UNIVERSIDADE FEDERAL DO RIO DE JANEIRO

INSTITUTO DE COMPUTAÇÃO

CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

AUGUSTO GUIMARÃES RODRIGUES DE LIMA

AUMENTO DE DADOS POR MASCARAMENTO NO BERT PARA DETECÇÃO DE COMENTÁRIOS OFENSIVOS

RIO DE JANEIRO

2026

AUGUSTO GUIMARÃES RODRIGUES DE LIMA

AUMENTO DE DADOS POR MASCARAMENTO NO BERT PARA DETECÇÃO DE COMENTÁRIOS OFENSIVOS

Trabalho de conclusão de curso de graduação apresentado ao Instituto de Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. João Carlos Pereira da Silva

RIO DE JANEIRO

2026

CIP - Catalogação na Publicação

Lima, Augusto Guimarães Rodrigues de
L732a Aumento de dados por mascaramento no BERT para detecção de comentários ofensivos / Augusto Guimarães Rodrigues de Lima. -- Rio de Janeiro, 2026.
71 f.

Orientador: João Carlos Pereira da Silva.
Trabalho de conclusão de curso (graduação) - Universidade Federal do Rio de Janeiro, Instituto de Computação, Bacharel em Ciência da Computação, 2026.

1. Processamento de linguagem natural. 2. BERT. 3. Detecção de linguagem ofensiva. I. Silva, João Carlos Pereira da, orient. II. Título.

Elaborado pelo Sistema de Geração Automática da UFRJ com os dados fornecidos pelo(a) autor(a), sob a responsabilidade de Miguel Romeu Amorim Neto - CRB-7/6283.

AUGUSTO GUIMARÃES RODRIGUES DE LIMA

AUMENTO DE DADOS POR MASCARAMENTO NO BERT PARA DETECÇÃO DE COMENTÁRIOS OFENSIVOS

Trabalho de conclusão de curso de graduação apresentado ao Instituto de Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em 07 de abril de 2026.

BANCA EXAMINADORA:

Documento assinado digitalmente
gov.br JOAO CARLOS PEREIRA DA SILVA
Data: 15/06/2026 17:07:53-0300
Verifique em https://validar.iti.gov.br

João Carlos Pereira da Silva
D.Sc (Instituto de Computação - UFRJ)

Documento assinado digitalmente
gov.br GISELI RABELLO LOPES
Data: 15/06/2026 17:18:12-0300
Verifique em https://validar.iti.gov.br

Giseli Rabello Lopes
D. Sc. (Instituto de Computação - UFRJ)

Documento assinado digitalmente
gov.br SILAS PEREIRA LIMA FILHO
Data: 16/06/2026 09:05:15-0300
Verifique em https://validar.iti.gov.br

Silas Lima Filho ,
D. Sc. (Instituto de Computação - UFRJ)

Dedico este trabalho à minha família, em especial aos meus pais, pelo apoio incondicional, incentivo constante e por acreditarem em mim ao longo de toda a minha trajetória acadêmica. Dedico também à minha irmã Ludmila, pelo carinho, apoio e motivação que foram fundamentais durante esse percurso. À minha namorada, Bruna Rieken, pelo amor, companheirismo, paciência e por tornar essa caminhada mais leve e especial

AGRADECIMENTOS

Agradeço, primeiramente, ao professor João Carlos, pela orientação, paciência, dedicação e valiosas contribuições ao longo do desenvolvimento desta monografia, que foram essenciais para a concretização deste trabalho.

Agradeço à instituição de ensino e ao Instituto de Computação (IC), pelos anos de aprendizado, pela formação acadêmica e pelas oportunidades proporcionadas ao longo desse período de estudos, que contribuíram de forma significativa para o meu crescimento acadêmico e profissional.

Por fim, agradeço a todos que, direta ou indiretamente, contribuíram para a realização deste trabalho.

RESUMO

A detecção automática de linguagem ofensiva em redes sociais tem se tornado um desafio relevante no contexto do Processamento de Linguagem Natural, especialmente em língua portuguesa, devido à escassez de recursos linguísticos rotulados e à natureza dinâmica da comunicação *online*. Este trabalho investiga o uso de técnicas de aumento de dados baseadas em mascaramento de palavras no modelo BERT como estratégia para aprimorar a detecção de comentários ofensivos em português brasileiro. O estudo utiliza o corpus HateBR, composto por comentários do *Instagram*, bem como um conjunto de dados adicional de comentários coletados da plataforma X (antigo *Twitter*), ampliando a diversidade de contextos discursivos analisados. Emprega-se o modelo BERTimbau como base para a geração de sentenças sintéticas semanticamente coerentes. Foram propostas e avaliadas três abordagens de enriquecimento de dados: o mascaramento tradicional de múltiplos *tokens*, o mascaramento controlado de palavras inteiras e uma estratégia conservadora baseada na substituição de um único *token* por iteração. Para garantir a preservação do significado original e dos rótulos de classe, as sentenças geradas foram filtradas por meio da similaridade do cosseno calculada com *embeddings* do Sentence-BERT, considerando diferentes faixas de similaridade semântica. Os experimentos demonstraram que, embora o BERTimbau apresente desempenho robusto mesmo sem aumento de dados, a estratégia de substituição de um único *token* aliada a uma faixa ampla de similaridade semântica produziu melhorias consistentes nas métricas de acurácia, precisão, *recall* e *F1-score*. Além disso, observou-se que os ganhos decorrentes do aumento de dados foram mais visíveis em modelos clássicos de aprendizado de máquina, como SVM, Regressão Logística e MLP, evidenciando que o enriquecimento semântico contribui de forma significativa para classificadores que não possuem mecanismos avançados de contextualização. Conclui-se que o mascaramento controlado de palavras, aliado à filtragem semântica, constitui uma estratégia eficaz para ampliar a variabilidade dos dados de treinamento, preservar a coerência linguística e melhorar o desempenho de modelos de detecção de linguagem ofensiva em português.

Palavras-chave: processamento de linguagem natural; linguagem ofensiva; aumento de dados; BERTimbau; aprendizado de máquina.

ABSTRACT

The automatic detection of offensive language on social media has become a relevant challenge in the context of Natural Language Processing, especially for the Portuguese language, due to the scarcity of labeled linguistic resources and the dynamic nature of online communication. This work investigates the use of data augmentation techniques based on word masking in the BERT model as a strategy to improve the detection of offensive comments in Brazilian Portuguese. The study employs the HateBR corpus, composed of Instagram comments, as well as an additional dataset of comments collected from the X platform (formerly Twitter), broadening the diversity of discourse contexts analyzed. The BERTimbau model is used as the basis for generating semantically coherent synthetic sentences. Three data augmentation approaches were proposed and evaluated: traditional multi-token masking, controlled masking of whole words, and a conservative strategy based on single-token substitution per iteration. To ensure the preservation of the original meaning and class labels, the generated sentences were filtered using cosine similarity computed from Sentence-BERT embeddings, considering different semantic similarity ranges. Experimental results show that, although BERTimbau presents robust performance even without data augmentation, the single-token substitution strategy combined with a broader semantic similarity range produced consistent improvements in accuracy, precision, recall, and F1-score. Furthermore, the gains obtained from data augmentation were more expressive for classical machine learning models, such as SVM, Logistic Regression, and MLP, indicating that semantic enrichment significantly benefits classifiers that do not incorporate advanced contextualization mechanisms. It is concluded that controlled word masking combined with semantic filtering is an effective strategy to increase training data variability, preserve linguistic coherence, and improve the performance of offensive language detection models in Portuguese.

Keywords: natural language processing; offensive language; data augmentation; BERTimbau; machine learning.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

API – Application Programming Interface

BERT – Bidirectional Encoder Representations from Transformers

HateBR – Corpus brasileiro para detecção de discurso de ódio

MLP – Multi-Layer Perceptron

NLTK – Natural Language Toolkit

PLN – Processamento de Linguagem Natural

SVM – Support Vector Machine

SUMÁRIO

1 INTRODUÇÃO

As redes sociais digitais consolidaram-se como um dos principais meios de comunicação, interação e compartilhamento de informações na sociedade contemporânea. Plataformas como *Instagram*, X e *Facebook* ampliaram significativamente o alcance das opiniões individuais e coletivas, transformando-se em espaços centrais de debate público. Entretanto, o crescimento expressivo do número de usuários e do volume de conteúdo gerado também intensificou problemas sociais, entre os quais se destaca a disseminação de linguagem ofensiva e tóxica, frequentemente associada a insultos, ameaças, discursos discriminatórios e manifestações de ódio direcionadas a indivíduos ou grupos.

A linguagem ofensiva pode ser compreendida como qualquer forma de comunicação inadequada ou agressiva, que varia desde o uso explícito de palavrões até ataques indiretos e implícitos. Já o discurso de ódio caracteriza-se por expressões públicas que promovem ou incentivam hostilidade, discriminação ou violência com base em atributos como raça, etnia, nacionalidade, religião, gênero ou orientação sexual. Esse tipo de conteúdo, além de comprometer a qualidade do debate público, pode gerar impactos sociais profundos e duradouros, configurando-se como uma preocupação relevante para plataformas digitais, governos e a sociedade em geral.

No contexto brasileiro, embora práticas discriminatórias e discursos de ódio sejam tipificados como crime, observa-se um crescimento expressivo desse tipo de manifestação no ambiente digital. O elevado volume de publicações, aliado à velocidade de propagação das mensagens e às estratégias criativas utilizadas pelos usuários para contornar sistemas de moderação, torna inviável a fiscalização exclusivamente humana. Assim, a identificação e a classificação automáticas de comentários ofensivos emergem como tarefas fundamentais, porém desafiadoras, no âmbito do Processamento de Linguagem Natural (PLN).

Grande parte das abordagens propostas para a detecção de linguagem ofensiva baseia-se em técnicas de Aprendizado de Máquina Supervisionado, que dependem de bancos de dados extensos e devidamente rotulados. Contudo, a construção manual desses conjuntos de dados é onerosa, demorada e fortemente influenciada por fatores subjetivos, como interpretações individuais, posicionamentos ideológicos e viés cultural dos anotadores. Anotações inconsistentes ou enviesadas podem comprometer diretamente o desempenho e a confiabilidade dos modelos treinados, tornando a escassez de dados de qualidade um dos principais obstáculos para o avanço da área, especialmente em língua portuguesa.

Nesse cenário, estratégias que visam mitigar a limitação de dados rotulados ganham destaque, entre elas o aumento artificial de dados e o uso de modelos de linguagem pré-treinados. Arquiteturas baseadas em transformadores, como o BERT (DEVLIN *et al*., 2019), representam um marco no PLN por sua capacidade de modelar relações semânticas complexas de forma contextualizada. A adaptação desses modelos para o português, como o BERTimbau (SOUZA; NOGUEIRA; LOTUFO, 2020), possibilitou avanços significativos em tarefas de classificação textual, incluindo a detecção de linguagem ofensiva.

Este trabalho insere-se nesse contexto e investiga o uso de técnicas de aumento de dados baseadas em mascaramento de palavras com o objetivo de aprimorar a detecção automática de comentários ofensivos em português brasileiro. Para isso, utiliza-se o corpus HateBR (VARGAS *et al.*, 2022), composto por comentários extraídos do *Instagram*. Adicionalmente, foi construído um conjunto de dados independente a partir de comentários coletados na plataforma X, empregado para fins comparativos e análise de robustez dos modelos. Explora-se o potencial do BERTimbau para a geração de sentenças sintéticas semanticamente coerentes. Além disso, emprega-se a similaridade semântica calculada a partir de *embeddings* do Sentence-BERT (REIMERS; GUREVYCH, 2019) como mecanismo de filtragem, visando preservar o significado original e os rótulos das classes.

Por fim, este trabalho busca contribuir para a área ao analisar o impacto do aumento de dados no desempenho de modelos baseados em aprendizado profundo (*Deep Learning*) e em algoritmos clássicos de aprendizado de máquina, oferecendo evidências de que estratégias controladas de enriquecimento semântico podem representar uma alternativa eficaz para contornar a escassez de dados rotulados e melhorar a robustez de sistemas de detecção de linguagem ofensiva em português.

Este trabalho está organizado da seguinte forma: o Capítulo 2 apresenta a fundamentação teórica que sustenta os conceitos e técnicas utilizados ao longo do estudo. O Capítulo 3 discute os trabalhos relacionados, contextualizando a pesquisa no estado da arte da detecção de linguagem ofensiva em português. O Capítulo 4 descreve a metodologia adotada, incluindo os materiais, o conjunto de dados e as estratégias de enriquecimento de dados empregadas. O Capítulo 5 apresenta e discute os resultados experimentais obtidos. Por fim, o Capítulo 6 apresenta as conclusões do trabalho e as perspectivas para pesquisas futuras.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os principais fundamentos teóricos que sustentam este trabalho, abordando conceitos relacionados à detecção de comentários ofensivos e ao uso de técnicas de mascaramento de palavras no modelo BERT (DEVLIN *et al*., 2019) como estratégia de aumento de dados (*data augmentation*). São discutidos a definição de linguagem ofensiva no contexto do conjunto de dados HateBR (VARGAS *et al.*, 2022), os princípios do Processamento de Linguagem Natural (PLN) e do aprendizado profundo, com destaque para as representações vetoriais de palavras (*word embeddings*) e sua função na representação semântica de palavras. Além disso, o capítulo explora o funcionamento do modelo BERT e sua arquitetura de transformadores (*transformers*) proposta por Vaswani *et al.* (2017), bem como técnicas de aumento de dados em PLN, como o método *Easy Data Augmentation* (EDA) proposto por Wei e Zou (2020). Por fim, aborda-se a similaridade do cosseno como métrica para comparar representações vetoriais, consolidando o embasamento teórico necessário para o uso do mascaramento de palavras como estratégia de aprimoramento na detecção de linguagem ofensiva.

2.1 LINGUAGEM OFENSIVA

No contexto dos estudos de detecção automática de conteúdo ofensivo em língua portuguesa, Vargas *et al.* (2022), autores do *corpus* HateBR, definiram linguagem ofensiva como qualquer forma de comunicação que contenha termos ou expressões com conotação pejorativa, incluindo palavrões, que podem ser explícitos ou implícitos. Enquadram-se nessa definição mensagens que apresentem insultos, xingamentos, expressões depreciativas, humilhações ou formas de agressividade verbal, independentemente de o alvo ser coletivo, como um grupo social, ou individual.

Essa concepção está alinhada a definições institucionais adotadas por órgãos como o Ministério dos Direitos Humanos e da Cidadania, que caracterizam manifestações ofensivas como formas de violência simbólica ou verbal que atentam contra a dignidade humana, frequentemente associadas à discriminação, desrespeito ou inferiorização de indivíduos ou grupos (BRASIL, 2018).

Essa definição foi adotada para a construção do HateBR, que reúne comentários do *Instagram* anotados por especialistas com formação em linguística, análise de discurso,

processamento de linguagem natural e aprendizado de máquina. Durante a anotação, cada texto foi examinado quanto à presença de termos, expressões ou marcas linguísticas com conotação depreciativa. Assim, comentários que apresentavam ao menos um termo ou expressão pejorativa foram classificados como ofensivos, enquanto aqueles que não continham qualquer referência ou inferência depreciativa foram rotulados como não ofensivos. Essa distinção pode ser observada na Tabela 1: nos comentários "*Mais um lixo*" e "*Porque é uma bandida*", as palavras *lixo* e *bandida* ilustram claramente o tipo de conteúdo pejorativo que caracteriza a categoria ofensiva. Já os demais exemplos não apresentam marcas linguísticas depreciativas e, portanto, foram classificados como não ofensivos.

Tabela 1: Exemplos de comentários classificados como ofensivos e não ofensivos extraídos do HateBR

COMENTÁRIO	OFENSIVO
Mais um lixo	1
Parabéns meu presidente!	0
Porque é uma bandida	1
Lula Livre!	0

Fonte: Elaborado pelo autor

2.2 Processamento de linguagem natural

O Processamento de Linguagem Natural (PLN) é uma área dentro da inteligência artificial e da linguística que se concentra em permitir que os computadores compreendam expressões ou palavras em línguas humanas, conhecidas como línguas naturais. Suas aplicações se expandiram em diversos campos, como tradução automática, detecção de *spam* em e-mails, extração de informações, sumarização e sistemas de diálogo, etc. (KHURANA *et al.*, 2017).

Os conceitos de PLN, aplicados aos princípios de inteligência artificial, são fundamentais neste trabalho, pois permitem a utilização de modelos baseados em transformadores, como o BERT, para a geração e ampliação de dados por meio de técnicas de mascaramento de palavras. Essa abordagem contribui diretamente para aprimorar a detecção automática de comentários ofensivos em textos.

2.2.1 Métodos de Classificação

A classificação de texto é uma tarefa essencial dentro do PLN, cujo objetivo é atribuir rótulos ou categorias a um texto com base em seu conteúdo (Vajjala *et al.*, 2020). Este processo de classificação é geralmente formulado como uma tarefa supervisionada, na qual modelos computacionais são ajustados com base em exemplos previamente anotados, de modo a aprender relações entre características textuais e classes associadas. O desempenho desses métodos depende de fatores como a forma de representação do texto, a complexidade do modelo adotado e sua capacidade de generalizar para dados não vistos (JURAFSKY; MARTIN, 2023).

Neste trabalho, são adotados métodos clássicos em tarefas de classificação textual com o objetivo de servir como base comparativa para a avaliação de modelos baseados em representações contextuais, em especial o BERT, os quais são descritos nas subseções seguintes

2.2.1.1 Regressão Logística

A Regressão Logística é uma técnica estatística que estima a probabilidade de uma instância pertencer a uma determinada classe com base em uma combinação linear das variáveis de entrada. É um método de classificação supervisionada amplamente utilizado em tarefas de categorização binária, ou seja, quando há duas classes possíveis. Apesar do nome "regressão", ela é usada para resolver problemas de classificação, e não de regressão linear no sentido tradicional que prevê valores contínuos (JAMES *et al.*, 2013).

O objetivo da Regressão Logística é encontrar uma equação que estime a probabilidade de uma determinada entrada pertencer a uma classe específica. Essa equação se baseia na função logística (ou sigmóide), que transforma valores numéricos em um intervalo entre 0 e 1 (BISHOP, 2006). Se o resultado for maior que 0,5, o modelo classifica como classe 1 (por exemplo, um comentário ofensivo); caso contrário, classifica como classe 0 (não-ofensivo).

A fórmula da função sigmóide é:

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

onde $z = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b$, sendo xi os atributos do texto, wi os pesos aprendidos pelo modelo, e b o viés.

