UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

ISADORA FERNANDES OLIVEIRA

IDENTIFICAÇÃO DE DISCURSO DE ÓDIO NAS REDES SOCIAIS POR MEIO DE REDES NEURAIS ARTIFICIAIS LSTM E ROBERTA

MEDIANEIRA

2025

ISADORA FERNANDES OLIVEIRA

IDENTIFICAÇÃO DE DISCURSO DE ÓDIO NAS REDES SOCIAIS POR MEIO DE REDES NEURAIS ARTIFICIAIS LSTM E ROBERTA

Identification of hate speech in social networks through Artificial Intelligence

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Ciência da Computação do Curso de Bacharelado em Ciência da Computação da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Jorge Aike Junior

MEDIANEIRA

2025

ISADORA FERNANDES OLIVEIRA

IDENTIFICAÇÃO DE DISCURSO DE ÓDIO NAS REDES SOCIAIS POR MEIO DE REDES NEURAIS ARTIFICIAIS LSTM E ROBERTA

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Ciência da Computação do Curso de Bacharelado em Ciência da Computação da Universidade Tecnológica Federal do Paraná.

Data de aprovação: 10/02/2025

Jorge Aikes Junior
Doutor
Universidade Tecnológica Federal do Paraná - Campus Medianeira

---

Juliano Rodrigo Lamb
Doutor
Universidade Tecnológica Federal do Paraná - Campus Medianeira

---

Nelson Miguel Betzek
Doutor
Universidade Tecnológica Federal do Paraná - Campus Medianeira

MEDIANEIRA
2025

Dedico este trabalho a minha mãe, pelo apoio
incondicional e amor constante.

AGRADECIMENTOS

Dedico este trabalho, antes de tudo, aos meus ancestrais, cujas histórias, força e sacrifícios permitiram que eu estivesse aqui hoje. Cada passo que dou é guiado pelo legado que deixaram, e carrego comigo sua sabedoria e resiliência.

A caminhada até aqui foi desafiadora, mas repleta de aprendizados e apoio incondicional de pessoas especiais, às quais dedico minha mais profunda gratidão.

À minha mãe Vera Lúcia, a pessoa mais importante da minha vida, que dedicou seus dias para que eu pudesse trilhar este caminho. Seu amor, esforço e sacrifício foram fundamentais para que este momento se tornasse realidade. Nada disso seria possível sem você.

Aos meus primos em especial a Niete, meu alicerce, que sempre incentivou meus estudos e acreditou no meu potencial. Seu apoio me fortaleceu em cada etapa dessa jornada.

Aos meus amigos, desde aqueles de infância, esporte e até os que a universidade me presenteou. Vocês foram essenciais nos momentos mais difíceis, oferecendo apoio, companhia e palavras de incentivo que fizeram toda a diferença.

Aos meus professores, que acreditaram no meu trabalho e me guiaram com paciência e sabedoria, transmitindo conhecimentos que levarei para toda a vida. Em especial, ao meu orientador Prof Dr. Jorge Aikes Junior, cujo conhecimento, dedicação e incentivo foram essenciais para a minha formação acadêmica. Sua orientação e apoio foram fundamentais para que eu pudesse evoluir e superar desafios deste trabalho.

Aos meus líderes e colegas de trabalho, que compreenderam minhas dificuldades e me apoiaram especialmente nos períodos de provas, permitindo que eu conciliasse o trabalho com os estudos.

Dedico também à espiritualidade que me guiou e fortaleceu em meio às dificuldades. Foi a fé, em suas diversas formas, que me deu forças para persistir e acreditar que cada obstáculo superado fazia parte de um propósito maior.

A todos vocês, meu mais sincero obrigado. Este trabalho também é de vocês.

"*A educação é a arma mais poderosa que você pode usar para mudar o mundo.*"
(Nelson Mandela)

RESUMO

O presente trabalho aborda a detecção de discurso de ódio nas redes sociais utilizando Rede Neural Artificial (RNA), com foco na comparação entre as arquiteturas *Long Short Term Memory* (LSTM) e a variante do *Bidirectional Encoder Representations from Transformers* (BERT), a RoBERTa. O crescimento alarmante de discursos prejudiciais nessas plataformas torna essencial o uso de Inteligência Artificial (IA) para a identificação automática desse tipo de conteúdo. O estudo busca analisar o desempenho dessas arquiteturas na classificação de discurso de ódio, avaliando suas métricas de eficácia e comparando os resultados com outros modelos que utilizaram o mesmo banco de dados. Para isso, foram coletados e pré-processados dados textuais, seguidos pelo treinamento e ajuste dos modelos, testando diferentes configurações de batch size e número de épocas. Os resultados indicaram que o modelo RoBERTa superou o LSTM apresentando maior acurácia na classificação dos conteúdos. No entanto, um modelo BERT Multilíngue demonstrou um desempenho ainda melhor do que o RoBERTa, possivelmente devido à sua capacidade de lidar com variações linguísticas e generalizar em diferentes contextos. Apesar da eficácia dos transformers, o alto custo computacional e o tempo de treinamento foram desafios para sua implementação em larga escala. Conclui-se que a Inteligência Artificial (IA), especialmente os modelos baseados em transformers, representa uma ferramenta promissora no combate ao discurso de ódio online. No entanto, otimizações são necessárias para equilibrar desempenho e eficiência computacional, tornando esses modelos mais viáveis para aplicações práticas.

Palavras-chave: inteligcia artificial; algoritmos; aprendizado de maquina.

ABSTRACT

This work addresses the detection of hate speech on social networks using RNA, focusing on the comparison between the LSTM architectures and the BERT variant, RoBERTa. The alarming growth of speeches on these platforms makes it essential to use Artificial Intelligence (AI) for the automatic identification of this type of content. The study seeks to analyze the performance of these architectures in the classification of hate speech, evaluating their effectiveness metrics and comparing the results with other models that used the same database. For this, textual data were collected and preprocessed, followed by training and tuning the models, testing different configurations of batch size and number of epochs. The results indicated that the RoBERTa model outperformed LSTM, presenting greater accuracy in the classification of content. However, a multilingual BERT model demonstrated even better performance than RoBERTa, possibly due to its ability to handle linguistic variations and generalize across different contexts. Despite the effectiveness of transformers, high computational cost and training time have been challenges for their large-scale implementation. We conclude that IA, especially transformer-based models, represent a promising tool in combating online hate speech. However, optimizations are straightforward to balance performance and computational efficiency, making these models more viable for practical applications.

Keywords: artificial intelligence; algorithms; machine learning.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

Siglas

AUC	*Area Under the Curve*
BERT	*Bidirectional Encoder Representations from Transformers*
BPE	*Byte Pair Enconder*
CNN	*Convolutional Neural Network*
CV	*Computer Vision*
GPT	*Generative pre-trained transformer*
GPU	*Graphics Processing Unit*
HSAN	Hierarchical Self-Attention Network
HTML	*Hypertext Markup Language*
IA	Inteligência Artificial
LSTM	*Long Short Term Memory*
MLM	*Masked Language Model*
NSP	*Next Sentence Prediction*
PLN	Processamento de Linguagem Natural
PTMS	*Transformer based pre-trained models*
ReLU	*Rectified Linear Unit*
RNA	Rede Neural Artificial
RNN	*Recurrent Neural Network*
RoBERTa	*Robustly Optimized BERT Approach*
ROC	*Receiver Operating Characteristic*
URL	*Uniform Resource Locator*
XML	Linguagem de Marcação Extensível

SUMÁRIO

1 INTRODUÇÃO

O discurso de ódio é caracterizado pela manifestação ou expressão motivada por preconceito ou intolerância, por meio da qual uma pessoa ou um grupo é discriminado, com base em suas características identitárias (SANTANA; BAPTISTA; GONÇALVES, 2022). Ele é um fenômeno abrangente que está relacionado com a liberdade de expressão, direitos individuais, de grupo e minoritários, bem como conceitos de dignidade, liberdade e igualdade. A interpretação do que é considerado discurso de ódio é uma tarefa complexa, que não possui uma definição única e consensual de modo uniforme nas diversas análises sobre o tema e que deve ser analisada sempre em um contexto social (TRINDADE, 2022). Isso significa que se faz necessário um olhar específico para avaliar quem são os grupos minoritários, o histórico de opressão, a política praticada no passado e no presente, a legislação vigente, dentre outros elementos (FARAH, 2018).

Os discursos de ódio, embora sejam recorrentes na Internet, violam não só as diretrizes das redes sociais como também a integridade das pessoas que utilizam essas plataformas. O crescente aumento do uso das redes sociais e a facilidade de acesso à Internet permitiram que as pessoas se expressassem livremente, muitas vezes sem pensar nas consequências de suas palavras. Além disso, a possibilidade de anonimato na Internet pode encorajar comportamentos agressivos e prejudiciais, incluindo o discurso de ódio (CASTAÑO-PULGARÍN *et al.*, 2021).

As redes sociais estão adotando medidas diferentes para combater o discurso de ódio, incluindo o uso de IA. A IA pode ajudar a identificar e remover contas falsas e conteúdos de caráter preconceituoso, além de detectar padrões de comportamento que possam indicar a presença de discurso de ódio e remover conteúdo que viole suas diretrizes de uso. No entanto, é importante ressaltar que a IA não é perfeita e pode cometer erros, por isso é importante que as plataformas de redes sociais continuem a trabalhar em conjunto com especialistas em direitos humanos e outras organizações para garantir que o discurso de ódio seja combatido de forma eficaz (SANTANA; BAPTISTA; GONÇALVES, 2022).

A IA pode contribuir para combater o discurso de ódio nas redes sociais de várias maneiras. Uma das principais maneiras é por meio da da detecção automática de conteúdo ofensivo, que pode ser feita por meio de algoritmos de aprendizado de máquina. Esses algoritmos podem ser treinados para identificar palavras e frases que são frequentemente usadas em discursos de ódio, bem como padrões de comportamento que possam indicar a presença de discurso de ódio. Além disso, a IA pode ser utilizada para analisar o contexto em que as palavras são usadas, a fim de determinar se elas são usadas de forma ofensiva ou não.

Atualmente, a subárea de IA que ajuda a identificar essas análises de texto é o Processamento de Linguagem Natural (PLN), encarregada de compreender a linguagem natural, disponibilizando, mecanismos automáticos capazes de extrair e categorizar informações provenientes da linguagem humana e do ambiente natural. Isso é realizado por meio da aplicação de modelos especializados (VICTORIA, 2022). A RNA é amplamente reconhecida por sua eficácia

na modelagem de sistemas dinâmicos (cujo estado evolui ao longo do tempo de acordo com um conjunto de regras ou equações) que lidam com dados temporais e informações dependentes, tais como vídeo, áudio e outros tipos de dados solicitados ao longo do tempo (SMAGULOVA; JAMES, 2019).

O modelo LSTM é uma categoria especial da *Recurrent Neural Network* (RNN), equipada com uma memória interna e portas multiplicativas, ela também demonstra habilidade para lidar com sequências de dados de extensão variável, destacando-se especialmente em modelar sistemas dinâmicos que envolvem informações temporais e de ordem dependente. Sua estrutura inclui células LSTM, que possuem portas de entrada, saída e esquecimento (SMAGULOVA; JAMES, 2019). Outra arquitetura de aprendizado de máquina que emergiu como destaque foi o *transformers*. Amplamente aplicada em diversas disciplinas, incluindo PLN, *Computer Vision* (CV) e processamento de fala. Inicialmente concebido como um modelo sequência-a-sequência para tradução automática, os *transformers* evoluíram para os *Transformer based pre-trained models* (PTMS) que apresentaram desempenho de destaque em diversas tarefas. Uma das versões mais notáveis da arquitetura *transformer* é o roBERTa, uma evolução do modelo BERT.Essa se distingue pela modificação em várias fases do treinamento, como o aumento do tamanho do lote, maior quantidade de dados e a remoção do mascaramento de palavras em algumas partes do treinamento, o que o torna mais eficiente e eficaz em várias tarefas de PLN.Deste modo, ambas as técnicas se apresentam como estado da arte para diversas tarefas de PLN (JAMIL; PIRAN; KWON, 2023) .

O discurso de ódio nas redes sociais, caracterizado por intolerância e preconceito, é um desafio complexo que envolve questões de direitos humanos e liberdade de expressão. As plataformas utilizam a IA com ferramentas de PLN para detectar esse tipo de conteúdo. Este trabalho tem como foco a avaliação das arquiteturas de RNA: LSTM e roBERTa para a classificação de conteúdos tóxicos considerando publicações na rede social X[1].

1.1 Objetivos

O objetivo geral deste trabalho é analisar, por meio de métricas, as arquiteturas LSTM e roBERTa para a identificação do discurso de ódio na rede social X. Este objetivo geral divide-se nos seguintes objetivos específicos:

- Ajustar um modelo LSTM e um modelo *Transformer* para a tarefa de detecção de discurso de ódio;
- Avaliar o desempenho das arquiteturas LSTM e *Transformer*;
- Comparar o desempenho das arquiteturas propostas nesse trabalho com arquiteturas estados da arte.

[1] https://x.com/login

1.2 Justificativa

De acordo com a Digital Brazil, 4,76 bilhões de pessoas utilizam redes sociais, sendo mais que a metade do total da população mundial (DIGITALBRAZIL, 2023). Denúncias de crimes envolvendo discurso de ódio nas redes sociais triplicaram nos últimos 6 anos, aponta levantamento (Jornal Nacional, 2023). Essas informações destacam que o discurso de ódio não é apenas um tópico relevante na atualidade, mas também um fenômeno persistente e alarmante. Ele impacta todas as pessoas que interagem nos diversos ambientes virtuais, seja de maneira direta ou indireta.

Diante desses dados, é perceptível que o ataques nas redes sociais é um problema crescente que afeta muitas pessoas em todo o mundo. Isso pode ter consequências graves para as pessoas que são alvo desse comportamento, incluindo danos psicológicos e até mesmo violência física. Além disso, o discurso de ódio pode ter um impacto negativo na sociedade como um todo, promovendo a intolerância e a discriminação.

A utilização de aprendizado de máquina para a detecção de discurso de ódio nas redes sociais é de extrema importância. A avaliação dos discursos em redes sociais, devido ao volume gigante de textos que seriam analisados, é inviável de ser realizada de maneira manual, por isso eles devem ser analisados de maneira automática, mesmo que seja apenas para uma triagem preliminar, podendo posteriormente ser realizada uma curadoria humana mais refinada e ajustada. Existem também razões sociais e de segurança, permitindo uma moderação mais eficaz e oportuna. Acredita-se que esse trabalho possa analisar e comprovar técnicas produtivas de reconhecimento dos discursos de ódio verificando o aprendizado contínuo e identificando esses ataques á medida que essas técnicas são expostas a mais dados.

1.3 Estrutura do trabalho

Este documento está estruturado da seguinte forma: Capítulo 1, "Introdução", oferece um contexto a problemática, define os objetivos gerais, específicos e a justificativa. No Capítulo 2, "Referencial Teórico", são apresentadas definições pertinentes a caracterização de discurso de ódio, bem como disponibilizada uma análise teórica sobre a IA e os modelos *Transformers* e LSTM, apresentando as origens, arquiteturas e a importância dos mesmos para a aplicação desta análise. Neste capítulo ainda são apresentados os trabalhos relacionados que fornecem uma revisão das pesquisas e discussões feitas por autores sobre o tema abordado no trabalho. No Capítulo 3, "Materiais e Métodos", disponibiliza as abordagens e ferramentas utilizadas para implementar e treinar os modelos dessas arquiteturas. No Capítulo 4, "Resultados", são apresentados os experimentos realizados e a análise comparativa entre as arquiteturas LSTM e RoBERTa. São discutidas métricas como acurácia, F1-score e tempo de treinamento, evidenciando as vantagens, limitações de cada modelo na detecção de discurso de ódio e a comparação dos modelos treinados a outros modelos utilizados no mesmo banco de dados BERTM. Por fim,

no Capítulo 5, "Conclusão", são sintetizados os principais achados da pesquisa, destacando a eficácia dos modelos baseados em transformers para a tarefa analisada e apontando desafios, como o alto custo computacional.

2 REFERENCIAL TEÓRICO

Neste capítulo são apresentados os conceitos sobre discurso de ódio, aprendizado de máquina, Redes Neurais Artificiais e Processamento de Linguagem Natural. O foco é apresentar a base teórica utilizada nas construções das arquiteturas LSTM e *Transformers*.

2.1 Discurso de ódio

O discurso de ódio se encontra em um complexo cruzamento com a liberdade de expressão, direitos individuais, de grupo e de minorias, assim como conceitos de dignidade, liberdade e igualdade. Sua definição é frequentemente debatida. De acordo com as leis nacionais e internacionais, o discurso de ódio se refere a expressões que incitam danos (especialmente discriminação, hostilidade ou violência) baseados nas características que identificam o alvo com um determinado grupo social ou demográfico. Isso pode incluir, mas não se limita a, discursos que defendem, ameaçam ou incentivam atos violentos. Para alguns, o conceito também abrange expressões que promovem um ambiente de preconceito e intolerância, sob a suposição de que isso pode alimentar discriminação direcionada, hostilidade e ataques violentos. No entanto, na linguagem cotidiana, as definições de discurso de ódio tendem a ser mais amplas, às vezes até se estendendo para incluir palavras que insultam aqueles no poder ou depreciam indivíduos particularmente visíveis (GAGLIARDONE *et al.*, 2015).

O discurso de ódio é um termo abrangente para diversos tipos de violência, dentre elas a LGBTQ+fobia que consiste em preconceito e exclusão social de grupos minoritários como pessoas lésbicas, gays, bissexuais, transexuais, pessoas *queer*, intersexo e assexuais (MEYER, 2015). Parte desses discursos também persiste o racismo, sendo um comportamento que se manifesta através de práticas contra grupos racialmente marginalizados, favorecendo e perpetuando o poder político, cultural e econômico nas mãos de indivíduos brancos (TYNES *et al.*, 2018). Essas práticas são intensificadas no ambiente virtual como ofensas explícitas em formatos de textos ou imagens. Dentre esses preconceitos, a xenofobia vem possuindo um espaço no ambiente virtual, se caracterizando como um preconceito ou aversão que englobam aspectos como a nacionalidade, a cultura e o país de origem de uma pessoa (VIEIRA, 2022). Similarmente a esses conjuntos de agressões, também persiste a misoginia, se caracterizando por um atitude negativa, de comportamento discriminatório baseado na idéia da inferioridade do gênero feminino (PIÑEIRO-OTERO; MARTÍNEZ-ROLÁN, 2021).

2.1.1 Redes sociais

O direito à liberdade de expressão não é apenas vital para a sobrevivência das sociedades democráticas, mas também é o que impulsiona o progresso dessas sociedades. É até

mesmo uma condição prévia para a realização de outros direitos. As ferramentas de comunicação da Internet democratizaram o poder de se comunicar publicamente, que antes era limitado aos meios de comunicação de massa (NICKEL, 2017).

A introdução da Internet desencadeou uma série de transformações na sociedade, uma das maiores transformações foi a oportunidade de expressão e interação social proporcionada pelas ferramentas de comunicação mediada por computador (RECUERO, 2009). A vinculação e a transmissão da comunicação de determinados grupos no meio virtual são compartilhadas por meio de redes sociais, na qual são ambientes onde o usuário comum contribui ativamente para a comunidade digital, seja através da criação de novos tópicos de discussão, compartilhamento de opiniões e experiências pessoais, ou respondendo a postagens de outros usuários. Alguns exemplos das redes sociais mais utilizadas no mundo são, *Facebook*[1], Instagram [2] Tik Tok[3] e X[4] que chegam na faixa de bilhões de usuários (BEAUSOLEIL, 2019).

Intermediários da Internet, como plataformas de redes sociais, provedores de serviços de internet ou motores de busca, definem em seus termos de serviço como podem intervir para permitir, limitar ou direcionar a criação e o acesso a conteúdos específicos. Uma grande parte das interações online ocorrem em plataformas de redes sociais que regulam as interações dos usuários com base em seus próprios termos de serviço e que também desenvolveram suas próprias definições de discurso de ódio e medidas para lidar com isso. Para um usuário que viole os termos de serviço, o conteúdo postado por ele pode ser removido da plataforma, ou seu acesso pode ser restrito para ser visualizado apenas por uma determinada categoria de usuários (por exemplo, usuários que vivem fora de um país específico) (BEAUSOLEIL, 2019).

O surgimento das redes sociais diminuiu as barreiras à comunicação e à formação de grupos, o que trouxe enormes potenciais de desenvolvimento, mas também desafiou instituições tradicionais que surgiram para garantir o exercício da cidadania nas sociedades democráticas. As redes sociais também ampliaram as ferramentas disponíveis para ações antissociais de alguns grupos e para a disseminação de discursos discriminatórios e agressivos. Por outro lado, indivíduos que antes não encontrariam ressonância para seus pensamentos de discriminação e agressividade agora encontram comunidades em que são acolhidos e não confrontados com argumentos divergentes. Portanto, as redes sociais têm um papel importante na formação de grupos e na circulação de informação, mas também exigem que as pessoas estejam atentas e críticas em relação ao conteúdo que consomem e compartilham.

Essas mesmas plataformas de mídias sociais estão sendo cada vez mais utilizadas para disseminar discursos de ódio e organizar atividades baseadas no preconceito. A possibilidade de anonimato e a facilidade de acesso que essas plataformas oferecem tornaram a criação e disseminação de discursos de ódio, que podem levar a crimes de ódio, uma tarefa simples no mundo virtual (ZHANG; LUO, 2019).

[1] https://pt-br.facebook.com/
[2] https://www.instagram.com/
[3] https://www.tiktok.com/pt-BR
[4] https://twitter.com/

2.2 Processamento de Linguagem Natural

O PLN é uma área de estudo que se concentra em criar modelos de computador capazes de entender e interagir com a linguagem humana de maneira eficaz. Seu propósito principal é capacitar os computadores a executarem uma variedade de tarefas úteis que envolvem a linguagem natural, como facilitar a comunicação entre humanos e máquinas, aprimorar a comunicação entre pessoas ou realizar processamentos úteis envolvendo texto ou fala (ANCHIÊTA *et al.*, 2021).

