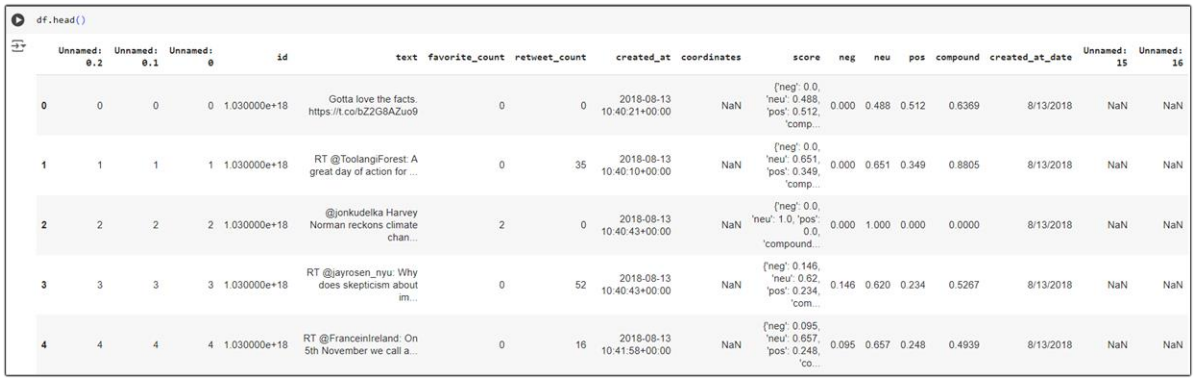
Tabela 1: Descrição do arquivo nltk_split.csv

#	Column	Non-Null	Count	Dtype
0	Unnamed: 0.2	non-null	163669	int64
1	Unnamed: 0.1	non-null	163669	int64
2	Unnamed: 0	non-null	163669	int64
3	id	non-null	163669	float64
4	text	non-null	163669	object
5	favorite_count	non-null	163669	int64
6	retweet_count	non-null	163669	int64

7	created_at	non-null	163669	object
8	coordinates	non-null	163669	object
9	score	non-null	163669	object
10	neg	non-null	163669	float64
11	neu	non-null	163669	float64
12	pos	non-null	163669	float64
13	compound	non-null	163669	float64
14	created_at_date	non-null	163669	object
15	Unnamed: 15	non-null	163669	float64
16	Unnamed: 16	non-null	163669	float64

Na Figura 8 pode ser observado um conjunto de 5 linhas do arquivo nltk_split.csv.

Figura 8: Exemplos de linhas do arquivo nltk_split.csv



```
df.head()
```

	Unnamed: 0.2	Unnamed: 0.1	Unnamed: 0	id	text	favorite_count	retweet_count	created_at	coordinates	score	neg	neu	pos	compound	created_at_date	Unnamed: 15	Unnamed: 16
0	0	0	0	1.030000e+18	Gotta love the facts. https://t.co/bZ2G8AZuo9	0	0	2018-08-13 10:40:21+00:00	NaN	{'neg': 0.0, 'neu': 0.488, 'pos': 0.512, 'comp...	0.000	0.488	0.512	0.6369	8/13/2018	NaN	NaN
1	1	1	1	1.030000e+18	RT @ToolangiForest: A great day of action for ...	0	35	2018-08-13 10:40:10+00:00	NaN	{'neg': 0.0, 'neu': 0.651, 'pos': 0.349, 'comp...	0.000	0.651	0.349	0.8805	8/13/2018	NaN	NaN
2	2	2	2	1.030000e+18	@jonkudelka Harvey Norman reckons climate chan...	2	0	2018-08-13 10:40:43+00:00	NaN	{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound...	0.000	1.000	0.000	0.0000	8/13/2018	NaN	NaN
3	3	3	3	1.030000e+18	RT @jayrosen_nyu: Why does skepticism about im...	0	52	2018-08-13 10:40:43+00:00	NaN	{'neg': 0.146, 'neu': 0.62, 'pos': 0.234, 'com...	0.146	0.620	0.234	0.5267	8/13/2018	NaN	NaN
4	4	4	4	1.030000e+18	RT @FranceinIreland: On 5th November we call a...	0	16	2018-08-13 10:41:58+00:00	NaN	{'neg': 0.095, 'neu': 0.657, 'pos': 0.248, 'co...	0.095	0.657	0.248	0.4939	8/13/2018	NaN	NaN

Foram aplicadas as seguintes regras de filtragem no arquivo nltk_split.csv:

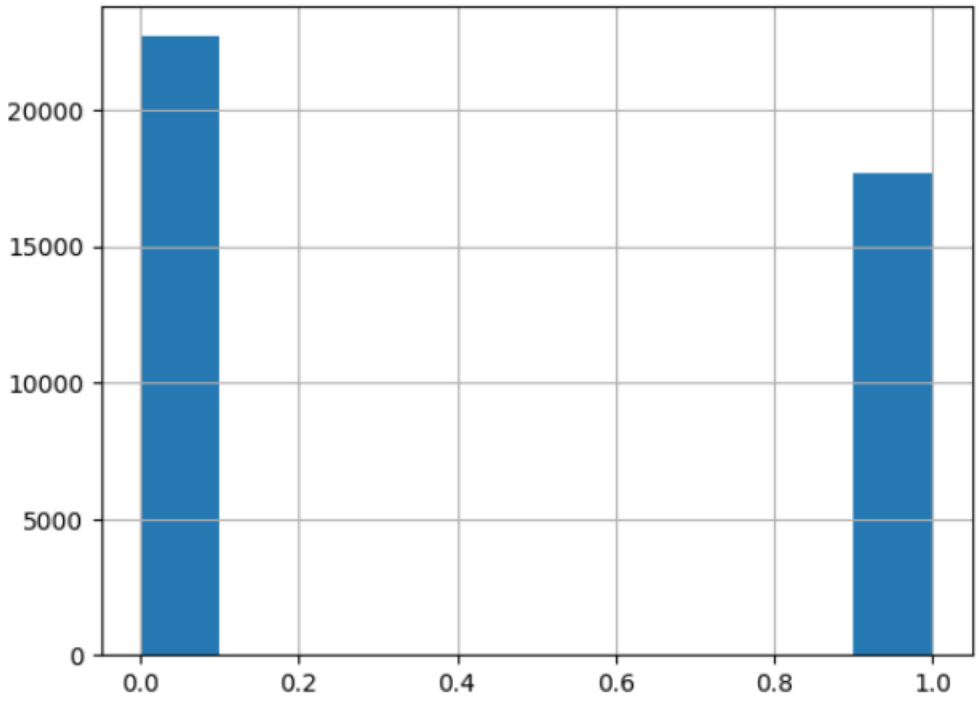
- Foram considerados textos positivos quando o valor da coluna compound fosse maior que 0,5. Foram considerados textos negativos quando o valor da coluna compound fosse menor que -0,5. E foram considerados textos neutros quando o valor da coluna compound fosse entre -0,5 e 0,5, inclusive.

```
def polarity_score(compound):
    if compound > 0.5:
        return "positive"
    elif compound < -0.5:
        return "negative"
    else:
        return "neutral"
```

- Foram realizados os seguintes mapeamentos: negative igual a -1, positive igual a 1 e neutral igual a 0.
- Foram removidos os textos classificados como neutral.
- Foi realizado um novo mapeamento: negative igual a 0, e positive se manteve com valor 1.
- Foi criada uma nova coluna chamada sentimento_nro que pudesse receber os valores 0 ou 1, de acordo com a classificação do texto, negativo ou positivo, respectivamente.
- Foram removidas as colunas: Unnamed: 0.2, Unamed: 0.1, Unnamed: 0, id, favorite_count, retweet_count, created_at, coordinates, score, neg, neu, pos, compound, created_at_date, Unnamed: 15, Unnamed: 16.

Dessa forma, o conjunto de dados que foram trabalhados são de 40.402 linhas (22.698 textos classificados como positivos e 17.704 textos classificados como negativos) e 2 colunas (text e sentiment_nro). O histograma pode ser observado na Figura 9.

Figura 9: Histograma com a quantidade de textos positivos e negativos



5.2 Configuração Experimental

O experimento foi realizado em linguagem de programação Python no Google Colab. O Google Colab é uma plataforma gratuita baseada em nuvem oferecida pelo Google. Ela fornece um ambiente de notebook que permite a criação e a execução de código diretamente no navegador. É voltada para o domínio da ciência de dados e aprendizado de máquina. Fornece acesso gratuito a unidades de processamento gráfico (GPUs) para tarefas que exigem computação intensiva, como treinamento de modelos de aprendizado de máquina. O Colab já tem várias bibliotecas pré-instaladas que são utilizadas em aprendizado de máquina, como Numpy, Pandas, TensorFlow, Keras entre outras. Desa forma, não é necessário configurar o ambiente de desenvolvimento.

