Trabalho de Conclusão de Curso

Identificação Automática de Comentários Tóxicos em Discussões Online: Uma Análise de Toxidade Utilizando Processamento de Linguagem Natural e Aprendizado de Máquina

Jadson César da Silva Santos
jcss@ic.ufal.br

Orientadores:

Leandro Dias da Silva
Marcos Antonio Barbosa Lima

Maceió, Abril de 2024

Jadson César da Silva Santos

Identificação Automática de Comentários Tóxicos em Discussões Online: Uma Análise de Toxidade Utilizando Processamento de Linguagem Natural e Aprendizado de Máquina

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Engenharia de Computação do Instituto de Computação da Universidade Federal de Alagoas.

Orientadores:

Leandro Dias da Silva

Marcos Antonio Barbosa Lima

Maceió, Abril de 2024

Catalogação na Fonte
Universidade Federal de Alagoas
Biblioteca Central
Divisão de Tratamento Técnico

Bibliotecário: Antonia Izabel da Silva Meyer – CRB-4 - 1558

S237i Santos, Jadson César da Silva.
Identificação automática de comentários tóxicos em discussões online : uma análise de toxidade utilizando processamento de linguagem natural e aprendizado de máquina / Jadson César da Silva Santos. – 2024.
43 f. : il.

Orientador: Leandro Dias da Silva.
Coorientador: Marcos Antonio Barbosa Lima.
Monografia (Trabalho de conclusão de curso em Engenharia da Computação) – Universidade Federal de Alagoas, Instituto de Computação. Maceió, 2024.

Bibliografia: f. 41-43.

1. Processamento de Linguagem Natural. 2. Aprendizado do computador. 3. Ambiente online - Comentários Tóxicos. I. Título.

CDU: 004.89

UNIVERSIDADE FEDERAL DE ALAGOAS/UFAL

Instituto de Computação – IC

Campus A. C. Simões - Av. Lourival de Melo Mota, BL 12
Tabuleiro do Martins, Maceió/AL - CEP: 57.072-970
Telefone: (082) 3214-1401

Trabalho de Conclusão de Curso - TCC

Formulário de Avaliação

Curso: Engenharia de Computação

Nome do Aluno

Jadson César da Silva Santos

Nº de Matrícula

18211019

Título do TCC (Tema)

Identificação Automática de Comentários Tóxicos em Discussões Online: Uma Análise de Toxicidade Utilizando Processamento de Linguagem Natural e Aprendizado de Máquina

Banca Examinadora

Leandro Dias da Silva
Nome do Orientador

Documento assinado digitalmente
gov.br LEANDRO DIAS DA SILVA
Data: 30/04/2024 15:13:21-0300
Verifique em https://validar.iti.gov.br

Assinatura

Marcos Antonio Barbosa Lima
Nome do Co-Orientador

Documento assinado digitalmente
gov.br MARCOS ANTONIO BARBOSA LIMA
Data: 06/05/2024 15:57:39-0300
Verifique em https://validar.iti.gov.br

Assinatura

Baldoino Fonseca dos Santos Neto
Nome do Professor

Documento assinado digitalmente
gov.br BALDOINO FONSECA DOS SANTOS NETO
Data: 02/05/2024 10:33:38-0300
Verifique em https://validar.iti.gov.br

Assinatura

Evandro de Barros Costa
Nome do Professor

Documento assinado digitalmente
gov.br EVANDRO DE BARROS COSTA
Data: 06/05/2024 13:19:53-0300
Verifique em https://validar.iti.gov.br

Assinatura

Data da Defesa

30/04/2024

Nota Obtida

9,5 (nove e meio)

Observações

Coordenador do Curso
De Acordo

Documento assinado digitalmente
gov.br LEANDRO DIAS DA SILVA
Data: 30/04/2024 15:14:43-0300
Verifique em https://validar.iti.gov.br

Assinatura

Agradecimentos

Com profunda gratidão, dedico esta seção de agradecimentos a Deus por tudo que fez e faz em minha vida, aos meus pais Geraldo e Maria Gisélia, principalmente à minha mãe, que sempre me incentivou para os estudos. Expresso também minha gratidão às minhas irmãs Gisele e Joelma pela influência na construção da pessoa que sou hoje, ao meu cunhado Tiago, cuja influência foi fundamental na escolha da minha área acadêmica, e à minha noiva Débora, que tem acompanhado de perto boa parte da minha trajetória e rotina em direção à conclusão do curso.

Quero manifestar minha gratidão aos colegas de graduação pelo apoio durante essa jornada. Sem a colaboração deles em diversos momentos, teria sido ainda mais desafiador. Agradeço também aos amigos que fiz dentro e fora da universidade, que compartilham comigo a alegria deste momento de encerramento de ciclo.

Durante a graduação, participei como membro dos programas PAESPE e PET Ciência e Tecnologia, cuja influência foi fundamental para meu desenvolvimento pessoal, além de terem contribuído para minha entrada na universidade quando ainda cursava o ensino médio. Sou profundamente grato a eles por isso.

Agradeço ao LCCV por ter exercido uma influência significativa na minha escolha de carreira e por contribuir continuamente para o meu desenvolvimento profissional.

Por fim, expresso minha gratidão ao Instituto de Computação, à UFAL como um todo, aos meus professores dentro e fora da graduação, ao meu orientador Leandro Dias e ao coorientador Marcos Lima por me guiarem nesta etapa final.

Que este trabalho não apenas marque o fim de uma jornada, mas também o início de novas conquistas e aprendizados que continuarão a enriquecer minha trajetória acadêmica e profissional.

Resumo

Este trabalho aborda a classificação de comentários tóxicos presentes em plataformas de mídia social e seu impacto, utilizando como base de dados da competição "Jigsaw Toxic Comment Classification Challenge"da plataforma Kaggle. O problema dos comentários tóxicos é discutido em relação ao ambiente online e sua influência negativa. O objetivo do estudo é contribuir para uma moderação eficaz da toxicidade nas plataformas digitais por meio de modelos de classificação de machine learning. A metodologia envolveu treinamentos iniciais, onde cada modelo foi designado para uma classe específica e selecionados aqueles com a maior pontuação F1-score para refinamento dos hiperparâmetros. Foram utilizados os algoritmos SVM, RF, LR e LSTM, juntamente com os vetorizadores TFIDF, CountVectorizer e Tokenizer (exclusivamente para LSTM). Os resultados demonstraram resultados satisfatórios, superando em alguns casos as expectativas da literatura, com validação cruzada que comprovou a robustez dos modelos. Conclui-se que os objetivos do trabalho foram alcançados, e sugere-se para análises futuras a incorporação de técnicas utilizadas na competição e na literatura para melhorar o desempenho na competição da base de dados.

Palavras-chave: Processamento de Linguagem Natural; Aprendizado de Máquina; Classificação de Textos; Comentários Tóxicos; Validação Cruzada

Abstract

This work addresses the classification of toxic comments present on social media platforms and their impact, using the "Jigsaw Toxic Comment Classification Challenge"dataset from the Kaggle platform. The issue of toxic comments is discussed in relation to the online environment and its negative influence. The study aims to contribute to effective moderation of toxicity on digital platforms through machine learning classification models. The methodology involved initial training, where each model was assigned to a specific class, and those with the highest F1-score were selected for hyperparameter refinement. The algorithms SVM, RF, LR, and LSTM were used, along with the vectorizers TFIDF, CountVectorizer, and Tokenizer (exclusively for LSTM). The results showed satisfactory outcomes, surpassing, in some cases, the expectations of the literature, with cross-validation confirming the robustness of the models. It is concluded that the objectives of the work were achieved, and it is suggested for future analyses to incorporate techniques used in the competition and literature to enhance performance in the competition dataset.

Key-words: Natural Language Processing; Machine Learning; Text Classification; Toxic Comments; Cross-validation

Conteúdo

1

Introdução

A proliferação de comentários tóxicos nas plataformas digitais é um grande desafio na atual era da Internet, onde a liberdade de expressão entra muitas vezes em conflito com a necessidade de manter um ambiente online seguro e respeitoso. A propagação de conteúdos nocivos, como discursos de ódio, cyberbullying e abusos, não só afeta a qualidade das interações online, mas também pode ter graves impactos emocionais e psicológicos nos utilizadores. Portanto, a identificação e a gestão eficaz de conteúdos tóxicos da internet são cruciais.

Nesse contexto, o campo do Machine Learning emerge como uma ferramenta poderosa para abordar esse problema complexo. A capacidade das técnicas de Machine Learning em analisar grandes volumes de dados e identificar padrões sutis tornou possível a criação de sistemas de classificação de comentários tóxicos de forma automatizada e eficiente. Essa abordagem oferece a oportunidade de extrair informações valiosas e agir proativamente contra conteúdos prejudiciais em tempo real.

Este trabalho de conclusão de curso tem como objetivo específico explorar e avaliar a eficácia de determinadas técnicas de Processamento de Linguagem Natural (PLN) e Aprendizado de Máquina comumente utilizadas na literatura para classificação de comentários tóxicos. A pesquisa destaca a importância de métodos acessíveis e de baixo custo, que podem ser implementados em um computador pessoal padrão, sem a necessidade de infraestrutura computacional complexa e custosa.

Nesse contexto, será utilizada a base de dados "Jigsaw Toxic Comment Classification Challenge", disponível na plataforma Kaggle, como parte fundamental dos experimentos e testes realizados neste estudo. Essa base de dados oferece um conjunto diversificado de comentários rotulados quanto à toxicidade, proporcionando uma oportunidade valiosa de treinar e avaliar modelos de Machine Learning.

Portanto, este TCC tem como objetivo geral fornecer resultados que podem ser utilizados na criação de sistemas de moderação de conteúdo online mais eficazes e acessíveis. Ao final deste estudo, espera-se contribuir para a promoção de ambientes online mais seguros e respeitosos,

disponibilizando conhecimento aplicável em futuras implementações de sistemas de moderação de conteúdo e, assim, enfrentar o desafio da toxicidade em comentários online de forma mais eficiente e acessível.

1.1 Contextualização

Comentários tóxicos em plataformas de mídia social causaram considerável perturbação entre indivíduos e grupos. Esses comentários não se limitam à violência verbal, abrangendo também mensagens rudes, desrespeitosas, comportamento negativo online e outras atitudes semelhantes que podem levar à interrupção de discussões (POOJITHA, K et al., 2023). A disseminação de comentários tóxicos em plataformas de mídia social tem sido uma questão de grande preocupação em um cenário cada vez mais digitalizado. Esses comentários não apenas afetam as interações entre os usuários, mas também têm impactos significativos na saúde mental e no bem-estar das pessoas envolvidas (LUISA I, 2021). Além disso, as consequências podem se estender para além do ambiente online, afetando relacionamentos interpessoais e até mesmo desencadeando conflitos em contextos offline. A natureza abrangente dos comentários tóxicos, que vai além da violência verbal, também levanta questões sobre a responsabilidade das plataformas de mídia social na moderação e controle desse comportamento prejudicial. De acordo com a notícia "Sob pressão, Facebook endurece sua política de moderação de conteúdos" (2020), o Facebook esteve sob pressão para implementar medidas mais eficazes de identificação e remoção de conteúdo tóxico, a fim de manter um ambiente digital mais seguro e saudável para seus usuários. Portanto, a identificação e mitigação de comentários tóxicos tornaram-se uma tarefa crucial tanto para os usuários individuais quanto para as empresas que operam essas plataformas.

1.2 Objetivos Gerais

Os objetivos gerais deste Trabalho de Conclusão de Curso são:

- Investigar e avaliar a eficácia das técnicas utilizadas, neste trabalho, na identificação de comentários tóxicos em discussões de plataformas digitais.
- Proporcionar ideias e resultados práticos que possam ser aplicados no desenvolvimento de sistemas de moderação de conteúdo online, enfatizando abordagens acessíveis e viáveis em ambientes com recursos computacionais limitados.
- Contribuir para a criação de ambientes virtuais mais seguros e respeitosos, promovendo uma interação online positiva e saudável.

1.3 Objetivos Específicos

Como objetivos específicos, o trabalho visa:

- Analisar e comparar diferentes técnicas de Processamento de Linguagem Natural (PLN) comumente encontradas na literatura para a classificação de comentários tóxicos.
- Desenvolver modelos de Aprendizado de Máquina capazes de identificar e classificar comentários tóxicos em plataformas digitais.
- Avaliar o desempenho dos modelos de PLN e Aprendizado de Máquina, considerando métricas relevantes para o contexto.

