Lígia Iunes Venturott

Detecção de Discurso de Ódio em Redes Sociais Utilizando Deep Learning

Brasil

Vitória, 2021

Lígia Iunes Venturott

Detecção de Discurso de Ódio em Redes Sociais Utilizando Deep Learning

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Engenharia Elétrica.

Universidade Federal do Espírito Santo

Programa de Pós-Graduação em Engenharia Elétrica

Orientador: Prof. Dr. Patrick Marques Ciarelli

Brasil

Vitória, 2021

Lígia Iunes Venturott

Detecção de Discurso de Ódio em Redes Sociais Utilizando Deep Learning

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Engenharia Elétrica.

Trabalho aprovado. Brasil, 25 de outubro de 2021:

Prof. Dr. Patrick Marques Ciarelli
Orientador

Prof. Dr. Jorge Leonid Aching Samatelo
Universidade Federal do Espírito Santo

Prof. Dr. Elias Silva de Oliveira
Universidade Federal do Espírito Santo

Brasil
Vitória, 2021

Agradecimentos

Agradeço primeiramente à minha família pelo apoio incondicional, por sempre apoiarem minhas escolhas e me ajudar por todo o percurso. Agradeço aos meus pais e meus irmãos pelo carinho e ajuda nos momentos difíceis e por serem meus exemplos de dedicação e determinação.

Agradeço aos meus professores, presentes e passados, que contribuíram para o meu aprendizado e para o meu desenvolvimento. Agradeço em especial ao meu orientador, Patrick Ciarelli, por aceitar me guiar para a elaboração desse trabalho e por me ajudar durante todo esse caminho.

Gostaria de agradecer também ao Programa de Pós-Graduação em Engenharia Elétrica (PPGEE) da UFES, por tornar possível a realização desse projeto, e à FAPES, pelo apoio financeiro.

Um agradecimento especial à todos os meus amigos, por me distraírem nos momentos difíceis e por deixarem os meus dias mais leves. Agradeço também aos meus colegas do laboratório CISNE, por me ajudarem na adaptação ao mestrado e por me ajudarem nas dificuldades que sempre surgiam no laboratório e na UFES.

Por fim, agradeço à todos que de alguma maneira me ajudaram nessa jornada. Se não fosse o apoio de todos nada disso teria sido possível.

"Do the scary thing first, and get scared later."
Lemony Snicket

Resumo

Na última década houve uma rápida expansão do uso das redes sociais online. Essas plataformas tem como objetivo permitir a comunicação e interação de pessoas de diferentes regiões, etnias, culturas e histórias. Porém, esse contato, somado à sensação de anonimato e impunidade do meio digital, gera nas redes sociais um ambiente propício para a disseminação do discurso de ódio, como xenofobia, racismo, sexismo, homofobia, entre outros.

A maior parte das plataformas, como Twitter e Facebook, proíbem esse tipo de comportamento. Porém, a grande quantidade de publicações diárias torna a análise manual uma tarefa praticamente impossível. Esse contexto leva à necessidade de criação de ferramentas que possam detectar automaticamente discursos de ódio em redes sociais. Contudo, a maioria dos trabalhos atuais foca na criação de ferramentas de classificação para a língua inglesa.

Esse trabalho propõe o desenvolvimento de um método para detecção de discurso de ódio em redes sociais para a língua portuguesa, usando como ferramenta principal redes neurais profundas. Para isso, foi feita uma análise tentando identificar os problemas envolvidos nessa tarefa, e foi detectada uma escassez em bases de dados rotuladas em português, o que dificulta a utilização de redes neurais profundas.

Para amenizar esse problema, foi proposta a utilização de técnicas de aumento de dados. Três técnicas foram selecionadas da literatura e foram aplicadas em diversos cenários, tentando identificar em que cenários essas técnicas trazem mais benefícios. Concluiu-se que as técnicas de aumento de dados selecionadas podem trazer resultados positivos para bases de dados muito limitadas, com tamanho entre 1.000 e 2.000 documentos.

Palavras-chave: Discurso de ódio, Redes Sociais, Redes Neurais Convolucionais, Redes Neurais Recursivas, Aumento de Dados.

Abstract

In the last decade, online social networks went through a quick expansion. The main goal of these platforms is to allow the communication between people from different backgrounds, religions, cultures and countries. However, this new form of contact, allied to the feeling of anonymity and impunity of the digital enviroment, turned social networks into a favorable enviroment for disseminating hate speech, such as xenophobia, racism, sexism, homophobia, and others.

Most platforms, such as Twitter and Facebook, explicitly forbid this kind of behaviour. However, the large volume of daily posts make manually detecting hate speech an almost impossible task. In this context, there is a need for automatic detection tools for hate speech in social networks, but most works focus on detecting of hateful content in English.

This work develops a method for detecting hate speech in social networks focused on Portuguese, using deep neural networks as the main resource. To that end, first we identified the main issues regarding hate speech detection in Portuguese, and it was observed that there is a lack of labeled datasets for hate speech and offensive language in Portuguese. The few existing datasets consist of few documents, which makes the application of deep learning techniques difficult.

In order to mitigate this problem, we propose using data augmentation techniques. Three techniques were selected from the literature and were applied in different scenarios, where we tried to identify in which cases these techniques would be the most beneficial. It was concluded that the data augmentation techniques selected can be helpful when applied to very reduced datasets, varying from 1,000 to 2,000 documents.