Entre as principais vantagens da Regressão Logística destaca-se sua simplicidade conceitual e facilidade de interpretação, uma vez que o modelo permite compreender a influência de cada variável de entrada sobre a decisão final. Além disso, apresenta bom desempenho em cenários nos quais as classes são aproximadamente linearmente separáveis, sendo amplamente utilizadas como método de base em tarefas de classificação textual. No entanto, a Regressão Logística possui limitações importantes, pois seu desempenho tende a ser prejudicado quando os dados apresentam relações não lineares complexas. Ademais, o modelo pode ser sensível à presença de atributos irrelevantes ou altamente correlacionados, o que pode impactar negativamente sua capacidade de generalização caso não haja um adequado processo de seleção ou regularização das características (HASTIE; TIBSHIRANI; FRIEDMAN, 2009).

2.2.1.2 Máquina de Vetores de Suporte

A Máquina de Vetores de Suporte (*Support Vector Machine* – SVM) é um método de classificação supervisionada amplamente empregado em tarefas de Processamento de Linguagem Natural, sobretudo em problemas de classificação textual. Seu objetivo é encontrar uma fronteira de decisão que separe instâncias pertencentes a classes distintas a partir de uma representação vetorial dos dados de entrada (CORTES; VAPNIK, 1995).

O princípio central do SVM consiste na determinação de um hiperplano que maximize a margem entre as classes. Essa margem corresponde à distância entre a fronteira de decisão e os exemplos mais próximos de cada classe, denominados vetores de suporte. Ao buscar a maximização dessa distância, o modelo procura estabelecer uma separação estável entre as classes, reduzindo a influência de variações individuais nos dados.

Em sua formulação linear, a função de decisão do SVM pode ser expressa como:

$$f(x) = w \cdot x + b$$

onde x representa o vetor de atributos da instância analisada, w corresponde ao vetor de pesos aprendido durante o processo de treinamento, e b é o termo de viés. A classificação de novas instâncias é realizada a partir do sinal do valor retornado por essa função, indicando a classe atribuída ao exemplo.

Em problemas de classificação textual, o SVM é frequentemente utilizado como modelo de base em estudos comparativos, especialmente por sua formulação baseada em margens e pela forma como utiliza apenas um subconjunto dos exemplos de treinamento para definir a fronteira de decisão. Entretanto, assim como outros classificadores lineares, seu desempenho pode ser limitado em cenários nos quais a separação entre as classes apresenta alta complexidade estrutural (HASTIE; TIBSHIRANI; FRIEDMAN, 2009).

2.2.1.3 Perceptron Multicamadas

O Perceptron Multicamadas (*Multilayer Perceptron* – MLP) é um método de classificação supervisionada amplamente utilizado em tarefas de classificação textual, especialmente em cenários nos quais a relação entre os atributos de entrada e as classes não pode ser adequadamente modelada por fronteiras de decisão estritamente lineares (BISHOP, 2006). O MLP pode ser entendido como uma generalização do Perceptron clássico, modelo de classificação supervisionada proposto por Rosenblatt (1958), cujo objetivo é separar instâncias pertencentes a duas classes por meio de uma combinação linear dos atributos de entrada. Embora conceitualmente simples, o Perceptron clássico é restrito à aprendizagem de fronteiras de decisão linearmente separáveis, o que limita sua capacidade de representação em problemas mais complexos (ROSENBLATT, 1958).

De forma geral, o MLP realiza a transformação dos vetores de atributos por meio de uma sequência de operações matemáticas compostas por combinações lineares seguidas da aplicação de funções não lineares. Essas transformações intermediárias permitem ao modelo capturar padrões mais complexos nos dados, tornando-o adequado para problemas em que as classes apresentam separações não triviais no espaço vetorial (HASTIE; TIBSHIRANI; FRIEDMAN, 2009).

Matematicamente, cada transformação intermediária pode ser expressa como:

$$h(x) = g(Wx + b)$$

onde x representa o vetor de entrada, W corresponde à matriz de pesos ajustada durante o treinamento, b é o termo de viés e $g( )$ é uma função não linear aplicada elemento a elemento. A saída final do modelo é obtida após a aplicação sucessiva dessas transformações, sendo então utilizada para estimar a classe associada à instância analisada.

Entre as principais vantagens do MLP destaca-se sua flexibilidade na modelagem de relações não lineares, o que pode resultar em melhorias de desempenho em conjuntos de dados com maior complexidade estrutural. Entretanto, o modelo também apresenta limitações relevantes, como maior sensibilidade à escolha de hiperparâmetros, risco de sobreajuste em bases pequenas e maior custo computacional quando comparado a métodos lineares. Assim como outros classificadores, seu desempenho depende fortemente da qualidade da representação vetorial adotada e da quantidade de dados disponíveis para treinamento (HASTIE; TIBSHIRANI; FRIEDMAN, 2009).

2.2.2 Aprendizado Profundo

Aprendizado profundo ou *deep learning* é um conjunto de algoritmos de aprendizado de máquina que permitem que modelos computacionais com múltiplas camadas de processamento aprendam representações de dados em vários níveis de abstração. Inspirado no funcionamento do cérebro humano, o aprendizado profundo utiliza redes neurais artificiais (RNA) para processar grandes quantidades de dados (DENG; YU, 2014).

Uma rede neural artificial é composta por diversas unidades de processamento, chamadas de neurônios, interconectadas por canais de comunicação associados a pesos. Esses pesos determinam a importância das conexões entre os neurônios e são ajustados durante o processo de aprendizado por meio de algoritmos como a retropropagação (GOODFELLOW; BENGIO; COURVILLE, 2016). Assim, a rede consegue aprender padrões complexos e não lineares presentes nos dados, aprimorando continuamente seu desempenho conforme é exposta a novos exemplos (LECUN; BENGIO; HINTON, 2015).

A utilização das técnicas e métodos de *deep learning* foi a base para o processamento computacional abordado nesta monografia, especialmente o uso de modelos baseados em transformadores. Em particular, empregou-se o BERTimbau como rede neural profunda para a predição de palavras mascaradas e para o posterior ajuste fino na tarefa de classificação de comentários ofensivos.

2.2.3 *Word Embedding*

No campo de Aprendizado de Máquina, as RNAs não são capazes de lidar com texto em sua forma bruta e, por isso, cada parte da frase deve ser previamente convertida em uma

representação numérica. Nesse processo, o texto é segmentado em *tokens*, menor parte da frase a ser processada individualmente, podendo ser palavras, expressões, sequências de letras, entre outras (JURAFSKY, 2000).

A partir desses *tokens*, são gerados vetores conhecidos como *word embeddings*, nos quais cada posição possui um valor numérico que estabelece relações do *token* com um contexto específico (BENGIO et al., 2003).

Um exemplo simplificado de embedding para o *token "rei"* poderia ser expresso como:

$$\text{Rei} = [0.30; 0.10; 0.95; 0.55; 0.65]$$

Nesse caso ilustrativo, o vetor possui apenas cinco dimensões, embora, na prática, modelos modernos utilizam vetores com centenas de componentes. Essas dimensões não correspondem a características linguísticas interpretáveis de forma isolada. São parâmetros aprendidos automaticamente durante o treinamento do modelo. O significado das palavras emerge da combinação dos valores em todas as dimensões, que refletem regularidades aprendidas a partir do uso das palavras durante o treinamento, e não de atributos linguísticos explícitos previamente definidos.

Em alguns modelos, como é o caso do BERT , que será abordado posteriormente neste trabalho, é possível utilizar um *token* especial para cada sequência a ser codificada, desta maneira a saída correspondente a este *token* traz reunidas informações correspondentes a sequência inteira e pode ser utilizada para a classificação do texto (DEVLIN et al., 2018).

2.2.4 Arquitetura do Transformador

O *transformer* é uma arquitetura de rede neural para dados sequenciais proposta por Vaswani *et al*. (2017), desenvolvida inicialmente para resolver tarefas de Processamento de Linguagem Natural. Esta arquitetura utiliza um tipo de mecanismo de atenção chamado de autoatenção (*self-attention*), que também foi proposto por Vaswan*i et al.* (2017).

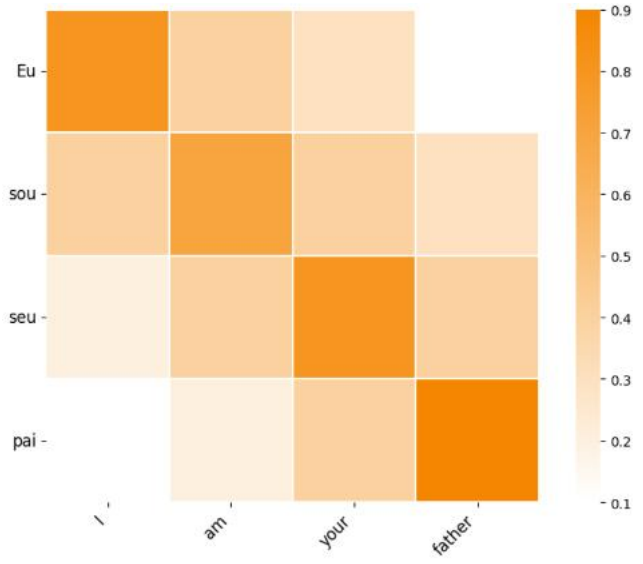
O mecanismo de atenção foi originalmente introduzido no contexto da tradução automática neural, com o objetivo de permitir que os modelos aprendessem a identificar, em cada etapa do processamento, quais partes da sequência de entrada são mais relevantes para gerar uma determinada saída (BAHDANAU; CHO; BENGIO, 2014). Este mecanismo calcula escores e pesos de atenção para cada elemento da sequência de entrada, refletindo o nível de

importância que deve ser atribuída a estes elementos. Os escores são normalmente calculados com base no relacionamento entre o elemento de entrada e os elementos da sequência que está sendo atendida.

Um exemplo simples ocorre na tradução da sentença *"I am your father!"* para *"eu sou seu pai!"*. Ao gerar a palavra "pai", o modelo deve atribuir maior peso de atenção à palavra "*father*", já que ambas carregam o mesmo significado. A Figura 1 ilustra esse processo por meio de uma matriz de atenção, na qual cada linha corresponde a um *token* da sequência de saída e cada coluna a um *token* da sequência de entrada. Os valores apresentados em cada célula representam os pesos de atenção atribuídos pelo modelo, indicando a contribuição relativa de cada *token* de entrada no processo de tradução.

Apesar de o mecanismo de atenção ter melhorado consideravelmente o desempenho dos modelos sequenciais tradicionais, eles ainda sofriam com problemas como desaparecimento do gradiente e limitações na captura de dependências distantes (GOODFELLOW; BENGIO; COURVILLE, 2016). Esses fatores levaram ao surgimento do mecanismo de *self-attention*.

Figura 1: Visualização dos Pesos de Atenção em Tradução



Fonte: Elaborado pelo autor

No caso da autoatenção, esses pesos são calculados entre os elementos da própria sequência, permitindo que o modelo estabeleça relações internas de dependência. Um exemplo ilustrativo pode ser visto na frase *"Você deveria acabar com os Siths, não se unir a eles!"*. No mecanismo de *self-attention*, espera-se que o *token* "*eles*" atribua um alto peso de

atenção ao token "*Siths*", pois ambos representam o mesmo grupo dentro do contexto da frase.

Um ponto importante é que a autoatenção é calculada de maneira independente para cada *token*, possibilitando processamento paralelo, ao contrário de modelos recorrentes, que dependem da ordem sequencial. Essa característica torna o treinamento mais eficiente e permite que o modelo aprenda dependências de longo alcance no texto, contribuindo diretamente para o desempenho superior dos *transformers* (VASWANI *et al.*, 2017).

2.3 BERT

Proposto por Devlin *et al*. (2018), o BERT, acrônimo de *Bidirectional Encoder Representations from Transformers*, é uma aplicação da arquitetura de transformadores. Trata-se de rede neural profunda projetada para treinar modelos de linguagem, através do processamento bidirecional de documentos não rotulados, considerando os contextos das palavras em ambas as direções.

Esse processamento bidirecional permite que o modelo considere simultaneamente informações provenientes de *tokens* anteriores e posteriores, resultando em interpretações mais precisas do significado de cada palavra dentro de uma sentença. Com isso, o BERT consegue identificar o sentido correto de termos que possuem múltiplas interpretações a partir do contexto da sentença. Por exemplo, na frase *"Ele foi ao banco sacar dinheiro"*, o BERT é capaz de compreender que o termo *"banco"* se refere a uma instituição financeira, e não ao ato de sentar-se, justamente porque avalia todo o contexto em torno da palavra.

O BERT adota um pacote padronizado para a entrada das sentenças usadas nas diversas tarefas que pode realizar. Tal pacote converte sentenças de entrada em sequências de *tokens*, que podem se referir a palavras ou subpalavras. Cada *token* é convertido em um identificador numérico (*input ID*) e, posteriormente, mapeado para um vetor denso (por exemplo, de 768 dimensões em versões base do BERT), gerando os *word embeddings* utilizados pela rede.

Esse processo pode ser visualizado na Figura 2, que apresenta a geração de word *embeddings* a partir da versão base do BERT: a frase "Luke, eu sou seu pai!" é tokenizada, convertida em *input* IDs e posteriormente transformada em vetores densos, os quais capturam informações semânticas e contextuais de cada palavra dentro do seu respectivo contexto na sentença.

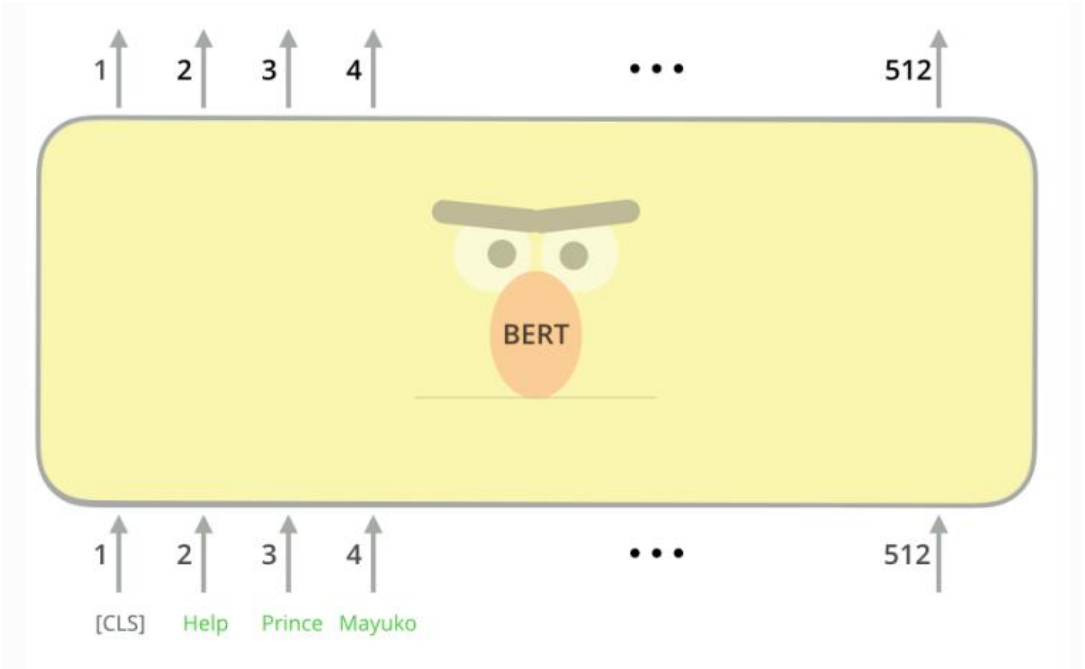
Figura 2: Geração de *embeddings* a partir da frase *"Luke, eu sou seu pai!"*

```
Sentence:  Luke, eu sou seu pai!
Tokens: ['[CLS]', 'Luke', ',', 'eu', 'sou', 'seu', 'pai', '!', '[SEP]']
Input IDs: [[101, 16529, 117, 2779, 7206, 347, 1568, 106, 102]]
word_embeds shape: torch.Size([1, 9, 768])
word_embeds: tensor([[[-0.0004,  0.0014,  0.0251,  ..., -0.0138, -0.0190,
0.0076],
         [-0.0259, -0.0454,  0.0560,  ..., -0.0062,  0.0031,  0.0233],
         [ 0.0242,  0.0097, -0.0009,  ...,  0.0237,  0.0058,  0.0201],
         ...,
         [ 0.0089, -0.0229, -0.0211,  ...,  0.0719, -0.0015, -0.0591],
         [ 0.0474,  0.0398,  0.0324,  ..., -0.0420, -0.0504, -0.0027],
         [ 0.0009,  0.0682,  0.0373,  ..., -0.0029,  0.0256,  0.0049]]])
```

Fonte: Elaborado pelo autor

A estrutura geral desse processo, desde a entrada dos *tokens* até a geração das representações vetoriais, pode ser observada na Figura 3.

Figura 3: Representação geral da arquitetura de entrada e geração de *embeddings* no BERT



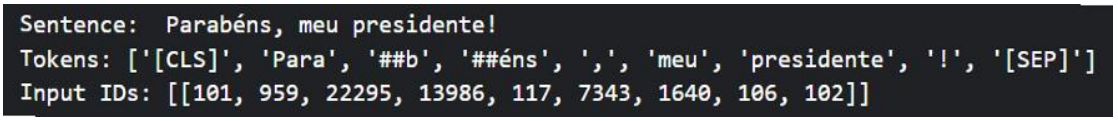
Fonte: Adaptado de Alammar (2018)

O sistema de *tokens* utilizado é o proposto por Wu et al. (2016) com o algoritmo *WordPiece*. Nesse processo, cada palavra pode ser representada integralmente por um único

token ou fragmentada em subpalavras, quando não consta no vocabulário ou é pouco frequente. O objetivo do *WordPiece* é reduzir a quantidade de *tokens* necessários para cobrir todo o *corpus*, permitindo que diferentes palavras sejam formadas pela combinação de subpalavras, o que melhora o tratamento de termos desconhecidos (WU *et al*., 2016).

Por exemplo, na Figura 4 observa-se a tokenização da frase *"Parabéns, meu presidente!"*. A palavra *"Parabéns"* não aparece integralmente no vocabulário do modelo e, portanto, é dividida em três subpalavras: *"Para"*, *"##b"* e *"##éns"*. Os dois sinais de cerquilha (##) indicam que o *token* faz parte de uma palavra iniciada por outro fragmento. Além disso, cada sentença é automaticamente envolvida por dois *tokens* especiais: [CLS], inserido no início, e [SEP], inserido no final. O *token* [CLS], proveniente da última camada do modelo, é utilizado como representação agregada da sentença em tarefas de classificação. Duas sentenças também podem ser concatenadas em uma única sequência de *tokens*, sendo separadas pelo *token* [SEP].