5.3 Resultados e Discussão

5.3.1 Bag-of-Words

A primeira técnica aplicada ao conjunto de dados foi a Bag-of-Words (BoW) para analisar os textos. Foram removidas as stopwords, foi feita a tokenização dos textos, foi realizada a limpeza dos textos (removendo caracteres especiais, pontuação, html, url, espaços extras) e os textos foram convertidos para letras minúsculas.

As palavras que expressam sentimentos posivitos podem ser observadas na Figura 10 e as que expressam sentimentos negativos podem ser observadas na Figura 11. Quando maior o tamanho das letras da palavra, mais vezes essa palavra apareceu no conjunto de textos.

Figura 10: Textos positivos

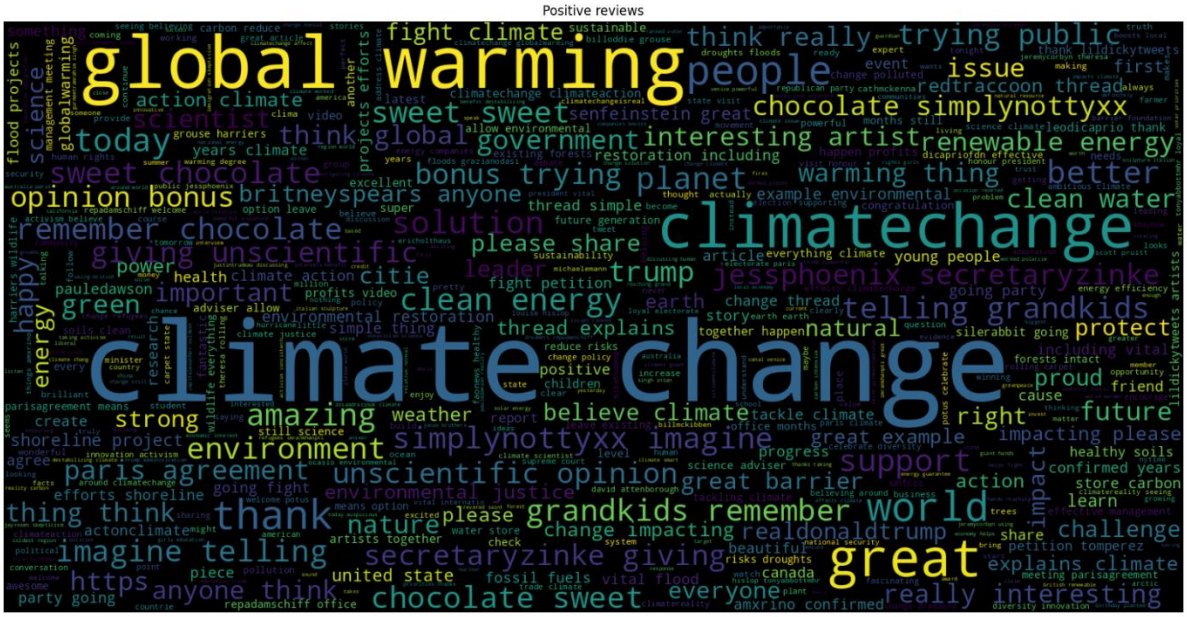


Figura 11: Textos negativos

5.3.2 Term Frequency-Inverse Document Frequency

A segunda técnica aplicada foi o Term Frequency-Inverse Document Frequency (TF-IDF) também para analisar os textos. É uma métrica que avalia a importância de uma palavra dentro de um conjunto de documentos, dada a frequência com que a palavra aparece nesses documentos.

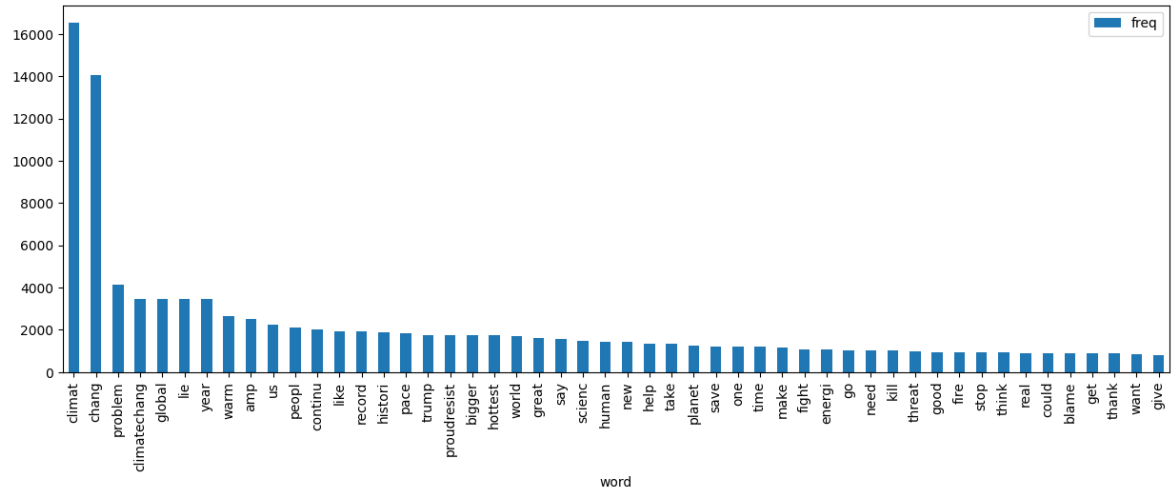
Foi utilizada a classe CountVectorizer do scikit-learn que transforma uma coleção de documentos de texto em uma matriz numérica de contagens de palavras. Esta classe possui vários parâmetros que também podem auxiliar em tarefas de pré-processamento de texto. Pode ser observada na Tabela 2, as 50 palavras que aparecem nos textos com maior frequência. Assim como, pode ser observado na Figura 12 o gráfico de barras com essas 50 palavras e as frequências com que aparecem nos textos.

Tabela 2: Palavra e frequência com que aparecem nos textos

indice	palavra	frequência
4882	climat	16537
4277	chang	14068
21093	problem	4130
4915	climatechang	3475
10699	global	3469
15589	lie	3464

29785	year	3463
28868	warm	2663
976	amp	2499
28244	us	2248
20146	peopl	2114
5598	continu	2016
15622	like	1930
21911	record	1917
11987	histori	1882
19691	pace	1843
27470	trump	1753
21256	proudresist	1744
2615	bigger	1740
12191	hottest	1734
29529	world	1716
11045	great	1604
23406	say	1571
23527	scienc	1489
12289	human	1436
18420	new	1423
11817	help	1349
26110	take	1337
20541	planet	1261
23354	save	1210
19323	one	1197
26940	time	1191
16217	make	1143
9486	fight	1075
8470	energi	1054
10781	go	1039
18298	need	1033
14743	kill	1025
26850	threat	985
10848	good	948
9552	fire	937
25453	stop	921
26784	think	912
21787	real	905
5775	could	898
2831	blame	893
10533	get	876
26469	thank	865
28861	want	840
10639	give	816

Figura 12: Gráfico de barras com as palavras e frequência com que aparecem nos textos



Foi utilizada a classe TfidfVectorizer do scikit-learn que também é uma técnica de pré-processamento usada para converter dados de texto em formato numérico. Porém, essa classe não conta apenas a frequência de cada palavra, mas atribui um peso a cada palavra com base em sua frequência no conjunto de documentos. Isso significa que atribui pesos maiores às palavras mais importantes no documento e pesos menores às palavras comuns. Isso é feito por meio de uma fórmula de frequência de documento inversa de frequência (TF-IDF) que equilibra a frequência de uma palavra em um documento com sua frequência em todos os documentos do conjunto de documentos.

Foi realizada uma ponderação do TF-IDF considerando as palavras que aparecem em no mínimo 3 textos e somando o TF-IDF de cada palavra de todos os textos para criar um raking de qual é a palavra com a maior soma de TF-IDF, com a intenção de melhorar a seleção das palavras mais importantes nos textos. Pode ser observada na Tabela 3, as 50 palavras que aparecem nos textos com as maiores somas do TF-IDF. Assim como, pode ser observado na Figura 13 o gráfico de barras com essas 50 palavras e a soma do TF-IDF.