Essa área possui alguns termos que formam a base para entender como os modelos de IA funcionam e interagem com a linguagem humana. Eles também ajudam a compreender melhor os conceitos e técnicas utilizadas (ANCHIÊTA *et al.*, 2021):

- Corpus ou Corpora: são conjuntos de dados linguísticos organizados de acordo com critérios específicos, sendo suficientemente amplos e detalhados para representar o uso da linguagem em sua totalidade ou em algum de seus aspectos (SÁNCHEZ *et al.*, 1995);
- Tokens: é o processo de dividir um texto em unidades menores, como palavras, números e símbolos;
- Normalização: é o processo de assegurar que os dados estejam consistentes e prontos para serem processados. Isso envolve várias etapas, como transformar todas as letras em minúsculas, corrigir erros gramaticais simples, eliminar elementos não essenciais como imagens, *Uniform Resource Locator* (URL) e *hashtags*. Também efetua a remoção de espaços extras e pontuação duplicada, além de descartar palavras comuns e letras repetidas que não contribuem significativamente para o conteúdo textual. Essas medidas garantem que os dados estejam em um formato uniforme e limpo, facilitando análises posteriores;
- *Stopwords*: são palavras consideradas irrelevantes para a compreensão do texto, como artigos, pronomes e preposições;
- *Stemmer*: é uma técnica que reduz um termo à sua forma básica, removendo desinências, afixos e vogais temáticas. É um processo de normalização de texto;
- *Lematizer*: é uma ferramenta utilizada para agrupar diferentes formas flexionadas de uma palavra em uma única forma básica, chamada de "lema". Essa técnica busca encontrar a raiz comum das palavras, removendo sufixos, prefixos e outras variações, para representá-las de forma mais concisa e significativa;
- Etiquetador: é uma ferramenta que atribui a classe gramatical a cada palavra de um texto, levando em consideração tanto sua definição quanto seu contexto, ou seja, palavras adjacentes (JURAFSKY; MARTIN, 2009);

- Analisador Sintático *(Parser)*: é uma ferramenta usada para analisar a combinação de regras gramaticais em uma frase, com o objetivo de gerar uma árvore que represente a estrutura sintática da frase.

Os desafios enfrentados pelos computadores na compreensão de textos são evidentes, levando à necessidade de transformar o texto em formatos que possam ser mais facilmente manipulados por máquinas. Uma abordagem comum para superar essa dificuldade é a representação textual por meio de números, mais precisamente, vetores. Nesse contexto, técnicas como *word embeddings* e o algoritmo word2vec surgem como soluções. Ao mapear palavras para vetores em um espaço dimensional, essas técnicas não apenas permitem que os computadores processem e entendam melhor o texto, mas também facilitam para uma ampla gama de aplicações em PLN.

2.2.1 *Word Embeddings*

Word embeddings são representações numéricas de palavras, onde cada palavra é mapeada para um vetor de números reais em um espaço vetorial de alta dimensão. Esses vetores capturam o significado e a semântica das palavras com base em seu contexto em um corpus de texto (BAKAROV, 2018). Essas relações são determinadas pela proximidade das palavras no espaço vetorial. Em outras palavras, se duas palavras estão próximas uma da outra nesse espaço, é possível calcular a distância entre seus vetores (por exemplo, usando a distância cosseno) para identificar relações de similaridade entre elas. Quanto mais próximas as palavras estiverem no espaço vetorial, maior será a similaridade entre seus significados ou usos em contextos diferentes (MIKOLOV *et al.*, 2013). A Figura 1 mostra uma representação abstrata de um espaço vetorial onde é possível observar a proximidade entre palavras que são semanticamente semelhantes.

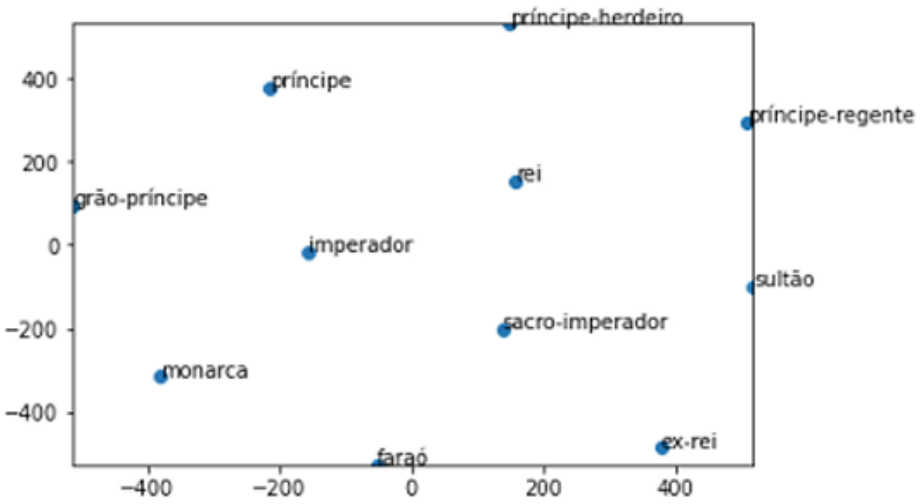
Na Figura 1, pode ser observado a proximidade entre palavras como "rei", "sultão", "imperador"e "faraó" no espaço vetorial, evidenciando sua similaridade semântica como sinônimos. Essa propriedade é explorada em diversas aplicações de PLN, incluindo análise de sentimentos e identificação de discurso de ódio.

2.2.2 *Word2vec*

O Word2Vec é um algoritmo popular de aprendizado de representações de palavras em modelos de linguagem. Também conhecido por sua eficiência computacional e sua capacidade de capturar informações semânticas e sintáticas das palavras. Ele se tornou uma técnica fundamental em várias aplicações de PLN, como tradução automática, análise de sentimentos e recomendação de palavras-chave (ANCHIÊTA *et al.*, 2021).

Esse modelo possui duas variações principais de treinamento (ANCHIÊTA *et al.*, 2021):

Figura 1 – Representação abstrata de espaço vetorial do algoritmo Word2vec



Fonte: Anchiêta *et al.* (2021).

- *Skip-gram*: o objetivo é prever as palavras vizinhas a partir de uma palavra de entrada. Em outras palavras, dada uma palavra de entrada, o modelo tenta prever as palavras ao seu redor no contexto. Este modelo é eficaz para capturar o contexto local de uma palavra e é frequentemente usado quando o objetivo é obter representações de palavras de alta qualidade, mesmo para palavras menos frequentes;

- *Continuous Bag of Words* (CBOW): nesse modelo, o objetivo é prever a palavra de destino com base em várias palavras de contexto ao seu redor. Ao contrário do Skip-gram, o *Continuous bag of words* utiliza um contexto maior para prever a palavra de destino. Este modelo é mais rápido para treinar e é frequentemente usado quando o objetivo é obter representações de palavras de maneira mais rápida e eficiente, especialmente em grandes conjuntos de dados.

Além das variações de treinamento do Word2Vec, os modelos baseados em *Transformers* representam uma evolução significativa no processamento de linguagem natural. Quando combinadas as técnicas de Word2Vec, os *Transformers* (discutidos no texto) podem aprimorar ainda mais a capacidade de reconhecimento de discurso de ódio, permitindo uma análise mais sofisticada e precisa do contexto linguístico.

Nas próximas seções, serão introduzidos modelos de RNA com foco em tarefas de PLN. Para tal, serão discutidos conceitos gerais de RNA, seguidos pela apresentação detalhada dessas arquiteturas específicas.

2.3 Inteligência Artificial

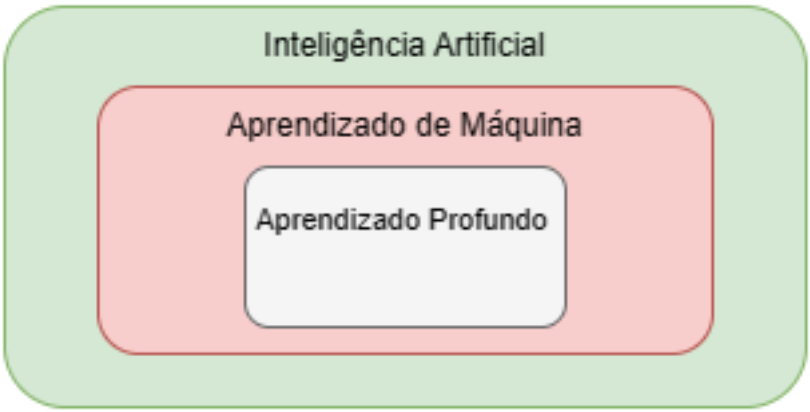
A IA é o campo de estudo que se concentra em criar elementos capazes de absorver informações do seu entorno e agir com base nessas informações. Essas entidades, chamadas de agentes, operam através de uma função que transforma sequências de percepções em ações. Nesse aspecto, eles utilizam as informações que coletam do ambiente, processam-nas e, em seguida, decidem a melhor ação a ser tomada com base nessas informações (NORVIG; RUSSELL, 1995). Os estudos da IA começaram a ganhar notoriedade por volta da década de 1950. O matemático britânico Alan Turing desempenhou um papel fundamental nesse início, com os testes de Turing, considerado um dos marcos mais importantes para a IA. Os testes de Turing consistia em avaliar a capacidade de inteligência de uma máquina. Para um computador ser considerado inteligente ele precisaria ter as seguintes habilidades:

- Aprendizado de Máquina: Permitindo que o computador se ajuste a novas situações e identifique e projete padrões;
- Processamento de Linguagem Natural: Isso permite que o computador se comunique efetivamente em uma linguagem natural;
- Representação de Conhecimento: Isso permite que o computador armazene o que aprendeu ou ouviu;
- Raciocínio Automatizado: Isso permite que o computador use as informações armazenadas para responder a perguntas e fazer novas inferências.

O aprendizado de máquina é uma abordagem de IA que permite que um sistema aprenda a realizar tarefas específicas a partir de dados de treinamento (MITCHELL, 2015). O objetivo é automatizar a construção de modelos analíticos para realizar tarefas cognitivas, como detecção de objetos ou tradução de linguagem natural. Isso é alcançado aplicando algoritmos que aprendem iterativamente a partir de dados de treinamento específicos do problema, permitindo que os computadores encontrem padrões ocultos e padrões complexos sem serem explicitamente programados previamente (DATABRICKS, 2021).

O aprendizado profundo é um subconjunto do aprendizado de máquina que envolve o uso de algoritmos de RNAs para aprender a representação de dados . Esse aprendizado, é capaz de aprender a partir de dados brutos, sem a necessidade de recursos humanos para extrair características relevantes. Ele é capaz de lidar com grandes quantidades de dados e é usado em várias áreas, como CV, análise semântica, PLN, recuperação de informações e gerenciamento de relacionamento com o cliente. O aprendizado profundo é uma área em constante evolução e que há ainda muito espaço para explorar suas aplicações (SHINDE, 2018). A relação entre os principais processos de aprendizagem de IA são demonstrados na Figura 2.

Figura 2 – Campos das Inteligência Artificial



Fonte: Adaptado de Shinde (2018).

A aprendizagem supervisionada é um tipo de aprendizado de máquina onde os algoritmos são treinados em um conjunto de dados rotulado, ou seja, um conjunto de dados onde cada exemplo está associado a uma saída desejada. O algoritmo aprende a mapear as entradas para as saídas com base nos exemplos de treinamento fornecidos (FONTANA, 2020). Na detecção de discurso de ódio, por exemplo, a aprendizagem supervisionada envolve treinar um modelo utilizando exemplos de texto rotulados como "discurso de ódio" ou "não discurso de ódio". O modelo aprende padrões nos dados rotulados para fazer previsões sobre se um novo texto contém discurso de ódio com base nas características que foram identificadas durante o treinamento. No aprendizado não supervisionado, por outro lado, a rede é deixada para encontrar padrões nos dados por conta própria (RAUBER, 2002).

A extração de características envolve a identificação e seleção das características mais relevantes dos dados para representá-los de forma mais eficaz (HAMAGUTI; BREVE, 2022). A aprendizagem de representação busca automaticamente aprender representações de alto nível dos dados, o que pode melhorar o desempenho em tarefas subsequentes de aprendizado de máquina (SCHLEDER; FAZZIO, 2021). A *Convolutional Neural Network* (CNN) ou Rede Neural Convolucional, originalmente desenvolvidas para processamento de imagens, também são aplicadas com sucesso em PLN, especialmente para tarefas como classificação de texto e análise de sentimento (ARAÚJO; VITTORAZZI, 2018). Elas são eficazes na extração de características locais e espaciais em dados sequenciais. Por fim, as RNNs ou Rede Neurais Recorrentes, foram projetadas para lidar com dados sequenciais, possuindo a capacidade de processar entradas de comprimento variável, mantendo uma memória de estados anteriores.

Essas técnicas permitem que os modelos capturem informações semânticas, sintáticas e contextuais e aprendam representações mais expressivas do discurso de ódio (**??**).

2.4 Redes Neurais Artificiais

RNA é um sistema projetado para modelar como o cérebro realiza uma tarefa específica. Geralmente, essa rede é construída usando componentes eletrônicos ou por meio de simulações por programação (KUBAT, 1999). Ela é composta por duas partes fundamentais: a estrutura e o método de aprendizado. Esta divisão é uma consequência natural da maneira como a rede é treinada. Diferentemente de um computador com arquitetura de Von Neumann, que é programado, a rede é treinada através de exemplos. O conhecimento sobre o problema em questão está contido nos exemplos, que devem estar disponíveis. O método de aprendizado generaliza esses dados e armazena o conhecimento nos parâmetros ajustáveis da rede, responsáveis para dar a importância das entradas, os pesos. Portanto, o construtor de um sistema baseado em RNA tem duas opções: a escolha do tipo de rede para resolver o problema em questão e o algoritmo para treinar a rede, ou seja, para ajustar os pesos da rede (RAUBER, 2002).

2.4.1 Modelo do Neurônio Cerebral

A rede neural do cérebro é formada pela interconexão de células chamadas neurônios. De maneira similar, as RNAs são uma simulação da rede neural biológica, onde os neurônios artificiais são interligados de forma semelhante à rede do cérebro (KOVáCS, 2023).

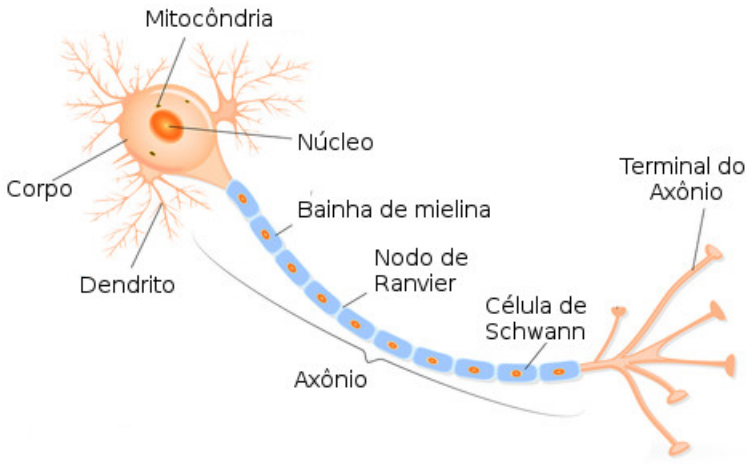
Um neurônio biológico (Figura 3) é formado por um corpo celular, um axônio e dendritos. Os dendritos funcionam como receptores de sinais eletroquímicos de outros neurônios, que são enviados para o corpo celular. Este último contém o núcleo e outras estruturas químicas vitais para a sobrevivência da célula. O axônio, por sua vez, é responsável por transmitir o sinal do neurônio para outros neurônios. A ligação entre os dendritos de dois neurônios, ou entre um neurônio e células musculares, é conhecida como sinapse (SHIRURU, 2016).

Em um neurônio biológico, os sinais são acionados quando a intensidade do sinal recebido pelos dendritos ultrapassa um determinado limite, conhecido como limiar de ativação. Quando isso ocorre, o neurônio emite um sinal elétrico ao longo do axônio, que é transmitido para outros neurônios por meio das sinapses. No caso de um neurônio artificial, a ativação é determinada por uma função de ativação. Quando a entrada do neurônio, combinada com o peso da conexão, ultrapassa um certo limite, o neurônio é ativado e produz uma saída (SHARMA; SHARMA; ATHAIYA, 2017).

2.4.2 Modelo do Neurônio Artificial

Uma RNA é composta por neurônios artificiais. Normalmente, o processamento de um único neurônio é a combinação linear das entradas com os pesos, seguida pela passagem da

Figura 3 – Neurônio Biológico



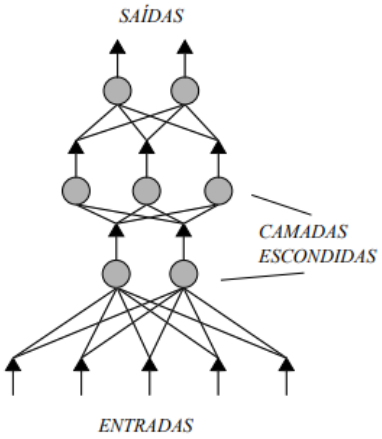
Fonte: Santos (2024).

combinação linear por uma função de ativação. O tipo de problema a ser resolvido geralmente define restrições em relação aos tipos de redes e métodos de aprendizado possíveis (RAUBER, 2002).

O modelo de neurônio artificial é uma representação matemática simplificada de um neurônio biológico Figura 5. O modelo de neurônio artificial mais simples é baseado no modelo proposto por McCulloch e Pitts em 1943. Este modelo busca simular as realidades biológicas que ocorrem dentro de uma célula do sistema nervoso. Ele descreve um neurônio como uma unidade de processamento que recebe entradas em D onde x_j são as sinapses. O processo envolve uma combinação linear das entradas, onde $net = w_1x_1 + w_2x_2 + ... + w_Dx_D = \sum_{j=1}^{D} w_jx_j = w^Tx$. Cada entrada possui um peso w_j que indica sua relevância de entrada x_j. O neurônio ativa sua saída em binário representado por y com o valor 1 se o limiar μ for ultrapassado. Caso contrário, a saída permanece inativa $y = 0$ (RAUBER, 2002). Esse modelo foi posteriormente aprimorado com a introdução de funções de ativação não-lineares, que permitem que os neurônios artificiais processem informações mais complexas. O modelo de neurônio artificial é a base para a construção de redes neurais artificiais, que são compostas por várias camadas de neurônios interconectados (SILVA *et al.*, 2019).

A potencialidade e flexibilidade do cálculo baseado em redes neurais artificiais vêm da interconexão de neurônios. Isso significa que, em uma rede neural, vários neurônios estão conectados entre si, formando uma estrutura complexa capaz de processar informações de forma paralela. Essa interconexão permite que a rede neural processe informações de forma distribuída, ou seja, cada neurônio processa uma parte da informação e a informação é combinada em conjunto para produzir uma saída. Esse paralelismo de processamento local é o que cria a "inteligência" global da rede, ou seja, a capacidade da rede de realizar tarefas complexas que seriam difíceis ou impossíveis para um único neurônio (RAUBER, 2002).

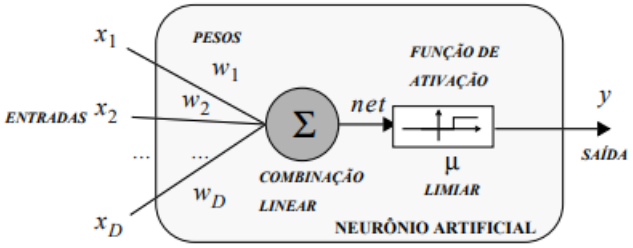
Figura 4 – Propagação da informação para frente entre neurônios



Fonte: Rauber (2002).

Figura 5 – Modelo de um neurônio de McCulloch e Pitts

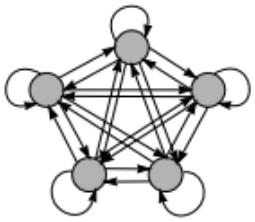
$$y = \Theta(\sum_{j=1}^{D} w_j x_j - \mu)$$



Fonte: Rauber (2002).

Uma categorização fundamental da topologia dos neurônios pode ser feita em relação ao método de propagação da informação recebida. Pode-se distinguir entre redes de propagação para frente *(feedforward)* Figura 4 e redes realimentadas *(recurrent)* demonstrado na Figura 6. No caso das redes de propagação para frente o fluxo de informação é unidirecional. Enquanto as redes alimentadas se caracterizam por retroalimentação dos resultados obtidos como entradas adicionais, promovendo a iteração e aprimoramento contínuos dos cálculos e saídas da rede, como demonstrado na Figura 6. Os Neurônios que recebem a informação simultaneamente agrupam-se em camadas (KOVáCS, 2023). A camada de entrada é composta pelos neurônios que recebem diretamente as informações iniciais da rede. Na sequência, obtêm-se a segunda camada, onde os neurônios recebem tanto as entradas da camada anterior quanto enviam saídas para a próxima camada. Essa progressão continua até chegar à camada final, conhecida como camada de saída. As camadas intermediárias, que estão entre as entradas e as saídas, são denominadas camadas ocultas (KOVáCS, 2023).

Figura 6 – Realimentação da informação entre neurônios



Fonte: Rauber (2002).

2.5 Processo de aprendizado

Ao estabelecer a estrutura da rede neural, é necessário treiná-la. Isso implica em ajustar os graus de liberdade que a rede possui para resolver a tarefa em questão de maneira ideal. Geralmente, isso significa que é preciso alterar os pesos entre os neurônios, seguindo um algoritmo específico. Durante a fase de treinamento da rede, é disposto um conjunto finito de exemplos de treino para ajustar os pesos (RAUBER, 2002).

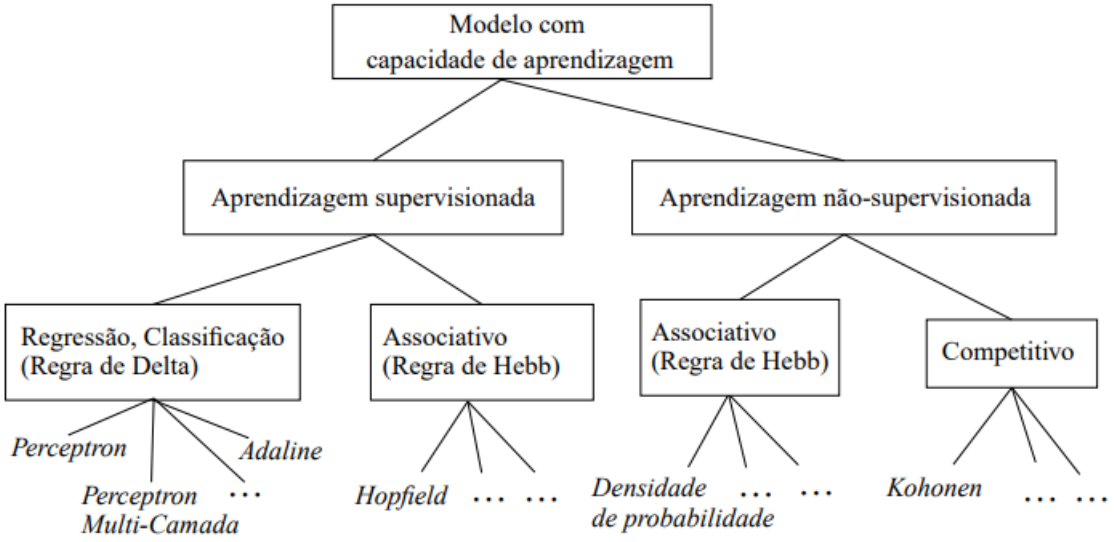
O aprendizado supervisionado é um processo em que a rede neural é treinada com um conjunto de exemplos rotulados. Cada exemplo consiste em um conjunto de entradas e uma saída desejada correspondente. Durante o treinamento, a rede ajusta os pesos sinápticos para minimizar a diferença entre a saída produzida pela rede e a saída desejada. Esse processo é repetido várias vezes até que a rede seja capaz de produzir saídas precisas para novos exemplos (RAUBER, 2002).