1.4 Base de Dados

A competição "Jigsaw Toxic Comment Classification Challenge"é um desafio de ciência de dados hospedado na plataforma Kaggle. O objetivo dessa competição é desenvolver modelos de Aprendizado de Máquina e Processamento de Linguagem Natural (PLN) capazes de classificar comentários em plataformas digitais como tóxicos ou não tóxicos. Essa classificação é crucial para a moderação de conteúdo online e para manter ambientes online seguros e respeitosos.

A base de dados associada a essa competição consiste em uma coleção de comentários extraídos da Wikipedia, os quais foram rotulados de acordo com sua toxicidade. Os rótulos indicam se um comentário é tóxico ou não, e os comentários tóxicos podem ser classificados em diferentes níveis de toxicidade, tais como tóxico, severamente tóxico, insultante, obsceno, ameaçador e de ódio à identidade. Essa variedade de rótulos permite a criação de modelos capazes de detectar diferentes tipos de comportamento prejudicial.

Neste trabalho, a base de dados será utilizada para treinar modelos de Aprendizado de Máquina e Redes Neurais, visando identificar e classificar comentários tóxicos em plataformas online.

2

Referencial Teórico

2.1 Machine Learning e suas Aplicações

O Machine Learning (Aprendizado de Máquina) é um ponto crucial para entender os avanços na inteligência artificial (IA) e suas aplicações em diversas áreas. O Machine Learning se diferencia da programação tradicional, pois capacita computadores a aprenderem a partir de dados, em vez de serem programados de forma explícita para tarefas específicas. Como mencionado por Domingos (2015), "os algoritmos de machine learning aprendem automaticamente através de dados e melhoram com mais dados. Agora, não é mais necessário programar computadores, pois eles se auto programam", e isso representa uma mudança fundamental na abordagem da resolução de problemas computacionais.

No cerne do Machine Learning está a capacidade de construir modelos matemáticos e estatísticos que podem aprender com os dados disponíveis (Goswami; Sinha, 2022). Isso significa que, em vez de programar regras rígidas para resolver um problema, podemos alimentar um algoritmo com exemplos de entrada e saída, permitindo que ele identifique padrões e tome decisões com base nesses padrões. Essa abordagem é particularmente poderosa quando lidamos com problemas complexos para os quais as soluções não podem ser facilmente codificadas de maneira convencional.

O aprendizado que as máquinas adquirem refere-se à capacidade dos modelos de melhorar seu desempenho em tarefas à medida que são expostos a mais dados. Isso envolve reconhecer padrões nos dados, fazer generalizações e tomar decisões com base nessas informações.

Existem três tipos principais de aprendizado de máquina que são:

- Aprendizado Supervisionado: envolve treinar um algoritmo em um conjunto de dados rotulados, onde as entradas estão associadas a saídas conhecidas. O algoritmo aprende a mapear as entradas para as saídas através do ajuste de seus parâmetros. Isso engloba tarefas como Classificação e Regressão, e todo o trabalho será fundamentado nesse tipo de aprendizado.

- Aprendizado não Supervisionado: o algoritmo é treinado em dados não rotulados e busca descobrir padrões ou estruturas nos dados. Isso engloba tarefas como Clusterização, Regras de Associação e Redução de Dimensionalidade.
- Aprendizado por Reforço: um agente aprende a tomar decisões sequenciais em um ambiente para maximizar uma recompensa cumulativa. O agente aprende através de tentativa e erro, explorando diferentes ações e observando as recompensas associadas.

As aplicações do Machine Learning são vastas e continuam a se expandir rapidamente. Um exemplo notável é a visão computacional, onde algoritmos de Machine Learning são usados para reconhecimento de objetos, detecção de padrões em imagens e até mesmo para conduzir veículos autônomos (IBM, 2023). Essas aplicações demonstram como os modelos de Machine Learning podem aprender a partir de dados visuais e tomar decisões em tempo real com base nesses dados.

Outra área de aplicação crucial é o Processamento de Linguagem Natural (PLN), que é a área na qual será trabalhada ao longo deste trabalho. No PLN, o Machine Learning é usado para capacitar sistemas a compreender, gerar e interagir com a linguagem humana. De acordo com Titenok (2022), isso inclui a tradução automática de idiomas, a análise de sentimentos em texto, a geração de respostas automáticas em chatbots e a classificação de textos, como identificar se um comentário é tóxico ou não, trazendo o exemplo para o que será utilizado neste trabalho. O PLN tem aplicações em atendimento ao cliente automatizado, análise de redes sociais e muito mais, desempenhando um papel central na melhoria da interação entre humanos e máquinas.

Além disso, o Machine Learning é amplamente utilizado em análise de dados e aprendizado estatístico. Ele permite que as empresas identifiquem tendências, façam previsões e tomem decisões baseadas em dados, impulsionando o avanço em áreas como finanças, medicina e ciência (CAMPOS, W et al., 2022) (AZEVEDO, C et al., 2023). A capacidade de processar grandes volumes de dados e extrair informações valiosas é uma vantagem competitiva significativa em muitos setores.

2.2 Processamento de Linguagem Natural

Processamento de Linguagem Natural (PLN) é uma abordagem computadorizada para analisar textos que envolve um conjunto de teorias e tecnologias. É uma área de pesquisa e desenvolvimento ativa, e embora não haja uma definição única aceita universalmente, existem aspectos comuns a qualquer definição informada. Como definido por Liddy (2001), o Processamento de Linguagem Natural é um conjunto de técnicas computacionais embasadas em teorias que se destinam a analisar e representar textos que ocorrem naturalmente em um ou mais níveis de análise linguística. Isso é feito com o objetivo de alcançar um processamento de linguagem que se assemelhe ao processamento feito pelos seres humanos, visando atender a diversas tarefas e aplicações.

Na abordagem da classificação, um conceito fundamental no PLN que aprimora a compreensão e eficácia desses processos é o pré-processamento do corpus. Para os propósitos deste estudo, corpus será definido como o conjunto organizado de textos utilizado para a análise linguística, ou seja, a base de dados mencionada no tópico 1.4 deste trabalho. O corpus é composto por documentos, que, por sua vez, representam as unidades individuais de texto. Neste contexto específico, cada documento será um comentário da base de dados.

Pré-processamento Textual: Antes de aplicar qualquer algoritmo de classificação, o pré-processamento textual emerge como um estágio crucial. Este estágio incorpora uma série de técnicas essenciais para a preparação adequada dos dados textuais, visando otimizar a performance dos algoritmos subsequentes.

- Tokenização: também chamada de segmentação de palavras, conforme descrito por Palmer (2010), é o processo de dividir a sequência de caracteres em um texto, destacando os limites entre as palavras, ou seja, indicando onde uma palavra termina e outra começa. No campo da linguística computacional, as palavras identificadas nesse processo são geralmente referidas como tokens. Essa abordagem facilita a manipulação e análise de componentes individuais do texto, constituindo um passo fundamental para o entendimento semântico.

- Remoção de Stopwords: segundo Savoy e Gaussier (2010), essa remoção ocorre por duas razões principais. Primeiramente, ela possibilita que a correspondência entre consultas e documentos se baseie somente em palavras com significado real. As stopwords, também conhecidas como palavras vazias por não trazerem muito significado, interferem no processo de recuperação e prejudicam o desempenho, já que não distinguem entre documentos relevantes e não relevantes. Em segundo lugar, a retirada de stopwords contribui para reduzir o tamanho de armazenamento da coleção indexada, buscando uma redução ideal entre 30% e 50%. Portanto, eliminar elementos redundantes e focar na extração de informações mais relevantes, contribui para a eficiência da classificação.

- Normalização de textos: é o processo de converter palavras não padronizadas em palavras padrão, a fim de tornar o texto mais legível e compreensível. Isso é necessário porque os textos reais contêm muitas palavras não padronizadas, como números, abreviações, datas, quantias de moeda e acrônimos, que não podem ser encontrados em um dicionário e não têm uma pronúncia padrão. De acordo com Sproat el al (2001), a normalização de textos é uma tarefa complexa, pois existem muitos tipos diferentes de palavras não padronizadas e cada caso requer um processamento especial. Como mencionado por Bertaglia (2017), "não há consenso na literatura sobre o que é de fato normalização", portanto, a normalização de textos não tem etapas definidas, podendo variar dependendo das necessidades de cada aplicação. Como exemplos de etapas de normalização, existem a conversão para letras minúsculas, remoção de acentos e lematização, sendo essa última a simplificação

de palavras flexionadas ou derivadas, reduzindo-as à sua forma base, chamada "lema". Por exemplo, no caso do verbo em inglês "DELIVER", as palavras "delivers", "deliver", "delivering"e "delivered"compartilham o mesmo lema "deliver"(Hippisley, 2010). Para este trabalho, as etapas de normalização utilizadas serão a remoção de caracteres não alfabéticos e a lematização.

2.3 Representação Vetorial de Textos

Com o corpus pré-processado, os documentos estarão menos sujeitos a ruídos e mais concisos, o que influencia positivamente nos resultados. A eficácia desses resultados está intrinsecamente ligada à representação numérica dos documentos, uma vez que os algoritmos operam exclusivamente com entradas numéricas. Surge, portanto, a necessidade de representar os textos em vetores númericos e um dos termos, pelo qual esse processo é conhecido na literatura, chama-se vetorização de textos. Na literatura, diversas técnicas de vetorização de texto são exploradas, e para este trabalho, optou-se por aquelas amplamente utilizadas em artigos sobre classificação de textos, tais como o CountVectorizer e TF-IDF. Especialmente para a arquitetura de rede neural, que será citada em 2.4.3, foram utilizadas as classes Tokenizer e pad_sequences da biblioteca Keras.

2.3.1 CountVectorizer

O CountVectorizer é uma técnica essencial no campo de PLN utilizada para a representação e vetorização de texto.

A ideia fundamental por trás do CountVectorizer é contar a frequência das palavras em um corpus. Cada documento é representado como um vetor onde cada componente do vetor corresponde a uma palavra única presente em todos os documentos da coleção. A contagem de frequência de uma palavra em um documento é usada como o valor daquela palavra na posição correspondente no vetor.

Utilizando o exemplo encontrado no portal GeeksforGeeks:

Figura 2.1: Lista representando um corpus com 3 documentos

```
document = [ “One Geek helps Two Geeks”, “Two Geeks help Four Geeks”, “Each Geek helps many other Geeks at GeeksforGeeks.”]
```

Fonte: GeeksforGeeks, 2022.

O CountVectorizer gera uma matriz em que cada palavra única é representada por uma coluna, enquanto cada string da lista "document"corresponde a uma linha. O valor em cada

célula representa a contagem da ocorrência da palavra na respectiva amostra de texto. Essa relação pode ser compreendida visualmente da seguinte maneira:

Figura 2.2: Matriz representando a contagem de palavras únicas em cada documento do corpus

	at	each	four	geek	geeks	geeksforgeeks	help	helps	many	one	other	two
document[0]	0	0	0	1	1	0	0	1	0	1	0	1
document[1]	0	0	1	0	2	0	1	0	0	0	0	1
document[2]	1	1	0	1	1	1	0	1	1	0	1	0

Fonte: GeeksforGeeks, 2022.

Observações:

1. Há 12 palavras únicas no corpus “document”, representadas como colunas da tabela.
2. Existem 3 documentos no corpus, sendo cada um representado como uma linha da tabela.
3. Cada célula contém um número que representa a contagem da palavra naquele documento específico.
4. Todas as palavras foram convertidas para minusculas.
5. As palavras nas colunas foram organizadas em ordem alfabética.

Dentro do CountVectorizer, essas palavras não são armazenadas como strings. Em vez disso, elas recebem um valor de índice específico. Neste caso, ’at’ teria o índice 0, ’each’ teria o índice 1, ’four’ teria o índice 2, e assim por diante. Segue a representação na tabela abaixo, conhecida como Matriz Esparsa:

Figura 2.3: Matriz com índices atribuídos às colunas

0	1	2	3	4	5	6	7	8	9	10	11
0	0	0	1	1	0	0	1	0	0	0	1
0	0	1	0	2	0	1	0	0	0	0	1
1	1	0	1	1	1	0	1	1	0	1	0

Fonte: GeeksforGeeks, 2022.

O CountVectorizer é uma técnica simples, mas poderosa, para a representação de texto, pois captura informações importantes sobre a frequência das palavras em documentos. No entanto,

ele tem como limitação o fato de não considerar a ordem das palavras, o que significa que perde informações sobre a estrutura do texto.

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

O Term Frequency - Inverse Document Frequency (TF-IDF), traduzindo para o português "Frequência do Termo - Frequência Inversa do Documento" é uma estatística numérica que reflete o quão importante uma palavra é para um documento (Munot; Govilkar, 2014).