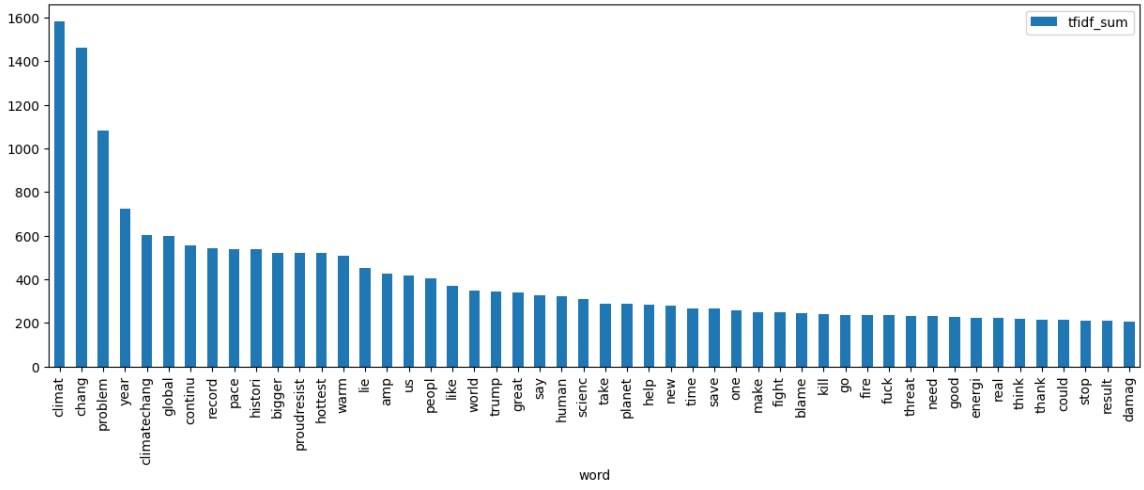
Tabela 3: Soma dos TF-IDF das palavras

index	word	tfidf_sum
1609	climat	1582.8432848601976
1411	chang	1462.4825011897892
6821	problem	1081.4154234745513
9659	year	724.0400278023351

1624	climatechang	604.535295449672
3610	global	596.8586639399999
1870	continu	555.8355302300655
7123	record	542.526003830107
6333	pace	539.2897782397075
4034	histori	539.102515597893
862	bigger	522.0709632006575
6893	proudresist	521.313100075493
4109	hottest	520.1873588702547
9372	warm	506.31814155155615
5125	lie	450.2028561146073
340	amp	426.01952873472317
9176	us	416.02338865835037
6471	peopl	404.41577023681157
5135	like	370.77342448927857
9585	world	348.55428513015227
8928	trump	342.1818836243755
3731	great	341.3773425754189
7611	say	325.72506573611986
4145	human	320.33116006079484
7655	scienc	309.50848803842456
8517	take	289.62211255955486
6606	planet	285.3354338845551
3982	help	283.8266724865214
5944	new	279.3777757930554
8763	time	264.9068424692005
7598	save	264.68416194079526
6198	one	255.92097885577112
5306	make	248.40814814108634
3219	fight	247.2210550264187
934	blame	244.8670782681535
4868	kill	242.08150196096926
3633	go	235.90799657478894

3250	fire	234.84172334678757
3447	fuck	234.6516183640363
8720	threat	233.25872657176905
5903	need	231.6558774059507
3653	good	227.6097572685795
2835	energi	224.20635806185373
7075	real	221.83246915824662
8702	think	217.3821924469924
8626	thank	215.5701572382033
1937	could	214.92142919221135
8298	stop	211.3239254862015
7294	result	207.9198007017611
2084	damag	207.14657973441032

Figura 13: Soma dos TF-IDF das palavras



E ainda foi realizada a geração de bigramas 2,2 que é o processo de criação de uma sequência de duas palavras consecutivas que aparecem em um texto. Para isso foi utilizada a classe TfidfVectorizer para encontrar os pares de palavras que apareceram em no mínimo 3 textos (TfidfVectorizer(tokenizer=meu_tokenizador, min_df=3,ngram_range=(2,2))). Pode ser observada na

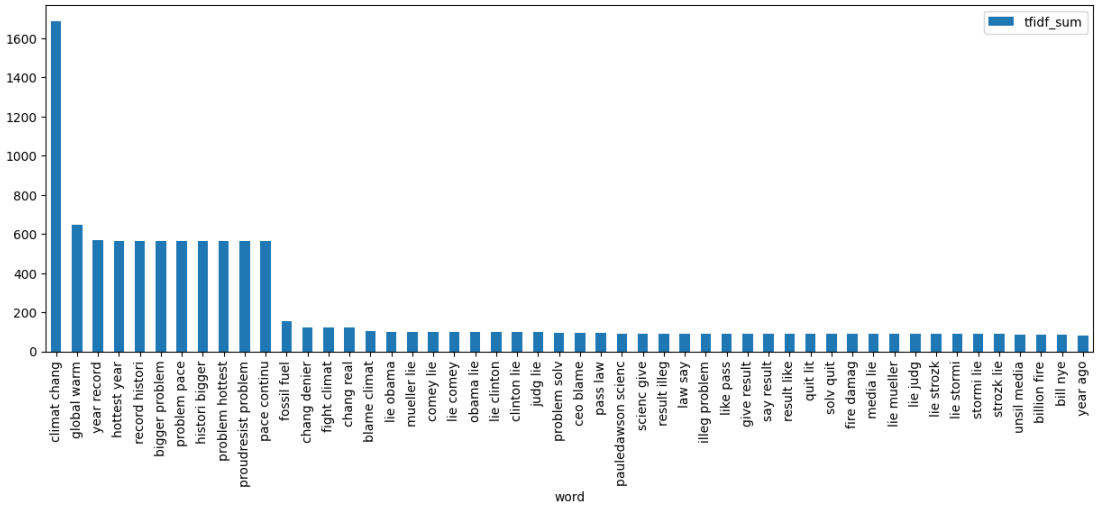
Tabela *4*, os 50 bigramas que aparecem nos textos com as maiores somas do TF-IDF. Assim como, pode ser observado na Figura 14 o gráfico de barras com esses 50 bigramas e a soma do TF-IDF.

Tabela 4: Bigramas 2,2 com respectiva soma de TF-IDF

	word	tfidf_sum
4224	climat chang	1687.237039
9272	global warm	645.405642
23838	year record	568.673378
10570	hottest year	565.035322
17377	record histori	564.253169
2123	bigger problem	564.090718
10407	histori bigger	563.220570
16600	problem hottest	563.220570
16606	problem pace	563.220570
15312	pace continu	562.921996
16803	proudresist problem	562.921996
8623	fossil fuel	156.088701
3350	chang denier	121.825980
8173	fight climat	120.306871
3723	chang real	120.220233
2257	blame climat	106.260539
12562	lie obama	99.984174
12545	lie comey	99.980953
4783	comey lie	99.980953
14085	mueller lie	99.980953
4600	clinton lie	99.619369
12544	lie clinton	99.619369
14853	obama lie	99.619369
11720	judg lie	99.196941
16611	problem solv	94.238746
3121	ceo blame	93.092380
15457	pass law	92.951902
15524	pauledawson scienc	92.288288
12676	like pass	92.148257
9189	give result	92.148257

12260	law say	92.148257
18338	say result	92.148257
17791	result illeg	92.148257
17793	result like	92.148257
10846	illeg problem	92.148257
18462	scienc give	92.148257
16963	quit lit	91.879529
19341	solv quit	91.879529
8284	fire damag	91.219326
13497	media lie	90.303889
12561	lie mueller	89.932443
12557	lie judg	89.725061
12567	lie strozk	88.982667
12566	lie stormi	88.982667
19848	strozk lie	88.982667
19792	stormi lie	88.982667
21845	unsil media	87.367631
2170	billion fire	86.351421
2151	bill nye	84.073217
23766	year ago	82.376142

Figura 14: Bigramas 2,2 com respectiva soma de TF-IDF



5.3.3 Word2Vec

A terceira técnica aplicada foi a Word Embeddings utilizando Word2Vec, que transforma palavras em vetores de alta dimensão. Esses vetores capturam semântica e contexto, permitindo que palavras com significados semelhantes estejam próximas no espaço vetorial. Embora existam vários modelos de treinamento, nesse projeto foi usado o Skip-Gram.