Figura 4: Tokenização da frase *"Parabéns, meu presidente!"* pelo algoritmo *WordPiece* utilizado no BERT.

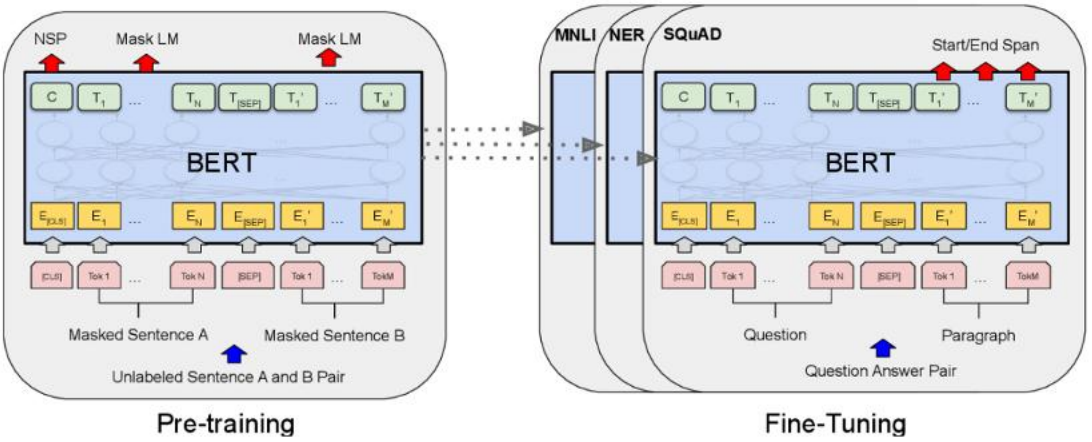


Fonte: Elaborado pelo autor

Devlin et al. (2018) projetaram o modelo BERT para possuir dois estágios: pré treinamento *(pre-training*) e ajuste fino (*fine-tuning*). Durante o pré-treinamento, o modelo é treinado em diferentes tarefas utilizando um corpus genérico na língua inglesa, especificamente a *Wikipedia* (2,5 bilhões de palavras) e o *Toronto Book Corpus* (800 milhões de palavras), com o objetivo de tornar o modelo capaz de compreender componentes linguísticos básicos. Para o ajuste fino, é feita uma modificação somente na última camada do BERT, que é então inicializado com os parâmetros do modelo pré-treinado. Em seguida, todos os parâmetros do modelo são ajustados, podendo ser aplicado para diferentes tarefas, dependendo do interesse de pesquisa. Cada tarefa possui modelos separados, apesar de haver uma diferença mínima nas arquiteturas durante o *fine-tuning* tanto entre si quanto entre elas e a arquitetura pré-treinada. Esta unificação das arquiteturas do BERT em diferentes tarefas torna o modelo bastante flexível. (DEVLIN *et al*., 2019).

A Figura 5 ilustra a arquitetura do BERT e seu treinamento em duas etapas: (i) à esquerda, o pré-treinamento e (ii) à direita, o ajuste fino. No pré-treinamento, o modelo é treinado com uma grande quantidade de dados. Durante o ajuste fino, o BERT é inicializado com os parâmetros do modelo pré-treinado, que são ajustados usando dados rotulados para resolver tarefas específicas, conhecidas como tarefas *downstream*. Essas tarefas correspondem à aplicação prática do modelo em problemas reais de PNL, como classificação de sentenças, análise de sentimentos, resposta a perguntas, reconhecimento de entidades nomeadas e análise de similaridade entre sentenças.

Figura 5: Procedimentos gerais de pré-treinamento e ajuste fino para o BERT.



Fonte: Devlin et al. (2019)

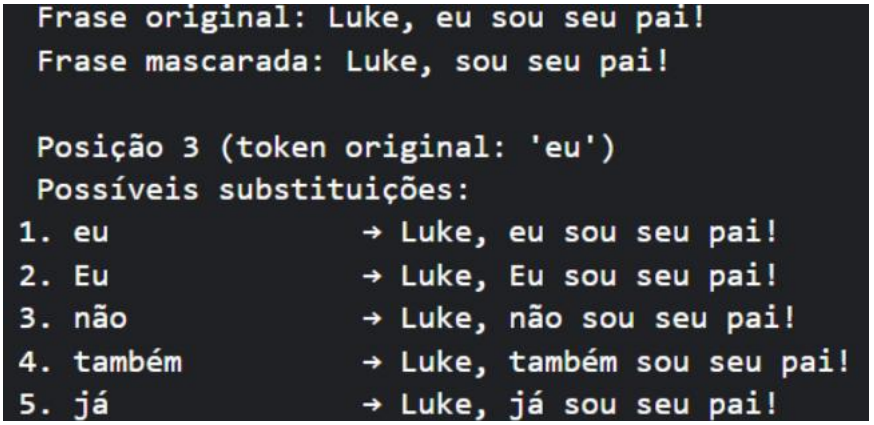
A etapa de pré-treino do BERT envolve duas tarefas de aprendizagem não supervisionadas: predição de palavras ocultas no modelo de linguagem mascarada (do inglês, *Masked Language Model* - MLM) e predição da próxima sentença (do inglês, *Next Sentence Prediction* - NSP).

O algoritmo do MLM consiste em selecionar aleatoriamente 15% dos *tokens* de cada sequência de entrada para serem mascarados por um *token* denominado [MASK]. O objetivo desta tarefa é prever qual é a palavra mascarada, fazendo com que o modelo utilize a informação dos *tokens* ao redor de [MASK] para alcançar tal objetivo. Dessa forma, esse procedimento força o BERT a aprender representações bidirecionais, pois a predição depende simultaneamente do contexto à esquerda e à direita de cada *token* (DEVLIN *et al.*, 2019).

A tarefa de NSP do pré-treino tem como objetivo ensinar o BERT a identificar relações discursivas entre pares de sentenças. Para isso, durante o treinamento, o modelo recebe dois segmentos e deve prever se a segunda sentença está conectada à primeira no documento original. Metade dos pares de entrada (50%) contém sentenças consecutivas reais, enquanto a outra metade contém sentenças escolhidas aleatoriamente, sem relação entre si. Para ensinar o modelo a distinguir entre as duas sentenças no treinamento, a saída do *token* [CLS], que sintetiza o contexto global dos dois segmentos processados, é transformada em um vetor de formato 2×1 usando uma camada de classificação. Em seguida, aplica-se a função *SoftMax*, responsável por converter esses valores em probabilidades normalizadas. Dessa forma, a classe que receber a maior probabilidade é escolhida como predição do modelo (DEVLIN *et al*., 2019).

Na frase "*Luke, eu sou seu pai!*" da figura 5, a palavra "*eu*" foi substituída pelo *token* especial [MASK], e o modelo é treinado para predizer o termo original analisando o contexto bidirecional, formado pelas palavras que aparecem antes e depois do *token* mascarado. A representação contextual gerada pelas camadas de *self-attention* é enviada a uma camada de classificação, que compara o [MASK] com todas as palavras conhecidas pelo modelo. Em seguida, aplica-se a função *SoftMax*, que organiza essas possibilidades de forma a destacar as palavras que mais fazem sentido no contexto. As cinco primeiras predições do modelo de linguagem mascarada foram: (1) *eu*, (2) *Eu*, (3) *não*, (4) *também* e (5) *já*, como mostrado na figura 6. Através desse processo de predição de *tokens* mascarados, o BERT aprende representações semânticas das palavras considerando os contextos em que aparecem.

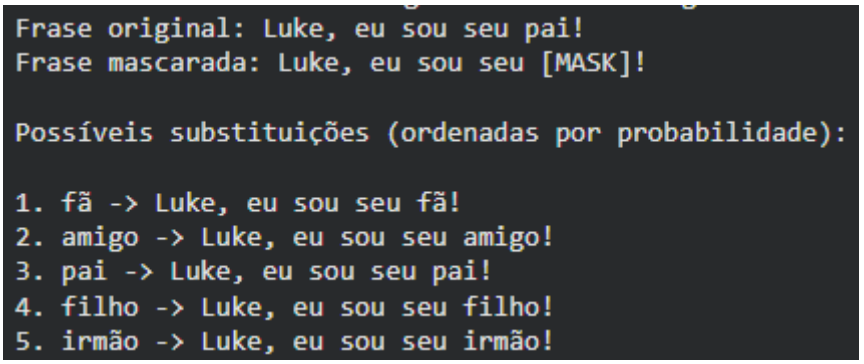
Figura 6: Exemplo de *Masked Language Modeling*



Fonte: Elaborado pelo autor

Ao mascarar a palavra “*pai*” na sentença “*Luke, eu sou seu pai!*”, observa-se que o modelo pode priorizar termos como “*fã*”, conforme ilustrado na Figura 7, em vez do *token* original. Esse comportamento ocorre porque o BERT realiza suas predições com base na probabilidade estatística de ocorrência das palavras no contexto fornecido, aprendida durante o treinamento em grandes bancos de dados. Assim, a expressão “*eu sou seu fã*” pode ser considerada mais frequente ou mais provável do ponto de vista linguístico do que “*eu sou seu pai*”, independentemente do significado específico da frase no universo narrativo em que ela é conhecida. Esse resultado evidencia que o modelo de linguagem mascarada busca maximizar plausibilidade contextual, e não preservar necessariamente o sentido original da sentença, reforçando a importância de mecanismos adicionais de controle semântico quando o mascaramento é utilizado para fins de aumento de dados.

Figura 7: Predição do *token* mascarado “*pai*” com BERTimbau



Fonte: Elaborado pelo autor

Nos textos (1.a) e (1.b), apresentados a seguir, é ilustrada a tarefa de predição da próxima sentença. No exemplo (a), a segunda sentença corresponde à continuação imediata da primeira no texto original, formando um par de sentenças consecutivas conforme aparecem em um mesmo trecho textual. Essa característica define o par como um exemplo positivo para a tarefa de NSP. Já no par (b), embora as sentenças pertençam ao mesmo universo temático e possam ser consideradas semanticamente plausíveis, a segunda sentença não ocorre como continuação direta da primeira no *corpus* de origem. Dessa forma, o par é rotulado como negativo no contexto da tarefa de NSP, uma vez que o objetivo dessa tarefa não é avaliar coerência semântica ou temática, mas identificar relações de continuidade estrutural entre sentenças adjacentes.

Texto 1: Exemplo de NSP

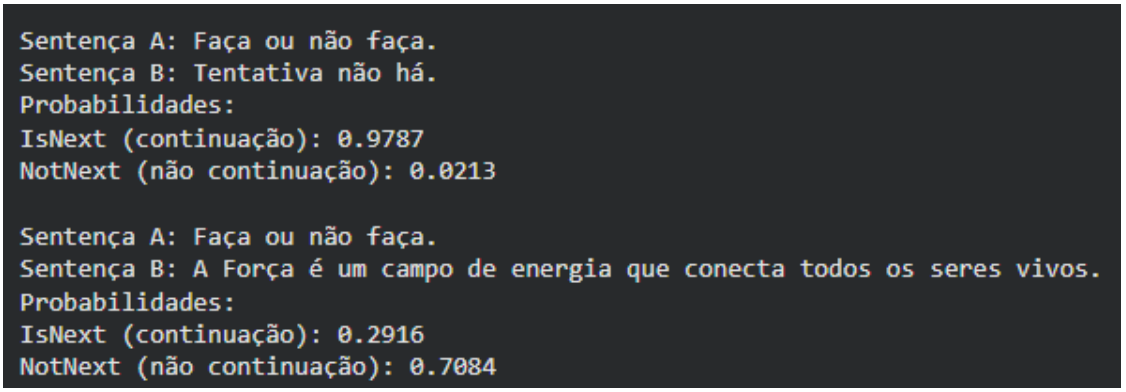
a) *Faça ou não faça.* ***Tentativa não há.***

b) Faça ou não faça. ***A Força é um campo de energia que conecta todos os seres vivos****.*

Fonte: Elaborado pelo autor

Durante o pré-treinamento, o BERT é exposto a exemplos desse tipo para aprender a distinguir quando duas sentenças realmente pertencem à mesma sequência textual. Ao processar cada par, o modelo analisa o conteúdo das duas sentenças em conjunto e avalia se existe ligação semântica ou progressão temática entre elas. O resultado dessa análise é utilizado pela camada de classificação, que, após a aplicação da *SoftMax*, indica a probabilidade de as sentenças serem ou não sequenciais. As probabilidades atribuídas pelo modelo para esses exemplos podem ser observadas na Figura 8. Assim, o treinamento com pares coerentes e incoerentes permite que o BERT internalize padrões de continuidade textual, ajudando-o a identificar relações discursivas em tarefas posteriores de compreensão textual.

Figura 8: Probabilidades atribuídas pelo BERTimbau na tarefa de *Next Sentence Prediction*



Fonte: Elaborado pelo autor

A combinação das duas tarefas durante o pré-treino possibilita que o modelo desenvolva representações profundas de linguagem, considerando tanto o contexto das palavras quanto a relação entre sentenças, tornando-o um codificador independente de tarefa (DEVLIN *et al.*, 2019).

O ajuste fino realiza o que é denominado de aprendizagem por transferência (do inglês*, transfer learning*), onde modelos pré-treinados são ajustados para um domínio e tarefas específicas (RUDER *et al*., 2019). Os dados usados no treinamento e ajuste fino são

mapeados em camadas ocultas dessas redes neurais, que podem ser usadas para diversas finalidades (ROGERS; KOVALEVA; RUMSHISKY, 2020), incluindo geração de *embeddings* contextualizados e classificação de textos.

O BERTimbau é uma adaptação do modelo BERT para o português brasileiro, construída por meio de aprendizado por transferência (SOUZA; LOTUFO; NOGUEIRA, 2020). O modelo foi desenvolvido com o propósito de gerar representações linguísticas profundas e contextualizadas específicas para o português. Nesse processo, um modelo BERT pré-treinado foi posteriormente ajustado utilizando o corpus brWaC (*Brazilian Web as Corpus*) (FILHO *et al.*, 2018), permitindo a aprendizagem de padrões linguísticos e semânticos característicos da língua.

No presente trabalho, o BERTimbau foi utilizado como base para explorar técnicas de mascaramento de palavras como estratégia de enriquecimento dos dados, com o objetivo de melhorar o desempenho na detecção de comentários ofensivos em português.

2.3.1 SBERT

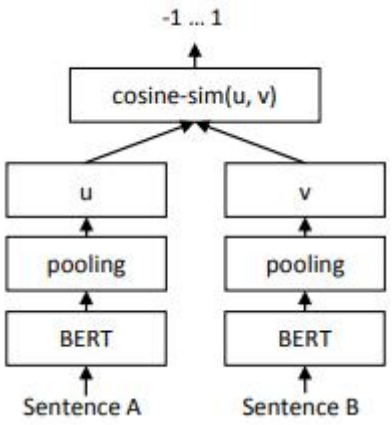
Embora o BERT gere representações vetoriais contextualizadas para cada *token* de uma sentença, o modelo não define, em sua arquitetura original, um mecanismo específico para produzir um único vetor que represente o significado global da sentença de forma independente (DEVLIN et al., 2019). Como consequência, em tarefas de similaridade semântica, a comparação entre sentenças depende do processamento conjunto de pares de textos, o que impede a reutilização de representações vetoriais previamente calculadas e acarreta elevado custo computacional quando se deseja comparar uma sentença com um grande conjunto de outras sentenças. Essa limitação dificulta a aplicação direta do BERT em cenários que exigem múltiplas comparações semânticas. Com o objetivo de contornar esse problema, Reimers e Gurevych (2019) propuseram o Sentence-BERT (SBERT), uma adaptação do BERT voltada à geração de representações vetoriais no nível de sentenças adequadas à comparação semântica eficiente.

No SBERT, cada sentença é processada de forma independente por um modelo BERT utilizando os mesmos parâmetros, conforme ilustrado na Figura 8. Para cada sentença, o modelo produz representações vetoriais para todos os seus *tokens*, que são posteriormente combinadas por meio de uma operação de *pooling*. Essa operação consiste em agregar os vetores correspondentes aos *tokens* em um único vetor de dimensão fixa, que representa o significado global da sentença, sendo o *mean pooling* a estratégia mais comumente empregada, na qual é calculada a média dos vetores dos *tokens* da sentença (REIMERS;

GUREVYCH, 2019). Os vetores resultantes podem então ser comparados diretamente por meio da similaridade do cosseno, permitindo estimar o grau de proximidade semântica entre sentenças de forma eficiente.

Para ilustrar a diferença entre o BERT e o SBERT em tarefas de similaridade semântica, considere uma sentença original S_0 e um conjunto de dez sentenças geradas a partir dela (S_1, S_2, …, S_{10}). Na arquitetura original do BERT, a comparação entre a sentença original e cada uma de suas variações exige que S_0 e S_i sejam processadas conjuntamente pelo modelo a cada nova comparação, não havendo a geração de representações vetoriais independentes que possam ser reutilizadas. Como consequência, o modelo precisa ser executado repetidamente para cada par de sentenças, mesmo quando uma das sentenças é sempre a mesma, o que torna o procedimento computacionalmente custoso em cenários com múltiplas comparações. Em contraste, no Sentence-BERT (SBERT) cada sentença é codificada de forma independente e apenas uma vez, produzindo um vetor que representa seu significado global. Após a geração desses vetores para S_0 e para cada S_i, a comparação entre a sentença original e suas variações é realizada fora do modelo, por meio da similaridade do cosseno, sem a necessidade de novas execuções do modelo para cada par de sentenças. Dessa forma, o SBERT permite uma comparação semântica mais eficiente e escalável, sendo particularmente adequado para estratégias de aumento de dados baseadas em similaridade semântica (REIMERS; GUREVYCH, 2019).

Figura 9: Funcionamento do Sentence-BERT para cálculo de similaridade semântica entre sentenças



Fonte: Reimers e Gurevych (2019)

2.4 AUMENTO DE DADOS

O enriquecimento de dados é uma técnica amplamente utilizada para incrementar o tamanho dos dados de treinamento, reduzindo o sobreajuste e aprimorando a robustez dos modelos de aprendizado de máquina em tarefas com poucos dados.

Em processamento de linguagem natural, diversos métodos baseados em substituição de palavras têm sido explorados para aumento de dados. Por exemplo, Wei e Zou (2019) sugeriram uma técnica chamada *easy data augmentation* (EDA) que utiliza um conjunto de quatro operações simples: substituição aleatória de sinônimos, inserção aleatória de sinônimos, remoção aleatória de palavras e troca aleatória de palavras na frase.