Keywords: Hate Speech, Social Networks, Convolutional Neural Networks, Recursive Neural Networks, Data Augmentation.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

BPTT	*Backpropagation Through Time*
CNN	*Convolutional Neural Network*
GloVe	*Global Vectors for Word Representation*
LSTM	*Long Short-Term Memory*
RNN	*Recurrent Neural Network*
SVM	*Support Vector Machine*
TF-IDF	*Term Frequency-Inverse Document Frequency*

Sumário

1 Introdução

Na última década foi observado um aumento exponencial do uso das redes sociais online. Sozinha, a rede social Twitter conta com aproximadamente 330 milhões de usuários ativos e 500 milhões de tweets publicados por dia (OMNICORE, 2021). Este espaço virtual permite a criação de comunidades formadas por pessoas que, sem o auxílio da internet, dificilmente se encontrariam (PERRY; OLSSON, 2009). Esse tipo de ambiente permite a comunicação entre pessoas de diferentes culturas, religiões e interesses. Porém, esses fatores, aliados à sensação de anonimidade, geram um ambiente ideal à propagação de discurso de ódio, como xenofobia, racismo, sexismo, homofobia, entre outros.

Na legislação brasileira, o termo discurso de ódio define a manifestação de pensamento que incita a violência contra vulneráveis. A condição de vulnerabilidade decorre de alguma característica física, étnica, religiosa, econômica, política e outras características que foram ultra-generalizadas na história e no cotidiano como um juízo tido como verdadeiro, criando esteriótipos e emergindo como preconceito, discriminação ou racismo (CARCARá, 2014).

Um dos problemas encontrados ao tentar suprimir o discurso de ódio é que as abordagens utilizadas podem esbarrar no direito à liberdade de expressão. O Artigo 19 da Declaração Universal dos Direitos Humanos (ONU, 1948) declara que:

> "Todo ser humano tem direito à liberdade de opinião e expressão; esse direito inclui a liberdade de, sem interferência, ter opiniões e de procurar, receber e transmitir informações e ideias por quaisquer meios e independentemente de fronteiras."

O discurso de ódio nos leva a uma situação nos limites do direito à liberdade de expressão, mas que também fere outros direitos fundamentais (CHETTY; ALATHUR, 2018). A violência causada pelo discurso de ódio em si já fere os direitos individuais, mas além disso o discurso de ódio pode influenciar ações fora das redes sociais. De acordo com a ONG *Words Heal the World*, em 2019 12.334 crimes de ódio foram cometidos no Brasil (BUARQUE; CRETTON, 2021).

Diante desses problemas, há uma pressão para que a disseminação do discurso de ódio seja restringida. Quando a propagação desse tipo de discurso se dá por plataformas online, a responsabilidade desse controle cai sobre as próprias plataformas. Empresas como o Facebook, Twitter e YouTube podem ser responsabilizados se falharem em restringir esse tipo de conteúdo (BBC, 2016). A maior parte dessas empresas proíbe explicitamente a disseminação de discurso de ódio (TWITTER, 2021; FACEBOOK, 2021; GOOGLE, 2021). Porém, como mencionado anteriormente, o volume de publicações diárias nessas plataformas é muito grande, o que torna a análise manual insuficiente.

Nesse contexto, surge a necessidade de criar ferramentas automáticas de detecção de discurso de ódio. Nos últimos anos surgiram diversos trabalhos com esse foco, muitos deles baseados em técnicas de aprendizado supervisionado.

Algoritmos de aprendizado supervisionado utilizam conjuntos de exemplos rotulados para permitir que a máquina reconheça padrões nos dados de forma automática. Entretanto, a construção de bases de dados rotuladas com qualidade é uma atividade trabalhosa, o que as tornam recursos escassos.

Bases de dados para detecção de discurso de ódio existem em diferentes formatos e com diferentes classificações. Algumas bases identificam a existência do discurso de ódio, enquanto outras classificam também o grupo alvo do discurso (sexismo, racismo, homofobia, entre outros).

A própria definição de discurso de ódio torna difícil a construção dessas bases, pois os especialistas têm que tomar cuidado para identificar o discurso de ódio sem afetar a liberdade de expressão dos autores.

A maior parte das bases existentes é formada por textos na língua inglesa. Algumas bases existem na língua portuguesa, porém essas tem um tamanho reduzido, o que limita as possibilidades de algoritmos de classificação, dado que técnicas como redes neurais profundas estão sujeitas a *overfitting* se a base de dados utilizada para treino for muito pequena.

1.1 Método Proposto

Este trabalho explora o uso de redes neurais profundas convolucionais e recursivas na tarefa de detecção de discurso de ódio em redes sociais para textos em português. São utilizadas cinco arquiteturas como classificadores: uma arquitetura convolucional simples, uma arquitetura convolucional com ramos paralelos, uma rede recursiva e duas arquiteturas que unem camadas convolucionais e camadas recursivas. São analisadas quais arquiteturas são mais apropriadas para o problema em questão e quais fazem melhor uso das bases de dados disponíveis. Observa-se que o modelo de rede convolucional com ramos paralelos produz bons resultados na maioria dos casos, quando comparado com os outros modelos testados.

Além disso, é proposto o uso de técnicas de aumento de dados para textos buscando amenizar o problema causado pelo tamanho reduzido das bases de dados disponíveis. São utilizadas três diferentes técnicas: Remoção Aleatória, Substituição por Sinônimos e Troca Aleatória.

Todas as técnicas de aumento de dados utilizadas são técnicas simples que requerem recursos fáceis de serem obtidos ou ainda nenhum recurso para serem aplicadas. Os

resultados mostram que, apesar da pouca complexidade, essas técnicas podem produzir um ganho em diversos casos. O código desenvolvido durante este trabalho está disponível na plataforma GitHub [1].

1.2 Objetivos

O objetivo geral deste trabalho consiste em desenvolver um método de detecção automática de discurso de ódio focado em textos de redes sociais na língua portuguesa. Para isso, pretende-se utilizar técnicas baseadas em aprendizado profundo e técnicas de processamento de linguagem natural, buscando solucionar problemas como a escassez de dados rotulados na língua portuguesa, por meio do uso de métodos de regularização, como técnicas de aumento de dados.