As 15 palavras mais próximas da palavra “climate” são:

[('change', 0.9945338368415833),
('looming', 0.9102566838264465),
('defining', 0.908141553401947),
('official', 0.9042466282844543),
('ewarren', 0.9034397602081299),
('chaos', 0.9005300402641296),
('gross', 0.8995162844657898),
('climatech', 0.8946119546890259),
('dread', 0.8938791751861572),
('planetary', 0.8894197344779968),
('chang', 0.8885343670845032),
('pentagon', 0.8860483169555664),
('proliferation', 0.8824966549873352),
('genuine', 0.8790950775146484),
('joesudbay', 0.876681387424469)]

Quanto mais próximo de 1 mais próxima a palavra está.

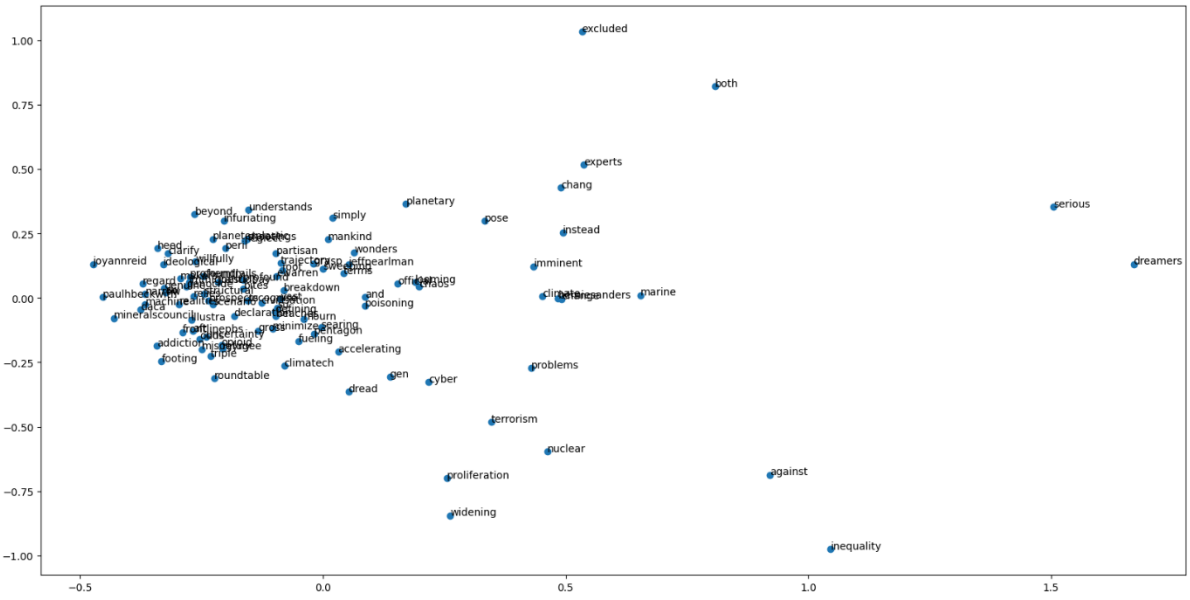
As 15 palavras mais próximas da palavra “change” são:

[('climate', 0.994533896446228),
('chaos', 0.9176994562149048),
('looming', 0.9152772426605225),
('ewarren', 0.9039932489395142),
('defining', 0.9018551707267761),
('pentagon', 0.8928453326225281),
('joesudbay', 0.8919951319694519),
('poisoning', 0.8873946070671082),
('gross', 0.886752188205719),

('dread', 0.8860046863555908),
('chang', 0.8858276605606079),
('official', 0.8857972621917725),
('proliferation', 0.8844568133354187),
('climatech', 0.8842229247093201),
('refugee', 0.8802300095558167)]

A distância vetorial em duas dimensões para a palavra “climate” pode ser observada na Figura 15.

Figura 15: Distância vetorial em duas dimensões para a palavra "climate"



5.3.4 Recursos léxicos: TextBlob

A quarta técnica aplicada foi de recursos léxicos usando a biblioteca TextBlob para analisar os textos e indicar a orientação do sentimento. Os resultados podem ser observados a seguir:

	precision	recall	f1-score	support
negative	0.79	0.80	0.80	22698
positive	0.74	0.73	0.74	17704
accuracy			0.77	40402
macro avg	0.77	0.77	0.77	40402
weighted avg	0.77	0.77	0.77	40402

5.3.5 Recursos léxicos: VADER

A quinta técnica aplicada foi de recursos léxicos usando a classe SentimentIntensityAnalyzer do vaderSentiment específica para análise de sentimentos, projetada para lidar com textos de redes sociais, utiliza dicionário de polaridade. Os resultados podem ser observados a seguir:

	precision	recall	f1-score	support
negative	0.99	0.99	0.99	22698
positive	0.99	0.99	0.99	17704
accuracy			0.99	40402
macro avg	0.99	0.99	0.99	40402
weighted avg	0.99	0.99	0.99	40402

5.3.6 Regressão Logística

A sexta técnica aplicada foi de aprendizado de máquina supervisionado - Regressão Logística, que consiste em um modelo estatístico de classificação utilizado para prever a probabilidade de ocorrência de um evento binário, isto é, um evento que pode ter apenas duas possibilidades, como "sim" ou "não", "1" ou "0". Foi utilizada a classe LogisticRegression do sklearn.linear_model. Os resultados podem ser observados a seguir:

	precision	recall	f1-score	support
0	0.95	0.94	0.95	2378
1	0.94	0.96	0.95	2498
accuracy			0.95	4876
macro avg	0.95	0.95	0.95	4876
weighted avg	0.95	0.95	0.95	4876

5.3.7 LSTM

A sétima técnica aplicada foi de modelo de aprendizado profundo: LSTM. É um tipo avançado das RNNs que inclui células de memória e portas para controlar o fluxo de informações, permitindo que a rede mantenha ou descarte informações conforme necessário. Foi utilizada a classe LSTM do tensorflow.keras.layers. O conjunto de dados (40402) foi divido entre treinamento (30301) e testes (10101). Os parâmetros para o modelo Sequencial do tensorflow.keras.models podem ser observados a seguir:

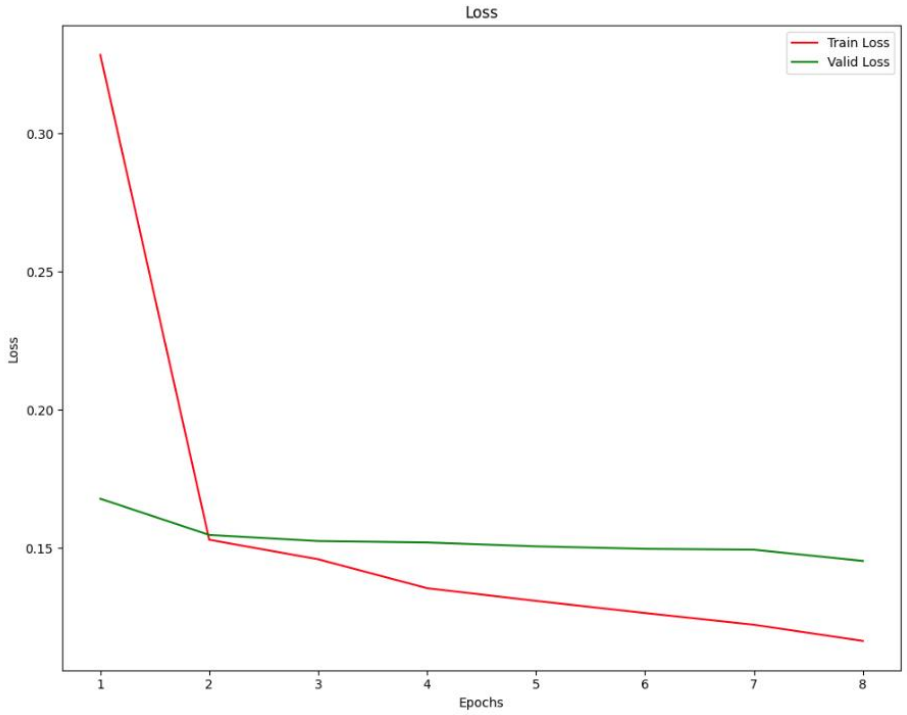
```
embedding_dim = 100

model = Sequential()
model.add(Embedding(max_words, embedding_dim))
model.add(SpatialDropout1D(0.2))
model.add(LSTM(100, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(1, activation='sigmoid'))
model.summary()
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
epochs = 8

batch_size = 264
history = model.fit(X_train, y_train, epochs=epochs, batch_size=batch_size,
                    validation_data=(X_test, y_test),
                    callbacks=[EarlyStopping(monitor='val_loss', patience=3, min_delta=0.0001)])
```