No entanto, esses métodos apresentam dificuldades em preservar corretamente os rótulos de classe. Por exemplo, ao aplicar um processo de aumento de dados que não leva em conta a classe atribuída à frase “um pequeno impacto com um grande filme”, o método pode gerar uma nova versão como “um pequeno filme com um grande impacto”. Embora semelhantes em estrutura, as duas frases transmitem sentimentos opostos: a original expressa uma avaliação negativa, enquanto a modificada tem conotação positiva. Se o novo texto for utilizado no treinamento mantendo o rótulo original (neste caso, negativo), o modelo aprenderá associações incorretas entre palavras e sentimentos, o que tende a comprometer seu desempenho na tarefa de classificação (KUMAR; CHOUDHARY; CHO, 2020).

Considerando a dificuldade de preservar adequadamente os rótulos, a geração de sentenças artificialmente modificadas que podem alterar o sentido original e a baixa capacidade de generalização entre diferentes domínios devido a variações de vocabulário e estilo, o presente trabalho propõe uma estratégia de aumento de dados que utiliza o modelo BERTimbau como base para garantir maior coerência semântica entre as sentenças originais e as versões aumentadas. A proposta incorpora a similaridade de cosseno em um *loop* iterativo ao final de cada *pipeline* de aumento, permitindo selecionar as sentenças mais próximas da original em termos de significado. Desse modo, busca-se preservar a integridade dos rótulos e aprimorar a qualidade dos dados expandidos, promovendo uma geração de exemplos mais consistente e contextualizada. Essa abordagem visa contribuir para o aprimoramento de modelos de detecção de linguagem ofensiva em português.

2.5 SIMILARIDADE DO COSSENO

Foi visto na Seção 2.2.3 que palavras e frases podem ser representadas através de vetores numéricos. Com isso, é possível comparar a relação semântica de duas unidades de texto determinando o cosseno entre os vetores das duas unidades. . Por exemplo, duas frases que usam exatamente os mesmos termos com as mesmas frequências terão um cosseno de 1, enquanto duas frases que não usam termos semanticamente relacionados tenderão a ter cossenos próximos de 0 ou abaixo disso. Em níveis intermediários, frases que contêm termos de significado relacionado, mesmo que nenhum deles seja o mesmo termo ou raiz, terão cossenos mais moderados (FOLTZ; KINTSCH; LANDAUER, 1998).

A Equação 1 apresenta a função de similaridade do cosseno, empregada para calcular o grau de similaridade entre os *embeddings* das sentenças, representados pelos vetores *A* e *B*:

$$similaridade = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \qquad (1)$$

Assim, a similaridade do cosseno é amplamente empregada em tarefas de Processamento de Linguagem Natural para mensurar o grau de proximidade semântica entre sentenças ou palavras representadas por vetores. Esse cálculo constitui uma etapa fundamental para a comparação de *embeddings* em modelos baseados em aprendizado de máquina.

Neste trabalho, utilizamos o Sentence-BERT (SBERT) como ferramenta para gerar *embeddings* de sentenças especificamente otimizados para cálculos de similaridade semântica. Diferentemente do BERT original, que não foi projetado para comparar sentenças de forma eficiente, o SBERT produz vetores densos que capturam o significado global de cada frase, permitindo que a similaridade do cosseno seja aplicada diretamente sobre esses *embeddings* (REIMERS; GUREVYCH, 2019). Assim, é possível identificar automaticamente quais frases geradas pelo processo de *data augmentation* são realmente próximas ao sentido da frase original, filtrando variações incoerentes e mantendo apenas aquelas cuja proximidade semântica se enquadra nas faixas desejadas.

3 TRABALHOS RELACIONADOS

Nos últimos anos, a detecção de discurso ofensivo e linguagem tóxica em português tem se consolidado como tema relevante em PLN, impulsionada pela expansão das redes sociais e pela necessidade de moderação automática de conteúdo. A seguir, destacam-se alguns trabalhos que servem de referência para o presente estudo.

Souza *et al.* (2023) propõem uma *pipeline* para detecção e censura de linguagem ofensiva em textos extensos em português brasileiro, utilizando o modelo BERTimbau (SOUZA; NOGUEIRA; LOTUFO, 2020) como base. O estudo emprega os *corpora* HateBR (VARGAS *et al.*, 2022) e OLID-BR (PEREIRA *et al.*, 2021) para treinar e avaliar classificadores que identificam comentários, postagens e artigos conforme o grau de ofensa, além de localizar termos ofensivos para posterior filtragem. Os autores ressaltam a escassez de recursos para o português e demonstram que modelos baseados em *transformers* têm bom desempenho nessa tarefa.

Assis *et al.* (2024) comparam o desempenho de modelos baseados em *encoders* como BERTimbau (SOUZA; NOGUEIRA; LOTUFO, 2020), AlBERTina (COUTINHO *et al.*, 2023) e BERTweet.BR (RODRIGUES *et al.*, 2021) com grandes modelos de linguagem generativos (LLMs). A pesquisa evidencia que modelos menores e ajustados de forma leve continuam superando LLMs em tarefas de classificação de discurso de ódio em português, principalmente em contextos de poucos recursos.

Outra contribuição importante vem de Neto *et al.* (2023), que apresentaram uma técnica semi-supervisionada baseada em grafos heterogêneos, estruturas que modelam diferentes tipos de nós e relações em uma mesma rede (SUN; HAN, 2012), para anotação automática de linguagem tóxica no contexto do português brasileiro. Nesse contexto, textos, termos e sinais provenientes de anotações automáticas são integrados em um grafo, possibilitando a propagação de rótulos a partir de um conjunto reduzido de dados anotados manualmente. Segundo os autores, essa modelagem contribui para mitigar limitações da anotação exclusivamente manual, como a subjetividade entre os anotadores. O método foi avaliado sobre o *corpus* ToLD-BR (LEITE *et al.,* 2020) e demonstrou alinhamento parcial com os rótulos de referência, conforme evidenciado pelas métricas de concordância apresentadas pelos autores, sugerindo a viabilidade de abordagens semi-supervisionadas para apoiar o processo de rotulagem de linguagem tóxica em português.

Já no campo de enriquecimento de dados, Garg e Ramakrishnan (2020) propõem uma abordagem para a geração de exemplos adversariais em tarefas de classificação textual, baseada no mecanismo de mascaramento do modelo BERT (DEVLIN *et al.*, 2019). A estratégia consiste na substituição ou inserção controlada de *tokens* no texto original, produzindo perturbações semanticamente coerentes, mas capazes de alterar a decisão de modelos classificadores. Segundo os autores, esse tipo de abordagem contribui para o fortalecimento e a robustez dos modelos, sendo conceitualmente alinhada a técnicas de enriquecimento de dados aplicadas ao processamento de linguagem natural.

De modo complementar, Wettig *et al.* (2023) analisam o impacto de diferentes taxas de mascaramento no pré-treinamento de modelos baseados em BERT (DEVLIN *et al.*, 2019) e demonstram que valores mais elevados podem melhorar o desempenho de modelos maiores, preservando até 95% da performance em tarefas de *fine-tuning* mesmo sob mascaramento extremo. Essa investigação fornece uma base empírica importante para compreender o impacto de diferentes taxas de mascaramento e apoia a utilização do *masking* como técnica de ampliação e robustez de dados em tarefas de modelagem de linguagem.

Esses estudos, em conjunto, mostram duas tendências principais que se alinham ao presente trabalho: (i) o fortalecimento de modelos baseados em *transformers* para detecção de linguagem ofensiva em português; (ii) o interesse crescente por técnicas de enriquecimento de dados e para robustez de modelos de PLN. O presente estudo insere-se nesse contexto ao propor especificamente a técnica de mascaramento de palavras no BERT (DEVLIN *et al.*, 2019) como estratégia de aumento de dados para detecção de comentários ofensivos no português do Brasil, utilizando o *corpus* HateBR (VARGAS *et al.*, 2022) e explorando como tal técnica pode ampliar a variabilidade dos dados de treinamento, reduzir sobreajuste e melhorar a generalização dos modelos.

4 METODOLOGIA

4.1 MATERIAIS

Todos os experimentos foram realizados em ambiente de programação hospedado na plataforma *Kaggle*[1], que fornece infraestrutura em nuvem voltada para execução de projetos de aprendizado de máquina e ciência de dados. O ambiente de execução apresentava as seguintes especificações: sistema operacional *Linux* 6.6.56+, versão do *Python* 3.11.13, memória RAM total de 31,35 GB, armazenamento total de 8.062,39 GB e GPU NVIDIA Tesla T4, utilizada para aceleração do treinamento dos modelos baseados em linguagem natural.

Todos os *scripts* foram implementados na linguagem de programação *Python*[2], amplamente utilizada em tarefas de Processamento de Linguagem Natural por sua vasta disponibilidade de bibliotecas e suporte à execução em GPU. Para o desenvolvimento dos experimentos, foram empregadas as bibliotecas *Hugging Face Transformers*[3], responsável pelo uso e *fine-tuning* de modelos pré-treinados como o BERTimbau, e NLTK *(Natural Language Toolkit)*[4]*,* utilizada para o pré-processamento textual e manipulação linguística dos dados.

4.2 CONJUNTO DE DADOS

4.2.1 HateBR

Foram utilizados dois conjuntos de dados neste estudo. O primeiro corresponde ao *corpus* HateBR (VARGAS *et al.*, 2022) e o segundo consiste em um banco de dados independente construído para este trabalho a partir de comentários coletados na plataforma X.

---

[1] GUIMARÃES, Augusto**.** *TCC – Código*. Disponível em: https://www.kaggle.com/code/augustoguimaraes/tcc-c-digo.

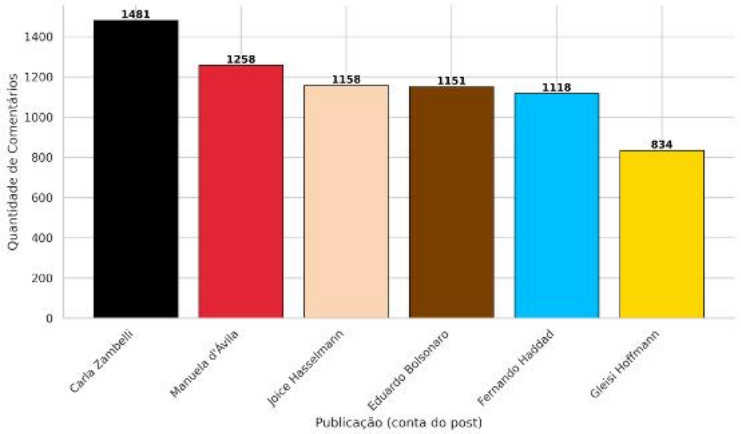
[2] Python Software Foundation. *Python Language Reference*, versão 3.11. Disponível em: https://www.python.org

[3] WOLF, Thomas et al. *Transformers: State-of-the-Art Natural Language Processing*. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2020.

[4] BIRD, Steven; KLEIN, Ewan; LOPER, Edward. *Natural Language Processing with Python*. O'Reilly Media Inc., 2009

O Hate BR contém 7.000 comentários coletados do *Instagram*. Os comentários foram selecionados de seis contas de políticos brasileiros: Carla Zambelli, Manuela d'Ávila**,** Joice Hasselmann**,** Eduardo Bolsonaro**,** Fernando Haddad e Gleisi Hoffmann. Vargas *et al.* (2022) informam que apenas especialistas com alto nível de escolaridade foram selecionados para realizar a anotação dos dados, com o objetivo de minimizar o viés e seus impactos negativos nos resultados. Dessa forma, o processo de anotação não corresponde a uma abordagem baseada exclusivamente em léxico, mas sim a uma avaliação contextual dos comentários. Ao final, 3.500 comentários foram classificados como ofensivos e 3.500 como não ofensivos, resultando em um *corpus* balanceado. A Figura 10 apresenta a proporção dos comentários entre as contas analisadas.

Figura 10: Distribuição do número de comentários coletados por conta pública no corpus HateBR



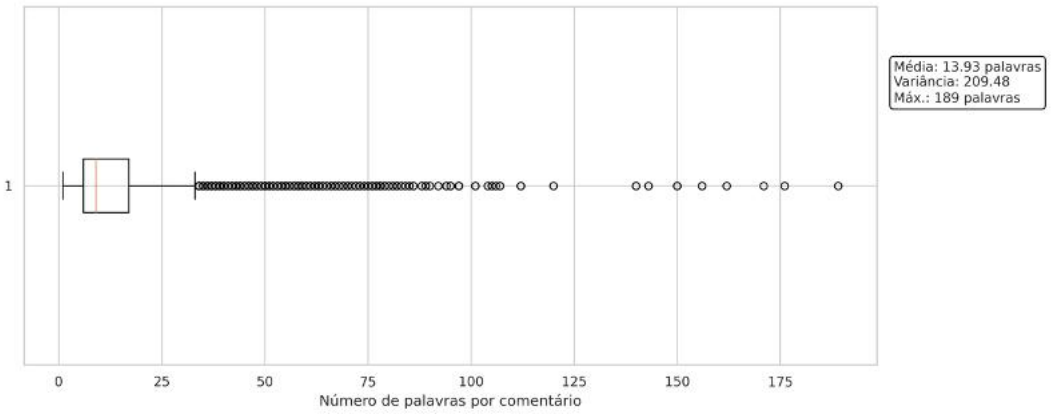
Fonte: Elaborado pelo autor

A Figura 11 apresenta um *boxplot* do comprimento dos comentários do *corpus* HateBR, medido em número de palavras. Observa-se uma distribuição fortemente assimétrica à direita, com concentração da maioria dos comentários em comprimentos curtos e a presença de uma cauda longa composta por comentários significativamente mais extensos. Esse comportamento indica que, embora o tamanho médio dos comentários seja relativamente baixo, há variação considerável no comprimento das postagens, com a ocorrência de *outliers*. Essa distribuição, caracterizada pela predominância de comentários curtos, possui implicações diretas tanto para a classificação automática quanto para a aplicação de técnicas de enriquecimento de dados. Comentários mais longos tendem a ser mais fáceis de classificar, pois apresentam maior quantidade de informação contextual. Em contrapartida, comentários muito curtos são mais sensíveis a transformações automáticas, como mascaramento ou

substituição de *tokens*, o que pode resultar em desvios semânticos relevantes e dificultar a preservação do rótulo original. Como consequência, o enriquecimento de dados aplicado a textos curtos pode introduzir ruído adicional ao conjunto de treinamento, impactando negativamente a qualidade dos exemplos gerados e, potencialmente, o desempenho dos modelos de detecção de conteúdo ofensivo.

No presente trabalho, o enriquecimento não foi aplicado a todas as postagens. Sentenças com menos de três *tokens* foram automaticamente excluídas do processo de geração de dados sintéticos. Esse critério foi adotado para evitar alterações semânticas excessivas em textos muito curtos, nos quais a substituição de um único token poderia comprometer significativamente o sentido original da sentença.

Figura 11: Análise do comprimento dos comentários no corpus HateBR



Fonte: Elaborado pelo autor

Figura 12: *WordCloud* do HateBR

Fonte: Elaborado pelo autor

4.2.2 Conjunto de dados montados a partir do X

Com o objetivo de comparar os resultados obtidos com o *corpus* HateBR e analisar os impactos das técnicas de enriquecimento de dados, construímos um conjunto de dados independente a partir de comentários publicados na plataforma X. A escolha dessa plataforma se justifica por sua ampla utilização para discussões públicas em tempo real, especialmente sobre temas políticos, sociais e culturais, o que a torna uma fonte relevante para a coleta de linguagem opinativa e potencialmente ofensiva. Esse conjunto foi desenvolvido especificamente para fins comparativos, permitindo avaliar a capacidade de generalização dos modelos e verificar se os ganhos observados com aumento de dados se mantêm quando aplicados a outro banco de dados.

A coleta foi realizada por meio de consultas baseadas nos assuntos mais comentados do dia, estratégia adotada em razão das limitações impostas pela API oficial da plataforma X, que restringe o volume de requisições e o acesso a dados históricos em planos gratuitos ou acadêmicos limitados. Os comentários foram coletados no intervalo de 27 de julho de 2025, até 27 de agosto de 2025, abrangendo aproximadamente um mês de coleta. Os termos utilizados como critério de busca encontram-se registrados no próprio banco de dados coletado, juntamente com a data e o horário de cada publicação, conforme exemplificado na visualização parcial apresentada na Figura 13, garantindo rastreabilidade e transparência metodológica.

Figura 13: Visualização parcial do banco de dados do X

	Data	Horário	Texto	Idioma	Ofensivo	Termo de Busca
0	2025-07-30 21:12:12	21:12:12	ALEXANDRE DE MORAES PREPARA DISCURSO PARA SEXT...	pt	0	Alexandre Moraes
1	2025-07-30 21:12:12	21:12:12	A esquerda está usando isso de forma canalha p...	pt	1	Alexandre Moraes
2	2025-07-30 21:12:12	21:12:12	SEM ANISTIA	pt	0	Alexandre Moraes
3	2025-07-30 21:12:12	21:12:12	TRUMP DITADOR	pt	1	Alexandre Moraes
4	2025-07-30 21:12:11	21:12:11	Não, Viviane Barci não foi sancionada diretame...	pt	0	Alexandre Moraes

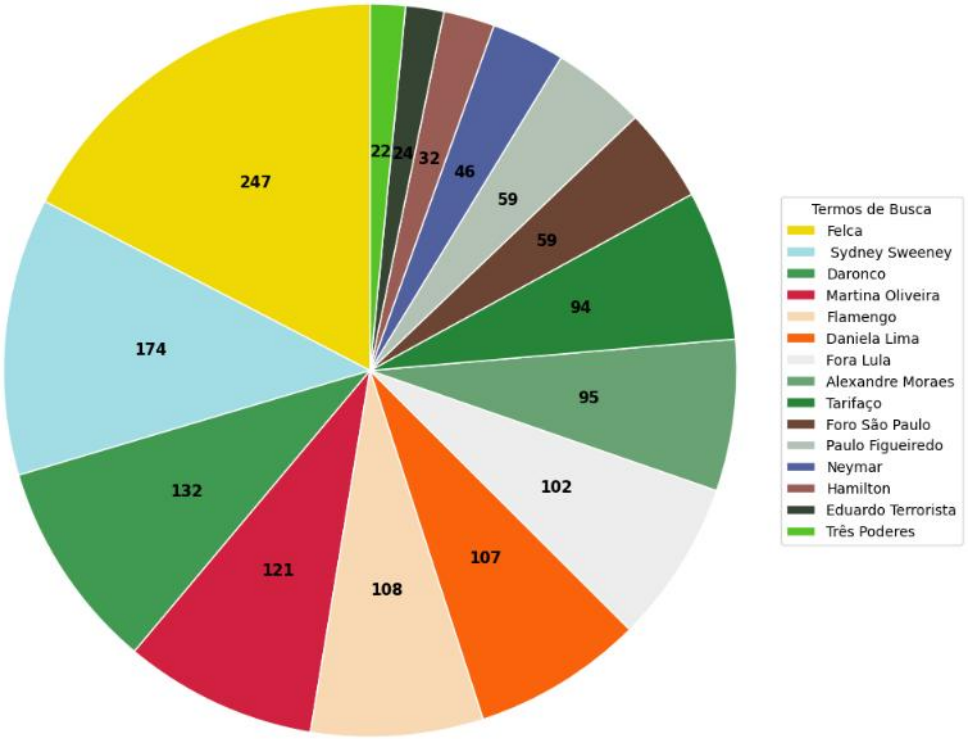
Fonte: Elaborado pelo autor

A seleção de comentários vinculados aos tópicos em alta permitiu obter amostras relevantes, contemporâneas e com elevado engajamento, maximizando a diversidade temática dentro das restrições técnicas existentes, conforme evidenciado nas Figuras 14 e 15.