O aprendizado não supervisionado é uma etapa em que a rede neural é treinada com um conjunto de exemplos não rotulados. O objetivo desse processo é encontrar padrões ou estruturas nos dados de entrada sem a necessidade de rótulos. Existem várias técnicas de aprendizado não supervisionado, como o agrupamento *(clustering)* e a redução de dimensionalidade (SINAGA; YANG, 2020). Na Figura 7 é representado a classificação dos principais processos de aprendizagem em IA.

2.5.1 Aprendizado pela regra de Hebb

A aprendizagem pela Regra de Hebb é um processo de aprendizagem não supervisionado em redes neurais artificiais. Essa regra foi proposta pelo psicólogo Donald Hebb em 1949 e é baseada na ideia de que a força da conexão sináptica entre dois neurônios é aumentada se eles são ativados simultaneamente. Se um neurônio A dispara um impulso elétrico e, em

Figura 7 – Classificação estrutural e funcional das redes neurais artificiais



Fonte: Rauber (2002).

seguida, um neurônio B dispara um impulso elétrico, a conexão sináptica entre A e B é fortalecida. Esse processo de fortalecimento da conexão sináptica é chamado de potenciação de longo prazo (MAZURKIEWICZ; ZAMOJSKI, 2002).

A regra de Hebb é utilizada em algumas redes neurais, como as redes de Hopfield, para armazenar e recuperar padrões de entrada. Durante o aprendizado, a rede é exposta a um conjunto de padrões de entrada e a regra de Hebb é utilizada para ajustar os pesos sinápticos entre os neurônios. Quando a rede é apresentada a um novo padrão de entrada, ela é capaz de recuperar o padrão armazenado mais semelhante a esse novo padrão (MAZURKIEWICZ; ZAMOJSKI, 2002).

A aprendizagem pela Regra de Hebb é um processo simples e eficaz de aprendizagem não supervisionado em redes neurais artificiais. No entanto, ela tem algumas limitações, como a tendência de levar a redes superespecializadas e a dificuldade de lidar com dados complexos e de alta dimensionalidade. Por isso, outras técnicas de aprendizagem, como o aprendizado supervisionado e o aprendizado por reforço, são frequentemente usadas em conjunto com a regra de Hebb para melhorar o desempenho da rede (RAUBER, 2002).

2.5.2 Aprendizado pela Regra de Delta

A aprendizagem na regra do delta é um aprendizado supervisionado em redes neurais artificiais. Essa regra de adaptação de pesos é também conhecida como regra de Widrow-Hoff, em homenagem aos seus criadores. Durante o treinamento, a rede neural é treinada com um conjunto de exemplos rotulados. Cada exemplo consiste em um conjunto de entradas e uma saída desejada correspondente. A rede neural produz uma saída para cada exemplo de entrada e o erro entre a saída produzida e a saída desejada é calculado (TAVARES, 2001).

A regra do delta é usada para ajustar os pesos sinápticos da rede neural de forma a minimizar o erro entre a saída produzida e a saída desejada. Essa regra é baseada no gradiente descendente, que é um método de otimização usado para encontrar o mínimo de uma função. Essa é uma das técnicas mais populares de aprendizagem supervisionada em RNA. Ela é usada em várias topologias de redes, como o *Perceptron*, o ADALINE e o *Perceptron* Multi-Camada (RAUBER, 2002).

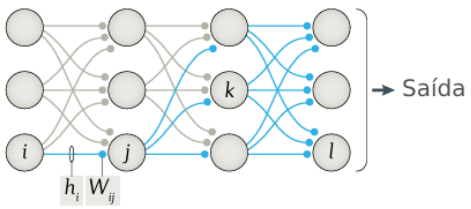
2.5.3 Retro propagação

O algoritmo de retropropagação é um dos mais populares algoritmos de aprendizados em RNAs (LEUNG; HAYKIN, 1991). Ele aprende rapidamente calculando atualizações sinápticas usando conexões de *feedback* para emitir sinais de erro (LILLICRAP *et al.*, 2020).

A abordagem desse algoritmo envolve a utilização da técnica de descida do gradiente para impulsionar o processo conhecido como gradiente descendente. Este método visa reduzir os erros gerados por uma rede neural, comparando a saída desejada com a saída atualmente produzida (HIROSE; YAMASHITA; HIJIYA, 1991).

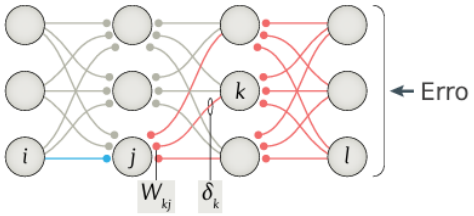
O treinamento de uma RNA utilizando o algoritmo de retropropagação é dividido em 2 etapas. A primeira etapa é utilizando a fase de *feedfoward*, onde os neurônios são inicializados na camada de entrada e ocorre a propagação do sinal de entrada para as camadas ocultas. A segunda etapa utiliza-se a fase *backward* que compara a saída atual com a saída desejada. O problema de aprendizagem consiste em encontrar uma ótima combinação de pesos para que a função da rede se aproxime de uma dada função *f* o mais próximo possível (ROJAS, 1996). Na Figura 8, as informações prévias são representadas pelas setas que fluem da esquerda para a direita, obtendo uma saída (forward). Na Figura 9, os resultados com erros são representadas pelas setas que fluem da direita para a esquerda e vão se ajustando de volta para a rede, combinando para a saída melhor esperada (backward). Onde *k* indica a camada atual onde o cálculo está acontecendo, *l* representa a camada anterior à camada atual, *w* são os pesos das conexões entre neurônios de diferentes camadas e *h* é a saída de ativação de um neurônio, resultado do cálculo com suas entradas.

Figura 8 – *Feedforward* - Movimentação do fluxo de informação das camadas da entradas para a saída



Fonte: Adaptado de Lillicrap *et al.* (2020).

Figura 9 – *Backforward - Erros propamultigados de volta através da rede*



Fonte: Adaptado de Lillicrap *et al.* (2020).

As equações 1 e 2 a seguir apresentam as fórmulas que atualizam os pesos, sendo calculados por equações lineares. O ajuste dos pesos em redes neurais *feedforward* é feito utilizando a taxa de aprendizado n, o valor da entrada x e o erro δ.

$$w_{ij} = w_{ij} + \eta\delta_j x_i \tag{1}$$

$$\epsilon = \frac{1}{2}\sum (y_{l-t_l})^2 \tag{2}$$

2.6 Funções de ativação

Em RNA, as funções de ativação determinam se um neurônio deve ser ativado com base na importância da informação que ele recebe (RIZZO; CANATO, 2020). Em suma, elas são como interruptores que decidem se um neurônio é "ativado" ou "desativado" dependendo da entrada que ele recebe. Essas equações são essenciais para energizar os neurônios, possibilitando a aprendizagem das RNA. São elas que determinam como os neurônios reagem aos diferentes estímulos e, assim, influenciam diretamente o processo de aprendizado.

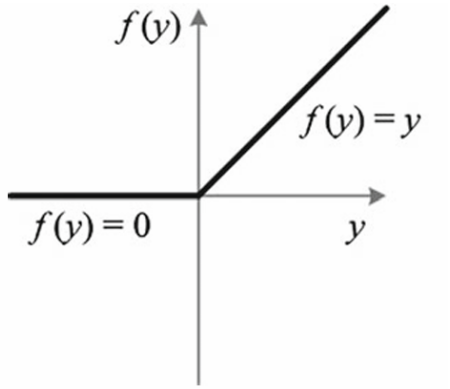
Existem diversos procedimentos para examinar uma rede neural, que vão desde a computação das entradas até a computação das saídas. Um amplo espectro de funções está disponível para ser empregado em redes neurais (BRITO; DAMETTO, 2023). As funções de propagação mais utilizadas são as *ReLU*, Tangente Hiperbólica (Tanh), Sigmóide, Softmax, ELU *(Exponential Linear Unit)*, Limiar (Degrau) e Gelu *(Gaussian Error Linear Unit)*. A escolha da função depende de acordo com as características dos dados e o problema a ser solucionado. Um dos pontos importantes dessas funções são elas obterem uma não linearidade, essa propriedade garante que a rede neural profunda não se degenere em uma operação linear. Sem ela, a rede seria apenas uma série de operações lineares, que não seria capaz de aprender relações complexas entre os dados (ZHU *et al.*, 2021). A seguir, algumas das principais funções são apresentadas.

2.6.1 Função ReLU (Unidade Linear Retificada)

Esta função se baseia na operação dos neurônios cerebrais, os quais retornam um valor positivo ou zero. Esta função de ativação obtêm seu cálculo descrito na Equação 3, onde x é o valor de entrada para essa função, que pode ser um escalar, vetor ou matriz. Para cada valor x da entrada, a função ReLU calcula a saída correspondente e y representa a saída da equação . Ela é representada graficamente pela Figura 10.

$$f(x) = \max(0, x) \tag{3}$$

Figura 10 – Função ReLu



Fonte: (DUBEY; JAIN, 2019).

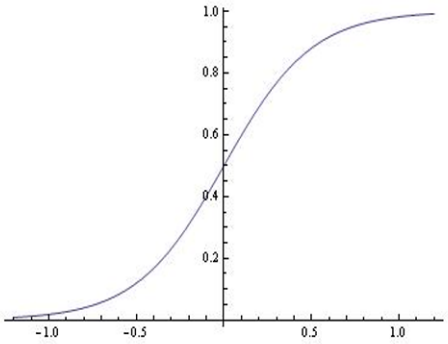
Segundo Snaily (2019), a ReLU tem a vantagem computacional de convergir rapidamente entre as redes neurais. No entanto, por mais que pareça que ela seja uma função linear, não é, pois tem uma função derivada que permite a propagação traseira. Entretanto, quando as entradas se aproximam de zero ou são negativas, o gradiente da função se torna zero impedindo a rede de executar a propagação de retorno.

2.6.2 Função Sigmóide

As funções sigmóides encontram múltiplas aplicações em redes neurais e modelos populacionais de crescimento celular (KYURKCHIEV; MARKOV, 2015). Essa função é representada pela Equação 4 e é representada graficamente pela Figura 11.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{4}$$

Figura 11 – Função Sigmóide



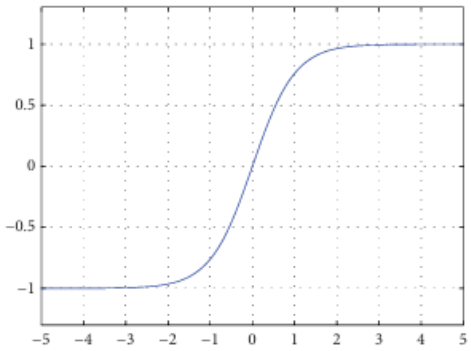
Fonte: (DUBEY; JAIN, 2019)

2.6.3 Função Tanh (Tagente Hiperbólica)

A função tangente hiperbólica tem uma estrutura parecida a função sigmoide. Porém essa função comprime o valor de entrada dentro do intervalo. no valor de $[-1, +1]$. Essa função é descrita pela Equação 5 e representada graficamente pela Figura 12.

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{5}$$

Figura 12 – Função Tangente Hiperbólica



Fonte: (HE *et al.*, 2015).

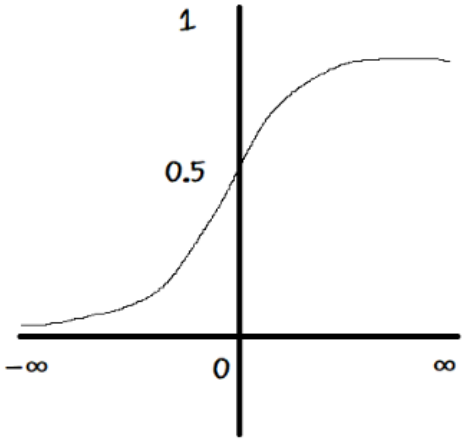
2.6.4 Função Softmax

A função softmax, representada pela Figura13, é amplamente utilizada em aprendizado profundo. Eficientes arquiteturas de hardware da camada softmax são desejadas para acelerar sistemas de aprendizagem profunda, porém a exponenciação e divisão dos cálculos na função softmax são bastante caros, especialmente em sistemas embarcados (WANG *et al.*, 2018). Ela também é frequentemente utilizada na última camada de RNAs de tarefas de classificação, já que representa a probabilidade da saída de cada classe.

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{6}$$

Na Equação 6, e^{z_i} é representada pela exponencial da pontuação associada à classe i. E Z é a soma das exponenciais de todas as pontuações do vetor z, garantindo que a soma das probabilidades de todas as classes seja igual a 1.

Figura 13 – Função Softmax



Fonte: Chiroya (2022).

As funções de ativação são estruturas fundamentais em redes neurais, obtendo-se a capacidade de aprender e modelar relações complexas nos dados. Elas introduzem não-linearidades essenciais para lidar com problemas e garantem a convergência eficiente dos algoritmos de treinamento. As redes neurais utilizam essas funções de ativação em conjunto com algoritmos de otimização para aprender a partir dos dados de treinamento.

2.7 Redes Neurais Recorrentes

A RNN é uma técnica de aprendizado de máquina particularmente útil para lidar com sequências de dados. Elas diferem de outras arquiteturas de redes neurais, como as CNN e as RNAs profundas, devido à sua estrutura única que permite o fluxo de informações em *loops*. Isso significa que a RNN têm a capacidade de manter uma espécie de “memória” dos dados de entrada anteriores ao processar os dados de entrada atuais (GALLICCHIO, 2018). No entanto, a RNN têm suas limitações. Uma delas é o problema do gradiente desvanecente, onde o gradiente da função de erro se torna muito pequeno à medida que é retropropagado através da rede. Isso pode tornar difícil para a rede aprender com dados que estão distantes no tempo. Para superar esse problema, foram desenvolvidas variantes mais avançadas das RNN, como as LSTM (SHINDE, 2018).

Essas arquiteturas avançadas têm mecanismos adicionais (como portões) que controlam o fluxo de informações através da rede, o que ajuda a mitigar o problema do gradiente desvanecente e permite que a rede aprenda efetivamente de sequências de dados mais longas.

2.7.1 LSTM

A *Long Short Term Memory* (LSTM) ou memória de longo curto prazo é uma versão especializada da RNN capaz de superar um dos maiores desafios do aprendizado de máquina: lembrar de informações por longos períodos (RAI *et al.*, 2022). Esse impedimento aborda a tentativa de superar o desafio do gradiente desvanecente. Este problema ocorre quando os gradientes usados para atualizar os pesos da rede durante o treinamento se tornam extremamente pequenos, impedindo a rede de aprender efetivamente.

Essa habilidade se deve à sua arquitetura interna única, composta por módulos repetidos chamados células. Cada célula possui uma estrutura que integra quatro redes neurais interligadas de maneira especial, permitindo que esse algoritmo memorize e utilize informações relevantes por extensos períodos (Figura 14). Para gerenciar o fluxo de informações dentro das células e garantir que apenas as informações relevantes sejam utilizadas, as LSTM empregam três mecanismos especiais chamados portões (RAI *et al.*, 2022) :

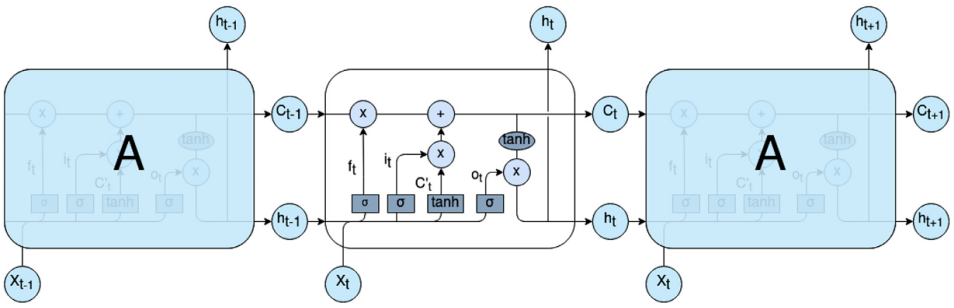
- Portão de Esquecimento f: Atua como um filtro seletivo, decidindo quais informações da memória celular serão "esquecidas" antes da atualização. Ele elimina dados irrelevantes ou desatualizados, otimizando o aprendizado e a memória da rede;

- Portão de Entrada i: Assume o papel de controlar a adição de novas informações à memória celular. Ele seleciona apenas as informações relevantes e as combina com o estado atual da célula, garantindo que apenas os dados mais importantes sejam armazenados;

- Portão de Saída o: Determina quais informações da memória celular serão utilizadas na próxima etapa do processamento. Ele decide quais dados serão expostos para a próxima célula, influenciando o fluxo de informações na rede e garantindo que apenas as informações relevantes sejam utilizadas na tomada de decisões.

O estado da célula (C_t) (Equação 11) em uma unidade LSTM representa a memória atual da unidade no tempo t. Esse estado é atualizado com base em várias entradas e cálculos. O valor de f_t (Equação 8) representa o "esquecimento" na unidade dessa arquitetura. Ele controla o quanto da informação anterior deve ser mantida ou esquecida na célula de memória. Este valor é determinado por uma função sigmoide que varia entre 0 (esquecer completamente) e 1 (manter completamente). Já C_{t-1} é o estado anterior da célula no tempo $t-1$. Ele fornece a

Figura 14 – Arquitetura da LSTM



Fonte: Rai *et al.* (2022).

memória anterior da célula, que é usada em conjunto com a entrada atual para calcular o novo estado da célula. A entrada atual i_t (Equação 7) na unidade desse algoritmo é uma parte dos dados de entrada que está sendo processada no tempo t. Este valor é modulado pela função sigmoide para determinar o quanto de informação nova deve ser adicionada ao estado da célula. Por fim, C'_t (Equação 10) representa a nova célula de memória proposta para o tempo t considerando a saída do portão O_t (Equação 9). Esta é a informação proposta que pode ser adicionada ao estado da célula, com base na entrada atual e no estado anterior da célula.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{7}$$

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{8}$$

$$o_t = \sigma(o_t \cdot [h_{t-1}, x_t] + b_o) \tag{9}$$

$$C'_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{10}$$

$$c_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{11}$$

Esses componentes juntos formam a equação para atualização do estado da célula em uma unidade LSTM, permitindo que a rede aprenda e armazene informações relevantes ao longo do tempo (RAI *et al.*, 2022).

As células LSTM podem ser comparadas aos neurônios do cérebro humano. A memória celular funciona como a memória de longo prazo dos neurônios, enquanto os portões e as funções de ativação representam os processos bioquímicos complexos que ocorrem nos neurônios para processar e transmitir informações.

2.8 *Transformers*

Os modelos de *Transformers* são uma classe de modelos de aprendizado de máquina utilizados em tarefas de PLN (LIN *et al.*, 2022). Eles foram introduzidos em 2017 e desde então se tornaram uma das arquiteturas mais populares para tarefas de PLN, incluindo tradução automática, sumarização de texto, resposta a perguntas e detecção de discurso de ódio (LIN *et al.*, 2022). Esses modelos são baseados em uma arquitetura de rede neural que utiliza atenção para permitir que o modelo se concentre em partes relevantes do texto de entrada durante o processamento. Eles são conhecidos por sua capacidade de capturar relacionamentos de longo alcance entre palavras e produzir resultados de alta qualidade em uma variedade de tarefas de PLN (RöTTGER *et al.*, 2021).

O *Transformer* é formado por duas partes essenciais: o codificador e o decodificador, cada um consiste em uma pilha de *L* com blocos idênticos conforme Figura 15.

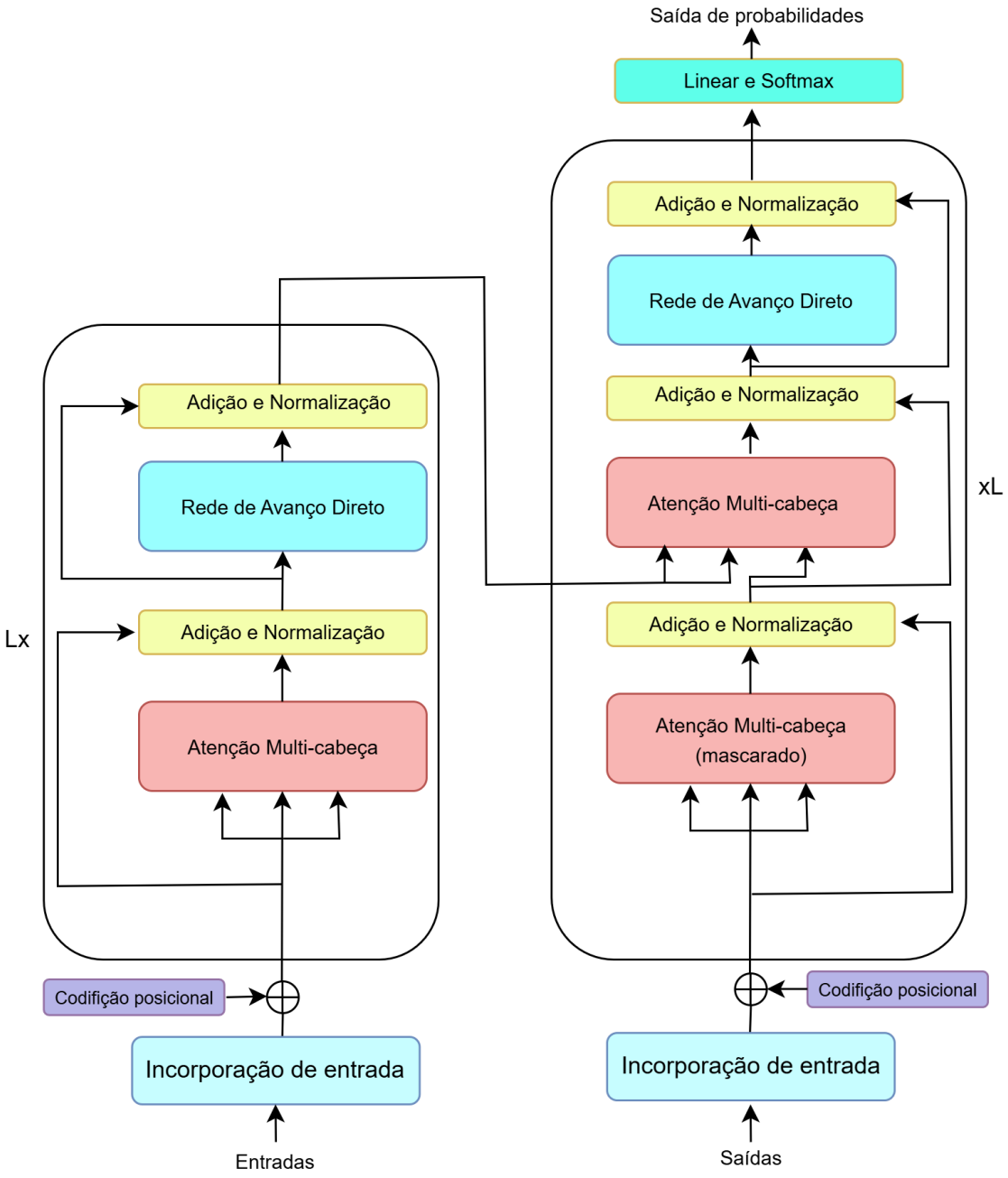
O codificador processa a entrada e cria uma representação latente, na qual essa representação se consiste onde a entrada é transformada em um espaço de características mais compacta e significativa, facilitando a posterior reconstrução ou decodificação da entrada original, sendo assim transmitida ao decodificador posteriormente. Este, por sua vez, produz a saída com base na representação fornecida pelo codificador. O codificador é formado por múltiplas camadas de blocos de auto atenção com vários cabeçalhos e camadas de rede de avanço *(feed-forward*). Cada bloco de auto atenção calcula a atenção entre todas as palavras da entrada, permitindo que o modelo se foque nas partes relevantes da entrada durante a codificação. A camada de avanço aplica uma transformação linear seguida de uma função de ativação não-linear para criar uma nova representação latente. O decodificador é parecido com o codificador, mas também contém uma camada extra de atenção que permite ao modelo focar nas partes relevantes da saída durante a decodificação. Além disso, o decodificador inclui uma camada de máscara de atenção que impede o modelo de acessar informações futuras durante a decodificação (TOPAL; BAS; HEERDEN, 2021).