Os resultados para accuracy, loss, val_accuracy e val_loss durante a execução das 8 épocas podem ser observados a seguir.

```
Epoch 1/8
123/123 ━━━━━━━━━━━━━━━━━━━━ 44s 298ms/step - accuracy: 0.7367 - loss: 0.4860 - val_accuracy: 0.9354 - val_loss: 0.1679
Epoch 2/8
123/123 ━━━━━━━━━━━━━━━━━━━━ 36s 292ms/step - accuracy: 0.9431 - loss: 0.1547 - val_accuracy: 0.9404 - val_loss: 0.1548
Epoch 3/8
123/123 ━━━━━━━━━━━━━━━━━━━━ 42s 304ms/step - accuracy: 0.9434 - loss: 0.1475 - val_accuracy: 0.9407 - val_loss: 0.1526
Epoch 4/8
123/123 ━━━━━━━━━━━━━━━━━━━━ 40s 294ms/step - accuracy: 0.9478 - loss: 0.1356 - val_accuracy: 0.9415 - val_loss: 0.1521
Epoch 5/8
123/123 ━━━━━━━━━━━━━━━━━━━━ 36s 287ms/step - accuracy: 0.9526 - loss: 0.1272 - val_accuracy: 0.9413 - val_loss: 0.1507
Epoch 6/8
123/123 ━━━━━━━━━━━━━━━━━━━━ 42s 293ms/step - accuracy: 0.9534 - loss: 0.1236 - val_accuracy: 0.9420 - val_loss: 0.1498
Epoch 7/8
123/123 ━━━━━━━━━━━━━━━━━━━━ 42s 301ms/step - accuracy: 0.9534 - loss: 0.1205 - val_accuracy: 0.9441 - val_loss: 0.1495
Epoch 8/8
123/123 ━━━━━━━━━━━━━━━━━━━━ 35s 283ms/step - accuracy: 0.9561 - loss: 0.1158 - val_accuracy: 0.9448 - val_loss: 0.1454
```

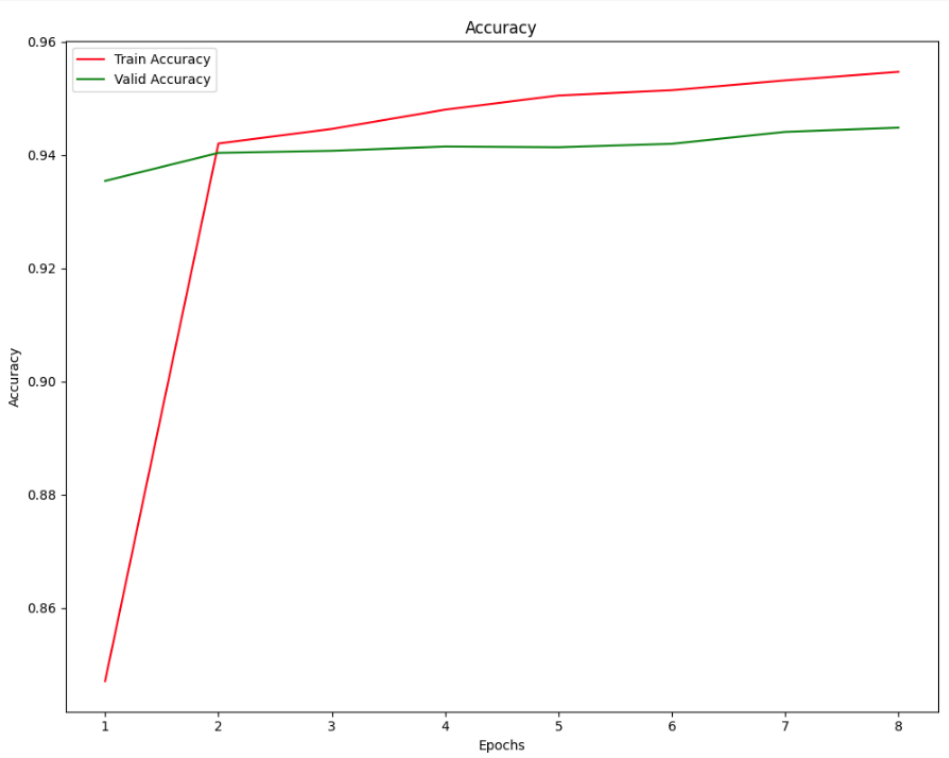
Na Figura 16 pode ser observado o valor de loss na curva de treinamento durante a execução das 8 épocas para o conjunto de dados para treinamento (em vermelho) e para testes (em verde).

Figura 16: Valor de loss na curva de treinamento durante a execução das 8 épocas



Na Figura 17 pode ser observado o valor da acurácia na curva de treinamento durante a execução das 8 épocas para o conjunto de dados para treinamento (em vermelho) e para testes (em verde).

Figura 17: Valor da acurácia na curva de treinamento durante a execução das 8 épocas



5.3.8 GRU

A oitava técnica aplicada foi outro modelo de aprendizado profundo: GRU. Desenvolvido para simplificar e melhorar a eficiência do treinamento das RNNs, é similar aos LSTM, mas com uma estrutura mais simples e menos parâmetros. Os parâmetros para o modelo Sequencial do tensorflow.keras.models podem ser observados a seguir:

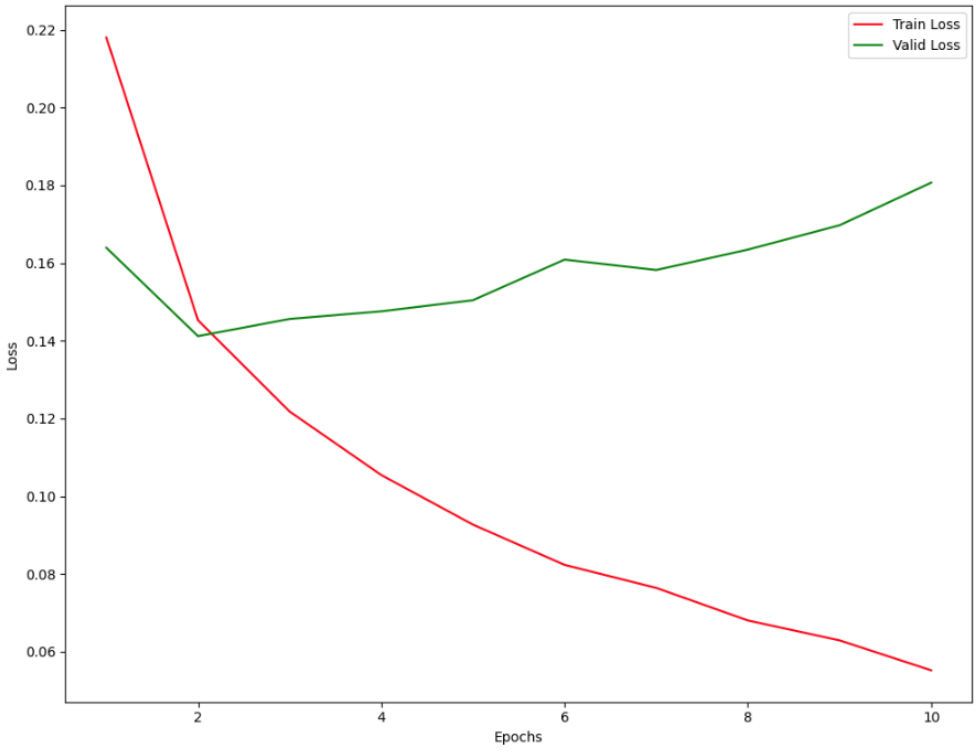
```
model_1 = Sequential()
model_1.add(Embedding(input_dim=max_words, output_dim=100))
model_1.add(GRU(100, return_sequences=True))
model_1.add(Dropout(0.5))
model_1.add(GRU(100))
model_1.add(Dropout(0.5))
model_1.add(Dense(units=1, activation='sigmoid'))
model_1.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
history_1 = model_1.fit(X_train, y_train, batch_size=32, epochs=10, validation_data=(X_test,y_test))
```