Figura 14: *WordCloud* do conjunto de comentários do X

Fonte: Elaborado pelo autor

Figura 15: Distribuição do número de comentários por termo de busca no novo conjunto de dados



Fonte: Elaborado pelo autor

Durante a coleta de dados, foi realizada a remoção de URLs, menções a usuários e identificadores de publicação, com o objetivo de preservar apenas o conteúdo textual relevante para a análise.

A anotação seguiu os mesmos critérios conceituais adotados no HateBR, considerando linguagem ofensiva como manifestações que atacam indivíduos ou grupos com base em características identitárias, posicionamentos ou atributos pessoais, sempre levando em conta o contexto da mensagem. Essa padronização metodológica foi essencial para permitir comparação direta entre os resultados obtidos nos diferentes conjuntos. Ressalta-se que o conjunto gerado não foi incorporado ao *corpus* HateBR durante o treinamento principal.

Entretanto, é importante destacar que o processo de anotação envolve, em certa medida, um julgamento subjetivo. A identificação de linguagem ofensiva nem sempre é trivial, especialmente em casos que envolvem ambiguidade semântica, linguagem figurada ou ausência de contexto explícito. Comentários como *"Sabe, fico até sem palavras pra tanta loucura em uma só pessoa!!!!"*, coletados a partir da plataforma X , ilustram casos reais nos quais a expressão pode ser interpretada tanto como uma crítica enfática quanto como um ataque direto à pessoa mencionada. O uso de termos como "*loucura*" pode assumir diferentes sentidos, variando entre uma metáfora comum e uma forma de desqualificação pessoal, o que torna a classificação dependente da interpretação individual. Outro exemplo é o comentário *"Vc fez inglês no Fisk?"*, que, à primeira vista, apresenta estrutura interrogativa neutra, mas pode assumir caráter irônico ou depreciativo dependendo do contexto discursivo. Nesse caso, a ambiguidade está associada à intenção do autor, podendo ser interpretada tanto como uma pergunta genuína quanto como uma forma de desqualificação indireta, o que também pode levar a divergências na anotação.

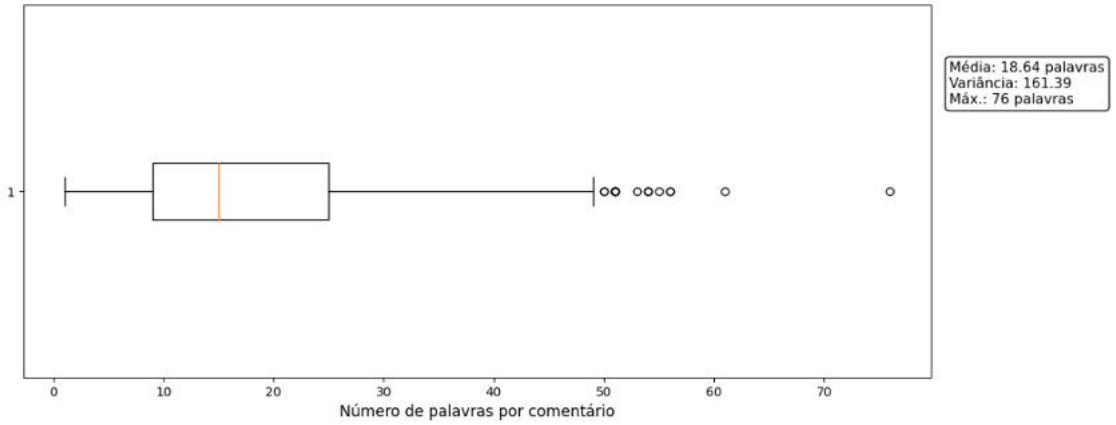
Para mitigar esse problema, foi adotado um critério conservador durante a anotação, no qual comentários considerados ambíguos ou cuja interpretação pudesse variar significativamente foram excluídos do conjunto final. Essa decisão teve como objetivo priorizar a qualidade dos dados em detrimento da quantidade, reduzindo a presença de ruídos no treinamento dos modelos.

A Figura 16 apresenta o *boxplot* do comprimento dos comentários do conjunto construído a partir da plataforma X, também medido em número de palavras. Diferentemente do *corpus* HateBR, que contém 7.000 instâncias, o conjunto do X é composto por 1.422 comentários, o que implica menor volume amostral e, consequentemente, menor probabilidade de ocorrência de valores extremos muito elevados.

Observa-se que a distribuição mantém assimetria à direita, porém menos acentuada em comparação ao HateBR. A média de comprimento é superior (18,64 palavras), indicando que, em termos gerais, os comentários do X tendem a ser ligeiramente mais extensos. No entanto, o valor máximo registrado (76 palavras) é substancialmente inferior ao máximo observado no HateBR (189 palavras), o que evidencia menor dispersão estrutural.

Do ponto de vista metodológico, essa diferença é relevante. O menor volume de dados no conjunto gerado a partir do X pode limitar a diversidade estrutural disponível para treinamento e avaliação, tornando os resultados mais sensíveis a variações pontuais. Por outro lado, a comparação entre um *corpus* amplo e um banco de dados menor e independente permite avaliar se os impactos do enriquecimento de dados se mantêm mesmo em cenários com menor quantidade de exemplos, contribuindo para a análise da robustez e da capacidade de generalização dos modelos.

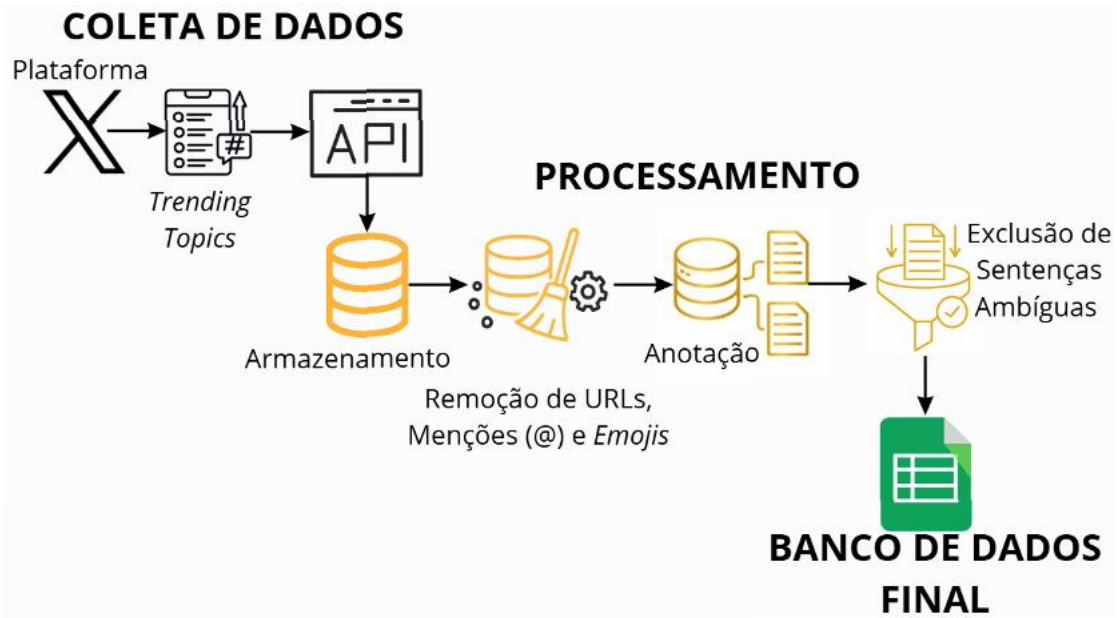
Figura 16: Análise do comprimento dos comentários do X



Fonte: Elaborado pelo autor

O processo completo de coleta, tratamento e preparação dos dados é sintetizado no *pipeline* apresentado na Figura 17:

Figura 17: *Workflow* da coleta e preparação do conjunto de dados da plataforma X



Fonte: Elaborado pelo autor

4.3 Limpeza lexical

4.3.1 HateBR

Antes da etapa de enriquecimento de dados e treinamento dos modelos de classificação, foi realizado um processo de limpeza léxica com o objetivo de remover ruídos textuais e padronizar os dados do *corpus* HateBR. Como os comentários desse conjunto de dados foram coletados do *Instagram*, muitos deles continham símbolos, menções, URLs e *emojis* que poderiam interferir na etapa de vetorização e prejudicar o aprendizado do modelo.

Diferentemente de abordagens tradicionais que convertem todo o texto para letras minúsculas, optou-se por não aplicar essa transformação neste trabalho, uma vez que o modelo utilizado foi o BERTimbau-base-cased. Esse modelo diferencia letras maiúsculas de minúsculas no processo de tokenização, e a perda dessa distinção poderia comprometer parte das informações linguísticas presentes nos comentários, além de prejudicar seu desempenho.

A limpeza léxica adotada baseou-se em um conjunto de expressões regulares aplicadas ao texto original, conforme apresentado na Tabela 2.

Tabela 2: Expressões regulares utilizadas no pré-processamento dos dados textuais.

REGEX	Substitui...	Por...
@\w+	Menções do tipo @usuario	*vazio*
[^\w\s,.!?áéíóúãõâêôçÁÉÍÓÚÃÕÂÊÔÇ]	Emojis, símbolos e caracteres não textuais	*vazio*
http\S+ | www\S+	URLs em geral	*vazio*
\s+	Sequências de espaços duplicados	*espaço simples*

Fonte: Elaborado pelo autor

Inicialmente, eliminou-se qualquer menção a usuários, representadas por nomes precedidos de "@", a fim de evitar que referências pessoais influenciem a distribuição dos dados ou introduzam ruídos artificiais. URLs também foram removidas por não contribuírem semanticamente para a análise textual. Além disso, símbolos não textuais, *emojis* e caracteres especiais foram filtrados por meio de uma expressão regular que preserva apenas letras, números, pontuação básica e caracteres acentuados válidos. Essa etapa reduz interferências visuais e garante maior consistência ao processo de tokenização subsequente. Após a remoção desses elementos, o texto foi normalizado seguindo o padrão Unicode NFKC, garantindo que caracteres equivalentes possuíssem uma única forma representacional, algo essencial para a estabilidade do modelo e para evitar duplicação de *tokens* semanticamente idênticos. Por fim, removem-se espaços duplicados originados pelas substituições, garantindo que o texto final permanecesse coerente e contínuo.

4.3.2 Conjunto de dados montados a partir do X

Diferentemente do *corpus* HateBR, no conjunto do X a remoção de elementos não textuais ocorreu previamente ao armazenamento das instâncias no banco de dados. Durante a coleta, foram eliminadas automaticamente URLs, menções a usuários e identificadores técnicos de publicação, preservando-se exclusivamente o conteúdo textual da mensagem. Essa decisão foi adotada com o objetivo de armazenar no banco apenas o texto semanticamente relevante para a tarefa de classificação, reduzindo a necessidade de múltiplas etapas posteriores de limpeza.

Após a coleta, aplicou-se o mesmo procedimento de normalização Unicode (NFKC) utilizado também no HateBR, garantindo padronização representacional entre os dois

conjuntos. Não foi realizada conversão para letras minúsculas, mantendo-se a coerência com a escolha do modelo BERTimbau-base-cased, que distingue maiúsculas e minúsculas durante o processo de tokenização.

Cabe destacar que, como o conjunto do X já foi armazenado sem URLs e menções, não foi necessário aplicar novamente as expressões regulares listadas na Tabela 2 de forma integral. Ainda assim, procedeu-se à verificação de possíveis resíduos textuais e à normalização de espaços, assegurando consistência estrutural entre os dois bancos de dados.

4.4 Estratégias de Enriquecimento de Dados

Após a etapa de limpeza léxica, o *corpus* foi dividido em 80% para treinamento e 20% para teste, assegurando que os dados destinados à avaliação permanecessem totalmente intactos. Somente após essa separação foram aplicadas as técnicas de enriquecimento de dados, de modo que apenas o conjunto de treinamento foi ampliado artificialmente. Essa abordagem garante que o modelo seja avaliado exclusivamente em exemplos reais e não modificados, preservando a independência e a confiabilidade do conjunto de testes.

Adicionalmente, para fins comparativos, realizou-se um segundo experimento com a divisão invertida, utilizando 20% dos dados para treinamento e 80% para teste. Essa configuração permitiu analisar o impacto do enriquecimento de dados em um cenário de menor disponibilidade de exemplos rotulados para aprendizado, possibilitando avaliar de forma mais rigorosa a contribuição das técnicas de aumento de dados no desempenho do modelo.

Os intervalos de similaridade semântica utilizados neste trabalho foram definidos de forma empírica, a partir de experimentações preliminares realizadas durante o desenvolvimento da pesquisa. A escolha das faixas considerou o equilíbrio entre preservação semântica das sentenças originais e diversidade linguística das frases geradas, buscando evitar tanto substituições excessivamente distantes quanto variações praticamente idênticas às do texto original.

4.4.1 Primeira abordagem

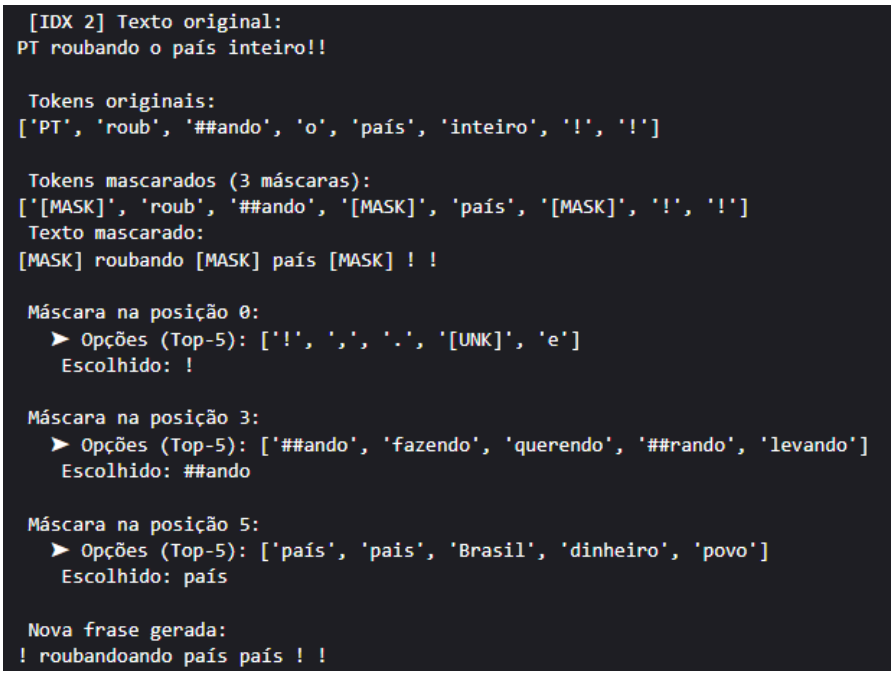
O primeiro método adotado foi baseado no próprio modelo BERTimbau Base Cased, utilizando seu mecanismo nativo de MLM. Nesse método, aproximadamente 15% dos *tokens*

de cada frase são mascarados de forma aleatória, e o modelo é então utilizado para prever possíveis substituições para cada posição mascarada. Para cada máscara, o BERT retorna uma distribuição de probabilidades sobre o vocabulário, e o *token* selecionado é aquele com maior probabilidade de ocorrência, segundo o modelo. Assim, cada frase aumentada é gerada substituindo-se os *tokens* mascarados pelos candidatos mais prováveis estimados pelo BERT.

Apesar de teoricamente promissora, a técnica apresentou limitações importantes quando aplicada ao domínio específico dos comentários do *Instagram*. Em diversos casos, o modelo gerou substituições pouco realistas ou linguisticamente inconsistentes, o que comprometeu a utilidade das frases aumentadas. Uma das principais causas desses problemas está relacionada ao processo de tokenização do BERT, que utiliza a segmentação em *subtokens* pelo algoritmo *WordPiece,* conforme descrito na Seção 1.3. Quando um *token* mascarado ou substituído é composto por múltiplos *subtokens*, o modelo nem sempre consegue reconstruir corretamente a unidade lexical completa, resultando em combinações truncadas, repetições indevidas ou recomposições morfológicas sem sentido.