O objetivo geral pode ser detalhado nos seguintes objetivos específicos:

- Desenvolver algoritmo de classificação automática de discurso de ódio em textos em português retirados de redes sociais;
- Realizar experimentos e comparar o desempenho de diferentes arquiteturas de aprendizado profundo;
- Utilizar técnicas de *data augmentation* para remediar a escassez de dados rotulados na língua portuguesa;

1.3 Estrutura da dissertação

Este trabalho está organizado da seguinte forma. O Capítulo 2 apresenta um embasamento teórico sobre o fenômeno do discurso de ódio e mostra os desafios da detecção automática desse tipo de conteúdo. No Capítulo 3 são apresentadas as tecnologias já existentes utilizadas como base para este trabalho. O Capítulo 4 explica a metodologia desenvolvida para detecção de discurso de ódio, e é feita a descrição das bases de dados utilizadas. No Capítulo 5 são expostas as métricas de avaliação e procedimentos experimentais executados, são exibidos os resultados obtidos e é feita a discussão desses resultados. Por fim, o Capítulo 6 consiste em uma conclusão e a possibilidade de trabalhos futuros.

[1] https://github.com/ligiaiv/deteccao_odio_DL_dissertacao

2 Detecção de Discurso de Ódio

2.1 Detecção de Discurso de Ódio

Com o crescimento da internet e de plataformas como as redes sociais, houve um aumento na disseminação de discurso de ódio online e do aparecimento de grupos de ódio nesses espaços. Discurso de ódio não é um fenômeno novo porém, devido à anonimidade e à sua natureza globalizada, a internet se tornou a ferramenta ideal para propagação desse tipo de conteúdo (BANKS, 2010).

Sendo um meio de comunicação globalizado e minimamente regulado, a internet permite a usuários se expressarem livremente. Apesar da liberdade de expressão ser um direito humano a ser preservado, a disseminação de ódio contra grupos e pessoas ultrapassa os limites desse direito (MACAVANEY et al., 2019).

Redes sociais como Facebook, Twitter e YouTube podem ser responsabilizadas por permitirem a disseminação de discurso criminoso em suas plataformas (BBC, 2016). As próprias plataformas tem políticas de uso que restringem ou proíbem o discurso de ódio (TWITTER, 2021; FACEBOOK, 2021; GOOGLE, 2021). Devido ao grande volume de publicações diárias, a análise manual dos textos se torna uma tarefa quase impossível, levando as empresas muitas vezes a dependerem de artifícios como denúncias de usuários. Nesse contexto surge a necessidade de desenvolver ferramentas capazes de analisar e identificar automaticamente textos contendo discurso de ódio.

Porém, reconhecer discurso de ódio é uma tarefa complexa. A primeira dificuldade vem em definir claramente o que é discurso de ódio. Não existe uma definição universal, e há discordância mesmo sobre os elementos que caracterizam esse discurso (MACAVANEY et al., 2019).

A rede social Twitter adota o seguinte posicionamento sobre discurso de ódio:

> "Conduta de propagação de ódio: não é permitido promover violência, atacar diretamente ou ameaçar outras pessoas com base em raça, etnia, origem nacional, orientação sexual, sexo, identidade de gênero, religião, idade, deficiência ou doença grave. Também não permitimos contas cuja finalidade principal seja incitar lesões a outros com base nessas categorias." (TWITTER, 2021)

Já o Facebook utiliza a seguinte definição:

> "Definimos discurso de ódio como um ataque direto a pessoas, e não a conceitos e instituições, baseado no que chamamos de características protegidas: raça, etnia, nacionalidade, religião, orientação sexual, casta, sexo,

> gênero, identidade de gênero e doença grave ou deficiência. Definimos ataques como discursos violentos ou desumanizantes, estereótipos prejudiciais, declarações de inferioridade, expressões de desprezo, repugnância ou rejeição, xingamentos e apelos à exclusão ou segregação. Também proibimos o uso de estereótipos prejudiciais, que definimos como comparações desumanizantes que têm sido historicamente usadas para atacar, intimidar ou excluir grupos específicos, e que muitas vezes estão ligadas à violência no meio físico. Consideramos a idade uma característica protegida quando referenciada juntamente com outra característica também protegida. Também protegemos refugiados, migrantes, imigrantes e pessoas que buscam asilo de ataques mais severos, embora permitamos comentários e críticas às políticas de imigração. Da mesma forma, fornecemos algumas proteções para aspectos como ocupação, quando estes são mencionados juntamente com uma característica protegida." (FACEBOOK, 2021)

Algumas definições consideram ataques direcionados a grupos como discurso de ódio, enquanto outras consideram apenas ataques direcionados a indivíduos. A dúvida também é gerada dependendo de como o ataque é feito. O uso de termos chulos ou ofensivos, como gírias racistas por exemplo, é claramente uma demonstração de discurso de ódio, porém ataques mais sutis, como relacionar um grupo a algum tipo de comportamento utilizando afirmações infundadas, pode se passar por um simples comentário ou opinião. Há também o problema de determinar se o ato de elogiar ou apoiar certos grupos, como a Ku Klux Klan, pode ser considerado como propagação de ódio ou incentivo à violência. Outra questão é se humor pode ou não ser considerado como discurso de ódio.

A falta de uma definição única sobre discurso de ódio dificulta tanto a análise manual quanto a classificação automática. Ao produzir uma base de dados, cada autor utiliza critérios próprios para a classificação manual dos textos, como será visto logo a seguir, o que dificulta a integração de bases de dados construídas por diferentes autores. Dadas as dificuldades citadas, a própria construção de uma nova base de dados se torna uma tarefa complexa, pois surgem entraves para que os anotadores cheguem a uma conclusão sobre a classificação de cada texto. Ross et al. (2017) mostra que o nível de concordância entre os anotadores para essa tarefa, medido utilizando o alfa de Krippendorff, é baixo.