Quando analisamos documentos, procuramos por palavras-chave significativas. A intuição inicial pode ser pensar que as palavras mais frequentes em um documento são as mais importantes. No entanto, isso geralmente está equivocado, já que as palavras mais comuns, como "o"e "e", não fornecem informações distintas sobre o conteúdo. Portanto, é comum remover essas stopwords antes da análise.

As palavras que realmente indicam o tópico são geralmente aquelas menos comuns. No entanto, nem todas as palavras raras são úteis. Algumas aparecem raramente, mas não contribuem para a compreensão do documento. Outras, mesmo sendo igualmente raras, são fundamentais para identificar o tópico, como o termo "chukker"indicando o esporte de polo.

A medida formal de quão concentradas são as ocorrências de uma palavra específica em poucos documentos é o TF-IDF. De acordo com Kao e Poteet (2007), a fórmula é:

$$tfidf(t_i, d_j) = tf(t_i, d_j) \cdot \log\left(\frac{N}{N(t_i)}\right) \tag{2.1}$$

e a versão normalizada é:

$$wi, j = \frac{tfidf(t_i, d_j)}{\sqrt{\sum_{k=1}^{|T|} tfidf(t_k, d_j)^2}} \tag{2.2}$$

onde N e $|T|$ indicam o número total de documentos e termos únicos contidos na coleção, respectivamente, e $N(ti)$ representa o número de documentos na coleção nos quais o termo ocorre pelo menos uma vez, e

$$tf(t_i, d_j) = \begin{cases} 1 + \log(n(t_i, d_j)), \text{ se } n(t_i, d_j) > 0; \\ 0, \text{caso contrário} \end{cases} \tag{2.3}$$

onde $n(t_i, d_j)$ representa o número de ocorrências do termo no documento. Na prática, o somatório na equação 2.2 leva em consideração apenas os termos presentes no documento em questão.

2.3.3 Tokenizer e pad_sequence no Keras

O Keras é uma biblioteca de aprendizado profundo popular em Python, que oferece uma API de alto nível para desenvolvimento rápido e fácil de modelos de redes neurais (Keras, 2024). No contexto de PLN, é possível utilizar o Keras para construir e treinar modelos de redes neurais para tarefas como classificação de texto.

A classe Tokenizer do Keras é uma ferramenta para pré-processamento de texto, permitindo a vetorização de um corpus. Ao instanciar um objeto Tokenizer, é possível convertê-lo em uma lista de inteiros, onde cada inteiro representa um token único no texto (Faroit, 2024). Além disso, é possível definir o parâmetro num_words para limitar o tamanho do vocabulário criado pelo Tokenizer (TensorFlow, 2024).

Por outro lado, pad_sequences é uma função do Keras usada para ajustar o tamanho de sequências de texto. Esta função preenche ou trunca as sequências de texto para um comprimento específico, garantindo que todas as sequências tenham o mesmo tamanho (TensorFlow, 2024). Essa padronização do tamanho das sequências é crucial ao treinar modelos de redes neurais, já que muitos deles exigem entradas de tamanho fixo.

2.4 Algoritmos de Classificação

A classificação de textos consiste em atribuir rótulos ou categorias a documentos com base em seu conteúdo textual. Para alcançar esse objetivo, são empregados algoritmos de aprendizado supervisionado, os quais aprendem a partir de exemplos previamente rotulados. A aplicação do PLN na classificação de textos envolve a extração de características relevantes dos documentos, bem como a escolha de algoritmos de classificação adequados. Dentre os algoritmos mais frequentemente mencionados na literatura para essa tarefa, destacam-se o Support Vector Machine (SVM), Random Forest (RF), Redes Neurais Long Short-Term Memory (LSTM) e Logist Regression (LR).

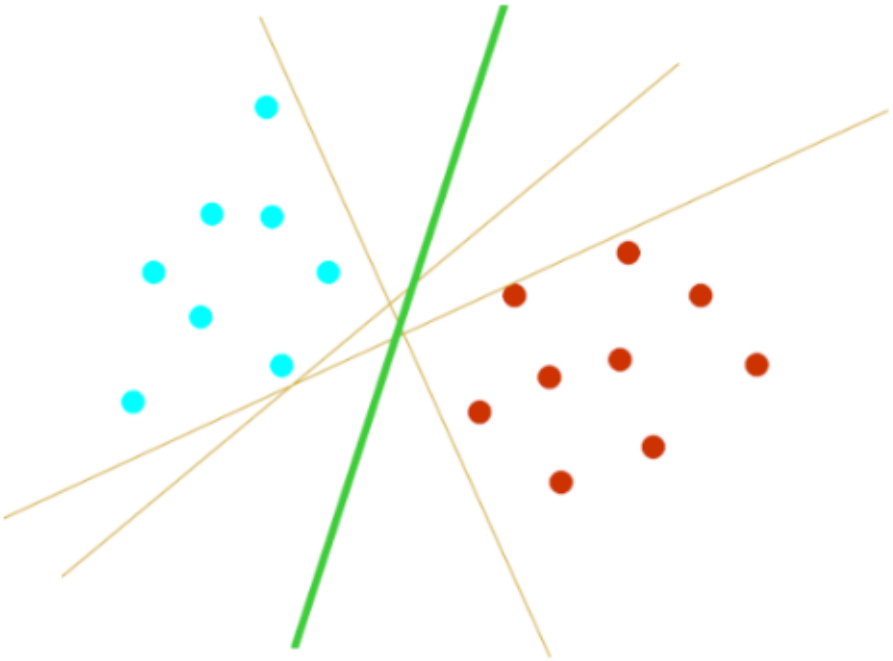
2.4.1 Support Vector Machine

Como consta no artigo de Gunn (1998) intitulado “Support Vector Machines for Classification and Regression”, o Support Vector Machine (Máquina de Vetores de Suporte ou SVM) é um algoritmo de Aprendizado de Máquina utilizado para classificação e regressão de dados sendo originalmente desenvolvido por Vapnik e colaboradores.

O problema de classificação pode ser simplificado ao considerarmos apenas duas classes, sem perder a generalidade. Nesse cenário, o objetivo é separar as duas classes por meio de uma função baseada nos exemplos disponíveis. Queremos criar um classificador que funcione bem mesmo para dados não vistos, ou seja, que seja capaz de generalizar corretamente. Considere o exemplo mostrado na Figura 2.4. Embora haja várias maneiras de traçar fronteiras lineares para

separar os dados, há apenas uma que maximiza a distância entre ela e os pontos de dados mais próximos de cada classe. Essa fronteira linear é chamada de hiperplano de separação ótimo. Intuitivamente, esperamos que essa fronteira seja mais eficaz na generalização do que outras possíveis fronteiras.

Figura 2.4: Hiperplano de separação ideal



Fonte: Gunn, 1998.

Considerando o problema de separar o conjunto de vetores de treinamento pertencentes a duas classes distintas,

$$D = \{(x_1, y_1), \ldots, (x_l, y_l)\}, x \in \mathrm{R}^n, y \in \{-1, 1\} \tag{2.4}$$

com um hiperplano,

$$< w, x > + \ b = 0. \tag{2.5}$$

onde w é um vetor de pesos, x é o vetor de características de entrada e b é o termo de viés. A função $< \cdot, \cdot >$ representa o produto escalar entre os vetores.

O objetivo é encontrar o hiperplano que separa as duas classes de forma ótima. Isso significa que queremos encontrar um hiperplano que maximize a margem entre as duas classes, ou seja, a distância entre o hiperplano e os pontos de cada classe.

Para encontrar o hiperplano separador ótimo, precisamos resolver o problema de otimização, onde tentamos maximizar a margem sujeito à restrição de que todos os pontos de treinamento estejam corretamente classificados. Isso pode ser formulado como:

$$\begin{aligned} \text{maximize} \quad & \rho \\ \text{sujeito a} \quad & y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq \rho, \quad i = 1, \ldots, l \end{aligned} \tag{2.6}$$

onde ρ é a margem entre as duas classes e l é o número de vetores de treinamento. A restrição garante que todos os pontos de treinamento estejam do lado correto do hiperplano separador.

A distância d de um ponto x_i ao hiperplano é dada por:

$$d(\mathbf{w}, b; \mathbf{x}_i) = \frac{|\langle \mathbf{w}, \mathbf{x}_i \rangle + b|}{\|\mathbf{w}\|} \tag{2.7}$$

A partir desta formulação, podemos deduzir que a margem ρ é igual a $\frac{2}{\|\mathbf{w}\|}$, pois a distância do hiperplano a qualquer ponto de treinamento é $\frac{1}{\|\mathbf{w}\|}$.

Portanto, o hiperplano que separa as duas classes de forma ótima é aquele que minimiza $\|\mathbf{w}\|^2$, sujeito às restrições da equação acima. Isso pode ser expresso como o seguinte problema de otimização:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{sujeito a} \quad & y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1, \quad i = 1, \ldots, l \end{aligned} \tag{2.8}$$

Uma característica notável do SVM é sua capacidade de lidar com dados não linearmente separáveis. Para fazer isso, o SVM usa truques de kernel, que mapeiam os dados originais para um espaço de maior dimensão onde eles se tornam linearmente separáveis. Os tipos de kernel mais comuns incluem o kernel linear, o kernel polinomial e o kernel radial (RBF). A escolha do kernel depende da natureza dos dados e da complexidade do problema.

2.4.2 Random Forest

Segundo Breiman (2001), “Random Forest é um classificador composto por uma coleção de classificadores em forma de árvore $h(x,k), k = 1, ...$ onde os k são vetores aleatórios independentes e identicamente distribuídos e cada árvore lança um voto unitário para a classe mais popular na entrada x”. O Random Forest (ou floresta aleatória em português) é um poderoso algoritmo de aprendizado de máquina utilizado em problemas de classificação e regressão. Este algoritmo pertence à classe de métodos ensemble, que combinam múltiplos modelos de aprendizado de máquina para melhorar o desempenho preditivo.

O algoritmo opera criando uma "floresta"de árvores de decisão, onde cada árvore é treinada em uma amostra aleatória dos dados de treinamento e usando um subconjunto aleatório das características. Durante a predição, as previsões de todas as árvores individuais são combinadas através de votação (no caso da classificação) ou média (no caso da regressão) para produzir a saída final. Uma das características distintivas do Random Forest é a seleção aleatória de

características em cada árvore de decisão. Isso garante que as árvores individuais sejam diversificadas, o que ajuda a reduzir a correlação entre elas e a aumentar a robustez do modelo.

Os principais parâmetros do Random Forest incluem o número de árvores na floresta, o critério de divisão de nó (como Gini impurity ou entropia), a profundidade máxima da árvore e o número mínimo de amostras necessárias para dividir um nó, entre outros. A escolha adequada desses parâmetros é crucial para o desempenho e a generalização do modelo.

O Random Forest oferece várias vantagens, como boa precisão em uma ampla gama de problemas, robustez a overfitting, eficiência computacional e é facilmente paralelizável (Breiman, 2001).

2.4.3 Redes Neurais LSTM

As Long Short-Term Memory (LSTM) ou Memória de Longo Curto Prazo em português, são um tipo especial de rede neural recorrente (RNN) desenvolvida para lidar com problemas de dependências de longo prazo em sequências de dados, como as encontradas em tarefas de PLN. As LSTMs foram propostas por Hochreiter e Schmidhuber em 1997 (Hochreiter; Schmidhuber, 1997).

O funcionamento das LSTMs é baseado em unidades de memória chamadas células, que possuem mecanismos de controle de fluxo de informações para armazenar e acessar informações relevantes ao longo do tempo. As células LSTM possuem três portas principais: a porta de entrada (input gate), a porta de saída (output gate) (Hochreiter; Schmidhuber, 1997) e a porta de esquecimento (forget gate), sendo a última proposta somente em 1999 por Gers, Schmidhuber e Cummins (Gers; Schmidhuber; Cummins, 1999). Cada porta é responsável por regular a interação entre a entrada atual, o estado anterior da célula e o estado atual da célula, permitindo que a LSTM aprenda a reter ou descartar informações com base no contexto da sequência.

As LSTMs são interessantes para o uso em tarefas de PLN, como tradução automática, análise de sentimentos e geração de texto, devido à sua capacidade de capturar dependências de longo prazo e reter informações relevantes ao longo de sequências extensas de texto.

2.4.4 Logistic Regression

A Logistic Regression (Regressão Logística) é um método estatístico amplamente empregado para examinar a relação entre uma variável de resultado binária ou dicotômica e uma ou mais variáveis independentes (Hosmer, Lemeshow e Sturdivant, 2013). Esse método é especialmente útil quando a variável de resultado é discreta, assumindo dois ou mais valores possíveis. O objetivo da regressão logística é estimar a probabilidade de que a variável de resultado pertença a uma categoria específica com base nos valores das variáveis independentes.