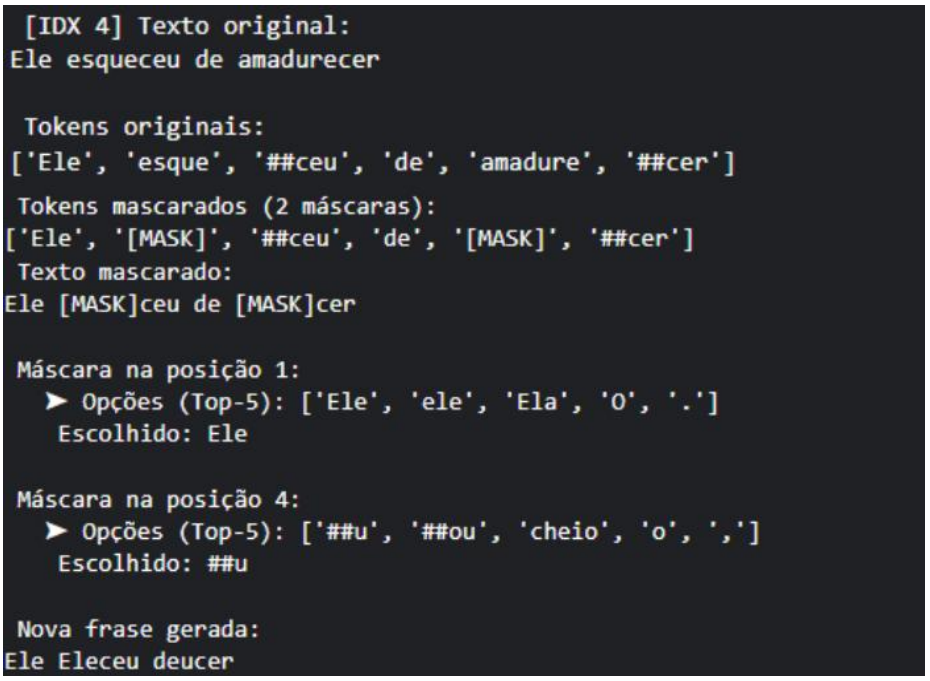
Observou-se, por exemplo, a criação de sequências morfologicamente incorretas e a inserção de *tokens* inválidos como [UNK], utilizado pelo BERT para representar palavras ou sequências que não conseguem ser corretamente mapeadas para o vocabulário do modelo, geralmente em decorrência de falhas no processo de recomposição de subtokens. Em um dos casos, a frase "*PT roubando o país inteiro!!*" foi transformada em *"! roubandoando país país !!"*, evidenciando a duplicação indevida do subtoken "##ando" e a recomposição semântica incoerente das partes da palavra. De modo análogo, a sentença *"Ele esqueceu de amadurecer"* foi convertida em *"Ele Eleceu deucer"*, demonstrando falhas tanto na etapa de substituição quanto na reconstrução dos *subtokens*, o que resultou na criação de formas inexistentes e na descaracterização completa da estrutura sintática original. Ademais, observaram-se duplicações inesperadas, substituições aleatórias ou destituídas de relação semântica com o texto de origem, bem como sentenças cuja coerência linguística foi significativamente comprometida, inviabilizando a utilização dessa abordagem como técnica de enriquecimento de dados. Esses problemas podem ser observados nos resultados apresentados na Figura 18 e na Figura 19:

Figura 18: Frase gerada pelo MLM: ‘! roubandoando país país !!’



Fonte: Elaborado pelo autor

Figura 19: Frase gerada pelo MLM: ‘Ele Eleceu deucer’



Fonte: Elaborado pelo autor

Esses resultados indicam que, embora o BERTimbau seja altamente eficaz em tarefas de linguagem natural, o uso direto do MLM como técnica de enriquecimento de dados não produziu frases adequadas nem semanticamente consistentes para o contexto da classificação de linguagem ofensiva. Por esse motivo, essa abordagem foi considerada apenas exploratória e acabou sendo descartada, abrindo espaço para a adoção de métodos mais controlados e adequados ao corpus, apresentada nas subseções seguintes.

4.4.2 Segunda abordagem

Considerando as limitações observadas na abordagem anterior, adotou-se uma segunda estratégia de aumento de dados voltada exclusivamente para palavras inteiras**,** evitando a fragmentação em *subtokens* característica do algoritmo *WordPiece*. Essa modificação teve como objetivo impedir que substituições incorretas gerassem formas truncadas, repetições indevidas ou *tokens* inexistentes. Na prática, antes de selecionar um termo para mascaramento, verifica-se se o *token* não inicia com ## e se não pertence a uma sequência fragmentada pelo tokenizador. Por exemplo, na frase *"Ele esqueceu de amadurecer"*, o tokenizador fragmenta a palavra "*amadurecer*" em subtokens como "ama", "##du" e "##recer". Nessa situação, a palavra "amadurecer" não é considerada elegível para o processo de mascaramento. Essa verificação garante que apenas unidades lexicais inteiras sejam manipuladas.

Além disso, para impedir que o modelo substituísse palavras de baixa relevância semântica, como artigos, preposições e conjunções, aplicou-se filtragem baseada em *stopwords*. Para isso, utilizou-se a biblioteca NLTK, empregada para o pré-processamento textual em tarefas de Processamento de Linguagem Natural. O NLTK fornece uma lista padronizada de *stopwords* para o português, permitindo descartar automaticamente termos funcionais que não contribuem para a geração de variações úteis durante o enriquecimento de dados.

Após a tokenização do texto, o sistema constrói uma lista de candidatos à substituição. Essa lista inclui todos os *tokens* que são palavras completas, compostos exclusivamente por caracteres alfabéticos e que não pertencem à lista de *stopwords* fornecida pelo NLTK. Cada *token* dessa lista é então avaliado individualmente com uma probabilidade fixa de ser mascarado, o que possibilita a criação de múltiplas variações a partir de um mesmo texto de forma parcialmente estocástica, porém controlada. A etapa seguinte consiste na geração das

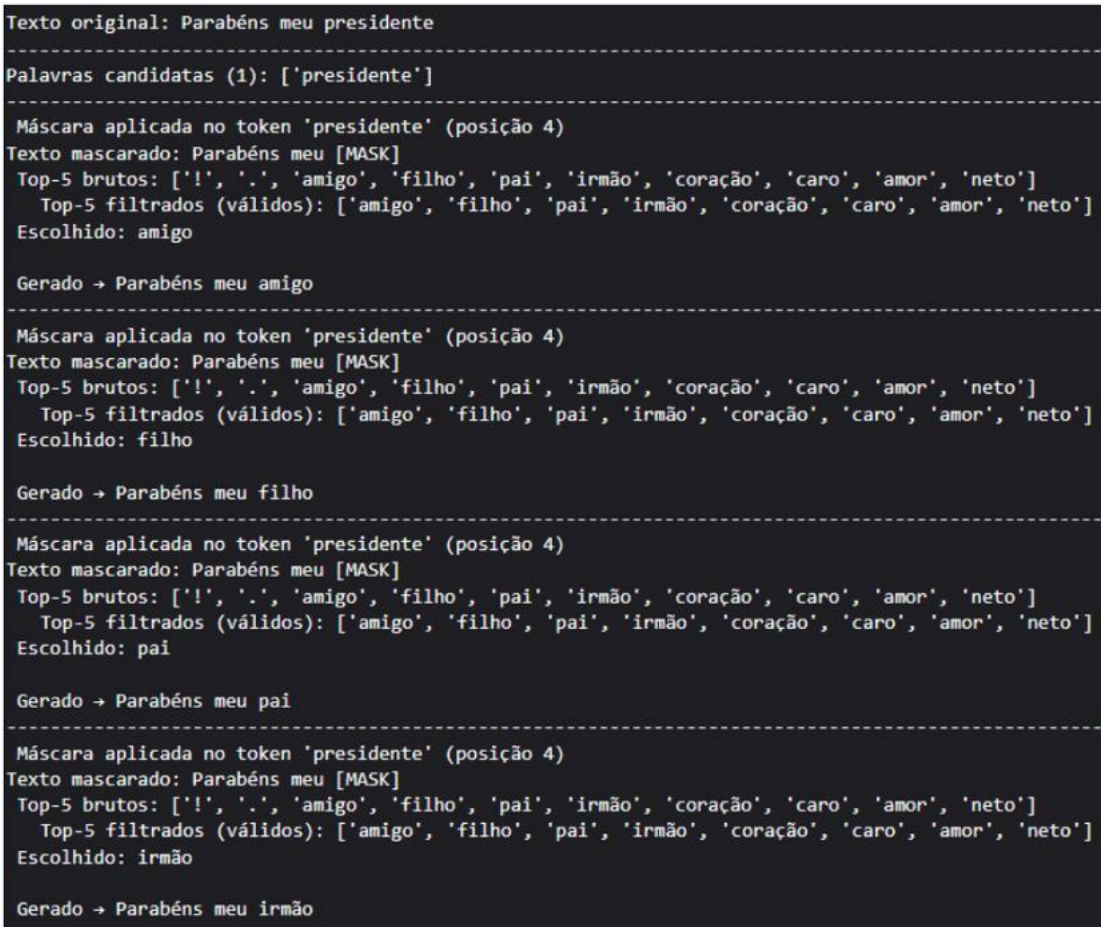
substituições, em que o modelo prevê um conjunto de possíveis palavras para cada posição mascarada. No entanto, para evitar o problema de repetição excessiva das mesmas substituições, empregou-se um mecanismo de rodízio na escolha das palavras finais. Assim, quando o modelo retorna os k melhores candidatos, esses termos passam por uma nova filtragem para remover *subtokens* e *stopwords*, e a seleção ocorre ciclicamente: na primeira variação utiliza-se o primeiro candidato, na segunda o próximo, e assim sucessivamente, retornando ao início da lista quando necessário.

A Figura 20 ilustra esse processo. Nela, observa-se que, para a frase original *"Parabéns meu presidente"*, apenas a palavra "presidente" foi identificada como candidata válida à substituição. Após aplicar a máscara, o modelo gera uma lista de possíveis substituições, como *"amigo"*, *"filho"*, *"pai"* e *"irmão"*. Graças ao mecanismo de rodízio, cada uma dessas opções é utilizada em variações sucessivas, resultando em sentenças distintas como *"Parabéns meu amigo"*, *"Parabéns meu filho"*, *"Parabéns meu pai"* e *"Parabéns meu irmão"*.

Outro aspecto relevante dessa abordagem diz respeito à quantidade de variações sintetizadas por frase. Em sua configuração final, o sistema tenta gerar até 10 frases novas para cada texto original, com um limite máximo de 20 tentativas por instância. Contudo, nem todas são necessariamente aceitas: frases duplicadas, incoerentes ou distantes semanticamente da original são descartadas com base na similaridade do cosseno (Seção 1.6) entre os *embeddings* das sentenças, de modo que o número final de amostras aumentadas varia conforme as características lexicais de cada texto. Esse controle impede ciclos infinitos quando nenhuma substituição válida pode ser obtida, garantindo que o processo permaneça eficiente e determinístico.

Esse procedimento garante maior diversidade lexical entre as frases geradas, distribui de forma equilibrada as diferentes possibilidades previstas pelo modelo e mantém um controle determinístico sobre o processo de substituição. Caso nenhum *token* válido seja encontrado após a filtragem, o sistema adota um mecanismo de contingência, selecionando o primeiro *token* alfabético disponível ou, se não houver alternativas adequadas, preservando a palavra original.

Figura 20: Variações produzidas pelo modelo a partir da substituição controlada de palavras



Fonte: Elaborado pelo autor

Essa mudança metodológica produziu impactos significativos na qualidade das frases geradas. As substituições passaram a manter coerência sintática e gramatical, preservando a estrutura das sentenças originais. Além disso, ao restringir a substituição às palavras com maior carga semântica, as frases aumentadas tornaram-se mais expressivas e úteis para o treinamento do classificador.

Embora mais estável que a primeira abordagem, essa técnica ainda poderia gerar substituições que, apesar de linguisticamente válidas, se afastassem do significado original. Para mitigar esse efeito, foi incorporada uma etapa adicional de validação baseada em similaridade do cosseno, garantindo que apenas as versões semanticamente próximas fossem mantidas no conjunto de treinamento.

Após gerar cada frase nova, o sistema calcula essa similaridade entre o *embedding* da frase original e o *embedding* da frase mascarada. Para ajustar o equilíbrio entre diversidade e fidelidade semântica, cinco faixas de similaridade foram testadas: 0.70–0.85, 0.70–0.90,

0.70–1.00, 0.65–0.90 e 0.65–1.00. Intervalos muito altos, como aqueles que atingem o limite superior de 1.00, tendem a gerar frases quase idênticas às originais, reduzindo o ganho de diversidade; já limites inferiores próximos de 0.65 aumentam o risco de produzir frases distorcidas ou semanticamente incoerentes. Após o filtro baseado nessas faixas, as frases aprovadas foram concatenadas à base de dados original, formando múltiplas versões expandidas do conjunto de treinamento.

Os resultados do processo de geração de frases para as diferentes faixas de similaridade na abordagem com mascaramento múltiplo são apresentados na Tabela 3, incluindo o número de frases originais, as variações geradas e o tamanho final das bases utilizadas nos experimentos.

Tabela 3*:* Quantidade de frases geradas pela abordagem *multi mask* no HateBR (80% treino / 20% teste)

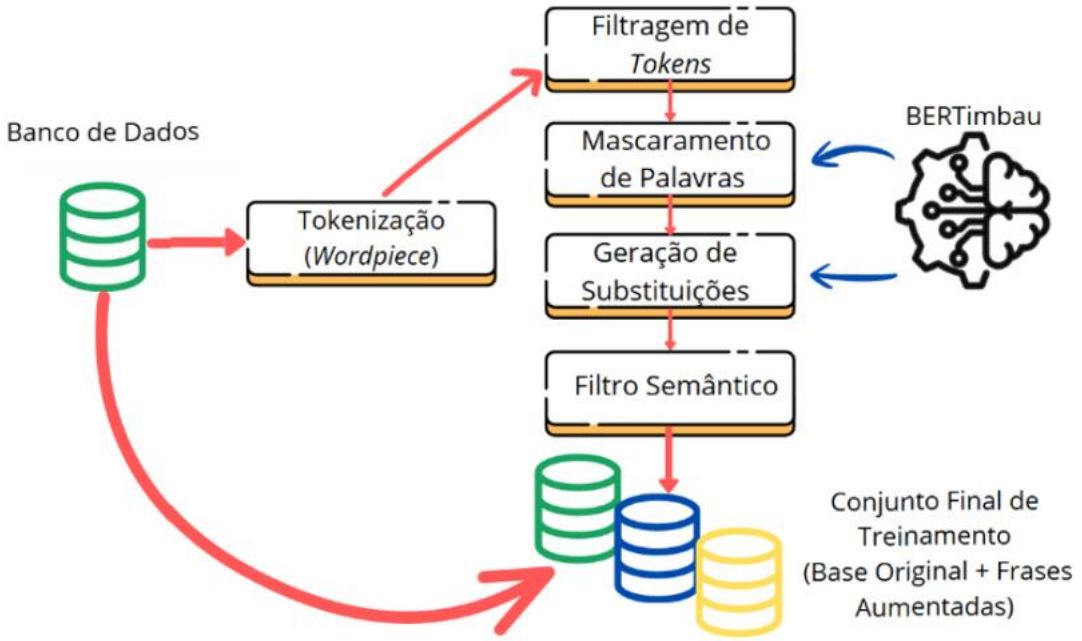
FAIXA DE SIMILARIDADE	FRASES ORIGINAIS	FRASES NOVAS	TAMANHO FINAL DA BASE
0.65 - 0.90	5322	10296	15618
0.65 - 1.00	5322	27328	32650
0.70 - 0.85	5322	5712	11034
0.70- 0.90	5322	9377	14699
0.70 - 1.00	5322	26280	31602

Fonte: Elaborado pelo autor

Em cada experimento, o modelo foi treinado utilizando a totalidade dessa base concatenada, permitindo avaliar empiricamente qual faixa de similaridade gerou o melhor equilíbrio entre diversidade e coerência, resultando no conjunto mais eficaz para o desempenho final do classificador. Importante destacar que todo o processo de treinamento do BERTimbau seguiu exatamente a mesma estrutura experimental e os mesmos hiperparâmetros adotados no estudo-base de Souza *et al.* (2023). Essa decisão buscou isolar completamente o impacto das técnicas de aumento de dados, garantindo que quaisquer diferenças observadas no desempenho fossem atribuídas exclusivamente ao enriquecimento de dados, e não a ajustes de arquitetura ou configuração de treinamento.

A Figura 21 apresenta uma visão consolidada do processo de enriquecimento de dados adotado a partir desta etapa do trabalho. O fluxo ilustrado evidencia como o processo de geração e validação das sentenças aumentadas foi estruturado de modo a preservar a coerência semântica e o rótulo original dos dados.

Figura 21: *Pipeline* do processo de enriquecimento de dados baseado em mascaramento controlado de palavras e filtragem semântica adotado neste trabalho.



Fonte: Elaborado pelo autor

4.4.3 Terceira abordagem

Como extensão das estratégias anteriores, explorou-se também uma terceira abordagem de aumento de dados baseada em mascaramento condicional, diferenciando-se principalmente pelo fato de apenas um único *token* ser mascarado por vez. Essa modificação foi introduzida como alternativa a abordagem apresentada na seção 4.4.2, cujo mascaramento simultâneo de múltiplas posições poderia gerar alterações excessivas, distorcendo o significado original das sentenças ou produzindo variações difíceis de controlar. Ao restringir o processo a apenas uma palavra por iteração, tornou-se possível exercer um controle mais refinado sobre o impacto semântico de cada substituição, reduzindo riscos de incoerência.

Assim como na abordagem anterior, o método inicia identificando candidatos válidos para substituição. Apenas *tokens* que correspondem a palavras completas e que não pertencem ao conjunto de *stopwords* do NLTK são considerados. Em cada tentativa, uma única palavra dessa lista é selecionada aleatoriamente para ser mascarada, garantindo que cada amostra gerada seja resultado de uma modificação pontual e isolada.

Os critérios de similaridade e filtragem semântica permanecem alinhados aos utilizados na segunda abordagem, garantindo consistência metodológica e permitindo

comparações diretas entre as estratégias. Adicionalmente, manteve-se o mesmo número de tentativas por sentença, o mesmo volume de frases sintéticas geradas e exatamente os mesmos hiperparâmetros utilizados no treinamento do modelo na segunda abordagem**.** Essa padronização operacional assegura que eventuais diferenças de desempenho observadas nas métricas possam ser atribuídas exclusivamente ao tipo de mascaramento empregado, sem interferências de variações no processo de geração ou no treinamento.

Os resultados obtidos com a abordagem baseada na substituição de um único *token* no *corpus* HateBR, considerando a divisão de 80% para treinamento e 20% para teste, são apresentados na Tabela 4.

Tabela 4*:* Quantidade de frases geradas pela abordagem *one token* no HateBR (80% treino / 20% teste)

FAIXA DE SIMILARIDADE	FRASES ORIGINAIS	FRASES NOVAS	TAMANHO FINAL DA BASE
0.65 - 0.90	5322	16871	22193
0.65 - 1.00	5322	18766	24088
0.70 - 0.85	5322	7193	12515
0.70- 0.90	5322	15229	20551
0.70 - 1.00	5322	16280	21602

Fonte: Elaborado pelo autor

Ainda que mais conservadora, essa técnica mostrou-se útil para produzir variações leves e semanticamente compatíveis, funcionando como alternativa às estratégias anteriores.

A abordagem *one token* foi adotada nos demais experimentos do trabalho, tanto no corpus HateBR quanto no conjunto alternativo construído a partir da plataforma X, em razão do seu desempenho superior observado no cenário do HateBR com divisão de 80% para treinamento e 20% para teste.

As Tabelas 5, 6 e 7 apresentam a quantidade de frases geradas pela abordagem baseada na substituição de um único *token* nos diferentes cenários experimentais, incluindo o banco de dados HateBR e o conjunto de dados alternativo. Em cada caso, são detalhados o número de frases originais, as variações geradas e o tamanho final das bases utilizadas nos experimentos.