Apesar de serem um ambiente propício para a propagação de discurso de ódio, e assim supostamente uma boa fonte de documentos para bases de dados, as redes sociais possuem políticas de uso de dados que podem dificultar a coleta de textos. Muitas bases de dados são baseadas em textos extraídos do Twitter, já que essa rede social possui regras mais brandas para a coleta de dados (MACAVANEY et al., 2019).

A limitação de fontes implica na limitação do tipo de texto que está sendo utilizado para formar a base de dados. As características de cada rede social influenciam na forma, e até nos traços estilísticos dos textos. Textos extraídos do Twitter por exemplo, tendem a ser mais curtos devido à restrição do número de caracteres.

Uma das formas mais básicas de detectar discurso de ódio é usando uma lista de

palavras-chave, como por exemplo termos ofensivos previamente conhecidos por expressar racismo, homofobia, ou outro tipo de ataque. Essa técnica possui vários entraves. Primeiramente, termos mudam de acordo com o tempo, frequentemente novos termos são inseridos no vocabulário e termos antigos se tornam obsoletos. Segundo, a propagação de ódio pode ocorrer sem o uso de palavras específicas. Terceiro, mesmo o uso desse tipo de termo não garante que o texto contenha discurso de ódio.

Termos como "lixo" e "sujo" são termos que podem ser utilizados em uma frase para atacar minorias, caso os termos estejam se referindo a esses grupos, porém não é produtivo inserir esses termos em uma lista de palavras-chave pois esses termos também são comumente utilizados em contextos mais cotidianos. Mesmo termos mais chulos, como "puta" e "viado", são frequentemente utilizados em conversas entre amigos para se referir uns aos outros.

Abaixo temos dois exemplos de textos identificados como discurso de ódio por Fortuna et al. (2019) retirados da rede social Twitter. Nesses dois exemplos, podemos perceber que não são utilizados termos chulos específicos para discurso de ódio. As palavras "gorda", "fedida" no primeiro exemplo, e "preto", no segundo exemplo, são frequentemente utilizadas no dia a dia para falar de assuntos comuns. Porém, no contexto em que estão inseridas no exemplo, elas são utilizadas para atacar grupos ou pessoas.

> "@g1 só gorda feia cabeluda fedida. Horror. Mulher que se preza nao tem tempo pra babaquice."
> (AUTOR ANÔNIMO)

> "Tinha que ser preto (nada contra))))"
> (AUTOR ANÔNIMO)

Outros algoritmos podem tentar fazer uso de metadados para realizar a tarefa em questão. Porém essa abordagem possui dois grandes problemas, o primeiro sendo a disponibilidade desses dados. O segundo está relacionado ao uso de algoritmos de inteligência artificial, que pode acabar gerando um viés contra grupos específicos (MACAVANEY et al., 2019).

Muitos métodos recentes utilizam aprendizado de máquina supervisionado para tentar identificar textos que contenham discurso de ódio, seja baseando-se apenas nos textos ou no conjunto texto + metadados. Esses métodos, porém, exigem dados rotulados para o treinamento. Pelos motivos já citados anteriormente, existe uma escassez de bases de dados rotulados de grande tamanho para a tarefa de detecção de discurso de ódio, principalmente para línguas diferentes da língua inglesa, como o português. No próximo subcapítulo são citados trabalhos relacionados à detecção de discurso de ódio e as bases de dados existentes atualmente para a língua portuguesa.

2.2 Trabalhos Relacionados

Recentemente, muitos esforços tem sido feitos para resolver o problema de detecção de discurso de ódio online. Apesar da maioria dos trabalhos se concentrarem na língua inglesa, alguns trabalhos focam na língua portuguesa. A seguir serão apresentados diversos trabalhos da literatura com o foco em detectar discurso de ódio online, tanto para a língua inglesa quanto para a língua portuguesa.

Waseem e Hovy (2016) criaram uma base de dados para detecção de discurso sexista e racista, com textos na língua inglesa retirados do Twitter e classificados manualmente. Os autores fizeram uma pesquisa manual dos termos mais comuns para expressar discurso de ódio contra minorias, e utilizaram esses termos para coletar 136.052 tweets durante um período de 2 meses. Para o processo de anotação, os autores definiram critérios para enquadrar um texto como discurso de ódio. De acordo com os critérios estabelecidos ataques mais subjetivos, como por exemplo fazer afirmações infundadas sobre uma minoria, também são considerados discurso de ódio. Os próprios autores classificaram a base de dados com ajuda de uma anotadora externa. Por fim 16.914 tweets foram anotados, sendo 3.383 considerados sexistas, 1.972 racistas e 11.559 não contendo nem racismo nem sexismo.

Para a validação da base de dados, os autores extraíram característica extra-linguísticas, como gênero dos autores e localização geográfica, para avaliar se tais informações auxiliariam os classificadores automáticos baseados em n-grams de caracteres. Porém, eles atestaram que essas informações não trazem grandes melhorias, quando trazem alguma.

Infelizmente, os autores disponibilizam apenas os IDs dos tweets, e não o texto em si. Sendo assim, os tweets estão sujeitos à remoção da rede social. Como parte dos textos contém discurso de ódio, grande parte dessa base não está mais disponível, seja por remoção dos próprios autores das mensagens ou por moderação da plataforma. Ao tentar coletar a base em 2019, haviam apenas 11.205 tweets dos 16.914 originais, sendo 8.278 sem conteúdo sexista ou racista, 2.915 sexistas, e apenas 12 racistas.

Badjatiya et al. (2017) utilizam diversas técnicas, incluindo redes neurais profundas, para classificar os tweets da base de Waseem e Hovy (2016). Eles utilizam técnicas não neurais para criar uma base de referência, sendo elas: regressão logística, árvores de decisão com *gradient boosting* e SVM (*Support Vector Machine*). Os autores variam as técnicas de codificação do texto, e para a construção da base de referência eles utilizam n-grams de caracteres, TF-IDF (*Term Frequency — Inverse Document Frequency*) e vetores *Bag-of-Words*.