Os resultados para accuracy, loss, val_accuracy e val_loss durante a execução das 10 épocas podem ser observados a seguir.

```
Epoch 1/10
1011/1011 ━━━━━━━━━━━━━━━━━━━━ 19s 16ms/step - accuracy: 0.8566 - loss: 0.3023 - val_accuracy: 0.9394 - val_loss: 0.1639
Epoch 2/10
1011/1011 ━━━━━━━━━━━━━━━━━━━━ 19s 15ms/step - accuracy: 0.9440 - loss: 0.1475 - val_accuracy: 0.9465 - val_loss: 0.1412
Epoch 3/10
1011/1011 ━━━━━━━━━━━━━━━━━━━━ 21s 15ms/step - accuracy: 0.9553 - loss: 0.1197 - val_accuracy: 0.9467 - val_loss: 0.1456
Epoch 4/10
1011/1011 ━━━━━━━━━━━━━━━━━━━━ 15s 15ms/step - accuracy: 0.9630 - loss: 0.0969 - val_accuracy: 0.9483 - val_loss: 0.1476
Epoch 5/10
1011/1011 ━━━━━━━━━━━━━━━━━━━━ 21s 15ms/step - accuracy: 0.9668 - loss: 0.0881 - val_accuracy: 0.9458 - val_loss: 0.1504
Epoch 6/10
1011/1011 ━━━━━━━━━━━━━━━━━━━━ 21s 16ms/step - accuracy: 0.9701 - loss: 0.0788 - val_accuracy: 0.9425 - val_loss: 0.1609
Epoch 7/10
1011/1011 ━━━━━━━━━━━━━━━━━━━━ 20s 16ms/step - accuracy: 0.9729 - loss: 0.0704 - val_accuracy: 0.9478 - val_loss: 0.1582
Epoch 8/10
1011/1011 ━━━━━━━━━━━━━━━━━━━━ 21s 16ms/step - accuracy: 0.9743 - loss: 0.0649 - val_accuracy: 0.9474 - val_loss: 0.1635
Epoch 9/10
1011/1011 ━━━━━━━━━━━━━━━━━━━━ 20s 16ms/step - accuracy: 0.9795 - loss: 0.0562 - val_accuracy: 0.9480 - val_loss: 0.1697
Epoch 10/10
1011/1011 ━━━━━━━━━━━━━━━━━━━━ 20s 16ms/step - accuracy: 0.9806 - loss: 0.0516 - val_accuracy: 0.9431 - val_loss: 0.1807
```

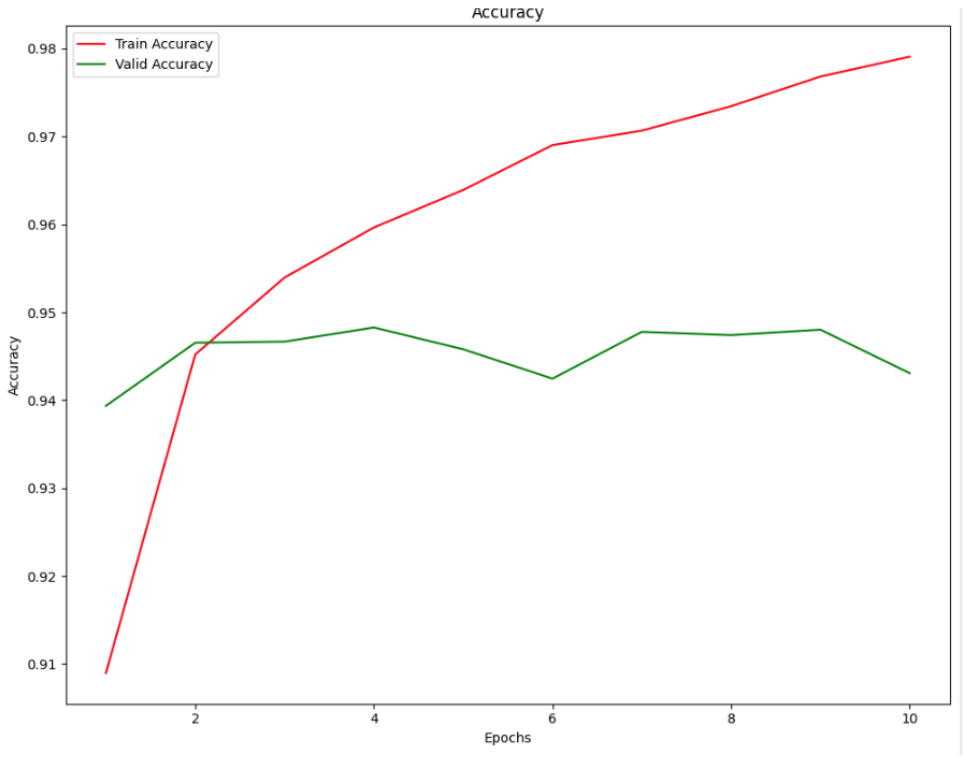
Na Figura 18 pode ser observado o valor de loss na curva de treinamento durante a execução das 10 épocas para o conjunto de dados para treinamento (em vermelho) e para testes (em verde).

Figura 18: Valor de loss na curva de treinamento durante a execução das 10 épocas



Na Figura 19 pode ser observado o valor de acurácia na curva de treinamento durante a execução das 10 épocas para o conjunto de dados para treinamento (em vermelho) e para testes (em verde).

Figura 19: Valor da acurácia na curva de treinamento durante a execução das 10 épocas



5.3.9 Transformers BERT

A nona técnica aplicada foi o BERT (Bidirectional Encoder Representations from Transformers), que é uma das variantes de Transformers amplamente utilizado em tarefas de compreensão de linguagem, como resposta a perguntas e análise de sentimentos. Foca no treinamento bidirecional para melhor compreensão contextual. Dessa forma, diferentemente de modelos como LSTM ou GRU que processam o texto em uma única direção (da esquerda para a direita ou vice-versa), BERT processa o texto em ambas as direções. Isso significa que o modelo considera o contexto das palavras antes e depois de uma palavra-alvo, proporcionando uma compreensão mais rica do significado.

O conjunto de dados (40402) foi divido entre treinamento (30301) e testes (10101). Foi utilizada a biblioteca ktrain, que usa o tf_keras no TensorFlow, para construir, treinar e depurar redes neurais de aprendizado profundo. Os parâmetros para o pré-processamento podem ser observados a seguir.

```
(x_train, y_train), (x_test, y_test), preproc = text.texts_from_df(df_train,
                                                                   'text',
                                                                   label_columns='sentiment',
                                                                   maxlen=64,
                                                                   preprocess_mode='bert',
                                                                   val_pct = 0.2,
                                                                   random_state=42
                                                                   )
```

O modelo de classificação para fine-turing do BERT pode ser observado a seguir.

```
model = text.text_classifier('bert', (x_train, y_train) , preproc=preproc)
classifier = ktrain.get_learner(
                                  model,
                                  train_data=(x_train, y_train),
                                  val_data=(x_test, y_test)
                                )
```

O treinamento do modelo fine turing pode ser observado a seguir.

```
classifier.fit(0.00002,2)
```

```
Epoch 1/2
758/758 [==============================] - 386s 483ms/step - loss: 0.1349 - accuracy: 0.9520 - val_loss: 0.1357 - val_accuracy: 0.9490
Epoch 2/2
758/758 [==============================] - 362s 478ms/step - loss: 0.0426 - accuracy: 0.9884 - val_loss: 0.0765 - val_accuracy: 0.9786
<tf_keras.src.callbacks.History at 0x7c4e1625a410>
```

E a avaliação do modelo pode ser observado a seguir.

	precision	recall	f1-score	support
negative	0.98	0.98	0.98	5720
positive	0.98	0.97	0.97	4381
accuracy			0.98	10101
macro avg	0.98	0.98	0.98	10101
weighted avg	0.98	0.98	0.98	10101

5.3.10 Considerações finais

Uma vez que o conjunto de dados utilizado nesse projeto estava balanceado entre as classificações de sentimentos positivos e negativos (22.698 positivos e 17.704 negativos), o resultado considerado para comparação entre os modelos de recursos léxicos (VADER e TextBlob), modelo de aprendizado de máquina supervisionado (Regressão Logística), os modelos de aprendizado profundo (LSTM e GRU) e o modelo transformers (BERT) foi a

acurácia. Em nenhum resultado desses modelos foi observado valores de precisão alta e revocação baixa ou precisão baixa e revocação alta. Além disso, o valor do F1-score também se apresentou alto para todos os modelos. Os valores da acurácia obtidos nos modelos aplicados em ordem de menor para maior acurácia podem ser observados na Tabela 5.