Tabela 5*:* Quantidade de frases geradas pela abordagem *one token* no HateBR (20% treino /80% teste)

FAIXA DE SIMILARIDADE	FRASES ORIGINAIS	FRASES NOVAS	TAMANHO FINAL DA BASE
0.65 - 0.90	1678	2576	4254
0.65 - 1.00	1678	4217	5895
0.70 - 0.85	1678	1817	3495
0.70- 0.90	1678	2425	4103
0.70 - 1.00	1678	4107	5785

Fonte: Elaborado pelo autor

Tabela 6*:* Quantidade de frases geradas pela abordagem *one token* no conjunto de dados alternativo (80% treino / 20% teste)

FAIXA DE SIMILARIDADE	FRASES ORIGINAIS	FRASES NOVAS	TAMANHO FINAL DA BASE
0.65 - 0.90	1137	4600	5737
0.65 - 1.00	1137	4850	5987
0.70 - 0.85	1137	2699	3836
0.70- 0.90	1137	3484	4621
0.70 - 1.00	1137	4648	5785

Fonte: Elaborado pelo autor

Tabela 7*:* Quantidade de frases geradas pela abordagem *one token* no conjunto de dados alternativo (20% treino / 80% teste)

FAIXA DE SIMILARIDADE	FRASES ORIGINAIS	FRASES NOVAS	TAMANHO FINAL DA BASE
0.65 - 0.90	284	837	1121
0.65 - 1.00	284	1519	1803
0.70 - 0.85	284	627	911
0.70- 0.90	284	845	1129
0.70 - 1.00	284	1499	1783

Fonte: Elaborado pelo autor

4.5 SÍNTESE DOS CENÁRIOS EXPERIMENTAIS

Esta seção consolida a organização dos cenários experimentais adotados neste trabalho, estabelecendo uma nomenclatura padronizada para identificação das diferentes configurações de divisão e enriquecimento dos conjuntos de dados. Essa padronização tem

como objetivo facilitar a referência e a análise comparativa dos resultados apresentados no capítulo 5 e discutidos no capítulo 6.

Foram consideradas duas bases distintas: o conjunto HateBR e a base alternativa (X). Para cada uma delas, avaliaram-se divisões nos formatos 80%/20% e 20%/80%, tanto na versão original quanto na versão com enriquecimento de dados aplicado exclusivamente ao conjunto de treinamento.

Nos cenários identificados pelo símbolo "+", o conjunto de treinamento foi ampliado com instâncias sintéticas geradas pelas estratégias de aumento de dados descritas anteriormente, mantendo-se o conjunto de testes integralmente original.

A tabela 8 apresenta a convenção de nomenclatura adotada:

Tabela 8: Nomenclatura dos Cenários Experimentais.

CONJUNTO	IDENTIFICADOR	DESCRIÇÃO
HateBR	HateBR_80/20	*HateBR com divisão original de 80% para treinamento e 20% para teste.*
HateBR	HateBR_20/80	*HateBR com divisão original de 20% para treinamento e 80% para teste.*
HateBR	HateBR+_80/20	*HateBR com 80% do conjunto original acrescido de instâncias sintéticas no treinamento e 20% original no teste.*
HateBR	HateBR+_20/80	*HateBR com 20% do conjunto original acrescido de instâncias sintéticas no treinamento e 80% original no teste.*
Base Alternativa (X)	X_80/20	*Base alternativa com divisão original de 80% para treinamento e 20% para teste.*
Base Alternativa (X)	X_20/80	*Base alternativa com divisão original de 20% para treinamento e 80% para teste.*
Base Alternativa (X)	X+_80/20	*Base alternativa com 80% do conjunto original acrescido de instâncias sintéticas no treinamento e 20% original no teste.*
Base Alternativa (X)	X+_20/80	*Base alternativa com 20% do conjunto original acrescido de instâncias sintéticas no treinamento e 80% original no teste.*

Fonte: Elaborado pelo autor

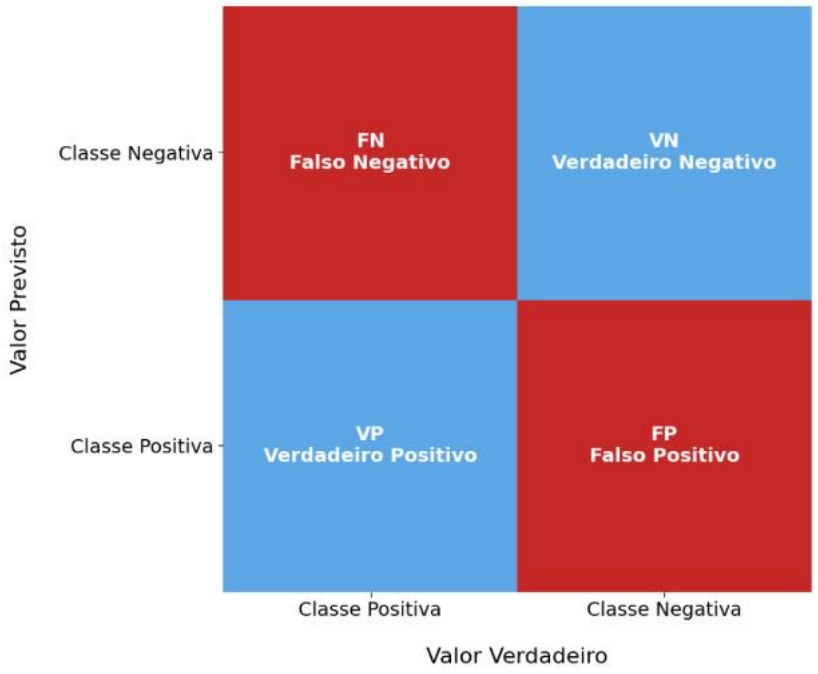
A adoção dessa convenção permite referenciar os experimentos de maneira objetiva e padronizada ao longo do texto, reduzindo ambiguidades e facilitando a comparação entre os diferentes cenários avaliados.

5 RESULTADOS E DISCUSSÕES

5.1 Medidas para Avaliação de Desempenho

Como critérios de avaliação de desempenho, foram utilizados quatro métodos, oriundos da matriz de confusão. Esta, por sua vez, trata-se de uma ferramenta visual que detalha o número de previsões corretas e incorretas, permitindo verificar o desempenho do método de aprendizagem como visto na Figura 22 (STEHMAN, 1997).

Figura 22: Exemplo de uma matriz de confusão



Fonte: Elaborado pelo autor

A partir da matriz de confusão, extrai-se então cada um dos métodos utilizados. São eles: Acurácia, Precisão, Sensibilidade ou *Recall* e F1-score. A acurácia pode ser definida como a proximidade dos resultados da predição do valor real, e é dada, matematicamente, por (TAYLOR, 1997):

$$acurácia = \frac{VP + VN}{VP + VN + FP + FN} \quad (2)$$

A Precisão mede o grau de confiança que um modelo pode depositar na previsão de que um exemplo pertence a uma classe específica, calculando o número de exemplos

corretamente classificados como pertencentes a essa classe dividido pelo número total de exemplos classificados como pertencentes a essa classe. Matematicamente, tem-se:

$$precisão = \frac{VP}{VP + FP} \quad (3)$$

O *Recall* é o número de exemplos corretamente identificados como pertencentes a uma classe dividida pelo número total de exemplos pertencentes a essa classe nos dados:

$$recall = \frac{VP}{VP + FN} \quad (4)$$

O F1-Score é a média harmônica entre a precisão e o *recall*, ou seja, é a média dos valores de precisão e *recall*, dando maior importância a valores baixos, visto que um valor de precisão ou *recall* muito baixo indica que o modelo não está equilibrando bem essas duas métricas, quando o ideal seria atribuir igual importância a ambas. O F1-score é dado pela seguinte equação:

$$F1 = 2 * \frac{precisao * recall}{precisao + recal} \quad (5)$$

5.2 CENÁRIO HATEBR_80/20

Neste cenário, avalia-se o desempenho dos modelos utilizando a divisão original do conjunto HateBR, com 80% das instâncias destinadas ao treinamento e 20% ao teste. São comparados o cenário *baseline* (HateBR_80/20) e suas variações com enriquecimento de dados (HateBR+_80/20).

5.2.1 BERTimbau

Após o término do pré-processamento e da preparação dos conjuntos de dados aumentados, iniciaram-se os experimentos de classificação. Esses focaram em analisar o impacto das estratégias de enriquecimento de dados baseadas em similaridade semântica via SBERT, empregando tanto substituições de um único *token* quanto variações *multi-mask*, aplicadas a diferentes faixas de similaridade. A intenção foi verificar até que ponto a

ampliação sintética do conjunto de dados poderia melhorar o desempenho do classificador sem introduzir ruído semântico prejudicial.

Utilizando o BERTimbau, na abordagem na qual os *tokens* tinham 15% de chances de serem mascarados, o intervalo de similaridade 0.65–0.90 apresentou o melhor desempenho entre os conjuntos enriquecidos, conforme evidenciado na Tabela 9. Nessa faixa, observou-se o equilíbrio mais adequado entre preservação semântica e diversidade lexical, evitando tanto substituições excessivamente próximas quanto alterações que distorcem o significado. Já no método que realiza apenas uma substituição por sentença, o melhor desempenho ocorreu no intervalo 0.65–1.00, indicando que faixas mais amplas beneficiam esse tipo de estratégia, pois ampliam a diversidade sem comprometer o sentido da frase, como pode ser verificado na tabela 10.

Tabela 9*:* Avaliação do BERTimbau na abordagem *multi-mask* sob múltiplos intervalos de similaridade

CONJUNTO DE DADOS	ACURÁCIA	PRECISÃO	*RECALL*	F1	INSTÂNCIAS DE TREINO	INSTÂNCIAS DE TESTE
HateBR_80/20	0.912546	0.912954	0.912454	0.912763	5322	1678
HateBR+_80/20 (0. 70-0.85)	0.908340	0.906667	0.912072	0.909361	11034	1678
HateBR+_80/20 (0. 70-0.90)	0.907588	0.895954	0.923994	0.909758	14699	1678
HateBR+_80/20 (0.65-0.90)	0.910594	0.903509	0.921013	0.912177	15618	1678
HateBR+_80/20 (0.65-1.00)	0.908340	0.903084	0.916542	0.909763	32650	1678
HateBR+_80/20 (0.70-1.00)	0.906086	0.892241	0.925484	0.908559	31602	1678

Fonte: Elaborado pelo autor

Tabela 10: Avaliação do BERTimbau na abordagem de substituição única para diferentes faixas

CONJUNTO DE DADOS	ACURÁCIA	PRECISÃO	*RECALL*	F1	INSTÂNCIAS DE TREINO	INSTÂNCIAS DE TESTE
HateBR_80/20	0.912546	0.912954	0.912454	0.912763	5322	1678
HateBR+_80/20 (0. 70-0.85)	0.907588	0.905325	0.912072	0.909369	12515	1678
HateBR+_80/20 (0. 70-0.90)	0.907588	0.904130	0.913562	0.908821	20551	1678
HateBR+_80/20 (0.65-0.90)	0.913599	0.916168	0.912072	0.914115	22193	1678
HateBR+_80/20 (0.65-1.00)	0.918107	0.915680	0.922504	0.919079	24088	1678
HateBR+_80/20 (0.70-1.00)	0.909842	0.902190	0.921013	0.911504	21602	1678

Fonte: Elaborado pelo autor

No BERTimbau, os resultados apresentaram vantagens marginais em relação aos trabalhos relacionados, principalmente porque o modelo já era altamente generalizado antes

da aplicação das técnicas de enriquecimento de dados. O BERTimbau, por natureza, possui forte capacidade de representação contextual e elevada robustez mesmo com quantidades moderadas de dados, o que reduz o espaço para ganhos substanciais decorrentes do aumento artificial do corpus. Por esse motivo, embora o aumento de dados tenha gerado melhorias reais, essas melhorias se manifestaram de forma limitada, refletidas em pequenas variações de F1 e *recall*, sem alterações drásticas no desempenho geral.

É importante destacar que apenas a abordagem de substituição por um único *token* (similaridade 0.65 - 1.00) apresentou métricas superiores às do trabalho relacionado. Em termos de acurácia, observou-se um aumento de 0,006, passando de 0,912 no *baseline* original para 0,918, o que representa uma melhoria relativa de aproximadamente 0,66% em relação ao modelo sem enriquecimento de dados. A estratégia *multi-mask* (0.65 -- 0.90), por outro lado, apresentou desempenho ligeiramente inferior ao *baseline* original, conforme mostrado na tabela 11.

Tabela 11: Desempenho do BERTimbau com diferentes estratégias de substituição para faixas de similaridade

CONJUNTO DE DADOS	TAMANHO DO TREINO	ACURÁCIA	PRECISÃO	*RECALL*	F1
HateBR_80/20	5.322	0.912546	0.912954	0.912454	0.912763
único *token*: HateBR+_80/20 (0.65-1.00)	24.088	0.918107	0.915680	0.922504	0.919079
multi-mask: HateBR+_80/20 (0.65-0.90)	15.619	0.910594	0.903509	0.921013	0.912177

Fonte: Elaborado pelo autor

5.2.2 Modelos Tradicionais

Entretanto, quando essas mesmas bases aumentadas foram aplicadas aos modelos clássicos, as melhorias se tornaram um pouco mais visíveis. Modelos como SVM, Logistic Regression e MLP não possuem mecanismos de contextualização profunda, dependendo fortemente da variação lexical presente nas representações TF-IDF ou *embeddings* simples. Assim, o aumento da diversidade do *corpus* impactou diretamente sua capacidade de separação das classes. Os ganhos foram especialmente nítidos nos aumentos de *recall* e F1, mostrando que nesses modelos o *augmentation* foi mais determinante que no BERTimbau.

Tabela 12: Impacto da estratégia *multi-mask* no desempenho dos classificadores tradicionais

CONJUNTO DE DADOS	**ACURÁCIA**	**PRECISÃO**	***RECALL***	**F1**	**MODELO**
HateBR_80/20	0.828700	0.843411	0.810730	0.826748	SVM
multi-mask: HateBR+_80/20 (0.65-0.90)	0.837716	0.823613	0.862891	0.842795	SVM
HateBR_80/20	0.811420	0.822086	0.798808	0.810280	LogReg
multi-mask: HateBR+_80/20 (0.65-0.90)	0.848234	0.830748	0.877794	0.853623	LogReg
HateBR_80/20	0.811420	0.820122	0.801788	0.810852	MLP
multi-mask: HateBR+_80/20 (0.65-0.90)	0.827949	0.810393	0.859911	0.834418	MLP

Fonte: Elaborado pelo autor

Tabela 13: Melhoria dos modelos clássicos com aumento de dados via único *token*

CONJUNTO DE DADOS	**ACURÁCIA**	**PRECISÃO**	***RECALL***	**F1**	**MODELO**
HateBR_80/20	0.828700	0.843411	0.810730	0.826748	SVM
one token: HateBR+_80/20 (0.65-1.00)	0.833959	0.815126	0.867362	0.840433	SVM
HateBR_80/20	0.811420	0.822086	0.798808	0.810280	LogReg
one token: HateBR+_80/20 (0.65-1.00)	0.839970	0.818942	0.876304	0.846652	LogReg
HateBR_80/20	0.811420	0.820122	0.801788	0.810852	MLP
one token: HateBR+_80/20 (0.65-1.00)	0.831705	0.816124	0.859911	0.837446	MLP

Fonte: Elaborado pelo autor

Esse comportamento reforça a interpretação de que o enriquecimento de dados por similaridade semântica funciona melhor como uma ferramenta de ampliação da variabilidade linguística em modelos que não possuem generalização contextual embutida. No caso do BERTimbau, ele opera mais como refinamento; nos modelos clássicos, atua como mecanismo estruturante, fornecendo padrões sintáticos e lexicais adicionais que esses classificadores não conseguem inferir por conta própria.

5.3 CENÁRIO HATEBR_20/80

Neste cenário, avalia-se o desempenho do modelo utilizando a divisão invertida do conjunto HateBR, com 20% das instâncias destinadas ao treinamento e 80% ao teste. São comparados o cenário *baseline* (HateBR_20/80) e suas variações com enriquecimento de dados (HateBR+_20/80).

Conforme observado na seção anterior, a estratégia de enriquecimento baseada na substituição de um único token por sentença foi a que apresentou os resultados mais consistentes no cenário HateBR_80/20. Em razão desse desempenho superior em relação às demais variações enriquecidas, optou-se por empregar exclusivamente essa abordagem no cenário HateBR_20/80, a fim de avaliar seu comportamento em uma condição de menor volume de dados para treinamento.

Cabe destacar que, neste cenário, os experimentos foram conduzidos exclusivamente com o BERTimbau. Não foram realizados testes com modelos tradicionais, uma vez que o objetivo principal desta etapa foi analisar o comportamento do modelo contextual profundo sob condição de restrição de dados, isolando o impacto do enriquecimento no modelo de maior capacidade representacional.

5.3.1 BERTimbau

Conforme apresentado na tabela 14, no cenário com 20% de dados para treino e 80% para teste, a estratégia de enriquecimento baseada na substituição de um único *token*, utilizando intervalo de similaridade entre 0.65 e 1.00, apresentou o melhor desempenho entre todas as configurações avaliadas. Essa abordagem alcançou acurácia de 0.893629, superando o modelo base (0.882896) em aproximadamente 1,07 pontos percentuais.

Tabela 14: Desempenho do BERTimbau no HateBR com Divisão 20%/80%

CONJUNTO DE DADOS	ACURÁCIA	PRECISÃO	*RECALL*	F1	INSTÂNCIAS DE TREINO	INSTÂNCIAS DE TESTE
HateBR_20/80	0.882896	0.888687	0.873764	0.881163	1678	5322
HateBR+_20/80 (0.70-0.90)	0.887404	0.876934	0.899262	0.887958	4103	5322
HateBR+_20/80 (0.65-0.90)	0.890699	0.886289	0.894465	0.890358	4254	5322

HateBR+_20/80 (0.65-1.00)	0.893629	0.890356	0.895941	0.893140	5895	5322
HateBR+_20/80 (0.70-0.85)	0.886672	0.882267	0.890406	0.886318	3495	5322
HateBR+_20/80 (0.70-1.00)	0.891432	0.895701	0.884133	0.889879	5785	5322

Fonte: Elaborado pelo autor

O impacto positivo tende a ser mais perceptível nesse cenário, possivelmente em função da menor quantidade de dados disponíveis para treinamento. Com apenas 20% das instâncias utilizadas para aprendizado, o modelo passa a ter maior dependência da diversidade e representatividade dos exemplos fornecidos. Nesse contexto, o enriquecimento de dados por meio da substituição controlada de um único *token* amplia a variabilidade linguística do conjunto de treino de forma moderada, preservando a estrutura semântica original das sentenças. Isso contribui para reduzir o risco de sobreajuste e favorecer a capacidade de generalização do modelo.