Na regressão logística, a variável de resultado assume dois resultados possíveis, como, por exemplo, presença ou ausência de uma doença. O modelo de regressão logística calcula a

probabilidade de que a variável de resultado esteja em uma das duas categorias em função das variáveis independentes. Essa relação entre as variáveis independentes e a probabilidade do resultado é modelada utilizando a função logística.

Ao contrário da regressão linear, que assume uma variável de resultado contínua e modela a relação de forma linear, a regressão logística modela a probabilidade de um resultado binário usando a função logística, resultando em uma relação curvilínea em forma de S. Ademais, a regressão logística parte do pressuposto de que os erros seguem uma distribuição binomial, ao invés de uma distribuição normal.

O processo de ajuste do modelo de regressão logística aos dados é realizado utilizando a estimativa de máxima verossimilhança. Essa técnica busca encontrar os valores dos parâmetros do modelo que maximizam a probabilidade de observar os dados fornecidos. O processo de estimação envolve a otimização iterativa dos parâmetros do modelo para que este se ajuste melhor aos dados observados.

2.5 Avaliação de modelos

A avaliação de modelos de aprendizado de máquina é uma etapa que visa selecionar e validar modelos que apresentem um bom desempenho em prever ou classificar dados não vistos. A escolha adequada de métricas de avaliação e a utilização de técnicas de validação são essenciais para garantir a eficácia e a generalização dos modelos desenvolvidos.

2.5.1 Métricas de Avaliação

As métricas de avaliação desempenham um papel crítico na avaliação de modelos de Aprendizado de Máquina, ajudando a medir o desempenho e a eficácia dos modelos em relação às tarefas específicas. Entre essas métricas, as que serão abordadas neste trabalho e que são comumente utilizadas em artigos científicos relacionados à toxicidade na web, são a Acurácia, Precisão, Cobertura, F1 Score e ROC AUC, cada uma delas desempenhando um papel específico na avaliação de diferentes aspectos do modelo.

Acurácia (Accuracy)

De acordo com Poojitha, K. et al. (2023), a acurácia indica a proporção de previsões corretas em relação ao total de previsões feitas pelos classificadores nos dados de teste. A pontuação máxima de acurácia é 1, indicando que todas as previsões feitas pelo classificador estão corretas, enquanto a pontuação mínima de acurácia pode ser 0. A acurácia pode ser calculada da seguinte forma:

$$Acurácia = \frac{Número\ de\ previsões\ corretas}{Número\ total\ de\ previsões} \tag{2.9}$$

Outra forma para calcular a acurária é utilizando:

$$Acurácia = \frac{TP + TN}{TP + FP + TN + FN} \tag{2.10}$$

onde TP é um verdadeiro positivo, TN é um verdadeiro negativo, FP é um falso positivo e FN é um falso negativo.

Precisão (Precision)

A precisão, também conhecida como valor preditivo positivo, indica a proporção de instâncias corretamente classificadas como positivas em relação ao total de instâncias classificadas como positivas pelo modelo. Um valor de precisão de 1 significa que todas as instâncias de dados categorizadas como positivas pelo modelo são realmente positivas.

$$Precisão = \frac{TP}{TP + FP} \tag{2.11}$$

Cobertura (Recall)

Cobertura representa a proporção de instâncias classificadas corretamente como positivas em relação a todas as instâncias que realmente são positivas. Em outras palavras, a cobertura mede a capacidade do modelo de identificar corretamente todas as instâncias positivas, sem deixar de considerar aquelas que foram erroneamente classificadas como negativas.

$$Cobertura = \frac{TP}{TP + FN} \tag{2.12}$$

F1 Score

A F1 Score é uma métrica que combina precisão e cobertura em uma única medida, fornecendo uma avaliação equilibrada do desempenho de um modelo de classificação, especialmente útil quando há desequilíbrio entre as classes do conjunto de dados. Quanto mais próxima de 1 for a F1 Score, melhor é o desempenho do modelo.

$$F1\ Score = \frac{2 \cdot Precisão \cdot Cobertura}{Precisão + Cobertura} \tag{2.13}$$

ROC AUC

A Curva ROC (Receiver Operating Characteristics) é uma medida de desempenho para problemas de classificação que analisa a capacidade de um modelo em distinguir entre diferentes classes (Narkhede, 2018). A área sob a curva ROC (AUC) é uma métrica que representa a capacidade de separação do modelo, onde valores mais altos indicam um melhor desempenho.

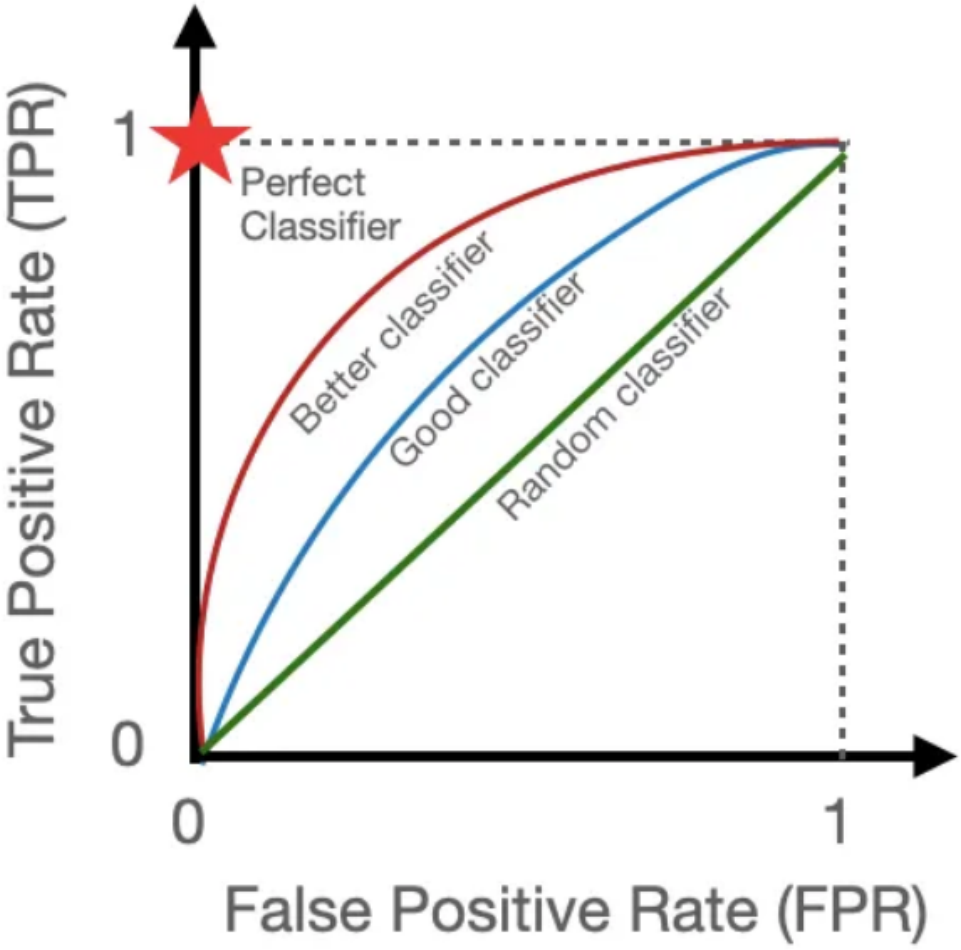
A curva ROC é construída plotando a taxa de verdadeiros positivos (TPR) no eixo y contra a taxa de falsos positivos (FPR) no eixo x. A TPR representa a proporção de exemplos positivos que foram corretamente classificados como positivos, enquanto a FPR representa a proporção de exemplos negativos que foram incorretamente classificados como positivos.

Um modelo ideal terá uma curva ROC que se aproxima do canto superior esquerdo do gráfico, indicando uma alta TPR e uma baixa FPR em vários pontos de corte. Isso significa que o modelo é capaz de distinguir com precisão entre as classes positiva e negativa.

A sensibilidade e especificidade são inversamente proporcionais: aumentar a sensibilidade geralmente leva a uma diminuição na especificidade e vice-versa. A sensibilidade mede a capacidade do modelo de identificar corretamente os verdadeiros positivos, enquanto a especificidade mede a capacidade de evitar falsos positivos.

Na competição da base de dados utilizada, a métrica avaliativa dos competidores é a ROC AUC. Portanto, para avaliar as predições feitas pelos modelos neste trabalho, serão feitas submissões, e a avaliação utilizando essa métrica será importante para a busca de pontuações mais altas.

Figura 2.5: Exemplo de melhor classificador em uma ROC AUC



Fonte: Di Sipio, 2021.

2.5.2 Validação de Modelos

A validação de modelos é um processo crucial para avaliar o desempenho e a capacidade de generalização de um modelo de aprendizado de máquina. Uma técnica amplamente utilizada para realizar essa validação é a validação cruzada. A validação cruzada divide o conjunto de dados em partes (folds) para treinamento e teste, permitindo uma avaliação mais robusta do modelo em dados não vistos.

K-Fold Cross Validation (Validação Cruzada K-Fold)

De acordo com Pedregosa et al. (2011), O K-Fold Cross Validation é uma técnica de validação cruzada que divide o conjunto de dados em k partes iguais. Em seguida, o modelo é treinado k vezes, cada vez usando uma parte diferente como conjunto de teste e as restantes como conjunto de treinamento. Isso permite avaliar o desempenho do modelo de forma mais robusta, pois cada observação é usada tanto para treinamento quanto para teste em diferentes iterações. Ao final, os resultados das k iterações são combinados para obter uma métrica de desempenho geral do modelo.

GridSearchCV

Também de acordo com Pedregosa et al. (2011), o GridSearchCV é uma técnica de busca exaustiva que encontra a melhor combinação de parâmetros para um estimador através da validação cruzada. Ele explora todas as combinações de valores de parâmetros especificados, ajusta o modelo para cada combinação e avalia o desempenho usando validação cruzada. Após a busca, refina o modelo com os melhores parâmetros encontrados e fornece acesso aos resultados, como os melhores parâmetros, pontuação média da validação cruzada e índice da melhor configuração de parâmetros.

3

Trabalhos Relacionados

Esta seção apresenta uma revisão dos estudos anteriores sobre a classificação de comentários tóxicos. Essa revisão é fundamental para embasar a abordagem metodológica e identificar lacunas na literatura. O objetivo é compreender os métodos, descobertas e desafios encontrados em pesquisas anteriores para orientar o trabalho e contribuir para avanços nessa área.

3.1 Machine learning methods for toxic comment classification: a systematic review

O artigo aborda uma revisão sistemática dos métodos de aprendizado de máquina para a classificação de comentários tóxicos (Androcec D, 2020). Inicialmente, destaca-se a importância desse problema, considerando a grande quantidade de comentários deixados em redes sociais, portais de notícias e fóruns, muitos dos quais podem ser tóxicos ou abusivos. A moderação manual desses comentários é inviável devido ao volume, levando à necessidade de sistemas automáticos de detecção de toxicidade.

A revisão analisou 31 estudos relevantes selecionados, investigando vários aspectos, como conjuntos de dados utilizados, métricas de avaliação, métodos de aprendizado de máquina empregados, classes de toxicidade consideradas e idiomas dos comentários. Destaca-se que a maioria dos estudos utiliza conjuntos de dados que contêm comentários rotulados manualmente por sua toxicidade, sendo o conjunto de dados Jigsaw's Toxic Comment Classification Challenge do Kaggle o mais utilizado. Além disso, os estudos empregam uma variedade de métodos de aprendizado de máquina, com destaque para arquiteturas de redes neurais profundas, como CNNs, LSTMs e BERT. As métricas de avaliação mais comuns incluem F1 score, acurácia e área sob a curva ROC (ROC AUC).

Os resultados indicam uma tendência recente de pesquisa nesse campo, impulsionada pelo aumento da disponibilidade de conjuntos de dados rotulados. No entanto, a revisão também

identifica lacunas nas pesquisas, como a necessidade de explorar mais o uso de transformers, como o BERT, e abordar a classificação multilíngue de comentários tóxicos.

3.2 Identification and classification of toxic comments on social media using machine learning techniques

Este estudo aborda a identificação e classificação de comentários tóxicos em mídias sociais utilizando técnicas de aprendizado de máquina (Ozoh, P. A. et al., 2019). Embora o conjunto de dados específico não seja mencionado explicitamente, é possível inferir que se refere ao Jigsaw's Toxic Comment Classification Challenge. Os comentários são submetidos a um processo de pré-processamento antes de serem alimentados aos modelos de classificação, utilizando a técnica TF-IDF. O modelo de regressão logística é empregado para treinar o conjunto de dados processado, distinguindo entre comentários tóxicos e não tóxicos. O modelo multirrótulo abrange todas as classes presentes no conjunto de dados, e sua performance é avaliada utilizando métricas como acurácia e precisão.