Os modelos básicos de *Transformers*, frequentemente referidos como *"vanilla Transformers*", estabeleceram os alicerces para inúmeras extensões e variações subsequentes (LIN *et al.*, 2022). Suas partes fundamentais possuem serão descritas na próxima seção.

2.8.1 Módulos de atenção

Essa arquitetura, adota um mecanismo de atenção com o modelo de Consulta-Chave-Valor (QKV). A representação matricial das consultas *Q* é uma matriz que contém informações sobre os tokens (unidades fundamentais de entrada e saída que representam parte do texto) que estão sendo consultados ou considerados para atenção. A matriz de chaves *K* contém informações sobre os tokens que estão sendo usados para calcular a relevância em relação às

Figura 15 – Arquitetura do *Transformer*



Fonte: Adaptado de Lin *et al.* (2022).

consultas e a matriz de valores contém informações detalhadas sobre cada token na sequência (BORJI, 2023).

A atenção é calculada usando o produto escalar entre as consultas e as chaves, o que resulta em uma matriz de pesos que indica a importância de cada token em relação aos outros. Para isso, é utilizada a Equação 12 (LIN *et al.*, 2022).

$$\text{Atenção} = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{12}$$

Essa arquitetura utiliza a atenção *multi-head* para melhor capturar relacionamentos complexos entre tokens em uma sequência. Isso significa que as consultas originais, chaves e valores são projetados em várias dimensões diferentes (denotadas como D,D_k e D_v). Cada conjunto de projeções aprendidas transforma as consultas, chaves e valores originais em representações em dimensões diferentes. Isso permite que o modelo capture uma variedade maior de características em diferentes níveis de abstração. Para cada conjunto de consultas, chaves e valores projetados, a atenção é calculada de acordo com uma função de atenção, resultando em várias saídas de atenção para cada conjunto de projeções (LIN *et al.*, 2022).

As saídas de atenção de todos os conjuntos de projeções são concatenadas e então projetadas de volta para uma representação dimensional original D_m. Essa etapa combina as informações de atenção de todas as cabeças em uma representação abrangente e rica dos relacionamentos entre os tokens na sequência original (LIN *et al.*, 2022). O modelo é de saída é representado pelas Equações 13 e 14, onde cada cabeça de atenção é denotada por $head_i$ e é calculada usando a função de atenção aplicada às projeções das consultas, chaves e valores específicas daquela cabeça. As saídas das várias cabeças de atenção são concatenadas e, em seguida, projetadas de volta para a dimensão original do modelo usando uma matriz de pesos.

$$\text{Atenção Multi-Cabeça}(\mathbf{Q},\mathbf{K},\mathbf{V}) = Concat(head_1,...,head_h)\mathbf{W}^o, \tag{13}$$

$$head_i = Attention(\mathbf{QW_i^Q},\mathbf{KW_i^K},\mathbf{VW_i^V}) \tag{14}$$

No Transformer, há três tipos diferentes de atenção, cada uma com uma abordagem única em termos de como as consultas e os pares de chave-valor são utilizadas (LIN *et al.*, 2022):

- Autoatenção: é implementada definindo as consultas Q, chaves K e valores Vcomo as saídas da camada anterior, ou seja, $Q = K = V = X$, onde X são as saídas da camada anterior. Desse modo, cada token na sequência é usado como consulta, chave e valor em relação a todos os outros tokens, permitindo ao modelo capturar as relações entre eles de forma contextualizada;

- Autoatenção mascarada: a autoatenção é mascarada para garantir que o modelo não viole a causalidade temporal durante a geração de saída. Isso significa que cada posição na sequência só pode atender a tokens anteriores a ela mesma (incluindo ela mesma) durante a geração da saída. Isso é feito utilizando uma matriz de atenção não normalizada, onde $\hat{A}$ representa a matriz de atenção não normalizada, Q é a matriz de consultas, K é a matriz de chaves e D_k é a dimensão dos vetores de chave. A função exp é a função exponencial, que é aplicada elemento a elemento à matriz resultante da multiplicação de Q por K^t dividido pela raiz quadrada de D_k. Isso é visualizado na Equação 15.

$$\hat{A} = \exp\left(\frac{QK^T}{\sqrt{D_k}}\right). \tag{15}$$

- Atenção Cruzada: as consultas são geradas a partir das saídas da camada anterior do decodificador, enquanto as chaves e os valores são gerados a partir das saídas do codificador.

Ao ajustar os parâmetros do módulo de atenção, é possível adaptar o modelo para atender às demandas específicas da tarefa a ser realizada. Enquanto os mesmos são cruciais para capturar relações de longo alcance em dados sequenciais, as redes de avanço direto são responsáveis por processar essas informações em camadas ocultas para aprender representações mais abstratas e complexas dos dados.

2.8.2 Rede de Avanço Direto

A rede de avanço direto, é um módulo *feedforward* totalmente conectado que opera separadamente e de forma idêntica em cada posição. Ela é representada pela Equação 16 (LIN *et al.*, 2022).

$$FFN(\mathbf{H}' = ReLU(\mathbf{H}'\mathbf{W^1} + \mathbf{b^1})\mathbf{W^2} + \mathbf{b^2} \tag{16}$$

Onde H' representa a saída da camada anterior, W_1 e W_2 são as matrizes de pesos, enquanto b_1, b_2 são os vetores de bias.Esses parâmetros são treinados durante o processo de treinamento do modelo.

2.8.3 Codificação posicional

Mecanismos adicionais são necessários para injetar informações posicionais nos transformadores. No Vanilla, as informações posicionais são codificadas usando codificações de posição sinusoidal absoluta. Isso significa que, em vez de usar uma representação absoluta de posição para cada token na sequência, o modelo utiliza uma representação relativa baseada em funções senoidais (seno/cosseno) do índice de posição.

Com o foco de utilizar a arquitetura do *Transformers*, ela pode ser aplicada de algumas maneiras segundo Lillicrap *et al.* (2020). Na modelagem de sequência para sequência, emprega-se o codificador-decodificador, como na tradução automática neural. Em certos contextos, apenas o codificador é empregado. Nessa abordagem, as saídas do codificador servem como representação para a sequência de entrada. Isso é comumente aplicado em tarefas de compreensão de linguagem natural, como classificação de texto e rotulagem de sequência. Por outro lado, em algumas situações, apenas o decodificador é utilizado, excluindo o módulo de atenção cruzada do codificador/decodificador. Essa técnica é utilizada na geração de sequências, como na modelagem de linguagem (LIN *et al.*, 2022).

2.8.4 Conexão residual e normalização

A arquitetura do *Transformers* obtêm uma conexão residual (HE *et al.*, 2023). Seguido por uma normalização de camada. Cada bloco de codificador pode ser escrito como na Equação 17 (LIN *et al.*, 2022). Onde a entrada x é um tensor que representa a sequência de entrada para o bloco do *transformer*. Este tensor pode conter *embeddings* de palavras. A camada de autoatenção é aplicada ao resultado da soma residual $(x + x)$, capturando dependências contextuais dentro da sequência. A normalização de camada representa a saída da camada de autoatenção, e é normalizada para estabilizar o treinamento e melhorar a convergência. O parâmetro h refere-se ao número de cabeças de atenção.

$$\mathbf{H}' = NormalizacaoCamada(AutoAtencao(\mathbf{X} + \mathbf{X})) \tag{17}$$

Uma das maiores vantagens do *Transformer* é a sua capacidade de capturar informações de contexto de longo alcance, o que é extremamente útil em tarefas de processamento de linguagem natural. Além disso, o *Transformer* pode ser paralelizado, o que permite que ele seja treinado de forma eficiente em grandes conjuntos de dados (WANG, 2021).

2.8.5 BERT

Em 2018, foi introduzido uma arquitetura chamada *Bidirectional Encoder Representations from Transformers* (BERT). Ele é um modelo de linguagem pré-treinado que utiliza uma arquitetura de transformador para aprender representações contextualizadas de palavras. Esse modelo é treinado em um grande corpus de dados de texto e pode ser ajustado para várias tarefas posteriores de PLN, como resposta a perguntas e classificação de texto (MULLAH; ZAINON, 2021). O BERT em seu estado atual da arte, apresenta um desempenho de última geração em diversas atividades de PLN (MÜLLER; SALATHÉ; KUMMERVOLD, 2023).

Durante a fase de pré-treinamento, o BERT possui dois estágios distintos: o pré-treinamento e o ajuste fino (Figura 16). No processo de pré-treinamento, o modelo é exposto a um grande conjunto de dados sem rótulos. Para o ajuste fino, ele começa com os parâmetros aprendidos durante o pré-treinamento e então esses parâmetros são otimizados usando dados rotulados para tarefas específicas (DEVLIN *et al.*, 2019). Esse modelo obtêm um processo de inicialização da representação de frases ou pares de frases seguindo as etapas (KENTON; TOUTANOVA, 2019):

- Tokenização: o BERT quebra as frases em tokens individuais. Ele utiliza o método de tokenização *WordPiece*, que divide as palavras em partes menores. Por exemplo, a palavra "pessoas" pode ser dividida em "pes" e "soas";

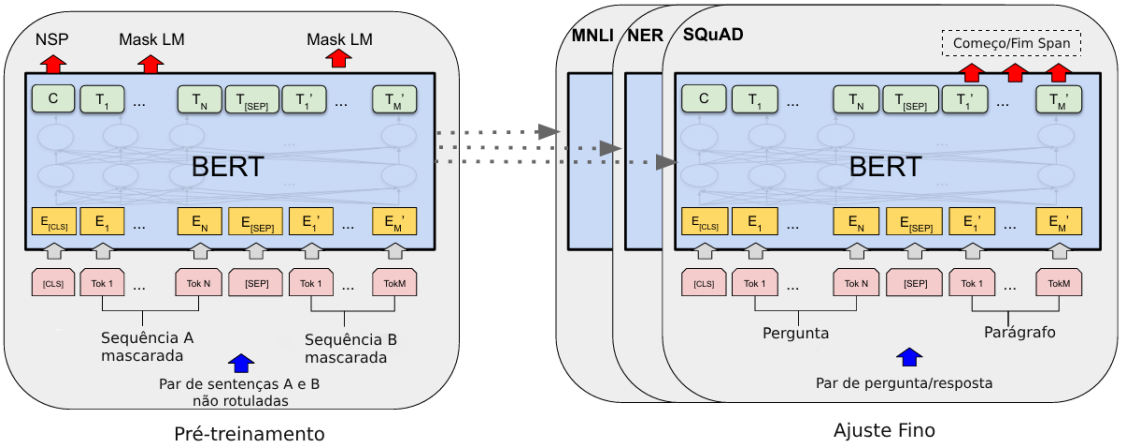
- Estrutura da Sequência: cada frase ou par de frases é representado como uma sequência de tokens. Se for um par de frases, essas frases são separadas pelo token especial;

- Tokens Especiais: adiciona tokens especiais ao início e entre as frases. O primeiro token é sempre "[CLS]", que é usado para representar a classificação da sequência (por exemplo, se é uma pergunta, uma afirmação, etc.). Os tokens "[SEP]" são usados para separar as frases em pares;

- Incorporação de Segmento: para distinguir entre as diferentes frases em um par, BERT adiciona uma incorporação de segmento a cada token, indicando se ele pertence à primeira frase ou à segunda. Isso ajuda o modelo a entender a estrutura e a relação entre as frases;

- Representação da Palavra: a representação final de cada token é construída combinando suas informações de tokenização, posição na sequência e incorporação de segmento. Isso cria uma representação rica e contextual de cada palavra na frase ou no par de frases, que o modelo BERT utiliza para realizar tarefas de PLN.

Figura 16 – Arquitetura do BERT



Fonte: Adaptado de Devlin *et al.* (2019).

No pré-treinamento ocorre a divisão em duas tarefas principais (DEVLIN *et al.*, 2019):

- *Masked Language Model* (MLM) ou Modelo de linguagem mascarada: Nesta etapa, uma certa porcentagem dos tokens de entrada é aleatoriamente mascarada, ou seja, substituída pelo token especial "[MASK]". Em seguida, o modelo é treinado para prever esses tokens mascarados com base no contexto das outras palavras na frase;

- *Next Sentence Prediction* (NSP) ou Predição da próxima sequência: Nesta tarefa, são fornecidas duas sentenças, chamadas de A e B. Metade das vezes, a sentença B é, de

fato, a próxima frase que segue logicamente a sentença A. Isso é rotulado como *"IsNext"* (é a próxima). Um exemplo, se a sentença A seja algo como "Todas as pessoas merecem respeito". Se a sentença B continuar nesse mesmo contexto, como "Independentemente de sua raça, religião ou orientação sexual", então ela seria rotulada como *"IsNext"*, indicando que é uma continuação lógica e respeitosa da primeira sentença. Por outro lado, metade das vezes, a sentença B poderia ser uma frase selecionada aleatoriamente do *corpus* de texto. Nesse caso, a sentença B pode conter conteúdo ofensivo, como "Alguns grupos étnicos são inferiores e merecem discriminação". Essa combinação seria rotulada como *"NotNext"*, pois não é uma continuação natural ou respeitosa da sentença A pois promove o discurso de ódio.

O BERT estabeleceu uma base sólida para o desenvolvimento de modelos de linguagem cada vez mais sofisticados. Sua arquitetura baseada em *Transformers* e seu treinamento em grandes quantidades de texto sem supervisão o tornam altamente adaptável a diferentes domínios e tarefas (DEVLIN *et al.*, 2019). O modelo RoBERTa é uma extensão do BERT que visa melhorar seu desempenho através de otimizações, como o aumento do conjunto de dados de treinamento e o uso de técnicas de pré-processamento mais avançadas, resultando em modelos mais robustos e precisos para uma variedade de tarefas de PLN (LIN *et al.*, 2022).

2.8.6 Roberta

O RoBERTa é um variante da arquitetura *transformer* do BERT (LIU *et al.*, 2019). Estudos recentes demonstraram que essa arquitetura ultrapassa consideravelmente o modelo padrão em otimização dos parâmetros sem modificar a estrutura da rede neural e com isso, vem se consolidando como uma das melhoras estruturas pré treinada no estado atual da arte (TIAN *et al.*, 2020). A utilização desse modelo promove as seguintes modificações:

- Remoção de NSP na predição da próxima palavra;
- Treinar o modelo por mais tempo;
- Treinar o modelo com o dados maiores;
- Aumento de *batches*, épocas, codificador de par de bytes *Byte Pair Enconder* (BPE) e taxas de aprendizado.

Tabela 1 – Comparação dos desempenhos dos modelos BERT e RoBERTa

Modelo	Dados	Batch Size	Passos	SQuAD (v1.1/2.0)	MNLI-m	SST-2
RoBERTa	160GB	8k	500K	94,6/89,4	90.2	96.4
BERT	13GB	256	1M	90,9/81,8	86.6	93.7

Fonte: Adaptado de Liu *et al.* (2019).

Na Tabela 1 pode ser visualizdo a comparação dos desempenhos das duas arquiteturas e foi observado uma notável melhoria nos resultados da arquitetura otimizada. São analisados os resultados em porcentagens de três modelos de bases de dados distintas, SQuAD, MNLI-m e SST-2.

As escolhas das alterações dos parâmetros da arquitetura fizeram esse modelo se sobressair em comparação com as estruturas básicas do BERT (LIU *et al.*, 2019).

2.9 Trabalhos Relacionados

Pulver e Lyu (2017) abordaram experimentos que compararam 2 arquiteturas: *LSTM* e a LSTM modificada. Para isso, envolveram diferentes tarefas, como reconhecimento de texto e classificação de números através de imagens enviadas. Foram testadas diferentes arquiteturas de rede, incluindo variações na largura das camadas e nas funções de ativação. Uma das principais tarefas desse estudo foi o reconhecimento de texto, para isso foi utilizado a base de dados chamada *"hunter challenge"*, que obtêm um conjunto de texto em Linguagem de Marcação Extensível (XML) da Wikipedia. Os primeiros 95% dos dados foram usados como um conjunto de treinamento e os últimos 5% para teste. A taxa de erro foi medido em bits por caractere (BPC). Também foi utilizado um *batch size* de 32, um tipo de *batch size* determina o número de exemplos de treinamento que são processados em paralelo pela rede neural em cada iteração durante o treinamento. Para reduzir a quantidade de tempo de treinamento necessário, foram utilizados comprimento 200 sequências para a primeira época e depois foi utilizada o comprimento 2.000 de sequências para as cinco épocas restantes. Os resultados o LSTM modificada assumiu uma ligeira vantagem, obtendo uma performance de menor tempo para reconhecer a próxima sequencia de caractere, com um mínimo de bits por caracteres 1725 até 1892, em comparação com a LSTM padrão com o mínimo de 1742 até 1893 bits.

Cao, Lee e Hoang (2020) compartilham um novo modelo de aprendizado profundo, chamado *DeepHate*, sendo composto por 3 componentes principais: Modelo de *Embeddings* de Palavras, Alocação Latente de Dirichlet (LDA) para descobrir automaticamente tópicos ocultos em um conjunto de documentos e Modelo de Recorrência Bidirecional (Bi-LSTM) combinando com representações de texto multifacetadas, como incorporação de palavras, sentimentos e informações atuais, para detectar discurso de ódio em plataformas sociais online. Para analisar os dados foram utilizados 3 bases disponíveis publicamente, WZ-LS, DT e o FOUNTA. Para a limpeza dos dados, foram banidos de publicações de spam e ocorreu a separação experimentos de 80% das informações foram utilizadas para treinamento e 20% usados para testes. Para as métricas de avaliação foi utilizada precisão de micro média (Micro-Prec), recall (Micro-Rec) e pontuação F1 (Micro-F1), O modelo DeepHate obteve seu melhor desempenho no conjunto de dados chamado *COMBINED* que obtêm agregação de todos os publicações considerados inadequados, incluindo publicações ofensivos e de ódio, juntamente com publicações normais. Obtendo uma Micro-Precisão de 92,48%, Micro-Recall de 92,45% e Micro-F1 de 92,43%.

Ayo *et al.* (2020) avalia a uma arquitetura genérica para classificação de discurso de ódio na rede social X para resolver problemas com fluxos de dados nessa plataforma. Os testes foram realizados utilizando técnicas de incorporação de palavras *(word embeddings)* e arquiteturas de aprendizado profundo, como CNN e LSTM. Além disso, foram empregadas técnicas de extração de características e classificação binária par. Observou-se que a arquitetura genérica de metadados desenvolvida teve melhor desempenho em todas as avaliações métricas para detecção de fala de ódio tendo 0,95%, 0,93%, 0,92% e 0,93% para exatidão, precisão, re-

call e Pontuação F1, respectivamente, quando comparado a métodos semelhantes. Da mesma forma, os metadados genéricos desenvolvidos arquitetura para classificação de sentimento de discurso de ódio teve melhor desempenho com pontuação F1 de 91,5% em comparação para métodos relacionados. A arquitetura de metadados genéricos desenvolvida também indica um teste mais perfeito tendo uma AUC (Área sob a curva ROC) de 0,97%, quando comparado a métodos semelhantes. A validação estatística dos resultados aponta a eficiência do sistema desenvolvido. Por fim, os resultados também mostraram que o sistema desenvolvido é muito bom para detecção e categorização automática de tópicos.

Alam, Khan e Alam (2020) avaliam a utilização de modelos o uso de modelos *Transformers* para classificar textos na linguagem Bangla em diferentes áreas, como análise de sentimentos, detecção de emoções, categorização de notícias e atribuição de autoria. Foram utilizados 6 conjuntos de dados de diversas fontes, abrangendo vários tópicos. Os modelos foram treinados com o critério de perda de entropia cruzada e o algoritmo de otimização Adam por 10 épocas. Os modelos *Transformer* apresentaram melhorias de 5% a 29% na precisão em comparação com métodos tradicionais de aprendizado de máquina e modelos CNN/LSTM. O modelo xlm-RoBerta-large se destacou com a maior precisão de 30%, enquanto o bert-base teve um dos piores resultados 0,25% de precisão. Conclui-se que os modelos *Transformer* são mais adequados para classificar textos em Bangla do que as abordagens tradicionais.

Gao *et al.* (2021) avaliam a comparação de 4 métodos de adaptação das linguagens BERT, CNN e *Hierarchical Self-Attention Network (HSAN)*. Esses métodos incluem: o número máximo de tokens, *pooling* máximo sobre confiança bruta, alvo de atenção, e atenção multi rótulo. Para os dados desse estudos, foram utilizadas 2 base de dados, o MIMIC-III e o SEER *Cancer Pathology*. Esses dados foram previamente limpos inserindo os textos em mínusculos, removendo símbolos unicode e substituindo valores hexadecimais em inteiros. Também foi feita uma otimização dos parâmetros utilizando um conjunto de validação de cada conjunto de dados. Outra estratégia utilizada foi alterar um único parâmetro por vez e treinado novamente até que o desempenho parasse de melhorar. Os resultados mostraram que o modelo BERT performou igual ou pior em comparação com o glshsan *baseline*, o mesmo ocorreu com o CNN. Na base de dados MIMIC-III, com a limitação de 510 peças de palavras, BERT foi o melhor na métrica de recall para a tarefa de código de procedimento, quando foi utilizado o tamanho de todo o documento o BERT performou melhor na métrica de recall na tarefa de diagnóstico de categoria. Porém em comparação com os outros modelos, o CNN padrão demostrou melhor desempenho de precisão entre 0,70% e 0,75%. Em outra base de dados de reporte de doenças de câncer, o modelo BERT não foi estatisticamente melhor do que o modelo HSAN em nenhuma das 6 tarefas que obteve as maiores acurácias, obtendo a acurácia de comportamento em 0,975%. Por fim também foi avaliado o tempo em segundo para prever 1000 documentos inteiros, no qual o BERT foi mais lento (75 mil segundos) que o CNN (entre 8 mil e 18 mil segundos) e o HSAN (média entre 8 mil segundos).