Os autores comparam essas referências com o desempenho de redes profundas CNN (*Convolutional Neural Networks*), LSTM (*Long short-term memory*) e FastText. Para esses modelos, são utilizados *embeddings* como codificação, tanto *embeddings* inicializados

aleatoriamente como *embeddings* do tipo GloVe (PENNINGTON et al., 2014) pré-treinados.

Por fim, os autores uniram as redes profundas com uma árvore de decisão. Os *embeddings* aprendidos durante o treinamento das redes profundas são utilizados no treinamento das árvores de decisão. Como esses vetores foram ajustados especificamente para a base de dados utilizada e para a tarefa em questão, eles supostamente seriam mais apropriados para as árvores de decisão. O modelo que trouxe melhores resultados foi o que utilizou *embeddings* inicializados aleatoriamente com a rede LSTM e a árvore de decisão com *gradient boost*.

O trabalho de Malmasi e Zampieri (2017) se destaca por tentar diferenciar discurso de ódio de linguagem ofensiva. De acordo com os autores, a linguagem ofensiva pode ser utilizada para diferentes fins, como para dar ênfase ou para propósitos estilísticos. De acordo com a maioria das definições de discurso de ódio, esse tipo de discurso é caracterizado pelo ataque a indivíduos ou a algum grupo baseado em características como cor, sexo, religião, entre outras. Caso a linguagem ofensiva não esteja sendo utilizada para esse propósito, o texto não constitui discurso de ódio.

Os autores utilizaram uma base de dados balanceada com 14.509 textos em inglês classificados em três categorias: discurso de ódio, linguagem ofensiva e conteúdo não ofensivo. Eles utilizaram SVM como classificador e obtiveram uma acurácia de 78%. A intenção dos autores era criar uma referência para trabalhos futuros que tivessem o mesmo objetivo.

A maior parte dos trabalhos existentes focam em textos na língua inglesa, devido à grande disponibilidade de base de dados nessa língua. Porém, como o presente trabalho foca no uso de aprendizado profundo para a detecção de discurso de ódio na língua portuguesa, são necessárias bases de dados rotuladas em português. Nesse âmbito destacamos 3 trabalhos: Pelle e Moreira (2017), Fortuna et al. (2019) e Alves Vargas et al. (2021).

A primeira base de dados encontrada para português foi desenvolvida por Pelle e Moreira (2017). Os autores coletaram comentários do site de notícias brasileiro G1[1], do Grupo Globo, que na época era o portal de notícias mais acessado do país. O site permite que usuários façam comentários e respondam comentários de outros usuários.

A coleta foi feita utilizando um *script* próprio e o texto dos comentários foi extraído do HTML das páginas coletadas. Dos 10.336 comentários coletados, 1.250 foram escolhidos aleatoriamente para o processo de anotação manual, em que cada comentário era classificado entre "ofensivo" e "não ofensivo" por 3 jurados distintos.

Duas bases de dados foram geradas a partir desse processo: OffComBr-2, em que a classe atribuída era a classe escolhida pela maioria dos jurados, e OffComBr-3, que contém apenas os comentários que obtiveram unanimidade entre os jurados. Sendo assim,

[1] g1.globo.com

a base OffComBr-2 contém 1.250 comentários, sendo 32.5% ofensivos, e a base OffComBr-3 contém 1.033 comentários, sendo que 19.5% são classificados como ofensivos.

O trabalho de Pelle e Moreira (2017) foi o primeiro a criar uma base de dados desse tipo para o Português. Apesar dos textos não serem oriundos de uma plataforma social, como Twitter ou Facebook, a funcionalidade dos comentários do site G1 é similar as das redes sociais mais comuns. Ainda assim, a base de dados é limitada, e contém apenas 1.250 comentários, mesmo na versão OffComBr-2.

A segunda base encontrada foi desenvolvida por Fortuna et al. (2019), constituída de 5.668 tweets. Os textos foram coletados utilizando 19 palavras-chave relacionadas à discurso de ódio e 29 perfis conhecidos por disseminar discurso de ódio contra minorias. A maioria dos tweets (95%) foram publicados entre Janeiro e Março de 2017.

O processo de anotação foi feito em duas etapas. Primeiramente, os tweets foram classificados por 18 anotadores não-especialistas em 2 categorias: discurso de ódio e sem discurso de ódio, que resultou em uma base de dados com 31,5% dos textos classificados como discurso de ódio.

Em seguida, os tweets foram organizados em uma estrutura hierárquica com múltiplos rótulos, sendo as raízes da estrutura hierárquica as seguintes classes: sexismo, corpo, origem, homofobia, racismo, ideologia, religião, saúde, outros estilos de vida. Por fim, resultaram 81 subclasses.

A terceira base, nomeada “HateBr”, foi desenvolvida por Alves Vargas et al. (2021). Os textos foram coletados da rede social Instagram, especificamente dos comentários feitos em publicações. Os autores focaram em contas de personalidades políticas brasileiras.

Foram escolhidas 6 contas e 30 publicações. Dessas publicações foram extraídos 15.000 comentários. As contas foram escolhidas de maneira a balancear o viés ideológico e de gênero, sendo 3 contas de personalidades esquerda e 3 contas de personalidades de direita, quatro homens e duas mulheres. As publicações foram selecionadas de forma aleatória, independente do tópico abordado.