O estudo ressalta a importância da classificação automática de comentários tóxicos, um problema fundamental de Processamento de Linguagem Natural (PLN). Ele explora também o uso de técnicas de aprendizado profundo (Deep Learning) para a classificação de comentários, visando aprimorar a detecção e controle do abuso verbal. Além disso, são discutidos métodos de processamento de texto, como a técnica TF-IDF e a regressão logística, assim como a utilização da matriz de confusão para avaliar o desempenho dos algoritmos.

Os resultados obtidos demonstram uma alta precisão na classificação de diferentes tipos de toxicidade. A precisão do modelo treinado atingiu valores superiores a 95% para maioria das categorias de comentários tóxicos.

Figura 3.1: Resultados do trabalho mencionado

Label	Training Accuracy	Precision
Toxic	96.38%	0.91
Severe-toxic	99.21%	0.99
Obscene	98.32%	0.95
Threat	99.81%	1
Insults	97.53%	0.95
Identity-hate	99.38%	0.99

Fonte: Ozoh P A et al, 2019.

O artigo não fornece detalhes sobre a divisão dos dados em conjuntos de treinamento e teste, assim como não menciona se foi realizada validação cruzada para validar os resultados.

3.3 Challenges for Toxic Comment Classification: An In-Depth Error Analysis

O artigo discute a classificação de comentários tóxicos, comparando abordagens de aprendizado profundo e superficial em conjuntos de dados, e propõe um modelo de conjunto que supera modelos individuais (van Aken et al, 2018). A validação em um conjunto de dados separado reforça essas descobertas. Uma análise de erros destaca desafios contínuos, incluindo nuances contextuais e rótulos inconsistentes.

O artigo também discute a falta de conjuntos de dados padronizados para essa classificação, apresentando duas bases de dados examinadas: a Toxic Comment Classification Challenge e outra do Twitter com rótulos de discurso de ódio. O trabalho destaca que as bases apresentam desequilíbrios de classe e complexidades linguísticas, como gírias, erros ortográficos e dependências de longo alcance. Os desafios comuns incluem palavras fora do vocabulário e frases de múltiplas palavras, exigindo técnicas robustas de classificação.

Além disso, são detalhados métodos e um modelo de conjunto proposto, que combina regressão logística, redes neurais recorrentes (RNNs), redes neurais convolucionais (CNNs) e embeddings de palavras (sub-) para abordar os desafios identificados. A análise experimental demonstra que o modelo de conjunto supera os métodos individuais, especialmente em casos de alta variância nos dados.

Por fim, uma análise detalhada dos erros revela diversas classes de falsos negativos e falsos positivos, destacando os desafios enfrentados na classificação de comentários tóxicos. Entre esses desafios, o trabalho encontrou na base de dados casos de rótulos duvidosos, nos quais os comentários podem não se encaixar claramente nas definições de toxicidade estabelecidas, bem como situações em que a toxicidade está presente sem a utilização de palavrões. Além disso, foi identificado o uso frequente de perguntas retóricas, metáforas e comparações sutis que podem escapar à detecção automática de toxicidade. O artigo também encontrou palavras idiossincráticas ou raras que desafiam a compreensão do modelo e casos de sarcasmo ou ironia que são difíceis de identificar automaticamente.

Figura 3.2: Resultados do trabalho mencionado utilizando duas bases de dados

Model	Wikipedia				Twitter			
	P	R	F1	AUC	P	R	F1	AUC
CNN (FastText)	.73	.86	.776	.981	.73	.83	.775	.948
CNN (Glove)	.70	.85	.748	.979	.72	.82	.769	.945
LSTM (FastText)	.71	.85	.752	.978	.73	.83	.778	.955
LSTM (Glove)	**.74**	.84	.777	.980	.74	.82	.781	.953
Bidirectional LSTM (FastText)	.71	.86	.755	.979	.72	.84	.775	.954
Bidirectional LSTM (Glove)	**.74**	.84	.777	.981	.73	**.85**	.783	.953
Bidirectional GRU (FastText)	.72	.86	.765	.981	.72	.83	.773	.955
Bidirectional GRU (Glove)	.73	.85	.772	.981	.76	.81	.784	.955
Bidirectional GRU Attention (FastText)	**.74**	**.87**	**.783**	**.983**	.74	.83	**.791**	**.958**
Bidirectional GRU Attention (Glove)	.73	**.87**	.779	**.983**	**.77**	.82	**.790**	.952
Logistic Regression (char-ngrams)	**.74**	.84	.776	.975	.73	.81	.764	.937
Logistic Regression (word-ngrams)	.70	.83	.747	.962	.71	.80	.746	.933
Ensemble	**.74**	**.88**	**.791**	**.983**	.76	.83	**.793**	.953

Fonte: van Aken et al, 2018.

Os resultados mostram que o conjunto (Ensemble) supera os classificadores individuais na medida F1. O classificador individual mais forte em ambos os conjuntos de dados é uma rede GRU bidirecional com camada de atenção (van Aken et al, 2018).

3.4 Toxic Comment Classification Challenge: 1st place solution overview

A solução vencedora do desafio de classificação de comentários tóxicos adotou uma abordagem meticulosa e eficaz que resultou em uma ROC AUC de 0.98856. Inicialmente, a equipe concentrou-se em explorar uma variedade de embeddings pré-treinados, especialmente FastText e Glove, que foram treinados em grandes conjuntos de dados como Common Crawl, Wikipedia e Twitter. Eles reconheceram que a escolha desses embeddings era crucial, já que a maior parte da complexidade do modelo reside na camada de embedding.

Para melhorar ainda mais os resultados, eles implementaram uma técnica chamada tradução como aumento de dados durante o treino/teste (TTA). Isso envolveu traduzir os comentários de treino e teste para outras línguas, como francês, alemão e espanhol, e depois de volta para o inglês. Essa abordagem inovadora teve um impacto significativo no desempenho dos modelos.

Além disso, foi utilizada uma técnica chamada pseudo-rotulagem grosseira (PL) para lidar com as diferenças nas distribuições de dados de treinamento e teste. Isso envolveu rotular amostras de teste usando o modelo com melhor desempenho e, em seguida, adicioná-las ao conjunto de treinamento para treinar o modelo até a convergência. Essa estratégia ajudou a corrigir discrepâncias na distribuição dos dados.

A equipe também desenvolveu um robusto framework de validação cruzada e empilhamento

de modelos. Eles usaram uma combinação de médias ponderadas e empilhamento de modelos, principalmente com o algoritmo LightGBM. A seleção de parâmetros foi otimizada utilizando técnicas de otimização bayesiana. Os modelos foram avaliados não apenas com base no desempenho no leaderboard público, mas também considerando métricas de desempenho da Validação Cruzada, como log loss e AUC.

Durante o processo, a equipe fez várias descobertas importantes, como a constatação de que pequenas mudanças na arquitetura do modelo tiveram pouco impacto no desempenho geral. Eles também observaram que as arquiteturas de redes neurais recorrentes (RNN) superaram outras abordagens, como redes neurais convolucionais (CNN)..

No geral, a abordagem da equipe demonstrou como a experimentação sistemática e a aplicação de técnicas avançadas podem levar a resultados excepcionais em desafios de classificação de texto.

4

Metodologia

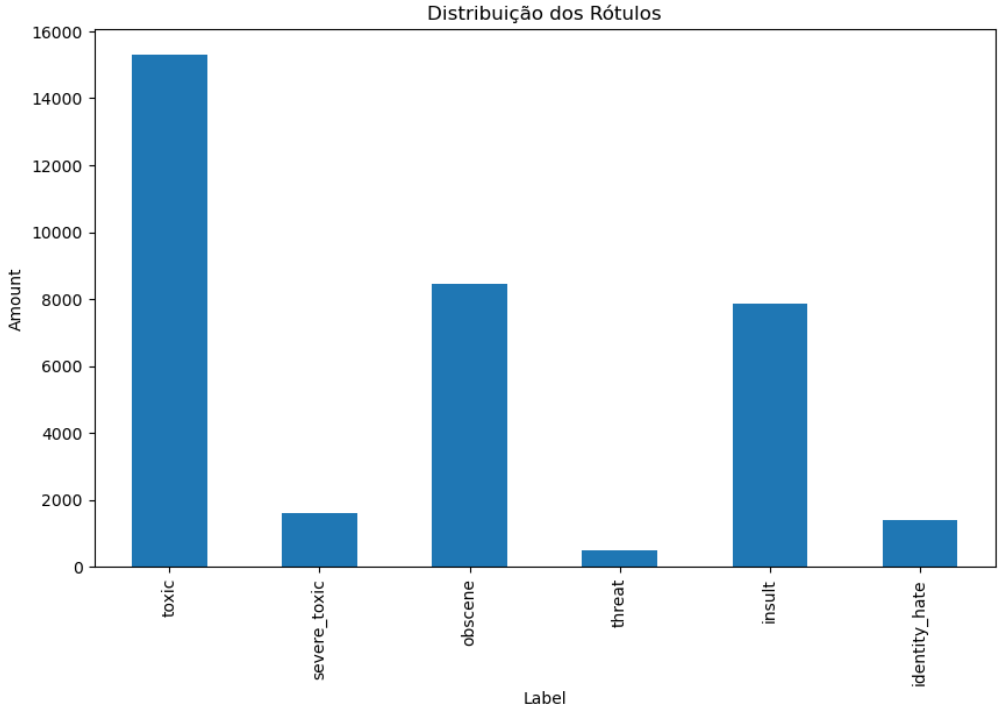
Neste capítulo, serão descritos os passos do processo de classificação, utilizando os dados já mencionados na Seção 1.4. Todo o trabalho foi realizado em um computador Ideapad3 com 12 GB de RAM e processador Ryzen 5 5500U, utilizando a linguagem Python na versão 3.10.9. Para executar e editar os códigos, foram empregados os softwares Anaconda, Visual Studio Code e uma extensão do Jupyter.

4.1 Análise Exploratória dos Dados

Os dados foram disponibilizados em formato zipado e, utilizando a biblioteca Pandas, foram carregados e transformados em um DataFrame. Isso permitiu a visualização das primeiras linhas e colunas, a contagem total de colunas, a listagem das colunas presentes e a identificação se possuíam dados nulos. Posteriormente, utilizando a biblioteca Matplotlib, foi realizada a plotagem do gráfico de barras para representar a distribuição dos rótulos, ou seja, a quantidade de comentários em cada classe.

Os rótulos encontrados são: toxic, severe_toxic, obscene, threat, insult e identity_hate, os quais foram traduzidos na seção 1.4, respectivamente. Com a plotagem da distribuição de rótulos, foi possível visualizar as discrepâncias presentes no conjunto de dados, conforme ilustrado na Figura 4.1. Essa discrepância será importante posteriormente para compreender o impacto nos resultados das classes com menos comentários.

Figura 4.1: Distribuição dos rótulos do conjunto de dados



Fonte: Autor, 2023.

Para compreender melhor o conteúdo e o tom dos comentários em cada categoria, foram criadas nuvens de palavras utilizando a biblioteca WordCloud para visualizar as palavras mais frequentes em cada classe do conjunto de dados.