Röttger *et al.* (2021) fornecem um estudo de um desenvolvimento e avaliação de um suite de testes funcionais chamado HateSpeech. Nele foi utilizado modelos de aprendizado basedo em Transformers, como o BERT *fine-tuned on binary* e o BERT *fine-tuned* e a interface de programação de aplicações chamada *perspective*, que utiliza modelos de aprendizado de máquina para avaliar o conteúdo das mensagens e identificar potenciais problemas, como discurso de ódio, assédio, linguagem tóxica, entre outros. Para os testes foram utilizadas 2 bases de dados: o Davidison contendo 24.783 publicações anotados como odiosos ou ofensivos ou nenhum dos dois e o FOUNT que registra 99.996 publicações anotados como odiosos, abusivos, spam e normal. Também foram divididos os conjuntos de dados utilizando uma divisão estratificada de treinamento/desenvolvimento/teste 80/10/10%. Foram avaliados o desempenho do modelo no HateCheck usando precisão, ou seja, a proporção de classes corretas, casos de teste especificados. Esse modelo, revelou fraquezas funcionais em todas as arquiteturas testadas. Nos resultados, o modelo BERT*fine-tuned on binary* obteve um precisão inferior de 50%em 8 dos 11 testes funcionais e o BERT *fine-tuned* obteve uma precisão inferior de 50% em 4 dos 11 testes funcionais para conteúdo não odioso e também tiveram baixo desempenho em casos de termos pejorativos resgatados, discursos de ódio negados e contra discursos. Em contra partida, o modelo Perspective obteve um desempenho superior aos modelos anteriores na maioria dos testes funcionais com precisão acima de 95% em 11 testes.

Alammary (2022) propõe a investigação e comparação das diferenças de modelos árabes de *BERT* e modelos originais em inglês para classificação de texto. Foram utilizados 9 modelos de aprendizado de máquina. Os conjuntos de dados incluíam publicações extraídos da rede social X, posts de outras plataformas de mídia social, avaliações de hotéis e livros, artigos classificados por assunto, entre outros. O BERT multilíngue foi pré-treinado em várias línguas com um grande conjunto de dados, ajustados e organizados. Os modelos árabes foram pré-treinados em dados de texto com linguagem pátria. Na maioria das 66 comparações realizadas nos estudos incluídos, modelos BERT árabes mostraram desempenho superior para classificar texto árabe em relação aos outros modelos de aprendizado de máquina. Considerando os resultados de desempenho de todos os 9 modelos, eles foram classificados em baixo modelos de desempenho (aqueles com pontuação F1 média < 0,7) e modelos de alto desempenho (aqueles com pontuação F1 média de 0,7). Os modelos BERT multilingue, XLM-RoBERTa, árabe ALBERT e GigaBERT tiveram um baixo desempenho. Os dois primeiros melhores modelos foram pré-treinados para que possam suportar mais de 100 idiomas cada e tinham pequenos corpus de pré-treinamento em árabe que afetavam seu desempenho. O mesmo se aplica ao árabe ALBERT, que foi pré-treinado em um corpus um pouco menor em comparação com os modelos de alto desempenho e obteve bons resultados sendo capaz de lidar com uma variedade de tarefas em árabe, mostrando competência em diferentes aspectos da linguagem, como análise de sentimentos, identificação de entidades, tradução automática. Em contrapartida, o desempenho do GigaBERT foi considerado inferior em termos de acurácia e eficácia em algumas tarefas específicas de classificação de texto em árabe.

Balkus e Yan (2023) discorrem sobre a avaliação de dois algoritmos principais: o algoritmo genético para seleção de melhores exemplos para o *Generative pre-trained transformer (GPT)* 3 *Completion Endpoint* com um conjunto de treinamento ideal escolhido usando um algoritmo genético e o GPT 3 *Classification Endpoint* com exemplos aumentados. Para treinar o modelo, coletaram um conjunto de dados de textos curtos da Universidade de Massachussets Dartmouth. O conjunto de dados continha 78 perguntas rotuladas anteriormente como “dados” ou “outros” para indicar se o tema estava relacionado à ciência de dados. Essas questões foram divididas em três conjuntos. Foram constados que o aumento dos dados melhorou significativamente a precisão de ambos os classificadores porém o GPT 3 *Classification Endpoint* atingiu a melhor precisão de cerca de 76%, em comparação com a precisão humana de 85%. Dessa forma, modelos de linguagem grandes, como o GPT 3, obtêm a capacidade de propor seus próprios exemplos de treinamento melhorando o desempenho da classificação de textos curtos.

3 MATERIAIS E MÉTODOS

Neste capítulo são apresentadas as descrições de hardware, software e também a metodologia utilizada nos procedimentos para alcançar o objetivo deste trabalho.

3.1 Equipamentos

O computador utilizado para o treinamento e comparação das técnicas é um notebook Dell. Ele é equipado com um processador *Intel Core I7-13650HX*, que possui 14 núcleos de processamento e um *clock* de 2.6GHz. Além disso, conta com 16GB de memória RAM e um SSD de 512 GB. A placa de vídeo é uma *NVIDIA GeForce RTX 3050*, que possui 6GB de memória. O sistema operacional utilizado é o Windows 11, que foi empregado para realizar pequenos testes, preparação do banco de dados e desenvolvimento do modelo de treinamento.

Para o ambiente de treinamento dos modelos foi utilizado o Google Collaboratory[1], esse serviço fornece uma plataforma que permite a criação e execução de códigos Python diretamente no navegador. No contexto de aprendizado de máquina, o Google Collaboratory oferece a capacidade de carregar um conjunto de dados, treinar dados classificados e avaliar a eficácia do modelo. Os documentos iterativos do Google Collaboratory são hospedados nos servidores em nuvem do Google, permitindo que seja utilizado o hardware do Google, incluindo as Unidades de Processamento Gráfico (GPUs) e Unidades de processamento de tensores (TPUs), independentemente da capacidade do computador utilizado. Além de permitir a hospedagem de diferentes máquinas de aprendizado de cada vez e com armazenamento em nuvem, também pode ser vinculado ao Google Drive [2].

3.2 Software

As especificações dos softwares utilizados estão na Seção 3.2.1, *frameworks* e bibliotecas na Seção 3.2.2 e o banco de dados na Seção 3.2.3.

3.2.1 Linguagem de Programação

Python[3] foi a linguagem de programação escolhida para o desenvolvimento deste trabalho. Ela é uma das linguagens de programação mais importantes para a ciência dos dados e possui um grande número de bibliotecas úteis desenvolvidas por sua comunidade (RASCHKA; MIRJALILI, 2017). Dentre as razões que levaram à seleção desta linguagem:

[1] https://colab.research.google.com/
[2] https://drive.google.com/
[3] https://www.python.org/

- Simplicidade e eficiência: Python é conhecido por suas ferramentas simples de utilização e eficientes para a extração dos dados. Também possui uma linguagem simples e de fácil compreensão;

- Interoperabilidade: é capaz de se integrar com outras linguagens de programaço Java, C# , e Python. Também com ferramentas e bibliotecas, como o NumPy [4], o SciPy[5] e o Matplotlib[6];

3.2.2 Bibliotecas e Frameworks

O PyTorch[7] foi utilizado para o treinamento dos modelos de aprendizado. Ele é uma biblioteca de aprendizado de máquina profundo, que fornece uma plataforma para a pesquisa e desenvolvimento de PLN. As arquiteturas utilizadas foram a de *Transformers* em comparação com a LSTM. Para a utilização de ambos os modelos foi empregada a técnica de ajuste fino, esse modelo destaca o ajuste do modelo pré-treinado em tarefas específicas de reconhecimento de texto pode economizar tempo e recursos de treinamento.

Em conjunto ao PyTorch, foi utilizada a biblioteca Pandas[8] que possui código aberto, fornece estruturas de dados de alto desempenho e ferramentas de análise de dados para a linguagem de programação Python. O Pandas é utilizado para analisar dados e é capaz de ler vários formatos de arquivo, como CSV, JSON e Excel. Outra biblioteca utilizada foi a Matplotlib[9], utilizada para a criação de gráficos em Python.

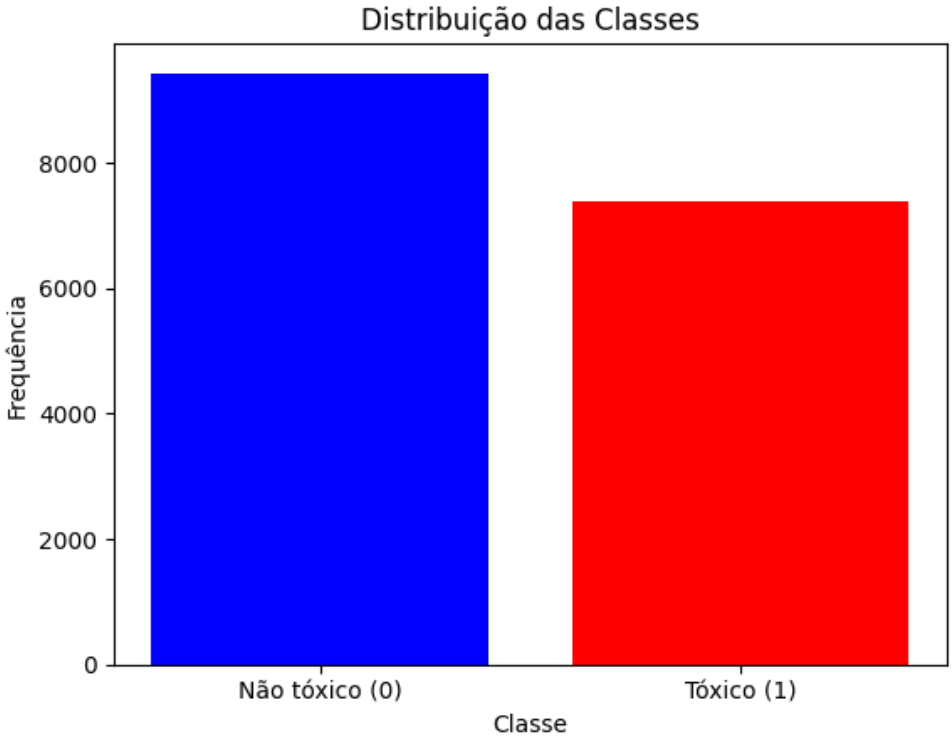
3.2.3 Banco de dados

Um banco de dados desempenha um papel fundamental no treinamento, avaliação e uso eficaz de modelos de aprendizado de máquina. Para treinar modelos LSTM e *Transformer* é necessário um grande conjunto de dados (tweets) de treinamento que seja relevante para a tarefa específica. Para este trabalho foi utilizado um banco de dados público, o told-br [10]. Esse conjunto de dados é considerado o maior de seu tipo e foi criado através de *crowdsourcing*, ou seja, esse método indica que uma tarefa específica seja distribuída para uma comunidade ou um público em geral. Essa coleta dos dados envolveu a contribuição de 42 anotadores cuidadosamente selecionados a partir de um grupo de 129 voluntários. O processo de seleção

[4] https://pytorch.org/docs/stable/generated/torch.Tensor.numpy.html
[5] https://pytorch.org/tutorials/advanced/numpy$_{e}xtensions_{t}utorial.html$
[6] https://matplotlib.org/
[7] https://pytorch.org/
[8] https://pytorch.org/tutorials/beginner/basics/data_tutorial/
[9] https://matplotlib.org/
[10] https://huggingface.co/datasets/told-br

dos anotadores foi projetado para refletir uma diversidade demográfica abrangente, incluindo diferentes etnias, orientações sexuais, faixas etárias e gêneros.

Figura 17 – Adaptação do told-br - Distribuição de conteúdo de texto na configuração binária



Fonte: Autoria própria (2025).

O conjunto de dados possui 2 tipos de configuração. A configuração binária que divide o cojunto em 2 classes: *text* (conteúdos de texto) e *label* 0 (tóxica) ou 1 (não tóxica). Essa configuração possui 16.800 conteúdos de textos nos dados de treinamento e essa distribuição é fornecida pela Figura 17

Tabela 2 – Distribuição de Tipos de Discurso

Tipo de Discurso	0	1	2	3
Homofobia	20.656	168	102	74
Linguagem Obscena	14.348	4.249	1.791	612
Insultos	16.615	2.516	1.352	517
Racismo	20.862	105	27	6
Misoginia	20.537	330	104	29
Xenofobia	20.849	109	27	15

Fonte: Autoria própria (2025).

A Tabela 2 fornece a distribuição da configuração do banco de dados no formato *multilabel*, obtendo 6 classes, sendo elas: homofobia, obsceno, insultos, racismo, misoginia e xenofobia que identifica os conteúdos de ódio em valores de texto com o total de 21 mil publicações. Instâncias que são tóxicas obtêm votos no valor de 0 a 3 nas classes específicas que foram votadas

com aquele tipo de conteúdo pelos anotadores (profissionais que rotulam os dados para garantir a qualidade e precisão das anotações), a maioria dos tweets obteve 0 votos. Por ser uma avaliação única possui um viés individual, sendo assim, outras publicações foram escolhidas por dois ou três anotadores proporcionando uma maior confiabilidade e consistência, permitindo uma avaliação mais equilibrada, pois as divergências podem ser discutidas e resolvidas, aumentando a precisão dos resultados. Nesse contexto do conjunto de dados, um conteúdo de texto pode ser multiclasse, ou seja, pertencer a vários discursos de ódio simultaneamente. Por fim, esse banco de dados é dividido em oitenta por cento de instâncias para treinamento, dez por cento de instâncias para validação e dez por cento de instâncias para testes, esses valores de porcentagens foram definidos por serem valores padronizados de treinamento de conjunto de dados e definidos pelos criadores do conjunto. A razão para essa separação é garantir a robustez, avaliar o desempenho e evitar o *overfitting* (captação de de ruídos e flutuações aleatórias nos dados) dos modelos de PLN.

3.3 Método

Nesta seção é apresentado o método empregado para a realização deste trabalho. O fluxo do trabalho está delineado na Figura 18, e cada etapa foi explicada por passos na ordem em que aparecem no diagrama, bem como a forma que serão implementadas.

3.3.1 Passos da metodologia utilizada para analisar os modelos

O Diagrama da Figura 18 contêm as etapas que serão inciadas com o pré-processamento dos dados, que inclui a remoção de ruídos, tokenização e normalização.

Em seguida, o conjunto de dados pré-processados está pronto para o uso. No treinamento, é utilizado o LSTM, que captura dependências temporais, e o *transformer*, com o RoBERTa, que utiliza atenção para analisar dependências de longo alcance. A avaliação é feita usando métricas como acurácia e F1-score, comparando previsões com os rótulos reais. Na comparação das técnicas, foi avaliada o desempenho, tempo de treinamento e complexidade dos modelos, identificando qual é mais eficaz para os dados específicos.

Etapa 1: Pré processamento da Base Dados: esta etapa consiste dos seguintes procedimentos:

- Remoção de caracteres indesejados (ex: *tags* de *Hypertext Markup Language* (HTML), emojis, espaços em branco, quebras de linhas, normalização do texto (letras minúsculas), links, menções, pontuação excessiva e etc...);
- Tokenização: O texto é dividido em palavras ou subpalavras, que são chamadas de tokens. Cada token é então mapeado para um número inteiro único, que é usado para representar o token no modelo;

- Criação de Tensores: Os tokens e as máscaras de atenção são então convertidos em tensores, que são a estrutura de dados básica usada pelo PyTorch.

Etapa 2: Configuração Binária

A configuração binária das arquiteturas LSTM e RoBERTa foi projetada para classificar texto em uma das duas categorias: "Tóxico" (1) ou "Não tóxico" (0). Isso foi alcançado por meio do fine-tuning do modelo pré-treinado, ajustando-o para tarefas de classificação binária, o modelos foram adaptados para classificação binária ao definir o número de *labels* igual a 2, indicando as duas classes.

Etapas 3: Treinamento do modelo LSTM.

Após a limpeza o modelo está pronto a ser treinado. Para a realização do ajuste da arquitetura todas as camadas do modelo ficaram ativas e suas respectivas variáveis de aprendizado (os pesos e os viéses), sendo atualizadas durante o treinamento. Foram variados durante o treinamento os seguintes hyperparâmetros:

- Definição do intervalo de valores dos hiperparâmetros: Os valores numéricos a seguir foram escolhidos com base em trabalhos anteriores que demonstraram bons desempenhos no treinamento de modelos LSTM:
  - Número de épocas de treinamento: É a quantidade de vezes que o algoritmo de aprendizado de máquina percorre todo o conjunto de dados durante o treinamento do modelo. Inicialmente foi implementado o valor de 100 épocas, porém ocorreu o overtting, após analisar outros trabalhos específicos para esse tipo de problema, ocorreu a diminuição dos valores das épocas, foi obtido melhores resultados entre 25, 28 e 30 épocas do tipo binário;
  - Tamanho do lote *(batch size)*: Refere-se ao número de exemplos de treinamento que são usados em uma única iteração de treinamento. Ocorre a divisão dos dados em lotes menores e atualização dos pesos após cada lote. Os melhores resultados foi ocorreu com os valores de 128 e 200. De acordo com os trabalhos relacionados esse valores foram utilizados nesse tipo de arquitetura para predição em texto;
  - Taxa de aprendizado *(learning rate)*: Controla o tamanho das atualizações feitas nos pesos do modelo durante o treinamento. Foi utilizado o algoritmo de otimização ADAM (*Adaptive Moment Estimation*) [11], esse algoritmo se sobressaiu no resultados em relação ao SGD (*Stochastic Gradient descent*) [12]. Essa ferramenta de otimização do algoritmo funciona de forma iterativa, buscando minimizar uma função de perda, que representa o erro entre a saída prevista do modelo e os valores reais de destino. Ao minimizar a função de

[11] https://pytorch.org/docs/stable/generated/torch.optim.Adam.html
[12] https://pytorch.org/docs/stable/generated/torch.optim.SGD.html

perda, esse algoritmo guia o modelo para um melhor desempenho. Inicialmente foi implementado o valor de 0,01, porém, comparando esse valor com outros trabalhos, foi inserido e analisado o valor de 0,002 e a acurácia obteve uma melhora signitiva;

- Definição da função perda *(loss function)*: utilizada com uma função de ativação sigmóide na última camada do modelo para produzir probabilidades de classe binárias. O cálculo da perda refere-se ao processo de calcular uma métrica que quantifica o quão bem o modelo está performando em prever os rótulos verdadeiros do conjunto de dados. Esse parâmetro foi inserido inicialmente em 0,01 para primeiras análises e fixado posteriormente com o valor de 0,0001 baseado na média de trabalhos anteriores que utilizaram esse algoritmo.

Etapa 4: Avaliação do modelo LSTM.

A avaliação do modelo proposto neste trabalho foi realizada por meio de métricas padronizadas para modelos de IA, com o objetivo de garantir uma análise do desempenho. Segundo Mariano (2021), as principais métricas são:

- Acurácia: Representa a proporção de previsões corretas em relação ao total de previsões, conforme indicado na Equação 18:

$$\text{Acurácia} = \frac{TP + TN}{TP + TN + FP + FN} \tag{18}$$

  - TP é o número de verdadeiros positivos,
  - TN é o número de verdadeiros negativos,
  - FP é o número de falsos positivos,
  - FN é o número de falsos negativos.

- Precisão: Mede a proporção de previsões positivas corretas em relação ao total de previsões positivas feitas pelo modelo, conforme indicado na Equação 19:

$$\text{Precisão} = \frac{TP}{TP + FP} \tag{19}$$

- Revocação *(recall)*: mede é a capacidade do modelo de identificar corretamente todas as instâncias positivas, conforme indicado na Equação 20:

$$\text{Revocação} = \frac{TP}{TP + FN} \tag{20}$$

- O F-*Score* (ou F1-*Score*) é a média harmônica da precisão e do *recall* (revocação). Ela é útil quando se busca um equilíbrio entre precisão e *recall*, conforme indicado na Equação 21:

$$\text{F1-Score} = 2 \cdot \frac{\text{Precisão} \cdot \text{Revocação}}{\text{Precisão} + \text{Revocação}} \tag{21}$$

- Curva ROC: É representação gráfica que ilustra a performance de um modelo de classificação binária ao variar o limiar de decisão. A curva é traçada plotando a taxa de verdadeiros positivos (eixo y) contra a taxa de falsos positivos (eixo x) para diferentes valores de limiar;

- *Area Under the Curve* (AUC): Ela resume a performance da curva ROC em um único valor numérico. A AUC varia de 0 (baixa classificação) a 1 (alta classificação).

Etapa 5: Treinamento do modelo *Transformer* Roberta

Após a limpeza e o código do modelo definido, foram efetuadas as seguintes etapas: a etapa de ajuste da arquitetura, onde os pesos do modelo são ajustados para a nova tarefa, utilizando um conjunto de dados rotulado relacionado à tarefa específica. No caso, classificação de texto, por meio de transferência de aprendizado. Nesse tipo de transferência de conhecimento, o modelo aprimora sua capacidade em abordar novas tarefas baseado em conhecimentos adquiridos em uma tarefa específica anterior (TORREY; SHAVLIK, 2010). Assim, o modelo é treinado onde apenas os pesos da última camada são atualizados. Os pesos das outras camadas são mantidos constantes. Isso permite que o modelo ajuste a última camada aos novos dados sem perder as características úteis aprendidas nas camadas anteriores.

Para o treinamento do modelo foram alterados diversos hiperparâmetros. A escolha dos hiperparâmetros, bem como o intervalo de valores de testes foram determinados com base em trabalhos anteriores [13] encontrados na literatura para treinamento de modelos *Transformers*. Os seguintes item foram alterados:

- Criação de Máscaras de Atenção: As máscaras de atenção são usadas para evitar que o modelo veja alguns tokens durante o treinamento. Ao treinar o modelo para prever a próxima palavra em uma frase, a máscara de atenção é usada para evitar que o modelo veja a palavra que está tentando prever. Foi implementado a função *tokenize* que utiliza o tokenizador da roBERTa, que automaticamente gera máscaras de atenção como parte do processo de tokenização.

- Inicialização dos parâmetros: Os pesos do modelo são inicializados utilizando um pré-treinamento de transferência *(fine-tunning)* ou ajuste da arquitetura;

[13] https://ieeexplore.ieee.org/document/10205892

- Número de épocas de treinamento: Foi implementado o número de 3, 5 e 10 épocas. Esses valores foram alterados devido a percepção de *overfitting* acima de 10 épocas para essa tarefa. Esses valores também foram fundamentados em estudos anteriores que utilizaram o treinamento do modelo para a detecção específica de conteúdos "não saudáveis" em textos;

- Tamanho do lote: Foi implementado o tamanho do lote com o valor de 16 e 32, baseando-se em trabalhos relacionados que demonstraram a eficácia desses tamanhos específicos nesses tamanhos para o treinamento de modelos semelhantes;

- Algoritmo de otimização e Taxa de aprendizado: Foi utilizado o algoritmo ADAM (*Adaptive Moment Estimation*) [14]. Em comparação com o SGD (*Stochastic Gradient Descent*) [15], o algoritmo ADAM demonstrou desempenho mais eficaz nos resultados de aprendizado durante o treinamento. O hiperparâmetro de taxa de aprendizado foi inserido primeiramente para 0,01, e foi diminuíndo com os passar dos testes, chegando no valor fixo de 0,0002, no qual obteve uma importância relevante para os valores de acurácia;

- Perda do peso: É uma técnica utilizada durante o treinamento para evitar o sobreajuste *(overfitting)*. Foi inserido o valor de 0,01 para os testes.