O corpus foi classificado de três maneiras distintas:

- Binário: os textos foram classificados entre “ofensivo” e “não ofensivo”;
- Intensidade: os textos foram classificados conforme a intensidade entre “altamente ofensivo”,“moderadamente ofensivo” e “levemente ofensivo”;
- Tipo: os textos foram classificados em grupos de ódio, os grupos estão expostos na Tabela 1;

O trabalho resultou em uma base de dados com 7.000 documentos, sendo 3.500 classificados como ofensivos. A Figura 2 mostra a distribuição das classes de intensidade

Tabela 1 – Grupos de ódio utilizados na base de dados HateBr

N	Grupos de Ódio	Total
1	Partido Político	496
2	Sexismo	97
3	Intolerância Religiosa	49
4	Apologia à ditadura	32
5	Gordofobia	27
6	Homofobia	17
7	Racismo	8
8	Antissemitismo	2
9	Xenofobia	1

Fonte: Tradução da autora de tabela retirada de Alves Vargas et al. (2021)

da base de dados para os comentários classificados como ofensivos.

Tabela 2 – Intensidade de ofensa nos comentários da base HateBr

Classes por Intensidade de Ofensa	Total
Levemente Ofensivos	1.678
Moderadamente Ofensivos	1.044
Altamente Ofensivos	778
Total	3.500

Fonte: Tradução da autora de tabela retirada de Alves Vargas et al. (2021)

Os autores apresentam a base de dados mais completa atualmente. Não só a base possui a maior quantidade de documentos das três (7.000 documentos contra 5.668 de FORTUNA et al.), mas também a base está classificada de três maneiras distintas.

Infelizmente, não foi encontrado um repositório onde a base estivesse armazenada, o que indica que talvez a base ainda não tenha sido disponibilizada livremente. Ademais, o trabalho apenas foi divulgado recentemente, em março de 2021, sendo assim não foi possível utilizar essa base de dados nesse trabalho de mestrado.

Há trabalhos na literatura que lidam com a detecção de discurso de ódio em português utilizando técnicas de aprendizado de máquina.

Paiva et al. (2019) utilizam características extraídas dos textos com classificadores mais simples para tentar detectar discurso de ódio em português. Os autores utilizaram a base de dados OffComBr-2 de PELLE; MOREIRA e realizaram algumas alterações de classificação em documentos onde não houve concordância com a classificação original. Além disso, para criar uma base balanceada, os autores reduziram o número de amostras da classe mais numerosa. Sendo assim, no final restaram 406 documentos de cada uma das classes.

Eles extraíram um conjunto de características linguísticas de cada documento,

como: quantidade de palavras ofensivas, quantidade de palavras, tamanho médio das palavras, quantidade de caracteres e quantidade de palavras escritas corretamente. Como classificadores foram testadas diversas técnicas, incluindo dois modelos de SVM, três modelos de *Naive Bayes* e uma árvore de decisão. Porém os autores atestaram que somente essas características linguísticas não foram suficientes para predizer se um comentário é ofensivo ou não. Por essa razão no próximo passo foram utilizados vetores *Bag-of-Words* juntamente com as características extraídas, o que levou a uma melhora nos resultados.

Por fim, os autores fazem uma análise de quais características extraídas mais auxiliaram os classificadores. Os resultados indicam que comentários ofensivos tendem a ter um número maior de palavras ofensivas, o que já era esperado, mas também que comentários ofensivos tem mais palavras escritas incorretamente, são geralmente mais curtos e tem um tamanho médio de palavras menor, o que leva a entender que eles são menos elaborados e possuem menos conteúdo.

Silva e Roman (2020) testaram diversas abordagens de aprendizado de máquina para a tarefa: *Support vector machines* (SVM), *Naïve Bayes* , Perceptron Multicamadas (MLP) e Regressão Logística. Os autores utilizam a base de dados de Fortuna et al. (2019), uma das bases utilizada nesse trabalho, e afirmam que conseguiram resultados superiores à rede LSTM utilizada pelos criadores da base. Para isso, os modelos de representação *Bag-of-words* e *n-grams*, tanto para palavras quanto para caracteres, foram utilizados para codificar os textos. Para cada modelo, foram feitos testes com a frequência simples dos termos e também calculando o TF-IDF. Os autores avaliam que a quantidade de textos na base de dados talvez não seja o suficiente para treinar a rede neural LSTM suficientemente bem.

As Tabelas 3 e 4 resumem os resultados encontrados na revisão de literatura.

Tabela 3 – Resumo dos resultados encontrados na revisão da literatura - Parte 1

Trabalho	Autores	Base de Dados	Classes
1	Badjatiya et al. (2017)	Waseem e Hovy (2016)	3
2	Malmasi e Zampieri (2017)	Davidson (DAVIDSON et al., 2017)	3
3	Paiva et al. (2019)	OffComBr-2 (Pelle e Moreira)	2
4	Silva e Roman (2020)	Fortuna	2

Tabela 4 – Resumo dos resultados encontrados na revisão da literatura - Parte 2

Trabalho	Melhor método	Métrica	Valor
1	LSTM+Random Embedding+GBDT	F1	93,00
2	Linear SVM + 4-grams de caracteres	Acurácia	78,00
3	Multinomial Naive Bayes + Características + Bag-of-Words	Acurácia	81,00
4	SVM	F1	71,30

3 Técnicas

Vários avanços recentes no campo do processamento de linguagem natural são devidos às redes neurais profundas. Duas arquiteturas recebem destaque por serem amplamente utilizadas nessa área: as redes neurais recursivas (RNN) e as redes neurais convolucionais (CNN) (YIN et al., 2017). Dentre os modelos de RNN, nesse trabalho é utilizada a arquitetura LSTM.

3.1 Processamento de Linguagem Natural e Classificação de Texto

São consideradas "linguagens naturais" aquelas que são utilizadas no dia a dia como meios de comunicação, como o português, o inglês, o chinês, entre outras. Essas línguas são meios de comunicação que foram passadas de geração em geração e evoluíram naturalmente através do uso diário. Essas línguas também são caracterizadas pela dificuldade de determinar com exatidão suas regras de uso, pois elas estão em constante mudança. A expressão "linguagem natural" é geralmente empregada para diferenciar essas linguagens de linguagens artificiais, como as linguagens de programação Python, Java, C++, entre outras (KUMAR, 2011).