Tabela 5: Valores da acurácia obtidos nos modelos aplicados

Técnica	Acurácia
TextBlob	0,77
Regressão Logística	0,95
LSTM	0,96
GRU	0,98
BERT	0,98
VADER	0,99

A técnica TextBlob apresentou a menor acurácia (0,77) dentre as técnicas executadas nesse projeto. É uma técnica mais simples com relação aos métodos baseados em aprendizado de máquina.

A técnica de Regressão Logística apresentou o valor de acurácia de 0,95. A eficácia dessa técnica está diretamente associada à qualidade do pré-processamento dos dados, como tokenização, remoção de stop words e lematização.

A técnica LSTM apresentou o valor de acuraria de 0,96. É um tipo avançado das RNNs desenvolvida para lidar com dados sequenciais ou temporais mais longos, pois possui uma estrutura de célula de memória e portas para controlar o fluxo de informações, o que permite a retenção de informações relevantes por mais tempo.

A técnica GRU apresentou valor de acurácia de 0,98. É tipo de variante das LSTM com uma estrutura mais simples. Usa portas de atualização e portas de reinicialização para controlar o fluxo de informações. Possui menos parâmetros do que LSTM, o que pode levar a um treinamento mais eficiente, que pode ser observado no resultado.

A técnica BERT apresentou valor de acurácia de 0,98. É uma das variantes de Transformers e amplamente utilizado em tarefas de compreensão de linguagem, como análise de sentimentos. Foca no treinamento bidirecional para melhor compreensão contextual.

A técnica VADER apresentou o maior valor de acurácia (0,99) dentre as técnicas executadas nesse projeto. Essa técnica é específica para análise de sentimentos e projetada para tratar textos de redes sociais, e o conjunto de dados utilizados nesse projeto é oriundo de redes sociais, o Twitter.

Com relação às extrações de recursos Bag-of-Words, TF-IDF e Word Embeddings utilizando Word2Vec, pode ser observado que as palavras que mais se destacaram na Bag-of-Words para textos positivos foram: climate change, global warming, climatechange, great. Como pode ser observado na Figura 10. E para textos negativos foram: climate change, problem continue, history bigger, problem hottest, years recorded. Como pode ser observado na Figura 11.

Esses resultados também podem ser observados na frequência com que essas palavras aparecem nos textos aplicando TF-IDF. Vide Tabela 6:

Tabela 6: Frequência com que as palavras aparecem nos textos aplicando TF-IDF

indice	palavra	frequência
4882	climat	16537
4277	chang	14068
21093	problem	4130
4915	climatechang	3475
10699	global	3469
15589	lie	3464
29785	year	3463
28868	warm	2663

Resultado esse muito semelhante quando foi aplicada a somatória do TF-IDF e cujas palavras apareciam em pelo menos 3 textos. Vide Tabela 7. E também quando foi aplicada a geração do bigrama 2,2. Vide a Tabela 8.

Tabela 7: Somatória do TF-IDF para palavras que aparecem em pelo menos 3 textos

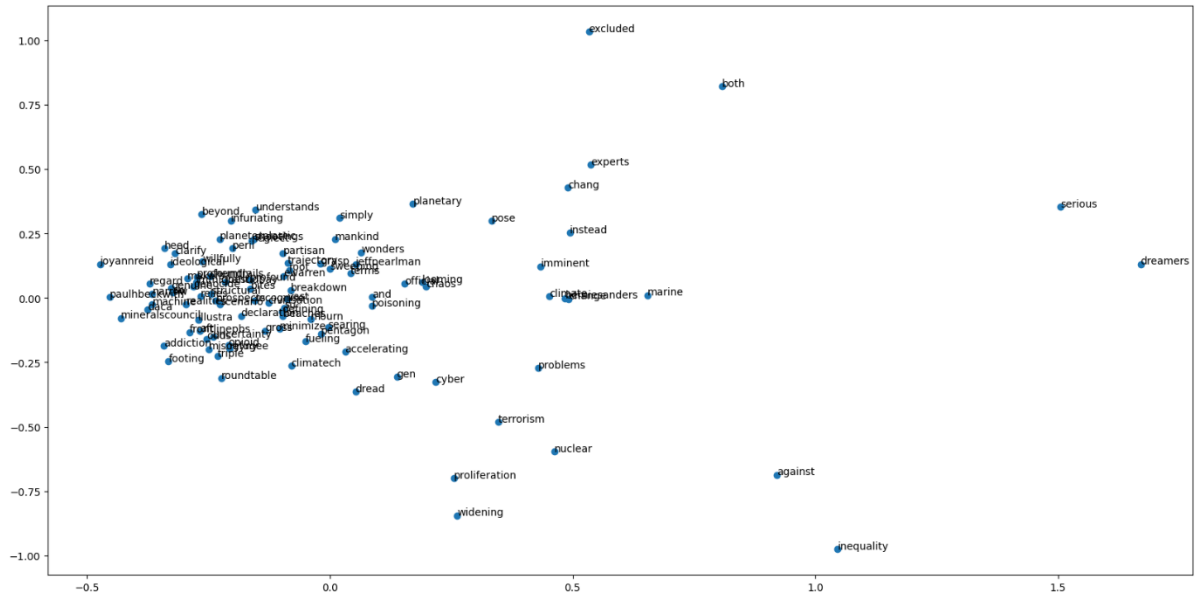
index	word	tfidf_sum
1609	climat	1582.8432848601976
1411	chang	1462.4825011897892
6821	problem	1081.4154234745513
9659	year	724.0400278023351
1624	climatechang	604.535295449672
3610	global	596.8586639399999
1870	continu	555.8355302300655
7123	record	542.526003830107

Tabela 8: Frequência dos bigramas

	word	tfidf_sum
4224	climat chang	1687.237039
9272	global warm	645.405642
23838	year record	568.673378
10570	hottest year	565.035322
17377	record histori	564.253169
2123	bigger problem	564.090718
10407	histori bigger	563.220570
16600	problem hottest	563.220570

Aplicando a técnica de Word Embeddings utilizando Word2Vec é possível observar a proximidade de palavras muito semelhante quando foi solicitada as palavras mais próximas de climate e change. A distância vetorial para a palavra climate pode ser observada na Figura 20.

Figura 20: Distância vetorial para a palavra “climate”



6 CONCLUSÕES

Diante da crescente importância das redes sociais na formação da opinião dos usuários sobre os mais abrangentes temas, que variam da compra de um produto, questões sociais até decisões políticas, é de suma importância direcionar os olhares para questões atuais que provocam impacto significativo na vida das pessoas e na sociedade, como é o caso das

mudanças climáticas. Dessa forma, analisar os sentimentos expressos nas redes sociais para melhor compreender as percepções das pessoas com relação a esse assunto é fundamental para formulação de políticas, campanhas de conscientização e avaliação da eficácia de ações de mitigação.

O objetivo desse trabalho foi aplicar abordagens de PLN para realizar a análise de sentimentos expressos nos textos de redes sociais acerca das mudanças climáticas e comparar a eficácia da classificação dos sentimentos em positivos e negativos. Dessa forma, foram executadas nove técnicas. As técnicas para extrações de recursos aplicando Bag-of-Words (BoW), TF-IDF (Term Frequency-Inverse Document Frequency) e Word2Vec. As técnicas de modelos de recursos léxicos: VADER e TextBlob. A técnica de modelo de aprendizado de máquina supervisionado: Regressão Logística. As técnicas de modelos de aprendizado profundo: LSTM e GRU. E a técnica BERT do modelo transformers. Essas técnicas foram aplicadas em uma base de dados procedente do Twitter, nos textos do arquivo nltk_split.csv, após o mesmo passar por um pré-processamento.

Realizar a classificação de sentimentos utilizando diferentes modelos, pois cada um tem seus pontos fortes e suas limitações, revelou quais ofereceram maior precisão e confiabilidade para o conjunto de dados.

Os resultados obtidos com as técnicas de extração de recursos são bem similares, pois as palavras que mais se destacaram na Bag-of-Words, apresentaram maior frequência no conjunto de dados quando aplicado o TF-IDF, assim como foi possível observar a proximidade dessas palavras quando aplicado o Word2Vec.