Etapa 6: Avaliação do modelo *Transformer*

As métricas de avaliação do modelo *Transformer* foram similares às utilizadas para o modelo LSTM, incluindo acurácia, precisão, revocação, F1-score, curva ROC e AUC. Esses parâmetros de avaliação foram escolhidos para proporcionar uma comparação direta e consistente entre as diferentes arquiteturas de rede neural, permitindo uma análise de seu desempenho relativo.

Etapa 7: Comparação das técnicas

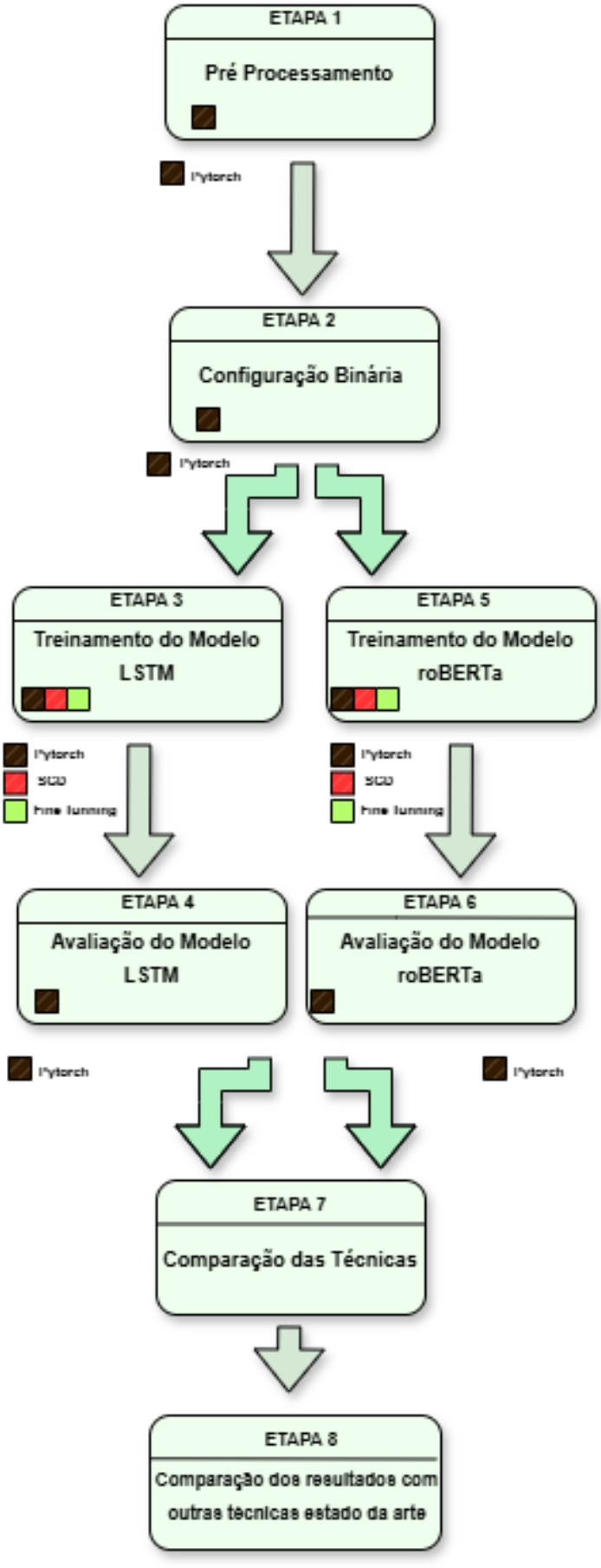
Após a avaliação das técnicas, foi validado o cálculo da métricas de avaliação, com base na previsão dos modelos e nos rótulos verdadeiros, também serão verificadas as métricas de precisão, *recall* e F1-score [16]. O desempenho com o uso de recursos (memória e tempo de resposta) e a eficácia de cada modelo na identificação de textos de discurso de ódio serão avaliados com base na métrica de precisão F1."

Etapa 8: Comparação das técnicas estado da arte

Nesta etapa, foi realizada a comparação dos resultados obtidos pelas arquiteturas LSTM e roBERTa no formato binário com outras arquiteturas de redes neurais apresentadas em artigos científicos que utilizaram o mesmo conjunto de dados. A comparação foi conduzida com base na métrica objetiva, a acurácia.

[14] https://pytorch.org/docs/stable/generated/torch.optim.Adam.html
[15] https://pytorch.org/docs/stable/generated/torch.optim.SGD.html
[16] https://learn.microsoft.com/pt-br/windows/ai/windows-ml/tutorials/pytorch-analysis-train-model

Figura 18 – Metodologia utilizada para o trabalho



Fonte: Autoria própria (2025).

4 RESULTADOS

Este capítulo apresenta os resultados obtidos durante o desenvolvimento do trabalho, com foco na avaliação de duas arquiteturas de redes neurais, LSTM e roBERTa, aplicadas à tarefa de identificação de discurso de ódio em textos curtos extraídos da rede social que formam o banco de dados told-br [1].

Além de descrever os modelos utilizados, são discutidos os principais desafios enfrentados, como limitações técnicas durante o treinamento, e os ajustes necessários para superar tais barreiras, especialmente no contexto do uso de GPUs (Unidades de processamento gráfico). São apresentadas a configuração do banco de dados no formato binário e as métricas de desempenho de cada modelo, como acurácia, precisão, *recall* e *F1-score*, bem como uma comparação entre as arquiteturas para destacar suas respectivas eficácias na classificação de textos ofensivos. Também é apresentado uma comparação dos resultados das arquiteturas mencionadas anteriormente junto com as arquiteturas [2] [3] de estado da arte que também utilizaram esse banco de dados , a fim de identificar os pontos fortes e fracos de cada abordagem, fornecendo informações sobre suas aplicações e limitações no contexto específico de detecção de conteúdo ofensivo no banco de dados proposto. Por fim, este capítulo também explora os dados obtidos e as reflexões sobre a aplicabilidade das redes neurais à detecção de conteúdos tóxicos na rede social X [4].

4.1 Configuração do banco de dados Bionário

O banco de dados utilizado, o told-br[5] como mencionado anteriormente, possui dois tipos de configuração para ser utilizado, a configuração binária e a multirrótulo. Foram efetuados testes nas 2 arquiteturas no tipo de configuração binária. Esse tipo de possui as classes disponíveis que representam os tipos tóxico (1) com 7.375 conteúdos do tipo *text* e não tóxico (0) com 9.425 conteúdos.

4.2 Implementação da Arquitetura LSTM

Durante o treinamento de um modelo LSTM para classificação binária, diversas etapas foram realizadas para assegurar o desempenho e a eficiência da arquitetura proposta. O processo incluiu a definição de hiperparâmetros essenciais, como o *batchsize*, a taxa de apren-

[1] https://huggingface.co/datasets/JAugusto97/told-br
[2] https://arxiv.org/abs/2410.15148
[3] https://huggingface.co/joaollm/xlm-roberta-base-finetuned-told-br
[4] https://x.com/
[5] https://huggingface.co/datasets/JAugusto97/told-br

dizado, o número de épocas e o tamanho das camadas ocultas, os quais foram otimizados iterativamente com base nos resultados observados.

A arquitetura foi projetada com uma rede LSTM de duas camadas ocultas, cada uma contendo 128 e 200 neurônios, e uma camada de saída para prever duas classes. Os dados de entrada foram configurados com um comprimento de sequência de 10 tokens, permitindo a análise de padrões temporais curtos. A função de perda utilizada foi a *CrossEntropyLoss* [6], adequada para tarefas de classificação, enquanto o otimizador escolhido foi o Adam, reconhecido por sua robustez e eficiência em ajustar pesos em arquiteturas profundas.

Para evitar o problema de *overfitting*, foi implementado o mecanismo de *early stopping*. Esse método monitora a perda de validação e interrompe o treinamento caso não sejam observadas melhorias após um número preestabelecido de épocas consecutivas. Sua utilização demonstrou ser fundamental no monitoramento do número das épocas específicas , evitando o gasto desnecessário de recursos computacionais e garantir a generalização do modelo.

A avaliação do modelo baseou-se em métricas amplamente aceitas na área de aprendizado de máquina, incluindo acurácia, precisão, recall e F1-score, que fornecem uma visão abrangente do desempenho em cenários desbalanceados. A acurácia mede a porcentagem de previsões corretas do modelo em relação ao total de previsões feitas. A precisão foca em como o modelo lida com as previsões positivas. Ela mede a proporção de previsões positivas feitas pelo modelo que, de fato, são positivas. Quando a precisão é alta, isso significa que o modelo comete poucos erros ao classificar exemplos negativos como positivos. O recall, por outro lado, mede a capacidade do modelo de identificar corretamente todos os exemplos positivos reais. Um recall alto indica que o modelo está sendo eficaz em capturar os positivos, mas sem levar em conta os falsos positivos.

Para equilibrar a precisão e o recall, utilizamos o F1-score, que é a média harmônica entre essas duas métricas. O F1-score é especialmente útil em cenários desbalanceados, pois ajuda a garantir que o modelo não apenas identifique bem os positivos (recall), mas também evite muitos falsos positivos (precisão). Essas métricas, em conjunto, fornecem uma visão mais detalhada e precisa sobre como o modelo está se saindo ao aprender e fazer previsões. Além disso, foi gerada a curva ROC, permitindo a análise da relação entre as taxas de verdadeiros positivos e falsos positivos, com o cálculo da área sob a curva AUC para quantificação do desempenho geral. A partir da curva, foi calculada a área sob a curva AUC, que quantifica o desempenho geral do modelo, oferecendo uma visão mais completa sobre sua capacidade de discriminar entre as classes.

4.2.1 Comparação entre as métricas

No processo de treinamento do modelo LSTM, foi realizada uma análise dos resultados obtidos para diferentes combinações de hiperparâmetros, como o número de épocas e o

[6] https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html

batchsize. Os experimentos visaram identificar a configuração que proporcionasse o melhor desempenho em termos de generalização e eficiência. Foram efetuados testes entre 25, 28 e 30 épocas.

Os resultados na Tabela 3 indicaram que o melhor desempenho foi alcançado com 27 épocas e *batchsize* 200, atingindo uma acurácia de 71,4 por cento, F1-score de 70,99 por cento, precisão de 71,43 por cento e recall de 70,56 por cento. A escolha do *batchsize* influenciou diretamente a estabilidade do treinamento. Utilizando um de 200, o modelo conseguiu obter gradientes mais suaves durante o processo de atualização dos pesos. O número de épocas foi um outro fator importante. Durante os experimentos, foi observado que, com 28 épocas, o modelo conseguiu capturar os padrões mais relevantes dos dados de treinamento, mantendo uma boa performance no conjunto de validação. Esse número foi suficiente para que o modelo alcançasse um bom aprendizado sem sinais de saturação ou *overfitting*

Esses resultados indicaram que o modelo apresentou um desempenho razoavelmente bom, especialmente considerando a harmonia entre as métricas. Também foi demonstrado que o modelo não está favorecendo excessivamente uma classe em detrimento da outra e obtendo a capacidade do modelo de identificar padrões nos dados com com bons resultados entre acurácia. Nos cenários verificados com 30 épocas, foi observada uma leve piora nos resultados da validação, sugerindo que treinar por mais tempo não traria benefícios adicionais e poderia até prejudicar a generalização. Essa piora é notada a partir da 28º época.

Tabela 3 – Métricas LSTM com 28 épocas e *batchsize* 200

Época	Perda Treino	Perda Validação	Acurácia	F1	Precisão	Recall
1	0,6931	0,6991	0,4940	0,5289	0,4913	0,5726
4	0,6876	0,7004	0,5760	0,6294	0,5556	0,7258
7	0,6775	0,7410	0,5520	0,5172	0,5556	0,4839
10	0,6702	0,7343	0,5880	0,5896	0,5827	0,5968
13	0,6630	0,7288	0,6060	0,6083	0,6000	0,6169
16	0,6529	0,7622	0,6120	0,5958	0,6164	0,5766
19	0,6409	0,7742	0,6260	0,6080	0,6332	0,5847
22	0,6180	0,7714	0,6520	0,6345	0,6623	0,6089
25	0,5893	0,8592	0,6700	0,6950	0,6416	0,7581
27	**0,5628**	**0,8209**	**0,7140**	**0,7099**	**0,7143**	**0,7056**
28	0,5628	0,8758	0,7080	0,7235	0,6821	0,7702

Fonte: Autoria própria (2025).

Por outro lado, em experimentos com o *batchsize* de 128 como na Tabela 4 o treinamento apresentou avanços mais lentos, e, em alguns casos, houve uma tendência ao *overfitting* nas últimas épocas com a acurácia diminuindo e o aumento do valor da perda de validação e treinamento. Isso reforça que tamanhos maiores de lote podem ser mais adequados para modelos como a LSTM, que se beneficiam de atualizações mais consistentes nos parâmetros.

Portanto, a configuração final escolhida na Tabela 3 com 27 épocas e batchsize 200, mostrou-se ideal para o problema em questão, apresentando uma combinação eficiente de estabilidade no treinamento, boa generalização e métricas de desempenho consistentes. Esses

Tabela 4 – Métricas LSTM 28 épocas e *batchsize* 128

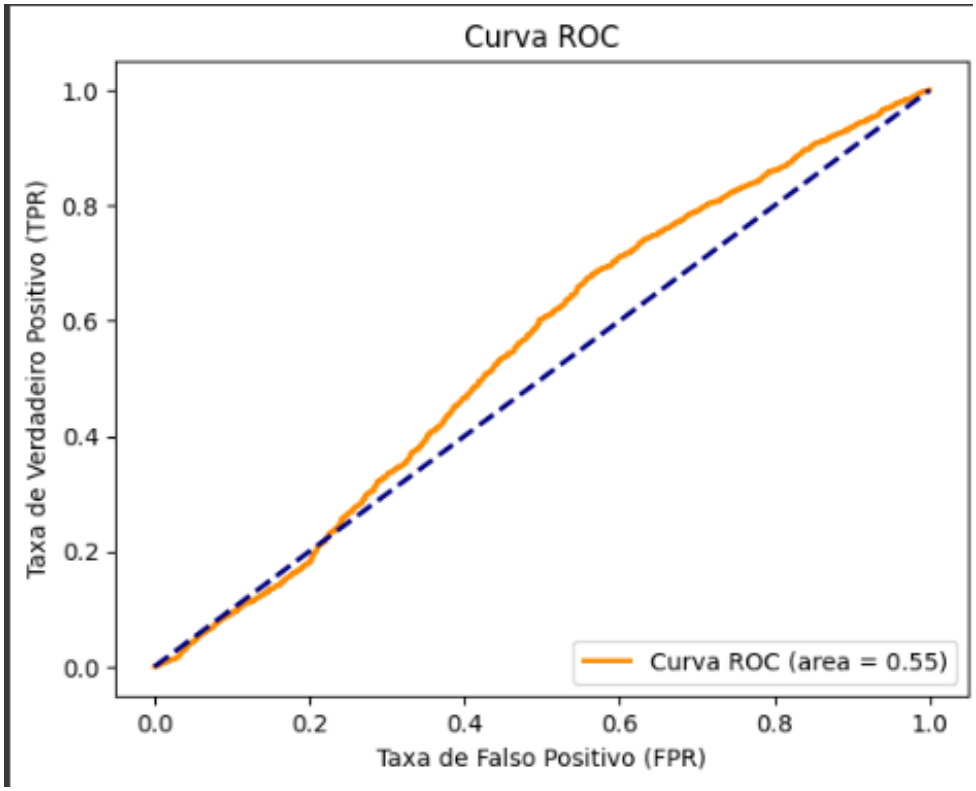
Época	Perda Treino	Perda Validação	Acurácia	F1	Precisão	Recall
1	0,6946	0,6930	0,4880	0,0000	0,0000	0,0000
4	0,6941	0,6930	0,5120	0,6772	0,5120	1,0000
7	0,6855	0,6917	0,5540	0,6177	0,5378	0,9189
10	0,6782	0,6973	0,5800	0,6097	0,5816	0,6406
13	0,6724	0,7045	0,5720	0,6198	0,5755	0,6250
16	0,6647	0,7028	0,5780	0,5887	0,5875	0,5898
19	0,6563	0,7018	0,6180	0,6470	0,6180	0,6836
22	0,6461	0,7108	0,6320	0,6571	0,6214	0,7054
25	0,6424	0,7475	0,6480	0,6440	0,6477	0,6403
28	**0,6068**	**0,7522**	**0,6500**	**0,6654**	**0,6517**	**0,6797**

Fonte: Autoria própria (2025).

ajustes refletem a importância do ajuste de hiperparâmetros na construção de modelos robustos e eficazes para tarefas de classificação binária.

O gráfico 19 da curva ROC apresenta uma área sob a curva de 0,53 por cento indicando que o desempenho do modelo é apenas marginalmente superior ao de um classificador aleatório. No apêndice A deste documento, serão apresentados os gráficos ROC de todos os experimentos obtidos.

Figura 19 – CURVA roC - LSTM 28 épocas a *batchsize* 200



Fonte: Autoria própria (2025).

4.3 Implementação da Arquitetura roBERTa

A etapa inicial envolve o carregamento do dataset, que contém textos e rótulos estruturados nas colunas text e label. Após verificar a consistência das amostras, o modelo foi preparado para processar as entradas textuais por meio da tokenização, utilizando o tokenizador *roberta-base*. Esse processo converte os textos em sequências numéricas compatíveis com o modelo, aplicando truncamento e preenchimento para atender ao limite de comprimento do modelo.

O modelo utilizado é o roBERTa pré-treinado, ajustado para classificação binária ao definir número de *labels* igual a 2. Essa abordagem de transferência de conhecimento permite adaptar um modelo robusto e previamente treinado a uma nova tarefa com menos esforço computacional, ajustando apenas as últimas camadas. Para o treinamento, foram configurados hiperparâmetros como taxa de aprendizado, número de épocas e *batchsize*, além de estratégias de salvamento e avaliação por época. O uso de logs e checkpoints foi implementado para monitorar e salvar o melhor modelo, evitando desperdício de recursos de armazenamento.

A avaliação do modelo foi realizada utilizando métricas como acurácia, precisão, revocação e F1-score, calculadas com base nas previsões do modelo no conjunto de validação. Essas métricas foram complementadas pela análise da curva ROC e da métrica AUC, que fornecem uma visão mais detalhada sobre a capacidade discriminativa do modelo.

Para o treinamento e *fine-tuning* da arquitetura roBERTa, inicialmente foram realizadas muitas épocas utilizando um ambiente de execução do tipo *Graphics Processing Unit* (GPU). No entanto, esse ambiente apresentou desafios significativos, como o tempo elevado de execução e o limite de memória insuficiente. Diante disso, foi necessário recorrer a recursos computacionais mais avançados, avaliando o modelo em um ambiente de execução do tipo AR100. Essa GPU de alta performance foi projetada especificamente para cargas de trabalho avançadas em IA e aprendizado de máquina, proporcionando uma melhora substancial no desempenho. Para utilizar a AR100, foi necessário atualizar o ambiente de execução e instalar o CUDA *(Compute Unified Device Architecture)*.

Durante os testes com mais de 10 épocas, o modelo apresentou sinais de *overfitting* logo nos primeiros experimentos. A partir disso, foi reduzida a quantidade de épocas, e, após uma série de testes, os valores de 3, 5 e 10 épocas foram definidos como padrões para as avaliações e comparados a trabalhos relacionados [7]. Após ajustar diversos hiperparâmetros, como o peso da perda, o *batchsize*, o número de épocas e a taxa de aprendizado, constatou-se que os fatores que mais impactavam a acurácia eram o *batchsize*, a quantidade de épocas e a taxa de aprendizado. Assim, os testes foram direcionados para otimizar esses hiperparâmetros.

[7] https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=arnumber=10205892

4.3.1 Comparação entre as métricas

Ao comparar os resultados do modelo roBERTa com *batchsize* 16 e 32, é observado algumas diferenças. Com o *batchsize* 16, a perda de treinamento diminui de 0,5525 para 0,2780, enquanto a perda de validação flutua entre 0,5474 e 0,7696. No caso do *batchsize* 32, a perda de treinamento também diminui, mas mais acentuadamente, de 0,5554 para 0,2454, e a perda de validação vai de 0,5388 para 0,7007.

A acurácia do modelo com *batchsize* 16 atinge 75,52 por cento na décima época, ligeiramente superior ao modelo com *batchsize* 32, que chega a 73,62 por cento na mesma época. O modelo com o batchsize 16 obteve sua maior acurácia na 4º época (0,7628) em comparação com o o modelo de 32 (0,7623). Em relação às métricas de F1, precisão e recall, o modelo com *batchsize* 16 se destaca em recall (em torno de 0,80 nas primeiras épocas), enquanto o *batchsize* 32 apresenta uma precisão ligeiramente melhor (0,7129 na quarta época), com o valor mais baixo e uma maior precisão.

Em resumo, o modelo com *batchsize* 32 demonstrou uma perda de validação mais estável e melhor precisão, enquanto o modelo com *batchsize* 16 tem um desempenho superior em recall e um aumento ligeiramente maior de 0,7628 por cento e 0,7552 na acurácia na última época e acurácia como mencionado anteriormente.

Tabela 5 – Métricas roBERTa com 10 épocas e *batchsize* 16

Época	Perda treino	Perda validação	Acurácia	F1	Precisão	Recall
1	0,552500	0,547439	0,723333	0,728885	0,666951	0,803498
2	0,493600	0,519572	0,733333	0,752431	0,659690	0,875514
3	0,460900	0,522029	0,741429	0,721966	0,718654	0,725309
4	**0,43620**	**0,537916**	**0,762857**	**0,765316**	**0,706087**	**0,835391**
5	0,389400	0,564441	0,741429	0,757047	0,669834	0,870397
6	0,338400	0,569350	0,759524	0,751109	0,720908	0,783951
7	0,331200	0,637503	0,747619	0,742468	0,703499	0,780864
8	0,276400	0,698313	0,752857	0,745214	0,712676	0,780064
9	0,272300	0,732058	0,753333	0,748054	0,709410	0,791152
10	0,278000	0,769634	0,755238	0,747296	0,715631	0,781893

Fonte: Autoria própria (2025).