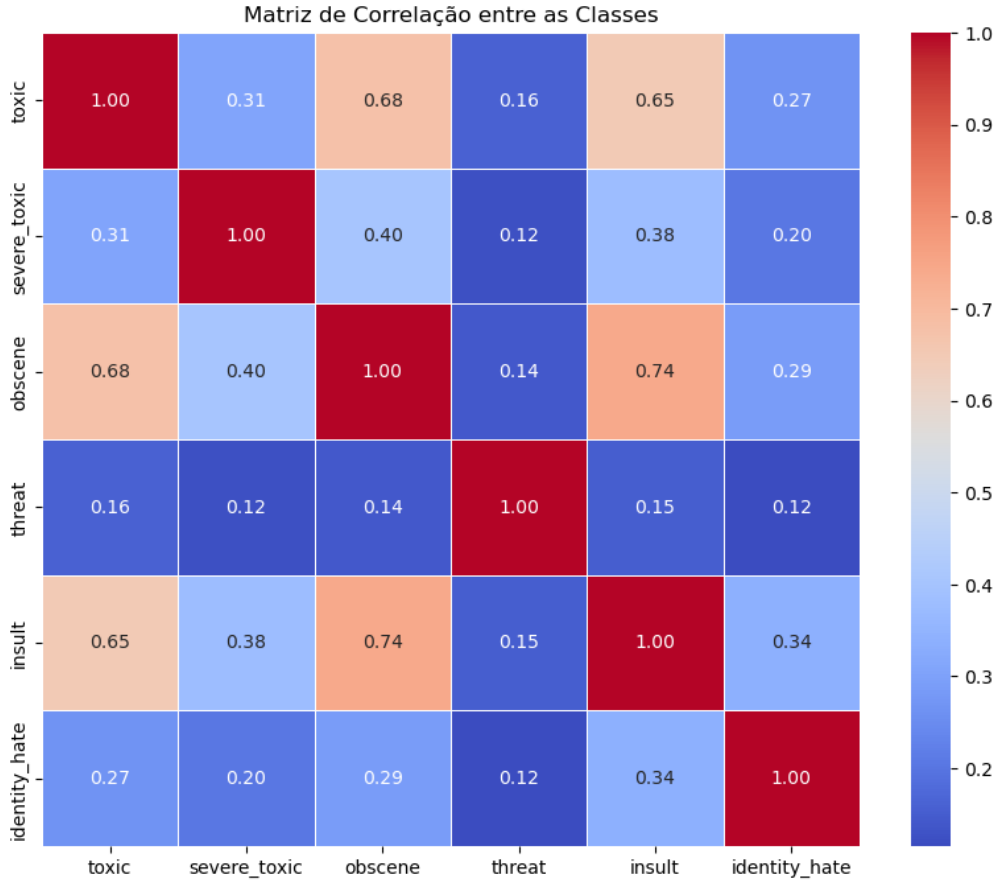
As nuvens de palavras foram plotadas utilizando a biblioteca Matplotlib, onde cada plotagem representa uma classe específica de comentários. Cada nuvem de palavras oferece uma representação visual das palavras mais comuns associadas a essa classe.

Figura 4.2: Nuvens das palavras mais frequentes em cada classe

Fonte: Autor, 2024.

Para criar um heatmap (mapa de calor) e visualizar a matriz de correlação entre as diferentes classes de comentários, foi utilizada a biblioteca seaborn. O objetivo era compreender as correlações entre essas classes. O heatmap gerado mostra as relações entre as classes, onde valores próximos de 1 indicam uma forte correlação positiva, valores próximos de -1 indicam uma forte correlação negativa, e valores próximos de 0 indicam pouca ou nenhuma correlação. O mapa de calor é uma ferramenta eficaz para identificar padrões de correlação entre as classes, auxiliando na análise exploratória dos dados. Essa análise é crucial para entender como as diferentes formas de toxicidade estão relacionadas entre si e como podem impactar o desempenho dos modelos de classificação.

Figura 4.3: Matriz de correlação entre as classes



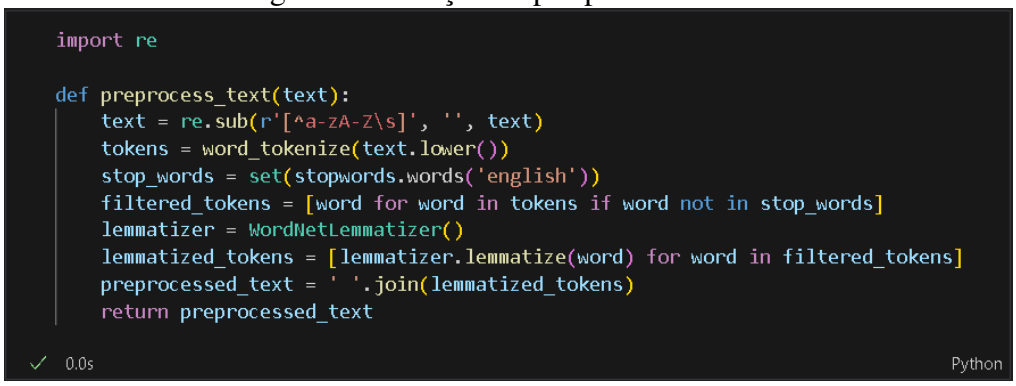
Fonte: Autor, 2024.

4.2 Pré-processamento

Para a aplicação de técnicas de PLN no pré-processamento do corpus, foi definida uma função que recebe um documento como entrada e realiza uma série de etapas. Primeiro, são removidos caracteres que não são letras ou espaços utilizando expressões regulares da biblioteca re. Em seguida, o documento é dividido em tokens usando a função *word_tokenize*. Posteriormente, as stopwords são removidas utilizando uma lista de stopwords. Então, cada token é lematizado usando o *WordNetLemmatizer*. Todos esses processos foram realizados utilizando a biblioteca NLTK. Por fim, os tokens lematizados são unidos de volta em um documento preprocessado que é retornado pela função.

A Figura 4.4 ilustra a implementação da função utilizada.

Figura 4.4: Função de pré-processamento



Fonte: Autor, 2023.

Na Figura 4.1, foi observado um desequilíbrio entre as classes no conjunto de dados, o que compromete a eficácia dos modelos de classificação. Equilibrar todas as classes para ter a mesma quantidade de dados não pareceu ser uma estratégia interessante, visto que a classe threat, por exemplo, possui apenas 478 comentários. Limitar todas as classes para que os modelos treinem com essa quantidade de dados pode resultar na perda de informações cruciais para a identificação de padrões. Portanto, para abordar essa questão, optou-se por treinar diferentes modelos para cada tipo de toxicidade, garantindo que cada um deles tivesse um balanceamento de documentos rotulados positiva e negativamente. Ou seja, a quantidade de documentos pertencentes ao tipo de toxicidade (classe alvo 1) deveria ser a mesma que a quantidade de documentos não pertencentes (classe alvo 0).

O processo de balanceamento de documentos em cada tipo de toxicidade foi dividido em três etapas:

1. Os dados foram divididos em duas partes:

   - Parte Positiva: Incluiu documentos com a classe alvo 1, ou seja, comentários rotulados com o tipo de toxicidade analisada.
   - Parte Negativa: Conteve documentos com a classe alvo 0, representando comentários não rotulados com o tipo de toxicidade analisada.

2. Foi realizada subamostragem da classe majoritária (negativa):

   - Os dados foram filtrados para selecionar apenas os primeiros documentos onde a classe alvo era igual a 0.
   - Em seguida, a seleção foi limitada para igualar a quantidade de amostras da classe alvo 1 nesse subconjunto de dados.

3. Os documentos com as classes alvo 0 e 1 foram concatenados, resultando em um conjunto de dados balanceado para cada tipo de toxicidade.

4.3 Treinamento dos Modelos

Para a obtenção dos melhores resultados, foram feitos testes e suposições. O processo de treinamento dos modelos foi feito em três etapas:

4.3.1 1ª Etapa de treinamento

Primeiramente foram definidas duas funções para treinar modelos de classificação: *train_svm_rf_and_lr* e *train_lstm*. Em cada uma delas foram feitas:

- Divisão dos Dados: Os dados foram divididos em conjuntos de treinamento e teste utilizando a função *train_test_split* da biblioteca scikit-learn, sendo 80% dos dados para treinamento e 20% para teste, com o parâmetro *random_state* ativado para garantir a reprodutibilidade dos resultados.

- Vetorização de Texto: Na função *train_svm_rf_and_lr*, um vetorizador é passado como parâmetro, podendo ser o TF-IDF ou CountVectorizer. Na função *train_lstm*, o vetorizador é definido dentro dela.

Para o treinamento na função *train_svm_rf_and_lr*:

- Os documentos de treino e teste são transformados em representações vetoriais usando o vetorizador fornecido. O método *fit_transform* ajusta o vetorizador aos dados de treino e os transforma em vetores, enquanto o método *transform* é utilizado apenas para os dados de teste, usando os parâmetros aprendidos durante o ajuste com os dados de treino.

- São utilizados os seguintes algoritmos:

  - **Support Vector Machine (SVM)**: com o modelo *SVC* do scikit-learn.
  - **Random Forest (RF)**: com o modelo *RandomForestClassifier* do scikit-learn.
  - **Logistic Regression (LR)**: com o modelo *LogisticRegression* do scikit-learn.

Para o treinamento da **Rede Neural LSTM** na função *train_lstm*:

- Na vetorização dos documentos, é utilizado o *Tokenizer* da biblioteca TensorFlow. O *Tokenizer* é configurado para considerar apenas as 2000 palavras mais frequentes, enquanto as outras são tratadas como out-of-vocabulary (OOV). Os documentos de treinamento são convertidos em sequências de números inteiros com base no vocabulário aprendido pelo *Tokenizer*. Essas sequências são padronizadas para terem o mesmo comprimento de 10 tokens, utilizando a função pad_sequences() da biblioteca Keras. O mesmo processo é aplicado aos dados de teste para garantir consistência nos dados de entrada para o modelo LSTM.

- É definida uma arquitetura de rede neural LSTM usando TensorFlow/Keras, incluindo uma camada de embedding, uma camada LSTM e uma camada densa de saída.
- O modelo é compilado com o otimizador *dam* e a função de perda *binary_crossentropy*.
- O treinamento é realizado durante 15 épocas.

Para avaliar os modelos, são feitas previsões usando os conjuntos de teste e calculadas as métricas: acurácia, precisão, cobertura, F1 Score e ROC AUC. Os resultados são armazenados em um dataframe, que é retornado para a chamada de cada função.

Os modelos foram treinados com os dados pré-processados, sendo treinados individualmente para cada classe de toxicidade. Isso resultou em um total de 7 modelos para cada classe, os quais foram gerados combinando os algoritmos SVM, RF e LR com os vetorizadores CountVectorizer e TF-IDF. A LSTM, por sua vez, foi treinada exclusivamente com o vetorizador Tokenizer.

4.3.2 2ª Etapa de treinamento

Com os resultados obtidos para cada modelo treinado, foi possível escolher o modelo com melhor desempenho em cada classe de toxicidade com base na métrica ROC-AUC, que foi utilizada na avaliação da competição. O objetivo desta etapa foi avaliar os melhores modelos utilizando o resultado obtido na submissão das predições. Para isso, foram criadas funções específicas para cada algoritmo de classificação.

Nessas funções, os dados não foram mais divididos em treinamento e teste. Em vez disso, todos os dados pré-processados de cada classe foram utilizados para o treinamento. Para as predições, foram utilizados os dados do arquivo test.csv disponibilizado na competição e por fim, as predições, em forma de probabilidades, do melhor modelo para cada classe de toxicidade foram concatenadas em um arquivo csv, e o arquivo foi submetido na competição.

O mesmo processo foi repetido, mas desta vez a escolha dos melhores modelos foi baseada na métrica F1 Score. O objetivo foi comparar os resultados das duas submissões e, a partir do melhor resultado, definir qual conjunto de modelos seria escolhido para melhorar seus parâmetros na próxima etapa de treinamento.

4.3.3 3ª Etapa de treinamento

Com o objetivo de aprimorar ainda mais os resultados dos modelos, o conjunto composto pelos 6 melhores modelos (um para cada classe) passou por uma etapa adicional de refinamento. Essa etapa envolveu a aplicação do GridSearch para encontrar os melhores hiperparâmetros para cada algoritmo.

Para tal, foram desenvolvidas funções individuais para treinar cada algoritmo, exceto a LSTM, que não se destacou como o melhor modelo em nenhuma das classes, seguindo a abordagem adotada na etapa anterior de treinamento, entretanto, desta vez, empregando o GridSearch.

Essas funções realizam uma busca em grade para otimizar os hiperparâmetros dos modelos de classificação SVM, RF e LR. Cada função é dedicada a um tipo específico de modelo, e para cada tipo, são definidos os parâmetros a serem avaliados na busca em grade. Os modelos são inicializados com os hiperparâmetros padrão e submetidos ao objeto *GridSearchCV*, que executa a busca em grade utilizando a métrica de avaliação f1-score, adotada com base nos resultados anteriores das submissões. O processo de validação cruzada com k-fold (k = 5) é realizado automaticamente durante a busca em grade. O valor de k foi escolhido levando em consideração a necessidade de um resultado rápido, pois valores maiores de k aumentariam significativamente o custo computacional do processo.

Os conjuntos de parâmetros selecionados para cada algoritmo foram os seguintes: para SVM, 'C': [1, 10, 12, 15], e 'kernel': ['linear', 'rbf']; para RF, 'n_estimators': [50, 100, 200], 'max_depth': [None, 10, 20]; e para LR, 'C': [1, 10, 12, 15], 'penalty': ['l1', 'l2']. É importante destacar que quanto maior o número de parâmetros e valores a serem analisados, maior será o custo computacional da busca em grade.

Todas as funções foram configuradas com o parâmetro n_jobs=-1 no objeto *GridSearchCV*, o que possibilita a utilização de todos os núcleos de processamento disponíveis no computador, resultando em uma otimização significativa do tempo de execução.