A área de processamento de linguagem natural é uma área de estudos que abrange o campo da ciência da computação e da linguística. Essa área estuda o processamento e a análise automática de linguagens humanas (YOUNG et al., 2018). Algumas aplicações típicas da área de processamento de linguagem natural são o reconhecimento de voz, a análise de sentimentos, tradução de máquina, geração automática de textos, entre outras (DENG; LIU, 2018).

Algumas dessas aplicações, como análise de sentimentos e detecção de discurso de ódio, podem ser enquadradas como tarefas de classificação de texto. Essa tarefa consiste em atribuir rótulos pré-definidos à documentos (SCOTT; MATWIN, 1999). Mais formalmente, considerando um conjunto de documentos $D = \{d_1, d_2, ..., d_m\}$ e um conjunto de rótulos $L = \{l_1, l_2, ..., l_n\}$, a tarefa de classificação de textos consiste em associar um subconjunto de L a cada documento d_i.

Um caso particular do problema de classificação é quando cada documento d_i pode ser associado à apenas um rótulo de L. Ou seja, cada documento pertence a uma classe pré-definida. Se houver apenas duas classes possíveis, o problema é considerado uma classificação binária (LORENA et al., 2008).

Com a popularização dos computadores pessoais, eles começaram a ser utilizados para funções organizacionais em empresas e órgãos governamentais, além da ser utilizados

por pessoas comuns para fins pessoais. Sendo assim, grande parte dos documentos antes em formato físico começaram a ser produzidos em formato digital, ou convertidos para esse formato, para que pudessem ser armazenados e compartilhados de maneira mais fácil e rápida. Atualmente, a maioria dos documentos em empresas privadas e órgãos governamentais está armazenada na forma digital, seja em documentos eletrônicos ou e-mails (DALAL; ZAVERI, 2011). Devido a essa grande quantidade de documentos, há a necessidade de métodos automáticos para a recuperação e classificação de documentos.

De acordo com Minaee et al. (2021a), existem duas vias principais para a classificação automática de documentos: sistemas baseados em regras e sistemas baseados em aprendizado de máquina (ou baseados em dados). Sistemas baseados em regras utilizam um conjunto de instruções descritas por especialistas para tomar decisões sobre os documentos. Porém, essa abordagem possui alguns problemas. Primeiramente, ela necessita de pessoas com conhecimento profundo da área. Segundo, esse tipo de método é difícil de generalizar para outras aplicações. Então, caso seja necessário criar um classificador similar, porém para documentos em outra língua, ou em outro contexto, o processo terá que ser reiniciado praticamente do princípio.

Já métodos de aprendizado de máquina operam buscando padrões dentro de dados já existentes. O lado negativo desses métodos é que eles geralmente requerem uma quantidade substancial de dados para obter resultados satisfatórios. Outro problema é que esses métodos também dependem da qualidade dos dados fornecidos. Caso o método escolhido seja de aprendizado supervisionado, outro requerimento é que os dados sejam rotulados, o que acarreta um custo adicional, já que o processo de rotular documentos é frequentemente manual e trabalhoso.

3.2 *Embeddings* GloVe

Redes neurais profundas são apenas capazes de processar dados numéricos. Por esse motivo é necessário converter documentos textuais para esse formato antes de utilizá-los para treinar uma rede neural.

Uma das maneiras de representar textos na forma numérica é com a utilização de *word embeddings*. Esses *embeddings* buscam representar cada palavra como um ponto em um espaço multidimensional, onde cada dimensão está relacionada a uma característica. Dessa forma, palavras com significados semelhantes tendem a ficar próximas umas das outras neste espaço D-dimensional, onde D é o tamanho do vetor que representa cada palavra. Com a utilização de *embeddings*, um documento contendo n palavras pode ser representado por uma sequência de n vetores de tamanho D, ou por uma matriz $n \times D$.

Esta representação é considerada um avanço em relação a outros métodos, como o *Bag-of-Words*, em que cada palavra é representada por um vetor do tamanho do vocabulário

do corpus. Ou seja, se o corpus possui 20.000 palavras, um documento de tamanho n seria representado por uma matriz de tamanho $n \times 20.000$, sendo que a maior parte dos valores da matriz seriam zeros, o que causa um dispêndio desnecessário de espaço.

Mikolov et al. (2013) introduziu técnicas para a criação de *word embeddings* baseadas em corpora com bilhões de palavras e com vocabulário com milhões de termos únicos. Os autores utilizaram redes profundas simples para a tarefa de construir modelos de linguagem, sendo que o aprendizado dos *embeddings* é feito durante o treinamento da rede. Os *embeddings* formados a partir dessas técnicas são chamados de Word2Vec.

Word embeddings bem treinados tornam possível a realização de operações algébricas com as palavras. Um exemplo clássico é a operação:

$$vetor(\text{“}rei\text{”}) - vetor(\text{“}homem\text{”}) + vetor(\text{“}mulher\text{”}) = vetor(\text{“}rainha\text{”}) \tag{3.1}$$

No exemplo acima, o vetor da palavra “rainha” pode ser encontrado pela Equação 3.1, considerando que ao subtrair o significado de “homem” do significado de “rei”, resta algum significado similar à característica “realeza”, e que ao adicionar o significado de “mulher” é obtido o significado de “rainha”. A resolução de analogias se tornou uma das maneiras mais comuns de se avaliar *word embeddings* (DROZD et al., 2016).

Pennington et al. (2014) apresentou uma técnica diferente para treinar *word embeddings*. Os autores argumentaram que o modelo introduzido por Mikolov et al. (2013) considerava apenas o contexto local ao invés de utilizar informações de co-ocorrências ao longo do corpus. Os vetores globais para representação de palavras (*Global Vectors for Word Representation*), ou simplesmente GloVe, utilizam uma matriz de co-ocorrências baseada em todo o corpus para calcular uma matriz de probabilidades.