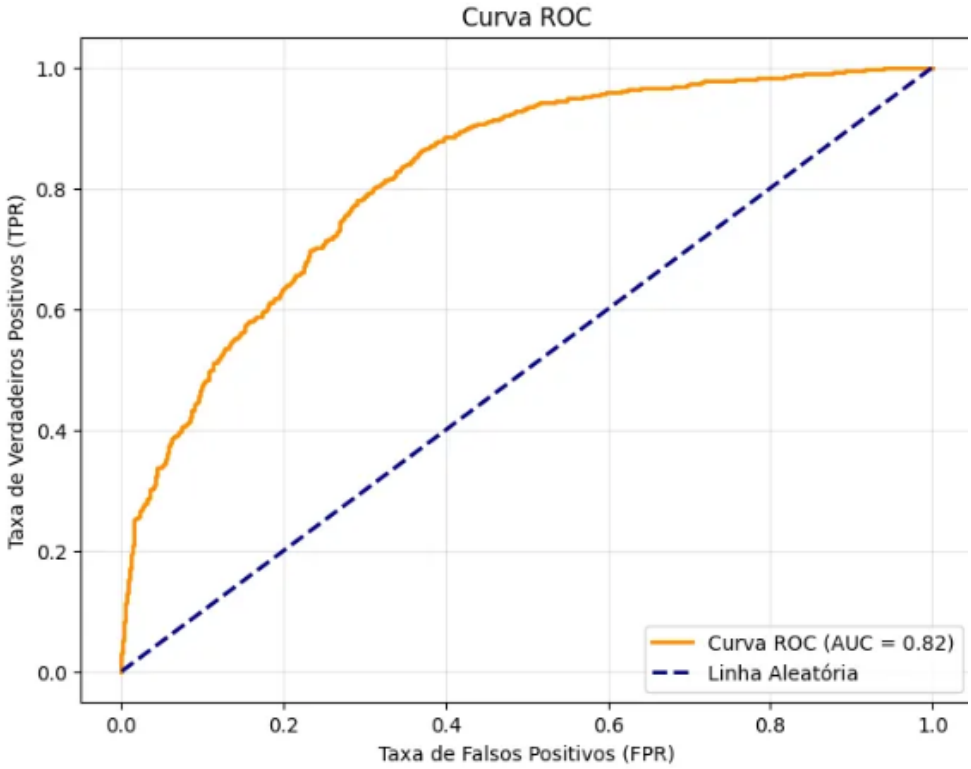
Por fim, a Figura 20 ilustra a curva ROC correspondente à arquitetura roBERTa que alcançou o melhor desempenho em termos de acurácia. Este gráfico caracteriza a relação entre a taxa de verdadeiros positivos (TPR) e a taxa de falsos positivos (FPR) ao longo de diferentes limiares de decisão. A área sob a curva ROC foi calculada como 0,82, o que denota que o modelo possui uma probabilidade de 82 por cento de diferenciar corretamente uma amostra positiva de uma negativa. Este resultado reflete um desempenho robusto na tarefa de classificação, com uma capacidade satisfatória de discriminação entre as classe.

Tabela 6 – Métricas roBERTa com 10 épocas e *batchsize* 32

Época	Perda treino	Perda validação	Acurácia	F1	Precisão	Recall
1	0.555400	0.538771	0.728571	0.715000	0.695525	0.735597
2	0.506500	0.513498	0.740952	0.749770	0.687037	0.838477
3	0.448400	0.530132	0.747143	0.754280	0.685450	0.838477
4	**0.414200**	**0.520023**	**0.762381**	**0.760442**	**0.712871**	**0.814815**
5	0.391400	0.546507	0.747143	0.759184	0.678832	0.861111
6	0.342200	0.602108	0.745238	0.746805	0.691499	0.811728
7	0.322900	0.604120	0.743333	0.745154	0.689414	0.810700
8	0.274400	0.640989	0.741905	0.738164	0.689511	0.786008
9	0.292900	0.656378	0.741429	0.744711	0.685714	0.818415
10	0.245400	0.700740	0.736190	0.730807	0.692449	0.773663

Fonte: Autoria própria (2025).

Figura 20 – CURVA roC- roBERTa 10 épocas com batch size 16 - roBERTa



Fonte: Autoria própria (2025).

4.4 Discussão

Após a análise dos resultados obtidos por meio desses dois modelos, foi possível observar as diferenças significativas no desempenho entre as arquiteturas. A acurácia foi a principal métrica utilizada para comparar a performance dos modelos, já que ela reflete diretamente a capacidade dos modelos de realizar previsões corretas.

4.4.1 Análise da acurácia entre os modelos LSTM e roBERTa

A arquitetura roBERTa obteve as melhores acurácias, tanto com batch size 32 quanto 16. A configuração da Tabela 5 com batch size 16 alcançou uma acurácia máxima de 0,762857 na 4ª época, enquanto o modelo na Tabela 6 com lote 32 obteve uma acurácia máxima de 0,762381 também na 4ª época. Observou-se que ambos os modelos apresentaram uma leve queda na acurácia nas últimas épocas, mas mantiveram bastante consistente, com uma performance superior a 0,70 durante todo o treinamento. O comportamento geral dessa arquitetura demonstra que, com poucas épocas, ela consegue alcançar bons resultados, destacando-se como uma arquitetura de destaque para tarefas de classificação, especialmente em tarefas de processamento de linguagem natural.

A arquitetura LSTM, por outro lado, apresentou resultados inferiores em termos de acurácia. Na Tabela 4 configuração com batch size 128 obteve uma acurácia máxima de 0,6500 na 28ª época, enquanto o modelo com lote 200 obteve 0,7140 na 27ª época. A performance dessa arquitetura, embora tenha melhorado com o aumento das épocas, ainda permaneceu aquém do que foi alcançado pelo *transformer*. Isso indica que, embora o LSTM tenha mostrado capacidade de aprendizado, ele não conseguiu superar as vantagens de um modelo pré-treinado como o roBERTa, que é mais eficiente e robusto em tarefas de classificação de textos.

A principal diferença entre as duas arquiteturas está no seu comportamento ao longo das épocas e na capacidade de generalização. Enquanto a arquitetura roBERTa apresentou um desempenho superior de forma constante e sem grandes variações, o LSTM teve um crescimento mais gradual e ainda apresentou dificuldades em atingir as mesmas altas acurácias. Essa diferença pode ser atribuída ao fato de que o roBERTa é um modelo pré-treinado com uma grande base de dados, o que facilita sua adaptação a diferentes tarefas de linguagem. Em contrapartida, o LSTM é uma rede neural recorrente que, apesar de ser poderosa em sequências temporais, não possui o mesmo nível de pré-treinamento e refinamento que o roBERTa.

Com base nos resultados apresentados, a arquitetura roBERTa demonstrou ser a melhor arquitetura em termos de acurácia, sendo a escolha mais eficiente para a tarefa de classificação de texto considerando o dataset de rede social utilizado. O modelo alcançou uma acurácia acima de 0,75 já nas primeiras épocas, enquanto o LSTM mostrou um desempenho mais modesto, especialmente quando utilizado com tamanhos de lote maiores. Portanto, para tarefas de classificação em grandes volumes de dados textuais, a arquitetura roBERTa se apresenta como a arquitetura mais indicada, evidenciando as vantagens dos modelos pré-treinados em relação às redes neurais recorrentes tradicionais, como a LSTM.

4.4.2 Comparação dos Modelos Testados com Modelos Propostos em Literatura Especializada

O banco de dados utilizado nesse trabalho também foi analisado em outros trabalhos como o de Schulte, Hamborg e Akbik (2024). Nesse estudo, foi analisada uma técnica de transferência de aprendizado, em que modelos de linguagem pré-treinados BERT foram inicialmente ajustados em tarefas intermediárias antes de serem aplicados a tarefas-alvo específicas, como reconhecimento de discurso de ódio.

O LogME (Ranking of Pre-trained Models) é uma abordagem que avalia e ordena a eficácia de diferentes modelos pré-treinados para transferência de aprendizado em tarefas específicas. Ele classifica as fontes com base na performance observada em um conjunto de validação, permitindo que se escolha o modelo que melhor se adapta à tarefa-alvo.

No contexto do bando de dados told-br, o LogME foi utilizado juntamente com outras metodologias para classificar e comparar a eficácia de várias fontes para tarefas de transferência de aprendizado. O documento menciona que diversos métodos de seleção de fontes foram avaliados, incluindo ESM-LogME, LogME, NCE, LEEP, entre outros.

Os resultados da pesquisa analisaram especificamente o modelo BERT *base-multilingual-uncased* (BertM). Esse modelo é uma versão multilingue do BERT que não é sensível a maiúsculas e minúsculas, permitindo assim a aplicação em diversas línguas para tarefas de PLN. No contexto desta pesquisa, o BERTM foi empregado na classificação de discurso de ódio, permitindo uma análise comparativa de seu desempenho em relação a outros modelos, como o RoBERTa e o LSTM. Essa avaliação possibilitou identificar a eficácia do modelo na detecção de padrões linguísticos associados ao discurso de ódio, evidenciando sua capacidade de generalização para diferentes idiomas e contextos.

Por fim, foi observada uma melhora significativa nos resultados de acurácia ao utilizar BERT em tarefas de classificação de sentimentos dos tweets, chegando a média de 80 por cento de acurácia.

A Tabela 7 apresentada fornece uma comparação entre diferentes modelos e suas configurações de treinamento, focando na acurácia máxima alcançada. Os modelos analisados incluem o roBERTa com diferentes tamanhos de lote (32 e 16) e duas variações do modelo LSTM com diferentes tamanhos de lote (128 e 200). A tabela destaca o número de épocas utilizadas e a acurácia máxima obtida por cada configuração, proporcionando uma visão clara do desempenho relativo dos modelos durante o treinamento.

Nesse cenário, a arquitetura BertM apresentou um desempenho superior ao RoBERTa e por consequência o da arquitetura LSTM na detecção de discurso de ódio,o que pode ser explicado por alguns fatores:

- Capacidade multilingue e Diversidade dos dados: o modelo BertM foi treinado em textos de 104 idiomas diferentes, o que o torna mais robusto para lidar com variações

Tabela 7 – Comparação de Acurácia Máxima entre Modelos e Configurações

Modelo	Épocas	Acurácia Máxima
roBERTa (*batchsize* 32)	4	0,7623
roBERTa (*batchsize* 16)	4	0,7628
LSTM (*batchsize* 128)	28	0,65
LSTM (*batchsize* 200)	27	0,71
BertM	3	0,80

Fonte: Autoria própria (2025).

linguísticas, expressões coloquiais e gírias que aparecem frequentemente no discurso de ódio online;

- Menos sensível e Viés Linguístico: modelos monolíngues como o da roBERTa podem apresentar viés para o tipo de linguagem presente nos dados de treinamento e é possível que ele tenha enfrentado dificuldades na identificação de termos ofensivos não padronizados.

Nesse cenário, O BertM alcançou 80 por cento de acurácia em apenas 3 épocas, enquanto o RoBERTa precisou de 4 épocas e ainda assim ficou abaixo de 76,3 por cento. Esses resultados apontam que o BertM conseguiu captar melhor os padrões do conjunto de dados mais rapidamente, possivelmente devido à sua robustez no pré-treinamento em múltiplos idiomas. O melhor desempenho dessa arquitetura, se deve à sua habilidade de generalizar melhor os padrões linguísticos, sua exposição a uma ampla variedade de idiomas e a um treinamento mais diversificado. Isso o tornou mais eficiente na identificação de discurso de ódio na rede social X, um ambiente caracterizado por grande variação linguística e uso informal da linguagem.

Sendo assim, a utilização do modelo RoBERTa poderia ser otimizada através de algumas abordagens que melhoram sua convergência, estabilidade e desempenho. Algumas melhorias são:

- Ajustes na taxa de aprendizado: valores 3e-5 ou 5e-5, pode resultar em uma melhor convergência do modelo, ajustando-se à complexidade dos dados e prevenindo problemas como estagnação ou oscilação na perda;

- Aumentar o tamanho do batch size: testar tamanhos como 64 ou 128 para auxiliar na estabilização do treinamento, reduzindo a variabilidade entre os gradientes e promovendo um ajuste mais eficiente dos parâmetros;

- Exploração de modelos alternativos: utilizar modelos a versões mais adequada ao idioma da tarefa, como modelos focados na língua portuguesa ou utilizar maiores modelos como o roBERTa *large*;

- Melhoria na tokenização: no código do modelo da arquitetura roBERTa foi feito uma tokenização limitada baseado no parâmetro *maxlenght*. Uma alternativa seria testar um truncagem dinâmica, permitindo que o texto seja ajustado sem perder certas partes.

Por fim, a otimização da arquitetura RoBERTa pode ser alcançada através de ajustes na taxa de aprendizado, tamanho do batch, exploração de modelos alternativos e melhorias na tokenização. A implementação dessas estratégias, combinada com uma avaliação dos resultados, poderá obter um modelo mais preciso e eficiente para a tarefa em questão.

5 CONCLUSÃO

Esse trabalho teve como objetivo analisar, por meio de métricas, o desempenho das arquiteturas LSTM e RoBERTa na detecção de discurso de ódio na rede social X [1]. A pesquisa atingiu seu objetivo ao comparar as arquiteturas LSTM e RoBERTa na identificação de discurso de ódio na rede social X. Os resultados obtidos demonstraram que a arquitetura RoBERTa apresentou desempenho superior em comparação ao LSTM, alcançando maior acurácia mesmo em cenários com menor número de épocas de treinamento. Porém o modelo BERTM se sobressaiu ao modelo roBERTa em relação a quantidade de épocas e valores de acurácia. Embora o RoBERTa seja otimizado para um melhor desempenho em algumas tarefas, o BertM pode ter se destacado devido à sua robustez em diferentes idiomas, capacidade de generalização e adaptação ao conjunto de dados utilizado no estudo. Além disso, a análise das métricas confirmou a eficácia dos modelos baseados em transformers para a classificação de conteúdos tóxicos, validando a hipótese de que essa abordagem é mais adequada para a tarefa proposta. Dessa forma, os resultados obtidos evidenciam que os objetivos estabelecidos foram plenamente alcançados, contribuindo para a área de PLN e IA no combate ao discurso de ódio nas redes sociais.

5.0.1 Limitações do trabalho

Este trabalho apresentou uma análise comparativa entre as arquiteturas LSTM e RoBERTa na detecção de discurso de ódio na rede social X. No entanto, algumas limitações devem ser destacadas para contextualizar os resultados obtidos. Primeiramente, a análise foi conduzida apenas na configuração binária do conjunto de dados, o que restringe a capacidade do estudo de avaliar a classificação de discurso de ódio em um contexto mais detalhado e multicategórico.

Além disso, apenas duas arquiteturas foram exploradas: LSTM e RoBERTa. Apesar de essas abordagens serem amplamente utilizadas, outras arquiteturas baseadas em transformers, redes neurais convolucionais ou híbridas poderiam ter sido consideradas para uma análise mais abrangente. Ademais, os experimentos foram conduzidos com um conjunto específico de batch sizes (2 valores para cada arquitetura), limitando a compreensão do impacto desse hiperparâmetro na performance dos modelos.

[1] https://x.com/

6 TRABALHOS FUTUROS

Com base nas limitações identificadas, algumas direções para trabalhos futuros podem ser consideradas. Primeiramente, seria interessante expandir a análise para a configuração multilabel do conjunto de dados, permitindo uma classificação mais detalhada dos diferentes tipos de discurso de ódio. Isso poderia proporcionar uma compreensão mais refinada sobre a eficácia dos modelos em identificar nuances específicas de conteúdo tóxico.

Outra direção importante seria a exploração de diferentes batch sizes para avaliar seu impacto na estabilidade e convergência dos modelos. Isso poderia ajudar a identificar configurações ótimas que balanceiem desempenho e eficiência computacional.

Por fim, recomenda-se a investigação de outras arquiteturas, incluindo modelos mais recentes e sofisticados, como variantes do BERT, modelos baseados em GPT, e híbridos entre redes recorrentes e transformers.

REFERÊNCIAS

ALAM, T.; KHAN, A.; ALAM, F. Bangla text classification using transformers. **arXiv preprint arXiv:2011.04446**, 2020. Disponível em: https://www.mdpi.com/2076-3417/12/11/5720 Acesso em: 27 de março de 2024.

ALAMMARY, A. S. Bert models for arabic text classification: A systematic review. **Applied Sciences**, v. 12, n. 11, 2022. ISSN 2076-3417. Disponível em: https://www.mdpi.com/2076-3417/12/11/5720.

ANCHIÊTA, R. *et al.* Pln: Das técnicas tradicionais aos modelos de deep learning. **Sociedade Brasileira de Computação**, 2021. Acesso em: 20 de abril de 2024.

ARAÚJO, G. R. d. O.; VITTORAZZI, W. d. O. A aplicação de redes neurais artificiais recorrentes no processamento de linguagem natural. 2018. Acesso em: 10 de março de 2024. Disponível em: https://dspace.uniube.br:8443/handle/123456789/535.

AYO, F. E. *et al.* Machine learning techniques for hate speech classification of twitter data: State-of-the-art, future challenges and research directions. **Computer Science Review**, v. 38, p. 100311, 2020. ISSN 1574-0137. Acesso em: 18 de abril de 2024. Disponível em: https://www.sciencedirect.com/science/article/pii/S1574013720304111?casa_token=zuGMMOHKv6gAAAAA:ztc1WFu7XpFSM9UzeKcdULrL-Y2OICnAzcA9PVmfxsCFdSiSMDe2AQ7Fa2aj6l3OFdbkZoR_I30.

BAKAROV, A. **A Survey of Word Embeddings Evaluation Methods**. 2018. Acesso em: 25 de abril de 2024. Disponível em: https://arxiv.org/abs/1801.09536.

BALKUS, S. V.; YAN, D. Improving short text classification with augmented data using gpt-3. **Natural Language Engineering**, p. 1–30, 2023. Disponível em: https://arxiv.org/abs/2205.10981.

BEAUSOLEIL, L. E. Free, hateful, and posted: rethinking first amendment protection of hate speech in a social media world. **BCL Rev.**, HeinOnline, v. 60, p. 2101, 2019.

BORJI, A. **Key-Value Transformer**. 2023. Data de Acesso em: 13 de abril de 2024. Disponível em: https://arxiv.org/abs/2305.19129.

BRITO, L. T. C. d.; DAMETTO, R. C. O problema xor usando redes neurais artificiais em python e com função de ativação tangente hiperbólica. **FIBiNOVA**, v. 3, p. 1, dezembro 2023. Artigo Original, Acesso em: 22 de abril de 2024. Disponível em: https://revistasfib.emnuvens.com.br/fibinova/article/view/666.

CAO, R.; LEE, R. K.-W.; HOANG, T.-A. Deephate: Hate speech detection via multi-faceted text representations. *In*: **Proceedings of the 12th ACM Conference on Web Science**. New York, NY, USA: Association for Computing Machinery, 2020. (WebSci '20), p. 11–20. ISBN 9781450379892. Disponível em: https://doi.org/10.1145/3394231.3397890.

CASTAÑO-PULGARÍN, S. A. *et al.* Internet, social media and online hate speech. systematic review. **Aggression and Violent Behavior**, Elsevier, v. 58, p. 101608, 2021.

CHIROYA. **Softmax Function**. 2022. Acesso em: 24 de abril de 2024. Disponível em: https://velog.io/@chiroya/13-Softmax-Function.

DATABRICKS. **Machine Learning on Databricks - Databricks Machine Learning**. Databricks, 2021. Acesso em: 18 de abril de 2024. Disponível em: https://databricks.com/resources/machine-learning-on-databricks.

DEVLIN, J. *et al.* Bert: Pre-training of deep bidirectional transformers for language understanding in: Proceedings of the 2019 conference of the north american chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers). **Minneapolis, MN: Association for Computational Linguistics**, p. 4171–86, 2019. Acesso em: 22 de abril de 2024. Disponível em: https://arxiv.org/abs/1810.04805.

DIGITALBRAZIL. Digital brazil 2023: Insights do report anual de plataformas digitais. **globalad**, 2023. Disponível em: https://globalad.com.br/blog/digital-brazil-2023/. Acesso em: 14 de setembro de 2023.

DUBEY, A. K.; JAIN, V. Comparative study of convolution neural network's relu and leaky-relu activation functions. *In*: MISHRA, S.; SOOD, Y. R.; TOMAR, A. (Ed.). **Applications of Computing, Automation and Wireless Systems in Electrical Engineering**. Singapore: Springer Singapore, 2019. p. 873–880. ISBN 978-981-13-6772-4. Acesso em: 20 de abril de 2024. Disponível em: https://link.springer.com/chapter/10.1007/978-981-13-6772-4_76.

FARAH, A. Hate speech digital: Em busca de respostas. **Acta Científica. Ciências Humanas**, v. 27, n. 2, p. 117–130, 2018. Disponível em: https://www.researchgate.net/publication/330571494_HATE_SPEECH_DIGITAL_EM_BUSCA_DE_RESPOSTAS.

FONTANA, É. Introdução aos algoritmos de aprendizagem supervisionada. **Departamento de Engenharia Química, Universidade Federal do Paraná**, 2020. Acesso em: 10 de abril de 2024. Disponível em: https://fontana.paginas.ufsc.br/files/2018/03/apostila_ML.pdf.

GAGLIARDONE, I. *et al.* **Countering online hate speech**. [*S.l.*]: Unesco Publishing, 2015. Acesso em: 10 de março de 2024. ISBN 978-92-3-100105-5.

GALLICCHIO, C. **Short-term Memory of Deep RNN**. 2018. Acesso em: 22 de abril de 2024. Disponível em: https://arxiv.org/abs/1802.00748.

GAO, S. *et al.* Limitations of transformers on clinical text classification. **IEEE journal of biomedical and health informatics**, IEEE, v. 25, n. 9, p. 3596–3607, 2021. Disponível em: https://ieeexplore.ieee.org/abstract/document/9364676. Acesso em: 18 de março de 2024.

HAMAGUTI, É. K.; BREVE, F. A. Introdução sobre machine learning e deep learning. 2022. Acesso em: 10 de março de 2024. Disponível em: https://www.fabriciobreve.com/artigos/2022/jornacitec2022expandido.pdf.

HE, C. *et al.* A new wavelet thresholding function based on hyperbolic tangent function. **Mathematical Problems in Engineering**, Hindawi, v. 2015, 2015. Acesso em: 17 de abril de 2024. Disponível em: https://www.hindawi.com/journals/mpe/2015/528656/.

HE, K. *et al.* Transformers in medical image analysis. **Intelligent Medicine**, v. 3, n. 1, p. 59–78, 2023. ISSN 2667-1026. Acesso em: 20 de abril de 2024. Disponível em: https://www.sciencedirect.com/science/article/pii/S2667102622000717.

HIROSE, Y.; YAMASHITA, K.; HIJIYA, S. Back-propagation algorithm which varies the number of hidden units. **Neural Networks**, v. 4, n. 1, p. 61–66, 1991. ISSN 0893-6080. Acesso em: 15 de abril de 2024. Disponível em: https://www.sciencedirect.com/science/article/pii/089360809190032Z.

JAMIL, S.; PIRAN, M. J.; KWON, O.-J. A comprehensive survey of transformers for computer vision. **Drones**, v. 7, n. 5, 2023. ISSN 2504-446X. Disponível em: https://www.mdpi.com/2504-446X/7/5/287.

Jornal Nacional. Denúncias de crimes envolvendo discurso de ódio nas redes sociais triplicaram nos últimos 6 anos, aponta levantamento. **G1 Globo**, 2023. Disponível em: https://g1.globo.com/jornal-nacional/noticia/2023/05/01/denuncias-de-crimes-envolvendo-discurso-de-odio-nas-redes-sociais-triplicaram-nos-ultimos-6-anos-apon ghtml. Acesso em: 14 de setembro de 2023.