Por fim, foi realizada a submissão final com os melhores resultados encontrados.

5

Resultados e Discussões

As etapas de treinamento foram conduzidas com base nos resultados obtidos em cada uma delas, baseando-se na ROC AUC e F1 Score.

5.1 1ª etapa de treinamento

Os resultados obtidos na 1ª etapa de treinamento foram:

Figura 5.1: Resultados dos modelos para toxic

Classe: toxic

	Acurácia	Precisão	Cobertura	F1 Score	ROC AUC
SVM com CountVectorizer	0.862373	0.871778	0.856136	0.863886	0.862502
Random Forest com CountVectorizer	0.870873	0.925520	0.812240	0.865188	0.872086
Logistic Regression com CountVectorizer	0.896371	0.921955	0.870554	0.895517	0.896905
SVM com TFIDF	0.898333	0.925724	0.870554	0.897292	0.898907
Random Forest com TFIDF	0.865806	0.909253	0.818648	0.861575	0.866781
Logistic Regression com TFIDF	0.892775	0.923103	0.861583	0.891283	0.893421
LSTM com Tokenizer	0.835077	0.876336	0.787888	0.829762	0.909824

Fonte: Autor, 2024.

Figura 5.2: Resultados dos modelos para severe_toxic

Classe: severe_toxic

	Acurácia	Precisão	Cobertura	F1 Score	ROC AUC
SVM com CountVectorizer	0.655172	1.000000	0.331307	0.497717	0.665653
Random Forest com CountVectorizer	0.932602	0.901685	0.975684	0.937226	0.931208
Logistic Regression com CountVectorizer	0.942006	0.974026	0.911854	0.941915	0.942982
SVM com TFIDF	0.949843	0.968454	0.933131	0.950464	0.950384
Random Forest com TFIDF	0.931034	0.903683	0.969605	0.935484	0.929786
Logistic Regression com TFIDF	0.945141	0.971154	0.920973	0.945398	0.945923
LSTM com Tokenizer	0.929467	0.935583	0.927052	0.931298	0.972526

Fonte: Autor, 2024.

Figura 5.3: Resultados dos modelos para obscene

Classe: obscene

	Acurácia	Precisão	Cobertura	F1 Score	ROC AUC
SVM com CountVectorizer	0.860947	0.967140	0.752469	0.846405	0.862974
Random Forest com CountVectorizer	0.923964	0.958647	0.889018	0.922520	0.924618
Logistic Regression com CountVectorizer	0.937278	0.963168	0.911679	0.936716	0.937756
SVM com TFIDF	0.929290	0.961970	0.896572	0.928120	0.929901
Random Forest com TFIDF	0.918639	0.959924	0.876816	0.916490	0.919421
Logistic Regression com TFIDF	0.925148	0.956468	0.893666	0.924001	0.925736
LSTM com Tokenizer	0.878994	0.890476	0.869262	0.879741	0.939876

Fonte: Autor, 2024.

Figura 5.4: Resultados dos modelos para threat

Classe: threat

	Acurácia	Precisão	Cobertura	F1 Score	ROC AUC
SVM com CountVectorizer	0.572917	0.523256	1.000000	0.687023	0.598039
Random Forest com CountVectorizer	0.911458	0.919540	0.888889	0.903955	0.910131
Logistic Regression com CountVectorizer	0.906250	0.909091	0.888889	0.898876	0.905229
SVM com TFIDF	0.911458	0.929412	0.877778	0.902857	0.909477
Random Forest com TFIDF	0.921875	0.912088	0.922222	0.917127	0.921895
Logistic Regression com TFIDF	0.916667	0.930233	0.888889	0.909091	0.915033
LSTM com Tokenizer	0.817708	0.766990	0.877778	0.818653	0.914815

Fonte: Autor, 2024.

Figura 5.5: Resultados dos modelos para insult

Classe: insult

	Acurácia	Precisão	Cobertura	F1 Score	ROC AUC
SVM com CountVectorizer	0.875595	0.925091	0.814862	0.866485	0.875041
Random Forest com CountVectorizer	0.895589	0.941261	0.841768	0.888739	0.895098
Logistic Regression com CountVectorizer	0.911774	0.929097	0.889814	0.909031	0.911574
SVM com TFIDF	0.908600	0.935661	0.875721	0.904699	0.908301
Random Forest com TFIDF	0.893685	0.927476	0.852018	0.888147	0.893305
Logistic Regression com TFIDF	0.902888	0.935461	0.863549	0.898068	0.902529
LSTM com Tokenizer	0.837195	0.835038	0.836643	0.835840	0.913157

Fonte: Autor, 2024.

Figura 5.6: Resultados dos modelos para identity_hate

Classe: identity_hate

	Acurácia	Precisão	Cobertura	F1 Score	ROC AUC
SVM com CountVectorizer	0.891459	0.938462	0.844291	0.888889	0.892841
Random Forest com CountVectorizer	0.884342	0.873333	0.906574	0.889643	0.883690
Logistic Regression com CountVectorizer	0.916370	0.935252	0.899654	0.917108	0.916860
SVM com TFIDF	0.927046	0.955882	0.899654	0.926916	0.927849
Random Forest com TFIDF	0.886121	0.871287	0.913495	0.891892	0.885319
Logistic Regression com TFIDF	0.921708	0.952030	0.892734	0.921429	0.922557
LSTM com Tokenizer	0.834520	0.860294	0.809689	0.834225	0.918165

Fonte: Autor, 2024.

Os melhores modelos obtidos foram:

Tabela 5.1: Melhores modelos baseando-se na ROC AUC

Categoria	**Modelo**
toxic	LSTM com Tokenizer (0.909824)
severe_toxic	LSTM com Tokenizer (0.972526)
obscene	LSTM com Tokenizer (0.939876)
threat	RF com TFIDF (0.921895)
insult	LSTM com Tokenizer (0.913157)
identity_hate	SVM com TFIDF (0.927849)

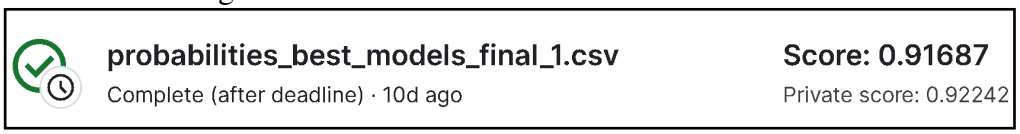
Tabela 5.2: Melhores modelos baseando-se na F1 Score

Categoria	**Modelo**
toxic	SVM com TFIDF (0.897292)
severe_toxic	SVM com TFIDF (0.950464)
obscene	LR com CountVectorizer (0.936716)
threat	RF com TFIDF (0.912088)
insult	LR com CountVectorizer (0.909031)
identity_hate	SVM com TFIDF (0.926916)

5.2 2ª etapa de treinamento

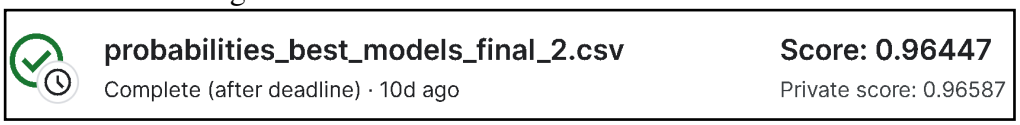
Na segunda etapa de treinamento, os resultados obtidos com as submissões dos melhores modelos foram:

Figura 5.7: Submissão baseando-se na ROC AUC



Fonte: Autor, 2024.

Figura 5.8: Submissão baseando-se na F1 Score



Fonte: Autor, 2024.

Conforme evidenciado nas figuras 5.7 e 5.8, os melhores resultados foram obtidos utilizando os modelos que apresentaram os melhores valores de F1 Score. Ao analisar as tabelas 5.1 e 5.2, observa-se que apenas os modelos treinados com RF e TF-IDF, na categoria 'threat', e com SVM e TF-IDF, na categoria 'identity_hate', permaneceram em ambas as tabelas.

Os demais modelos listados na tabela 5.1 foram treinados com LSTM e Tokenizer, e não alcançaram os melhores resultados em termos de F1 Score para nenhuma das classes. Além disso, ao analisar outras métricas desses modelos, como acurácia, precisão e cobertura, observou-se que eles mantiveram valores significativamente mais baixos em comparação com os valores de ROC AUC.

Por outro lado, os modelos que permaneceram em ambas as tabelas demonstraram bons resultados em todas as métricas avaliadas, indicando uma bom desempenho não somente na ROC AUC.

5.3 3ª etapa de treinamento

No processo de otimização dos modelos da tabela 5.2 utilizando *GridSearchCV*, os hiperparâmetros que obtiveram melhores resultados baseando-se na F1 Score são mostrados na figura 5.9.

Figura 5.9: Melhores hiperparâmetros para cada modelo

```
Melhores parâmetros para a classe toxic (SVM com TFIDF)
C: 1
Kernel: linear


Melhores parâmetros para a classe severe_toxic (SVM com TFIDF)
C: 1
Kernel: linear


Melhores parâmetros para a classe obscene (Logistic Regression com CountVectorizer)
C: 1
penalty: l2


Melhores parâmetros para a classe threat (Random Forest com TFIDF)
n_estimators: 200
max_depth: 10


Melhores parâmetros para a classe insult (Logistic Regression com CountVectorizer)
C: 1
penalty: l2


Melhores parâmetros para a classe identity_hate (SVM com TFIDF)
C: 1
Kernel: linear
```

Fonte: Autor, 2024.

A seleção dos melhores hiperparâmetros para cada classe de toxicidade revela informações interessantes sobre a natureza dos dados:

- SVM para toxic, severe_toxic e identity_hate:
  - C = 1: indica que o algoritmo busca um equilíbrio entre a minimização do erro de treinamento e a maximização da margem entre as classes. Isso sugere que os dados dessas classes são relativamente separáveis, permitindo que o SVM linear encontre uma boa solução sem a necessidade de regularização forte.
  - Kernel Linear: indica que os dados das classes após serem transformados pelo TF-IDF, possuem uma relação linear entre as features numéricas geradas. Isso significa que a classificação se baseia em uma combinação linear dessas features, sem a necessidade de transformações complexas no espaço vetorial.

- Logistic Regression para obscene e insult:
  - C = 1: Similarmente ao SVM, indica um equilíbrio entre a minimização do erro e a margem entre as classes. Isso sugere que os dados das classes também são relativamente separáveis e que a regularização forte não é necessária.
  - Penalty L2: introduz uma regularização que evita o overfitting, penalizando coeficientes de grande magnitude. Isso sugere que os dados das classes podem ser sensíveis ao overfitting, e que a regularização L2 ajuda a controlar a complexidade do modelo e a melhorar a generalização.
- Random Forest para threat:
  - n_estimators = 200: o número de árvores indica que o modelo utiliza um conjunto diversificado de árvores de decisão para realizar a classificação. Isso sugere que os dados da classe são complexos e que a combinação de múltiplas árvores de decisão com diferentes perspectivas melhora a precisão da classificação.
  - max_depth = 10: a profundidade máxima das árvores de decisão limita a complexidade das árvores individuais, evitando o overfitting. Isso sugere que os dados da classe podem ser bem representados por árvores de decisão com profundidade moderada, sem a necessidade de árvores excessivamente ramificadas.

Os resultados das métricas obtidas com os melhores modelos e hiperparâmetros encontrados neste trabalho são apresentados na Figura 5.10.

Figura 5.10: Melhores resultados encontrados

	Classe	Algoritmo	Vetorizador	Acurácia	Precisão	Recall	F1-score	ROC-AUC
0	toxic	SVM	tfidf	0.962733	0.970101	0.956424	0.963214	0.962863
1	severe_toxic	SVM	tfidf	0.995298	0.990964	1.000000	0.995461	0.995146
2	obscene	Logistic Regression	countvectorizer	0.985799	0.991192	0.980825	0.985981	0.985892
3	threat	Random Forest	tfidf	0.958333	0.955556	0.955556	0.955556	0.958170
4	insult	Logistic Regression	countvectorizer	0.984449	0.982143	0.986547	0.984340	0.984469
5	identity_hate	SVM	tfidf	0.991103	0.989655	0.993080	0.991364	0.991045

Fonte: Autor, 2024.

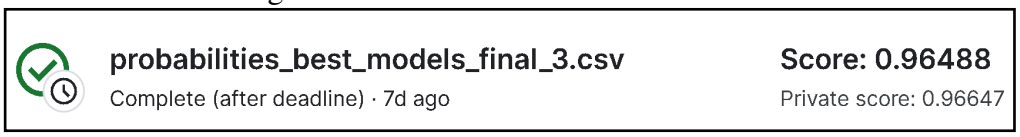
Fica evidente que, em todas as classes, os modelos apresentaram melhorias muito satisfatórias após a otimização dos hiperparâmetros, mesmo com a inclusão da validação cruzada, a qual geralmente pode diminuir os resultados, porém garante uma melhor generalização. A vantagem clara em comparação com os modelos não otimizados é demonstrada na Tabela 5.3.