Dentre as outras técnicas consideradas, o maior e o menor valor de acurácia foram identificados nos modelos de recursos léxicos: VADER com 0,99 e TextBlob com 0,77. A técnica TextBlob é simples e possui algumas limitações como, por exemplo, inversores de sentimentos. Por outro lado, a técnica VADER foi desenvolvida especificamente para classificar sentimentos e analisar textos oriundos de redes sociais e é capaz de captar bem nuances da linguagem como gírias, negações e emoticons.

As demais técnicas implementadas (Regressão Logística, LSTM, GRU e BERT) apresentaram valores bem próximos ou iguais (0,95, 0,96, 0,98 e 0,98, respectivamente). A Regressão Logística é um modelo de classificação e não de regressão (apesar do nome). É um modelo estatístico que prevê a probabilidade da ocorrência do evento ser binário, no caso, positivo ou negativo. E o bom resultado se deve à qualidade do pré-processamento dos dados. A LSTM é uma variante das RNNs e, portanto, é aprendizado profundo e o modelo é treinado. A GRU também é uma variante das RNNs, similar à LSTM, porém tem uma estrutura mais

simples e requer menos parâmetros. Por essa simplicidade, a GRU pode convergir mais rapidamente durante o treinamento. E, por último, o BERT, que é uma variante do Transformers, é amplamente utilizado em tarefas de análise de sentimentos. O treinamento bidirecional melhora a compreensão contextual.

As principais contribuições desse trabalho foram:

- as implementações das técnicas
    - foi utilizado o ambiente do Google Colab e as bibliotecas pré-instaladas. Porém, foi necessário instalar e importar a biblioteca ktrain, que facilita o aprendizado profundo. Um ponto de atenção foi a verificação da compatibilidade das versões das bibliotecas, principalmente com a keras, que nesse caso precisou ser a tf_keras.
    - é altamente recomendado usar GPU para executar as implementações, principalmente as que fazem aprendizado profundo, pois precisam de CPU e memória. O Google Colab fornece tempo limitado para uso das GPUs.
    - experimentar na prática a execução das técnicas sobre um mesmo conjunto de dados e observar as diferenças e as semelhanças.
    - observar na prática, os conceitos teóricos das abordagens de análise de sentimento.
- os resultados obtidos
    - analisar os resultados obtidos em cada uma das técnicas de acordo com a precisão, revocação, acurácia e F1-Score.
    - interpretar os resultados de cada técnica e avaliar qual apresentou melhor resultado de acordo com as métricas utilizadas.
- O levantamento bibliográfico
    - foi realizada uma pesquisa sobre a definição de PLN e de Análise de Sentimentos. Assim como, foi realizada uma pesquisa para descrever as principais características, vantagens e limitações para cada um dos métodos considerados nesse trabalho, que podem ser utilizados para realizar a classificação dos sentimentos sobre um conjunto de textos. Foram incluídas as abordagens baseadas em regras, recursos léxicos, aprendizado de máquina e aprendizado profundo.

Esse trabalho pode ser continuado considerando outras abordagens para a classificação de sentimentos, e ainda, alterando parâmetros utilizados nesses experimentos. Assim como,

pode evoluir para aplicações de inteligência analítica a fim de promover formulações de políticas e campanhas de conscientização.

7 REFERÊNCIAS

ARAÚJO, M et al. Métodos para análise de sentimentos no twitter. In: Proceedings of the 19th Brazilian symposium on Multimedia and the Web (WebMedia'13), p. 19, 2013.

BURGES, C. J. C. A Tutorial on Support Vector Machines for Pattern Recognition. Data Mining and Knowledge Discovery, 2(2), 121-167, 1998.

CASELI, H.M.; NUNES, M. G. V. (org.) Processamento de Linguagem Natural: Conceitos, Técnicas e Aplicações em Português. 2 ed. BPLN, 2024. Disponível em: https://brasileiraspln.com/livro-pln/2a-edicao.

CHOMSKY, N. Syntactic Structures. The Hague: Mouton, 1957.

CRAWFORD, K. Atlas of AI: Power, Politics, and the Planetary Costs of Artificial Intelligence. Yale University Press. 2021.

DEVLIN, J.; CHANG, M. W.; LEE, K., & TOUTANOVA, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv preprint arXiv:1810.04805. 2019.

HAYKIN S. Neural Networks and Learning Machines. 3rd Edition. Pearson.2009.

HIRSCHBERG , J.; MANNING, C. D. Advances in Natural Language Processing. Science, 349(6245), 261-266. 2015.

HOCHREITER, S.; SCHMIDHUBER, J. Long Short-Term Memory. Neural Computation, 9(8), 1735-1780. 1987.

HSU, Chih-Wei. A Practical Guide to Support Vector Classification. Department of Computer Science, National Taiwan University, 2003.

HUTTO, C; GILBERT, E. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In: Proceedings of the international AAAI conference on web and social media. p. 216-225. 2014.

ISLAM, M. S. et al. Challenges and future in deep learning for sentiment analysis: a comprehensive review and a proposed novel hybrid approach. Artificial Intelligence Review, v. 57, n. 3, p. 62, 2024.

JIANG, Y., ZHANG, Y. A Review on Evaluation Metrics of Classification Algorithms in Machine Learning. International Journal of Computer Applications, 975, 8887. 2019.

JURAFSKY, D., MARTIN, J. H. Speech and Language Processing (3rd ed.). Pearson. 2023.

https://www.kaggle.com/datasets/thedevastator/social-media-sentiment-and-climate-change

LOH, W. Y. Classification and Regression Trees. Wiley Interdisciplinary Reviews: Computational Statistics, 3(2), 193-203. 2011.

LIU, B. Sentiment analysis and opinion mining. Springer Nature, 2022.

LIU, B. et al. Sentiment analysis and subjectivity. Handbook of natural language processing, v.2, n.2010, p.627–666, 2010.

MIENVE, D.; SWART, T.; OBAIDO, G. Recurrent Neural Networks: A Comprehensive Review of Architectures, Variants, and Applications. Information. 15. 517. 10.3390/info15090517. 2024.

MIKOLOV, T.; CHEN, K.; CORRADO, G.; DEAN, J. Efficient Estimation of Word Representations in Vector Space. arXiv preprint arXiv:1301.3781. 2013.

NOSOUHIAN, S.; NOSOUHIAN, F.; KAZEMI, K. A. A Review of Recurrent Neural Network Architecture for Sequence Learning: Comparison between LSTM and GRU. Preprints 2021, 2021070252.

REZENDE, S.O. Sistemas Inteligentes: fundamentos e aplicações. Editora Manole Ltda. 2003.

SCHMIDHUBER, J. Deep Learning in Neural Networks: An Overview. Neural Networks, 61, 85-117. 2015

GU, J.; WANG, Z.; KUEN, J.; MA, L.; SHAHROUDY, A.; SHUAI,B.; LIU, T.; WANG, X.; WANG, G.; CAI, J.; CHEN, T. Recent advances in convolutional neural networks. Pattern Recognition, 77, 354-377. 2018.

SUNDERMEYER, M.; SCHLÜTER, R.; NEY, H. Lstm neural networks for language modeling. In: Annual conference of the international speech communication association. [S.l.: s.n.], 2012.

TURING, A. M. Computing Machinery and Intelligence. Mind, 59(236), 433-460. 1950

VASWANI, A.; SHARDLOW, T.; 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 (Vol. 30). 2017.

WEIZENBAUM, J. ELIZA – A Computer Program For the Study of Natural Language Communication Between Man and Machine. Communications of the ACM, 9(1), 36-45. 1966.

WINOGRAD, T. Understanding natural language. Cognitive psychology, v. 3, n. 1, p. 1-191, 1972.

ZHANG, Y.; JIN, R.; ZHOU, Z.-H. Understanding bag-of-words model: a statistical framework. International Journal of Machine Learning and Cybernetics, Springer, v. 1, n. 1-4, p. 43–52, 2010.

ZHANG, L.; WANG, S.; LIU, B. Deep learning for sentiment analysis: A survey. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 8(4), p.e1253. 2018.

ZHU, W.; ZHANG, W.; LI, G.-Z.; HE, C.; ZHANG, L. A study of damp-heat syndrome classification using word2vec and tf-idf. In: IEEE International Conference on Bioinformatics and Biomedicine (BIBM). [S.l.], p. 1415–1420. 2016.

ZONG, C.; XIA, R.; ZHANG, J. Chapter 8: Sentiment Analysis and Opinion Mining. In Text Data Mining (pp. 163-199). Tsinghua University Press. 2021.