JURAFSKY, D.; MARTIN, J. H. Naïve bayes classifier approach to word sense disambiguation. **Computational Lexical Semantics**, 2009. Acesso em: 07 de abril de 2024. Disponível em: https://www.let.rug.nl/nerbonne/teach/rema-stats-meth-seminar/presentations/Olango-Naive-Bayes-2009.pdf.

KENTON, J. D. M.-W. C.; TOUTANOVA, L. K. Bert: Pre-training of deep bidirectional transformers for language understanding. *In*: **Proceedings of naacL-HLT**. [*S.l.*: *s.n.*], 2019. v. 1, p. 2.

KOVáCS, Z. L. **Redes Neurais Artificiais**. [*S.l.*]: Editora Livraria da Fisica, 2023. Acesso em: 15 de março de 2024. ISBN 8588325144.

KUBAT, M. Neural networks: a comprehensive foundation by simon haykin, macmillan, 1994, isbn 0-02-352781-7. **The Knowledge Engineering Review**, Cambridge University Press, v. 13, n. 4, p. 409–412, 1999. Acesso em: 25 de março de 2024. Disponível em: https://dl.acm.org/doi/abs/10.5555/521706.

KYURKCHIEV, N.; MARKOV, S. **Sigmoid Functions Some Approximation and Modelling Aspects: Some Moduli in Programming Environment MATHEMATICA**. [*s.n.*], 2015. Acesso em: 14 de abril de 2024. ISBN 978-3-659-76045-7. Disponível em: https://www.researchgate.net/publication/308898939_Sigmoid_Functions_Some_Approximation_and_Modelling_Aspects_Some_Moduli_in_Programming_Environment_MATHEMATICA.

LEUNG, H.; HAYKIN, S. The complex backpropagation algorithm. **IEEE Transactions on Signal Processing**, v. 39, n. 9, p. 2101–2104, 1991. Acesso em: 13 de abril de 2024. Disponível em: https://ieeexplore.ieee.org/abstract/document/134446.

LILLICRAP, T. P. *et al.* Backpropagation through time and the brain. **Nature Reviews Neuroscience**, Springer Nature Limited, v. 21, p. 346, 2020. Acesso em: 22 de abril de 2024. Disponível em: https://www.sciencedirect.com/science/article/pii/S0959438818302009.

LIN, T. *et al.* A survey of transformers. **AI Open**, v. 3, p. 111–132, 2022. ISSN 2666-6510. Acesso em: 22 de abril de 2024. Disponível em: https://www.sciencedirect.com/science/article/pii/S2666651022000146.

LIU, Y. *et al.* **RoBERTa: A Robustly Optimized BERT Pretraining Approach**. 2019.

MARIANO, D. Métricas de avaliação em machine learning: acurácia, sensibilidade, precisão, especificidade e f-score. **BIOINFO – Revista Brasileira de Bioinformática**, Edição 01, Julho 2021.

MAZURKIEWICZ, J.; ZAMOJSKI, W. Systolic-based hardware realisation of hopfield neural network. **Eletrónica e Telecomunicações**, Citeseer, v. 3, n. 5, p. 392–399, 2002. Acesso em: 18 de abril de 2024. Disponível em: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=34bff4a2abe6594024513adfb90c0669435c2c8c.

MEYER, D. **Violence Against Queer People: Race, Class, Gender, and the Persistence of Anti-LGBT Discrimination**. Rutgers University Press, 2015. Acesso em: 10 de março de 2024. ISBN 0813573157. Disponível em: https://doi.org/10.1080/19361653.2017.1392917.

MIKOLOV, T. *et al.* Distributed representations of words and phrases and their compositionality. *In*: BURGES, C. *et al.* (Ed.). **Advances in Neural Information Processing Systems**. Curran Associates, Inc., 2013. v. 26. Acesso em: 22 de abril de 2024. Disponível em: https://proceedings.neurips.cc/paper_files/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf.

MITCHELL, T. M. **Machine learning: Trends, perspectives, and prospects**. IEEE Press, 2015. Disponível em: https://ieeexplore.ieee.org/document/8375630.

MULLAH, N. S.; ZAINON, W. M. N. W. Advances in machine learning algorithms for hate speech detection in social media: A review. **Information**, Multidisciplinary Digital Publishing Institute, v. 12, n. 2, p. 70, 2021. Acesso em: 20 de março de 2024. Disponível em: https://ieeexplore.ieee.org/abstract/document/9455353.

MÜLLER, M.; SALATHÉ, M.; KUMMERVOLD, P. E. Covid-twitter-bert: A natural language processing model to analyse covid-19 content on twitter. **Frontiers in artificial intelligence**, Frontiers Media SA, v. 6, p. 1023281, 2023. Acesso em: 20 de abril de 2024. Disponível em: https://www.frontiersin.org/articles/10.3389/frai.2023.1023281/full.

NICKEL, B. Desafios para o combate à violação de direitos humanos na internet: premissas e casos. **Friedrich Ebert Stiftung Brasil**, 2017. Acesso em: 20 de março de 2024. Disponível em: https://library.fes.de/pdf-files/bueros/brasilien/13194.pdf.

NORVIG, P.; RUSSELL, S. **Inteligência Artificial**. Campus, 1995. Acesso em: 10 de março de 2024. ISBN 8535237011. Disponível em: https://dspace.scz.ucb.edu.bo/dspace/bitstream/123456789/417/3/1609.pdf.

PIÑEIRO-OTERO, T.; MARTÍNEZ-ROLÁN, X. Eso no me lo dices en la calle. análisis del discurso del odio contra las mujeres en twitter. **El profesional de la información**, El Profesional de la Informacion, v. 30, n. 5, 2021.

PULVER, A.; LYU, S. Lstm with working memory. *In*: IEEE. **2017 International Joint Conference on Neural Networks (IJCNN)**. [*S.l.*], 2017. p. 845–851.

RAI, N. *et al.* **Fake News Classification using transformer based enhanced LSTM and BERT**. 2022. 98-105 p. Acesso em: 20 de abril de 2024. Disponível em: https://www.sciencedirect.com/science/article/pii/S2666307422000092.

RASCHKA, S.; MIRJALILI, V. **Python Machine Learning**. [*S.l.*]: Packt Publishing Ltd, 2017.

RAUBER, T. W. Redes neurais artificiais. **Universidade Federal do Espírito Santo**, 2002. Acesso em: 18 de março de 2024. Disponível em: https://www.researchgate.net/profile/Thomas-Rauber-2/publication/228686464_Redes_neurais_artificiais/links/02e7e521381602f2bd000000/Redes-neurais-artificiais.pdf.

RECUERO, R. **Redes Sociais na Internet**. Sulina, 2009. Acesso em: 20 de março de 2024. ISBN 978-85-205-0525-0 1. Disponível em: https://e-compos.emnuvens.com.br/e-compos/article/view/28.

RIZZO, I. V.; CANATO, R. L. C. Inteligência artificial: Funções de ativação. **Revista Prospectus**, v. 2, n. 2, p. 51–65, Ago/Fev 2020. Acesso em: 10 de março de 2024. Disponível em: https://www.prospectus.fatecitapira.edu.br/index.php/pst/article/view/37.

ROJAS, R. The backpropagation algorithm. *In*: ____. **Neural Networks: A Systematic Introduction**. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996. p. 149–182. ISBN 978-3-642-61068-4. Acesso em: 15 de abril de 2024. Disponível em: https://doi.org/10.1007/978-3-642-61068-4_7.

RÖTTGER, P. *et al.* HateCheck: Functional tests for hate speech detection models. *In*: ZONG, C. *et al.* (Ed.). **Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)**. Online: Association for Computational Linguistics, 2021. p. 41–58. Disponível em: https://aclanthology.org/2021.acl-long.4.

RöTTGER, P. *et al.* Hatecheck: Functional tests for hate speech detection models. *In*: **Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)**. Association for Computational Linguistics, 2021. Acesso em: 09 de abril de 2024. Disponível em: http://dx.doi.org/10.18653/v1/2021.acl-long.4.

SÁNCHEZ, A. *et al.* Cumbre: corpus lingüístico del español contemporáneo: fundamentos, metodología y aplicaciones. **(No Title)**, 1995. Acesso em: 18 de abril de 2024. Disponível em: https://cir.nii.ac.jp/crid/1130282269262014848.

SANTANA, A.; BAPTISTA, G.; GONÇALVES, M. O discurso de ódio nas redes (anti) sociais: um estudo sobre sua ocorrência. Universidade Presbiteriana Mackenzie, 2022. Disponível em: https://dspace.mackenzie.br/items/24a4e342-fbd5-40f9-82cd-ec58e9f74d22.

SANTOS, V. dos. Neurônios. 2024. Acesso em: 10 de abril de 2024. Disponível em: https://brasilescola.uol.com.br/biologia/neuronios.htm.

SCHLEDER, G. R.; FAZZIO, A. Machine learning na física, química, e ciência de materiais: Descoberta e design de materiais. **Revista Brasileira de Ensino de Física**, SciELO Brasil, v. 43, p. e20200407, 2021. Acesso em: 18 de abril de 2024. Disponível em: https://www.scielo.br/j/rbef/a/qzcfSKw4nzBK5Mddr8ZXy4v/?lang=pt.

SCHULTE, D.; HAMBORG, F.; AKBIK, A. **Less is More: Parameter-Efficient Selection of Intermediate Tasks for Transfer Learning**. 2024. Disponível em: https://arxiv.org/abs/2410.15148.

SHARMA, S.; SHARMA, S.; ATHAIYA, A. Activation functions in neural networks. **Towards Data Sci**, v. 6, n. 12, p. 310–316, 2017. Acesso em: 05 de abril de 2024. Disponível em: https://www.ijeast.com/papers/310-316,Tesma412,IJEAST.pdf.

SHINDE, P. P. A review of machine learning and deep learning applications. *In*: IEEE. **2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)**. 2018. p. 1–4. Acesso em: 10 de abril de 2024. Disponível em: https://ieeexplore.ieee.org/abstract/document/8697857?casa_token=TrM2v7CY3AoAAAAA:w0Oegrc4nLPmEyrMEWQ94b50Uh6icPf0Qu5hKqvhkYw11SLLkw3eZrTewgCb6b9HFKdHmkK8ufY.

SHIRURU, K. An introduction to artificial neural network. **International Journal Of Advance Research And Innovative Ideas In Education**, v. 2, n. 5, p. 26–29, 2016. Acesso em: 15 de abril de 2024. Disponível em: https://www.researchgate.net/publication/319903816_AN_INTRODUCTION_TO_ARTIFICIAL_NEURAL_NETWORK.

SILVA, J. R. *et al.* Ferramenta de detecção de padrões de planicidade utilizando redes neurais. **Rem: Revista Escola de Minas**, SciELO, v. 72, n. 4, p. 347–352, 2019. Acesso em: 05 de abril de 2024. Disponível em: https://www.aedb.br/seget/arquivos/artigos16/23424234.pdf.

SINAGA, K. P.; YANG, M.-S. Unsupervised k-means clustering algorithm. **IEEE Access**, v. 8, p. 80716–80727, 2020. Acesso em: 08 de abril de 2024. Disponível em: https://ieeexplore.ieee.org/abstract/document/9072123.

SMAGULOVA, K.; JAMES, A. P. A survey on lstm memristive neural network architectures and applications. **The European Physical Journal Special Topics**, Springer, v. 228, p. 2313–2324, 2019. Disponível em: https://link.springer.com/article/10.1140/epjst/e2019-900046-x.

SNAILY. **What, Why and Which Activation Functions**. 2019. Acesso em: 08 de abril de 2024. Disponível em: https://medium.com/@snaily16/what-why-and-which-activation-functions-b2bf748c0441.

TAVARES, D. M. Redes neurais. **Campinas: Instituto de Computação da Unicamp**, 2001. Acesso em: 10 de abril de 2024. Disponível em: https://www.dca.ufrn.br/~lmarcos/courses/robotica/notes/RNAs.pdf.

TIAN, H. *et al.* SKEP: Sentiment knowledge enhanced pre-training for sentiment analysis. *In*: JURAFSKY, D. *et al.* (Ed.). **Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics**. Online: Association for Computational Linguistics, 2020. p. 4067–4076. Acesso em: 19 de abril de 2024. Disponível em: https://aclanthology.org/2020.acl-main.374.

TOPAL, M. O.; BAS, A.; HEERDEN, I. van. Exploring transformers in natural language generation: Gpt, bert, and xlnet. *In*: SPRINGER. **International Conference on Interdisciplinary Applications of Artificial Intelligence**. 2021. Acesso em: 18 de abril de 2024. Disponível em: https://arxiv.org/abs/2102.08036.

TORREY, L.; SHAVLIK, J. Transfer learning. **Handbook of research on machine learning applications and trends: algorithms, methods, and techniques**, 2010.

TRINDADE, L. V. **Discurso de ódio nas redes sociais**. [*S.l.*]: Editora Jandaíra, 2022.

TYNES, B. *et al.* From racial microaggressions to hate crimes: A model of online racism based on the lived experiences of adolescents of color: Influence and implications. *In*: ____. [*s.n.*], 2018. p. 194–212. ISBN 9781119420040. Acesso em: 27 de abril de 2024. Disponível em: https://www.researchgate.net/publication/327957993_From_Racial_Microaggressions_to_Hate_Crimes_A_Model_of_Online_Racism_Based_on_the_Lived_Experiences_of_Adolescents_of_Color_Influence_and_Implications.

VICTORIA, J. S. Conceptos y aplicaciones docentes de la inteligencia artificial (ia) y el procesamiento del lenguaje natural (pln). **Preprint**, 2022. Disponível em: https://www.researchgate.net/publication/365797450_CONCEPTOS_Y_APLICACIONES_DOCENTES_DE_LA_INTELIGENCIA_ARTIFICIAL_IA_Y_EL_PROCESAMIENTO_DEL_LENGUAJE_NATURAL_PLN.

VIEIRA, P. S. T. **Xenofobia no Brasil: revisão de literatura e relato de experiência**. 2022. Monografia apresentada ao Instituto de Estudos em Saúde Coletiva, da Universidade Federal do Rio de Janeiro. Acesso em: 10 de abril de 2024. Disponível em: https://pantheon.ufrj.br/handle/11422/17057/.

WANG, M. *et al.* A high-speed and low-complexity architecture for softmax function in deep learning. *In*: **2018 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)**. [*s.n.*], 2018. p. 223–226. Acesso em: 27 de abril de 2024. Disponível em: https://ieeexplore.ieee.org/document/8605654.

WANG, X. Attention mechanism, transformers, bert, and gpt: Tutorial and survey. **arXiv preprint arXiv:2101.01069**, 2021. Acesso em: 20 de abril de 2024. Disponível em: https://paperswithcode.com/paper/attention-mechanism-transformers-bert-and-gpt.

ZHANG, Z.; LUO, L. Hate speech detection: A solved problem? the challenging case of long tail on twitter. **Semantic Web**, IOS Press, v. 10, n. 5, p. 925–945, 2019. Acesso em: 14 de abril de 2024. Disponível em: https://content.iospress.com/articles/semantic-web/sw180338.

ZHU, H. *et al.* Logish: A new nonlinear nonmonotonic activation function for convolutional neural network. **Neurocomputing**, v. 458, p. 490–499, 2021. ISSN 0925-2312. Acesso em: 10 de abril de 2024. Disponível em: https://www.sciencedirect.com/science/article/pii/S0925231221009917.

APÊNDICES

APÊNDICE A – Gráficos da Comparação entre LSTM e RoBERTa

Essa seção proporciona uma compreensão clara de como os dois modelos se comportaram em termos de capacidade discriminativa, sendo um ponto importante para ilustrar as diferenças no desempenho dos modelos.

Este apêndice contém todos os gráficos gerados durante a comparação entre os modelos LSTM e RoBERTa, conforme discutido no texto principal da tese. Os gráficos apresentados são resultados das análises experimentais realizadas e fornecem uma visão detalhada sobre o desempenho de ambos os modelos em diferentes métricas de épocas e tamanhos de lote. Esses gráficos são apresentados para fornecer uma visão mais clara das diferenças de desempenho entre os dois modelos e permitir uma comparação visual das suas características e resultados.

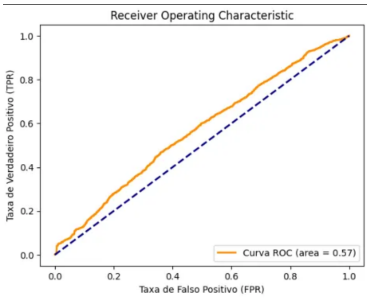
A.1 Tempo de processamento

Durante o treinamento, o tempo de execução da arquitetura roBERTa durou algumas horas devido à necessidade de processar grandes volumes de dados e realizar cálculos complexos, como operações de atenção multi-cabeça e atualizações nos gradientes para otimizar a função de perda. Além disso, o uso de modelos maiores e a necessidade de ajustar hiperparâmetros, como o número de épocas de treinamento e o tamanho do lote, contribuíram para a duração do treinamento. Dada a complexidade computacional do desse modelo, o uso de *hardware* especializado tornou-se essencial para reduzir significativamente o tempo de execução e permitir que o modelo fosse aplicado de maneira eficiente. A GPU A100, foi especialmente eficazes para tarefas de treinamento de modelos de aprendizado profundo. Oferecendo alta capacidade de memória e alto poder de processamento. Em contrapartida, a LSTM têm uma arquitetura relativamente simples em comparação com os modelos baseados em *Transformer*. Isso se traduziu em um menor custo computacional para o treinamento.

A.2 Curvas ROC LSTM

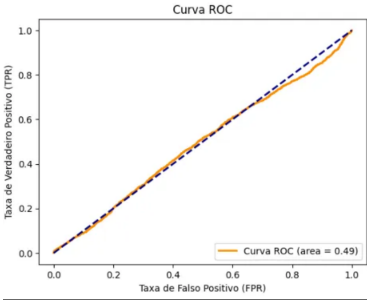
A.3 Curvas ROC roBERTa

Figura 21 – LSTM - 20 épocas e batch size 128



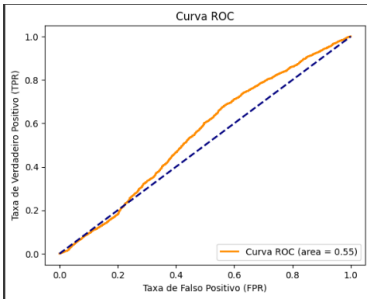
Fonte: Autoria própria (2025).

Figura 22 – LSTM - 25 épocas e batch size 128



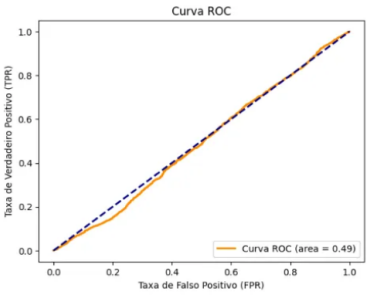
Fonte: Autoria própria (2025).

Figura 23 – LSTM - 28 épocas e batch size 200



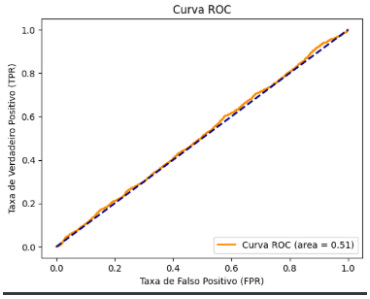
Fonte: Autoria própria (2025).

Figura 24 – LSTM - 20 épocas e batch size 200



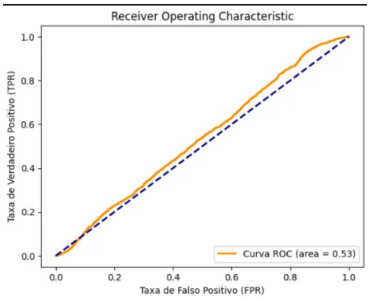
Fonte: Autoria própria (2025).

Figura 25 – LSTM - 25 épocas e batch size 200



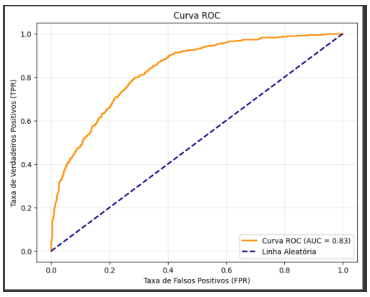
Fonte: Autoria própria (2025).

Figura 26 – LSTM - 28 épocas e batch size 200



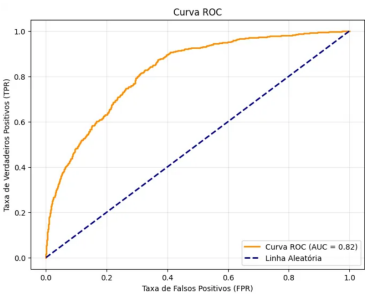
Fonte: Autoria própria (2025).

Figura 27 – RoBERTa - 3 épocas e batch size 16



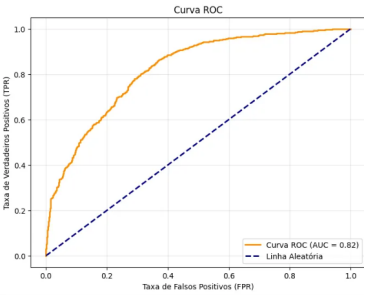
Fonte: Autoria própria (2025).

Figura 28 – RoBERTa - 5 épocas e batch size 16



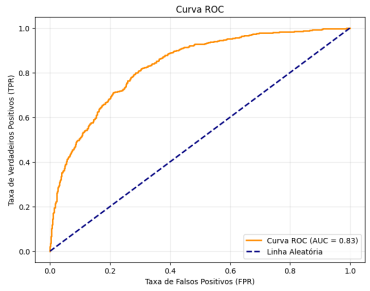
Fonte: Autoria própria (2025).

Figura 29 – RoBERTa - 10 épocas e batch size 16



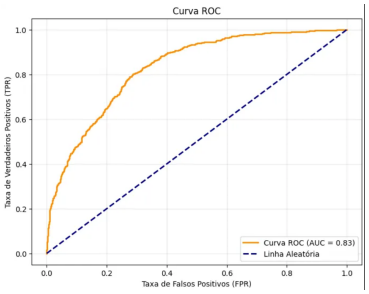
Fonte: Autoria própria (2025).

Figura 30 – RoBERTa - 3 épocas e batch size 32



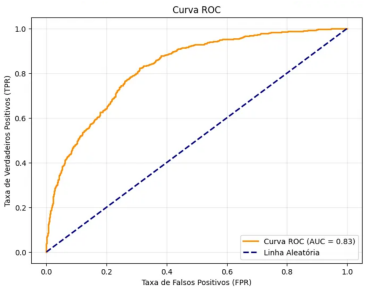
Fonte: Autoria própria (2025).

Figura 31 – RoBERTa - 5 épocas e batch size 32



Fonte: Autoria própria (2025).

Figura 32 – RoBERTa - 5 épocas e batch size 32



Fonte: Autoria própria (2025).