Tabela 5.3: F1 Score dos modelos não otimizados x modelos otimizados

Categoria	**Modelo**	**Não otimizado**	**Otimizado**
toxic	SVM com TFIDF	0.897292	0.963214
severe_toxic	SVM com TFIDF	0.950464	0.995461
obscene	LR com CountVectorizer	0.936716	0.985981
threat	RF com TFIDF	0.912088	0.955556
insult	LR com CountVectorizer	0.909031	0.984340
identity_hate	SVM com TFIDF	0.926916	0.991364

Por fim, o resultado da submissão final, contendo os melhores modelos e hiperparâmetros, é apresentado na Figura 5.11.

Figura 5.11: Resultado da submissão final



Fonte: Autor, 2024.

Dada a Figura 5.11, pode-se perceber uma melhora em relação à submissão apresentada na Figura 5.8, porém não tão grande quanto era esperado devido à melhora significativa no desempenho da F1 Score.

6

Conclusão

Neste trabalho, foram investigados diversos aspectos relacionados ao desafio dos comentários tóxicos em plataformas digitais e à aplicação de técnicas de Machine Learning para abordá-lo. Inicialmente, foram examinados trabalhos realizados anteriores para contextualizar a abordagem metodológica adotada. Foram utilizadas bibliotecas para a leitura dos dados, seguida de análises exploratórias, incluindo visualizações da distribuição de rótulos e matriz de correlação entre as classes de comentários. Para uma análise mais aprofundada, foram geradas nuvens de palavras para cada classe. Na etapa de pré-processamento, os documentos foram limpos e preparados usando técnicas de PLN. Os modelos de classificação foram treinados com os algoritmos SVM, RF, LR e uma Rede Neural LSTM, sendo refinados com otimização de hiperparâmetros aqueles com maiores potenciais. Por fim, foram submetidos na competição os melhores resultados obtidos para avaliação final.

Para este estudo, optou-se por utilizar os algoritmos comumente encontrados na literatura para a classificação de comentários tóxicos, conforme o trabalho mencionado na seção 3.1 relata. No entanto, foi escolhida uma abordagem que priorizou a simplicidade e ajustes menos refinados para alcançar melhor desempenho. Isso explica a utilização de apenas um tipo de arquitetura de rede neural, além da hipótese de que o significativo desequilíbrio entre as classes teria um impacto maior no desempenho desses classificadores em comparação com algoritmos mais simples.

Os resultados destacam a importância de considerar métricas além da ROC AUC para maximizar a pontuação na competição. Notavelmente, a F1 Score foi mais eficaz na seleção dos modelos, resultando em pontuações mais altas do que aquelas baseadas apenas na ROC AUC. Embora quatro dos seis modelos com as melhores pontuações ROC AUC, obtidos com a 1ª etapa de treinamento, tenham sido treinados com LSTM, suas F1 Score não foram tão satisfatórias. Essa disparidade nos resultados pode ter levado a uma generalização menos eficaz na competição em comparação com os melhores modelos em termos de F1 Score. É importante ressaltar que três das seis classes tinham menos de 1600 exemplos positivos, sendo que a menor

delas tinha apenas 478 exemplos. Isso afetou o treinamento da rede neural, e a equipe vencedora da competição destacou a importância de aumentar o número de dados nessas classes para melhorar as probabilidades. Portanto, em análises futuras, essa estratégia pode ser adotada para comparar os algoritmos utilizados neste estudo. Além disso, um possível aprimoramento para a LSTM pode envolver um ajuste mais refinado no processo de vetorização dos dados.

A alta correlação entre algumas classes, com os maiores valores de correlação alcançando 0.74, 0.68 e 0.65, compromete a eficácia da abordagem adotada neste estudo, que consiste no treinamento independente dos modelos para cada classe. Essa abordagem não é capaz de capturar as correlações, o que pode resultar em uma queda no desempenho dos modelos. Essa forte correlação pode estar relacionada aos casos de rótulos duvidosos presentes na base de dados, como observado no estudo mencionado da seção 3.3. É importante ressaltar que o desequilíbrio significativo de dados entre as classes torna o treinamento multirrótulo uma opção insatisfatória, especialmente prejudicando o desempenho nas classes com poucos dados. Equilibrar os dados treinando com a classe que possui menos exemplos também não é uma solução adequada, pois o modelo teria uma quantidade insuficiente de dados para aprendizado. No entanto, em análises futuras, a utilização de uma abordagem que aumenta a quantidade de dados das classes pode permitir a adoção de um treinamento multirrótulo.

Apesar dos desafios mencionados anteriormente, a abordagem de treinamento independente resultou em modelos robustos. Estes foram validados com k-fold no gridsearch e alcançaram resultados bastante satisfatórios. Os resultados foram equivalentes a estudos anteriores que empregaram abordagens diferentes. Por exemplo, no estudo referenciado na seção 3.2, a média de acurácia foi apenas 0.6% maior do que a deste trabalho, mas a média de precisão foi 1.3% menor.

Para abordar a lacuna identificada na literatura, conforme mencionado no trabalho da seção 3.1, e aprimorar as representações vetoriais dos textos, a utilização de transformers e embeddings pré-treinados apresenta-se como uma estratégia promissora para análises futuras.

A fim de aprimorar os resultados obtidos na competição, é crucial realizar testes com as modificações propostas anteriormente. Apesar do desempenho alcançado de 0.96488, é importante notar a diferença significativa em relação à pontuação de 0.9885 da equipe vencedora.

Conclui-se que a abordagem metodológica deste trabalho foi bem-sucedida na realização de seus objetivos, proporcionando ideias valiosas e resultados replicáveis em ambientes com recursos computacionais limitados. Além disso, demonstrou-se como uma metodologia eficaz na identificação e classificação de comentários tóxicos em plataformas digitais. Esses resultados não apenas contribuem para o desenvolvimento de sistemas de moderação de conteúdo online mais robustos, mas também para promover ambientes digitais mais saudáveis e fomentar a preservação da democracia por meio de discussões online construtivas.

A pesquisa sobre a classificação de comentários tóxicos é um campo dinâmico e em constante evolução, com um enorme potencial para tornar a sociedade digital mais justa, democrática e inclusiva. À medida que continuamos a aprofundar nosso entendimento e aprimorar nossas

técnicas nessa área, estamos pavimentando o caminho para uma experiência online mais segura e enriquecedora para todos os usuários.

Referências bibliográficas

[1] Darko ANDROCEC. Machine learning methods for toxic comment classification: a systematic review. *Acta Universitatis Sapientiae, Informatica*, 12(2):205–216, 2020.

[2] Caio Oliveira AZEVEDO and et al. Avanço da ciência de dados nos estudos de economia: uma análise bibliométrica do uso de machine learning. Disponível em: https://www.anpec.org.br/nordeste/2023/submissao/arquivos_identificados/060-1927970dbd3dc9a41a96300c8beb88e6.pdf. Acesso em: 31 de out de 2023.

[3] T BERTAGLIA. Normalização textual de conteúdo gerado por usuário. [s.l: s.n.]. URL https://www.teses.usp.br/teses/disponiveis/55/55134/tde-10112017-170919/publico/ThalesFelipeCostaBertaglia_revisada.pdf. Acesso em: 10 nov. 2023.

[4] Leo BREIMAN. Random forests. *Machine Learning*, 45:5–32, 2001.

[5] Weslley Pina CAMPOS, Renata Mirella FARINA, and Fabiana FLORIAN. Inteligência artificial: Machine learning na gestão empresarial. *RECIMA21-Revista Científica Multidisciplinar*, 3(6):e361617–e361617, 2022. ISSN 2675-6218.

[6] Jeffrey Sorensen CJADAMS and et al. Toxic comment classification challenge. https://kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge, 2017.

[7] R DI SIPIO and et al. A quick guide to auc-roc in machine learning models. *Towards Data Science*, 2021. Acesso em: 09 de abril de 2024.

[8] Pedro DOMINGOS. *The master algorithm: How the quest for the ultimate learning machine will remake our world*. Basic Books, United States, 2015.

[9] FAROIT. Keras preprocessing text, 2024. URL https://faroit.com/keras-docs/1.2.2/preprocessing/text/.

[10] GeeksforGeeks. Using countvectorizer to extracting features from text, 2024. URL https://www.geeksforgeeks.org/using-countvectorizer-to-extracting-features-from-text/.

[11] Felix A GERS, Jürgen SCHMIDHUBER, and Fred CUMMINS. Learning to forget: Continual prediction with lstm. *Neural Computation*, 12(10):2451–2471, 1999.

[12] Felix A. GERS, Jürgen SCHMIDHUBER, and Fred CUMMINS. Learning to forget: Continual prediction with lstm, 1999.

[13] Tilottama GOSWAMI and G. R SINHA. *Statistical Modeling in Machine Learning: Concepts and Applications*. Academic Press, United States, 2022.

[14] Steve R GUNN and et al. Support vector machines for classification and regression. Technical Report 14, ISIS, 1998.

[15] A HIPPISLEY. Lexical analysis. In *Handbook of Natural Language Processing*. Chapman & Hall/Crc, Boca Raton, Fl, 2010.

[16] Sepp HOCHREITER and Jürgen SCHMIDHUBER. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[17] D. W. Jr. HOSMER, S. LEMESHOW, and R. X. STURDIVANT. *Applied Logistic Regression*. John Wiley & Sons, New Jersey, 3 edition, 2013.

[18] IBM. What is computer vision?, 2023. URL https://www.ibm.com/topics/computer-vision.

[19] ISTOÉ. Sob pressão, facebook endurece sua política de moderação de conteúdos, 2020. URL https://istoedinheiro.com.br/sob-pressao-facebook-endurece-sua-politica-de-moderacao-de-conteudos/.

[20] Anne KAO and Steve R. POTEET, editors. *Natural language processing and text mining*. Springer Science & Business Media, 2007.

[21] KERAS. About keras, 2024. URL https://keras.io/about/.

[22] E. D. LIDDY. Natural language processing. In E. D. LIDDY, editor, *Encyclopedia of Library and Information Science, 2nd Ed.* Marcel Decker, Inc., New York, 2001.

[23] Ingrid LUISA. A internet está tóxica! e isso pode mexer com a nossa saúde mental: Críticas podem virar discurso de ódio nas redes sociais, gerando problemas emocionais e sociais. como se blindar e não cair nesse tipo de comportamento? *Veja*, page 1, 2021. URL https://saude.abril.com.br/mente-saudavel/a-internet-esta-toxica-e-isso-pode-mexer-com-a-nossa-saude-mental. [S. l.].

[24] Nikita MUNOT and Sharvari S. GOVILKAR. Comparative study of text summarization methods. *International Journal of Computer Applications*, 102(12), 2014.

[25] Sarang NARKHEDE. Understanding auc-roc curve. *Towards Data Science*, 26(1): 220–227, 2018.

[26] P. A. OZOH, Adepeju Abeke ADIGUN, and M. O. OLAYIWOLA. Identification and classification of toxic comments on social media using machine learning techniques. *International Journal of Research and Innovation in Applied Science (IJRIAS)*, 4(11): 142–147, 2019.

[27] D. PALMER. Text preprocessing. In *Handbook of Natural Language Processing*. Chapman & Hall/Crc, Boca Raton, Fl, 2010.

[28] K et al. POOJITHA. Classification of social media toxic comments using machine learning models. *ArXiv: CS - Computers and Society*, 2023. URL https://arxiv.org/ftp/arxiv/papers/2304/2304.06934.pdf. DOI arxiv-2304.06934.

[29] J. SAVOY and E. GAUSSIER. Information retrieval. In *Handbook of Natural Language Processing*. Chapman & Hall/Crc, Boca Raton, Fl, 2010.

[30] R. et al. SPROAT. Normalization of non-standard words. *Computer Speech & Language*, 15(3):287–333, July 2001.

[31] TENSORFLOW. pad_sequences, . URL https://www.tensorflow.org/api_docs/python/tf/keras/utils/pad_sequences.

[32] TENSORFLOW. Tokenizer, . URL https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/text/Tokenizer.

[33] Y. TITENOK. Natural language processing vs text mining, 2022. URL https://sloboda-studio.com/blog/natural-language-processing-vs-text-mining.

[34] Betty et al. VAN AKEN. Challenges for toxic comment classification: An in-depth error analysis. arXiv preprint arXiv:1809.07572, 2018.