5.4 CENÁRIO X_80/20

5.4.1 BERTimbau

Conforme apresentado na tabela 15, os experimentos realizados no banco de dados alternativo com divisão de 80% para treinamento e 20% para teste evidenciam impacto positivo do enriquecimento de dados no desempenho do modelo.

Tabela 15 *:* Desempenho do Modelo no Banco de Dados Alternativo (80% Treino / 20% Teste)

CONJUNTO DE DADOS	ACURÁCIA	PRECISÃO	*RECALL*	F1	INSTÂNCIAS DE TREINO	INSTÂNCIAS DE TESTE
X_80/20	0.781404	0.845000	0.698082	0.764361	1137	284
X+_80/20 (0. 70-0.85)	0.757895	0.818182	0.678082	0.741573	3836	284
X+_80/20 (0. 70-0.90	0.750877	0.790698	0.698630	0.741818	4621	284
X+_80/20 (0.65-0.90)	0.846491	0.888462	0.796575	0.839855	5737	284

X+_80/20 (0.65-1.00)	0.777404	0.835000	0.702782	0.762381	5987	284
X+_80/20 (0.70-1.00)	0.767895	0.818000	0.701781	0.755387	5785	284

Fonte: Elaborado pelo autor

A melhor configuração foi obtida com o intervalo de similaridade entre 0.65 e 0.90, que alcançou acurácia de 0.846491. Em comparação com o modelo base, cuja acurácia foi de 0.781404, observa-se um ganho absoluto de aproximadamente 6,51 pontos percentuais. Esse resultado evidencia que o enriquecimento de dados contribuiu para a melhoria da capacidade de generalização do modelo nesse conjunto.

5.5 CENÁRIO X_20/80

5.5.1 BERTimbau

Conforme apresentado na Tabela 16, no cenário com divisão invertida (20% para treinamento e 80% para teste), observa-se que o enriquecimento de dados manteve impacto positivo no desempenho do modelo.

Tabela 16 *:* Desempenho do Modelo no Banco de Dados Alternativo (20% Treino / 80% Teste)

CONJUNTO DE DADOS	ACURÁCIA	PRECISÃO	*RECALL*	F1	INSTÂNCIAS DE TREINO	INSTÂNCIAS DE TESTE
X_20/80	0.701775	0.714395	0.683656	0.703436	284	1137
X+_20/80 (0.65-0.90)	0.734622	0.796009	0.630931	0.703922	1121	1137
X+_20/80 (0.65-1.00)	0.745167	0.781818	0.680141	0.727444	1803	1137
X+_20/80 (0.70-0.85)	0.745167	0.779559	0.683656	0.728464	911	1137
X+_20/80 (0.70-0.90)	0.739895	0.781443	0.666081	0.719165	1129	1137
X+_20/80 (0.70-1.00)	0.759561	0.766318	0.746380	0.746215	1783	1137

Fonte: Elaborado pelo autor

A melhor configuração foi obtida com o intervalo de similaridade entre 0.70 e 1.00, que alcançou acurácia de 0.759561. Em comparação com o modelo base (0.701775), verifica-se um ganho absoluto de aproximadamente 5,78 pontos percentuais. Esse resultado indica que, mesmo em um cenário mais restritivo em termos de dados de treinamento, a estratégia de enriquecimento contribuiu para ampliar a capacidade de generalização do classificador.

De forma comparativa, a acurácia apresentou ganhos em ambos os cenários com o uso do enriquecimento de dados, sendo mais elevada na divisão com maior quantidade de amostras para treinamento, o que indica melhor aproveitamento das instâncias adicionais durante o aprendizado. No cenário com menos dados de treino, embora o aumento tenha sido um pouco menor, ainda houve melhoria em relação ao modelo base, sugerindo que os exemplos gerados ajudaram a reduzir a limitação causada pelo baixo volume de dados.

6 Conclusão

Este trabalho investigou o uso de técnicas de aumento de dados baseadas em mascaramento de palavras no modelo BERT como estratégia para aprimorar a detecção automática de comentários ofensivos em português do Brasil. Os resultados experimentais demonstraram que o modelo BERTimbau apresenta desempenho robusto mesmo sem a aplicação de técnicas de aumento de dados, confirmando sua forte capacidade de representação contextual. Ainda assim, observou-se que a estratégia de substituição de um único *token*, aliada a uma faixa ampla de similaridade semântica, produziu melhorias consistentes nas métricas de acurácia, precisão, *recall* e F1-score. O mascaramento de múltiplos *tokens* apresenta como principal vantagem a capacidade de gerar variações mais diversificadas das sentenças originais, promovendo maior riqueza lexical e estrutural no conjunto de treinamento. Quando combinado com faixas de similaridade semântica intermediárias, esse método contribui para expor o modelo a padrões linguísticos mais variados, o que pode favorecer a generalização e reduzir a dependência de termos específicos.

Além disso, verificou-se que os ganhos proporcionados pelo enriquecimento de dados foram significativamente mais expressivos em modelos clássicos de aprendizado de máquina, como Regressão Logística, SVM e MLP. Esse resultado indica que o aumento de dados semânticos exerce um papel fundamental para classificadores que não dispõem de mecanismos avançados de contextualização, contribuindo para melhorar sua capacidade de generalização e reduzir o impacto da escassez de dados rotulados.

No experimento com inversão da proporção no HateBR (20% treino / 80% teste), observou-se que o impacto do enriquecimento de dados tornou-se relativamente mais evidente. Com a redução significativa das instâncias disponíveis para treinamento, o modelo passou a depender mais fortemente da diversidade e da representatividade do conjunto ampliado. Nesse cenário, a estratégia de substituição de um único token na faixa de similaridade 0.65–1.00 apresentou desempenho superior ao baseline, indicando que o aumento controlado da variabilidade lexical pode mitigar parcialmente os efeitos da limitação de dados. Esse resultado reforça que o enriquecimento tende a ser mais relevante quando o volume de exemplos rotulados é reduzido.

No banco de dados alternativo, os efeitos do aumento de dados mostraram-se ainda mais expressivos, especialmente na divisão 80% treino / 20% teste, em que os ganhos de acurácia ultrapassaram seis pontos percentuais em relação ao modelo base. Mesmo na divisão

invertida (20% treino / 80% teste), o enriquecimento manteve ganhos consistentes, evidenciando sua contribuição para a capacidade de generalização do classificador. Esses resultados indicam que, em conjuntos menores ou menos consolidados que o HateBR, a ampliação sintética controlada exerce papel mais determinante no desempenho final do modelo.

De modo geral, conclui-se que o mascaramento controlado de palavras, quando aliado a uma etapa de filtragem semântica eficiente, constitui uma estratégia viável e eficaz para ampliar a variabilidade dos dados de treinamento sem comprometer a coerência linguística. O estudo reforça a importância de abordagens cuidadosamente controladas de aumento de dados em tarefas sensíveis ao significado, como a detecção de linguagem ofensiva, especialmente no contexto do português brasileiro.

Apesar dos resultados obtidos, este trabalho apresenta algumas limitações importantes. Embora a filtragem baseada em similaridade semântica tenha contribuído para reduzir a geração de exemplos incoerentes, não há garantia absoluta de que todas as sentenças produzidas pelas estratégias de aumento de dados preservem integralmente o significado original e, consequentemente, o rótulo associado à frase. Dessa forma, investigações futuras podem incluir análises qualitativas e inspeções manuais das sentenças geradas, com o objetivo de avaliar mais rigorosamente o impacto das transformações semânticas na manutenção das classificações originais.

Além disso, a construção e anotação do conjunto de dados alternativo estão sujeitas à subjetividade inerente à identificação de linguagem ofensiva, especialmente em casos envolvendo ironia, ambiguidade contextual e linguagem implícita. Embora tenham sido adotados critérios conservadores durante o processo de anotação, trabalhos posteriores podem incorporar múltiplos anotadores no processo de classificação dos comentários, permitindo avaliar a concordância entre diferentes julgamentos e reduzir possíveis vieses individuais na atribuição dos rótulos.

Como trabalhos futuros, sugere-se também a investigação de técnicas híbridas que combinam mascaramento com outras formas de enriquecimento semântico, como a retrotradução (*back-translation*), que consiste em traduzir uma sentença para outro idioma e posteriormente traduzi-la novamente para o idioma original, gerando variações sintáticas mantendo o significado semântico. Ademais, estudos complementares podem investigar o desempenho da abordagem em diferentes conjuntos de dados e contextos de aplicação, visando avaliar sua capacidade de generalização e sua robustez em cenários variados.

REFERÊNCIAS

ASSIS, Lucas; FERREIRA, João; SILVA, Mariana; PEREIRA, André; SOUZA, Fábio. **Exploring Portuguese Hate Speech Detection in Low Resource Settings: Lightly Tuning Encoder Models or In-Context Learning of Large Models?** In: *Proceedings of the 15th International Conference on Computational Processing of Portuguese* (PROPOR), 2024. Disponível em: https://aclanthology.org/2024.propor-1.?. Acesso em: 8 nov. 2025.

BEDDIAR, Djamila Romaissa; JAHAN, Md Saroar; OUSSALAH, Mourad. **Data Expansion using Back Translation and Paraphrasing for Hate Speech Detection. Online Social Networks and Media**, v. 24, p. 100153, 2021. Disponível em: https://doi.org/10.1016/j.osnem.2021.100153. Acesso em: 7 nov. 2025.

BENGIO, Yoshua; DUCHARME, Réjean; VINCENT, Pascal; JAUVIN, Christian. **A neural probabilistic language model.** *Journal of Machine Learning Research*, v. 3, p. 1137–1155, 2003. Disponível em: http://www.jmlr.org/papers/volume3/bengio03a/bengio03a.pdf. Acesso em: 6 nov. 2025.

BRASIL. Ministério dos Direitos Humanos. **Guia de enfrentamento ao discurso de ódio e à violência nas redes sociais**. Brasília: MDH, 2018. Disponível em: https://repositorio.ufsc.br/handle/123456789/187510. Acesso em: 5 maio 2026.

BISHOP, C. M. Pattern Recognition and Machine Learning. New York: Springer, 2006. Disponível em: https://link.springer.com/book/10.1007/978-0-387-45528-0. Acesso em: 8 jan. 2026.

CORTES, C.; VAPNIK, V. **Support-vector networks**. Machine Learning, v. 20, n. 3, p. 273–297, 1995. Disponível em: https://link.springer.com/article/10.1007/BF00994018. Acesso em: 8 jan. 2026.

DENG, Li; YU, Dong. **Deep learning: Methods and applications.** *Foundations and Trends in Signal Processing*, v. 7, n. 3–4, p. 197–387, 2014. Disponível em: https://doi.org/10.1561/2000000039. Acesso em: 6 nov. 2025.

DEVLIN, J.; CHANG, M.-W.; LEE, K.; TOUTANOVA, K. **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 (NAACL-HLT)*, 2019, Minneapolis. p. 4171–4186.

FOLTZ, P. W.; KINTSCH, W.; LANDAUER, T. K**. The measurement of textual coherence with latent semantic analysis. Discourse Processes**, Routledge, v. 25, n. 2–3, p. 285–307, 1998.Disponível em: https://doi.org/10.1080/01638539809545029. Acesso em: 11 nov. 2025.

GARG, Siddhant; RAMAKRISHNAN, Goutham. **BAE: BERT-based Adversarial Examples for Text Classification.** In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2020. p. 6174–6181. Disponível em: https://aclanthology.org/2020.emnlp-main.498/. Acesso em: 8 nov. 2025.

GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. **Deep Learning**. Cambridge, MA: MIT Press, 2016. Disponível em: https://aikosh.indiaai.gov.in/static/Deep+Learning+Ian+Goodfellow.pdf. Acesso em: 10 dez. 2025.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. **The Elements of Statistical Learning: Data Mining, Inference, and Prediction.** 2. ed. New York: Springer, 2009.
Disponível em: https://hastie.su.domains/ElemStatLearn/.
Acesso em: 8 jan. 2026.

JAMES, G.; WITTEN, D.; HASTIE, T.; TIBSHIRANI, R. **An Introduction to Statistical Learning: with Applications in R**. New York: Springer, 2013.
Disponível em: https://www.statlearning.com/.
Acesso em: 8 jan. 2026.

JURAFSKY, D.; MARTIN, J. H. **Speech & Language Processing: An Introduction to Natural Language Processing**. [S.l.]: Pearson Education India, 2000. Disponível em: https://pages.ucsd.edu/~bakovic/compphon/Jurafsky,%20Martin.-Speech%20and%20Language%20Processing_%20An%20Introduction%20to%20Natural%20Language%20Processing%20(2007).pdf. Acesso em: 8 dez. 2025.

KHURANA, Diksha; KOLI, Aditya; KHATTER, Kiran; SINGH, Sukhdev. **Natural Language Processing: State of The Art, Current Trends and Challenges.** *International Journal of Advance Research in Computer Science*, v. 8, n. 1, p. 1–10, 2017. Disponível em: https://ijarcs.info/index.php/Ijarcs/article/view/4545. Acesso em: 5 nov. 2025.

KUMAR, Varun; CHOUDHARY, Ashutosh; CHO, Eunah. **Data Augmentation Using Pre-trained Transformer Models.** *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI (EMNLP 2020)*. Association for Computational Linguistics, 2020. Disponível em: https://aclanthology.org/2020.nlp4convai-1.5/. Acesso em: 7 nov. 2025.

LECUN, Yann; BENGIO, Yoshua; HINTON, Geoffrey. **Deep learning.** *Nature*, v. 521, p. 436–444, 2015. Disponível em: https://doi.org/10.1038/nature14539. Acesso em: 6 nov. 2025.

LORENA, A. C.; CARVALHO, A. C. P. L. F. **Uma introdução às máquinas de vetores de suporte. Revista de Informática Teórica e Aplicada**, v. 14, n. 2, p. 43–67, 2007.
Disponível em: https://seer.ufrgs.br/rita/article/view/rita_v14_n2_p43.
Acesso em: 8 jan. 2026.

NETO, Pedro H.; SANTOS, Lívia; CUNHA, Rafael; MENDES, Victor. **Abordagem Semi-Supervisionada para Anotação de Linguagem Tóxica.** In: *Proceedings of the 12th Brazilian Conference on Intelligent Systems (BRACIS)*. Porto Alegre: SBC, 2024. Disponível em: https://sol.sbc.org.br/index.php/bracis/article/view/30156. Acesso em: 8 nov. 2025.

REIMERS, N.; GUREVYCH, I. **Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks.** In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, Hong Kong. p. 3982–3992.

ROGERS, Anna; KOVALEVA, Olga; RUMSHISKY, Anna.**A Primer in BERTology: What We Know About How BERT Works.***Computational Linguistics*, v. 46, n. 4, p. 842–866,

2020. Disponível em: https://direct.mit.edu/coli/article/46/4/842/97382. Acesso em: 7 fev. 2026.

RUDER, S.; PETERS, M. E.; SANH, V.; WOLF, T. **Transfer Learning in Natural Language Processing.** *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials*, Minneapolis, MN, 2019. p. 15–18. Disponível em: https://aclanthology.org/N19-5004.pdf. Acesso em: 7 nov. 2025.

SOUZA, F. C. de; NOGUEIRA, R.; LOTUFO, R. **BERTimbau: pretrained BERT models for Brazilian Portuguese.** Campinas: Universidade Estadual de Campinas (Unicamp), 2020. Disponível em: https://repositorio.unicamp.br/handle/REPOSIP/342981. Acesso em: 7 nov. 2025.

SOUZA, Matheus; LIMA, Renan; ALMEIDA, Thales; LOPES, Gabriela; GOMES, Daniel. **Detection and Censorship of Offensive Language in Extended Texts in Portuguese.** In: *Proceedings of the International Conference on Computational Linguistics (COLING)*. Gyeongju, South Korea: International Committee on Computational Linguistics, 2023. p. 1885–1896. Disponível em: https://aclanthology.org/2023.coling-main.163/. Acesso em: 8 nov. 2025.

SUN, YIZHOU; HAN, JIAWEI. **Mining heterogeneous information networks: principles and methodologies.** Synthesis Lectures on Data Mining and Knowledge Discovery, v. 3, n. 2, p. 1–159, 2012. Disponível em: https://www.morganclaypool.com/doi/10.2200/S00442ED1V01Y201209DMK005. Acesso em: 8 nov. 2025.

VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, Ł.; POLOSUKHIN, I. Attention Is All You Need. Advances in Neural Information Processing Systems (NeurIPS), v. 30, 2017. Disponível em: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf. Acesso em: 7 nov. 2025.

VAJJALA, Sowmya; MAJUMDER, Bodhisattwa; VARGAS, Francielle; SANTOS, Leandro; RIBEIRO, Manoel. **Practical Natural Language Processing**. Sebastopol: O'Reilly Media, 2020. Disponível em: https://www.oreilly.com/library/view/practical-natural-language/9781492054047/titlepage01.html. Acesso em: 7 fev. 2026.

VARGAS, Francielle; CARVALHO, Isabelle; GOES, Fabiana; PARDO, Thiago A. S.; BENEVENUTO, Fabrício. **HateBR: A Large Expert Annotated Corpus of Brazilian Instagram Comments for Offensive Language and Hate Speech Detection.** In: *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, 13., 2022, Marseille. *Anais...* Marseille: European Language Resources Association (ELRA), 2022.

WAGNER FILHO, J. A.; WILKENS, R.; IDIART, M.; VILLAVICENCIO, A. **The brWaC Corpus: A New Open Resource for Brazilian Portuguese.** In: *Proceedings of the 11th International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japão: ELRA, 2018.

WETTIG, Alexander; GAO, Tianyu; ZHONG, Zexuan; CHEN, Danqi. **Should You Mask 15% in Masked Language Modeling?** In: *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2023)*. Dubrovnik, Croatia: Association for Computational Linguistics, 2023. p. 2985–3000. Disponível em: https://aclanthology.org/2023.eacl-main.220/. Acesso em: 8 nov. 2025.

WU, Yonghui; SCHUSTER, Mike; CHEN, Zhifeng; LE, Quoc V.; NOROUZI, Mohammad; MACHEREY, Wolfgang; KRIKUN, Maxim; CAO, Yuan; GAO, Qin; MACHEREY, Klaus et al. **Google's neural machine translation system: Bridging the gap between human and machine translation.** *Transactions of the Association for Computational Linguistics*, v. 4, p. 339–351, 2016. Disponível em: https://aclanthology.org/Q16-1024/. Acesso em: 8 dez. 2025.