Os autores demonstraram que o modelo GloVe apresenta melhores resultados que o modelo Word2Vec na resolução de analogias e também em reconhecimento de entidades nomeadas (*named entity recognition*).

Em uma arquitetura profunda, a camada de *embedding* geralmente precede outras camadas, como a LSTM por exemplo. Essa camada é responsável em converter os documentos utilizados como entrada da rede para a representação por *word embeddings*.

Considerando um vocabulário total $W = \{w_1, w_2, ..., w_v\}$, onde v é o total de palavras no vocabulário e os números de 1 a v são os índices de cada palavra nesse vocabulário, um documento de tamanho L pode ser representado como uma sequência de L índices.

A camada de *embedding* contém uma matriz de tamanho $D \times v$, onde estão contidas todas as representações em *word embeddings* das palavras do vocabulário. A camada de *embedding* substitui cada palavra do documento por seu vetor correspondente, tendo como saída uma matriz $D \times L$ que representa o documento.

3.3 LSTM

As técnicas mais tradicionais de classificação de textos geralmente não consideram a ordem das palavras, porém essa informação é relevante para identificar certas ideias expressas no texto.

Redes neurais simples não levam em consideração a ordem dos dados. Sendo assim, qualquer informação sequencial é perdida durante o processo. Os modelos convolucionais apresentam algum nível de sensibilidade quanto à ordem, porém esses modelos estão limitados a identificar padrões locais, e ignoram padrões com maior distanciamento no texto (GOLDBERG, 2017).

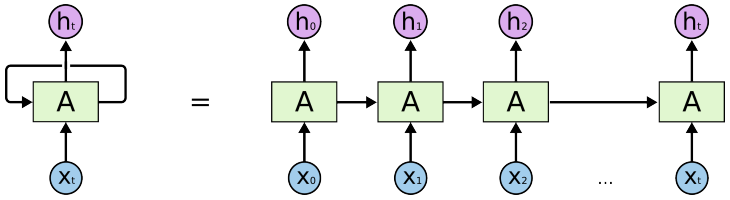
Já as redes neurais recorrentes (RNN) foram desenvolvidas para lidar com dados sequenciais. Nesse tipo de arquitetura, todas as palavras do texto passam por uma mesma camada profunda, que analisa cada palavra passo a passo levando em consideração todas as palavras vistas anteriormente. De maneira mais formal, considere uma frase $S = \{x_1, x_2, ..., x_t, ..., x_n\}$, sendo x_t o vetor que codifica a t-ésima palavra na frase, que será a entrada da RNN na t-ésima iteração de tempo (considerando uma palavra de entrada por vez). A camada recorrente processa o vetor x_t e gera uma saída h_t. Esta saída e o vetor x_{t+1} servem de entrada para a camada na próxima iteração $t + 1$. Ao utilizar esse mecanismo, a rede tem acesso não só a informação de uma palavra, mas também à informação do contexto em que a palavra está inserida.

Para ajustar os pesos da camada recorrente, utiliza-se o algoritmo de *backpropagation through time* (BPTT), na qual a camada pode ser representada como diversas camadas em sequência, uma camada para cada instante de tempo t (PASCANU et al., 2013). A Figura 1 mostra a camada "desenrolada". O algoritmo então calcula o erro acumulado em cada instante de tempo e ajusta os pesos dos neurônios da camada.

Um dos problemas desse método é o custo computacional. Como o cálculo em cada instante t depende dos outros instantes, esse processo não pode ser paralelizado. Além disso, o processamento tanto da propagação dos dados pela rede quanto da retropropagação do erro são proporcionais ao tamanho do texto. Se a frase contiver 20 termos, o custo computacional de processar uma única camada RNN será equivalente a processar 20 camadas simples (SUTSKEVER, 2013).

Apesar de apresentar uma teoria interessante, na prática a arquitetura RNN só apresenta resultados interessantes quando a dependência entre as entradas é de um curto período de tempo. Devido à BPTT, os sinais tendem a diminuir rapidamente até desaparecer, ou a aumentar repentinamente, o que se agrava em situações de longa dependência temporal (várias iterações no tempo) (HOCHREITER; SCHMIDHUBER, 1997). Buscando reparar esse problema, Hochreiter e Schmidhuber (1997) propuseram um novo modelo de RNN: a arquitetura LSTM (*Long Short-Term Memory*), ou memória

Figura 1 – Uma camada "desenrolada" de uma RNN

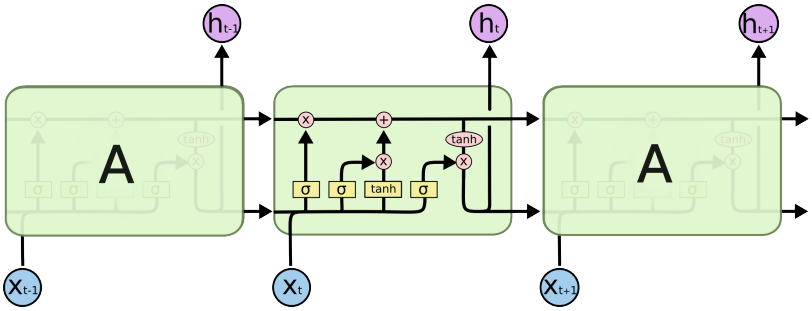


Fonte: Olah (2015)

longa de curto prazo, em tradução livre.

Na arquitetura LSTM, os nerônios simples que realizam a soma ponderada da entrada e utilizam função de ativação são substituídos por estruturas mais complexas chamadas células, sendo que cada uma possui três portas. Estas portas permitem que a informação seja conservada por mais tempo, procurando remediar o problema das RNN (GRAVES, 2012). A estrutura de uma célula LSTM é mostrada na Figura 2.

Figura 2 – Célula LSTM



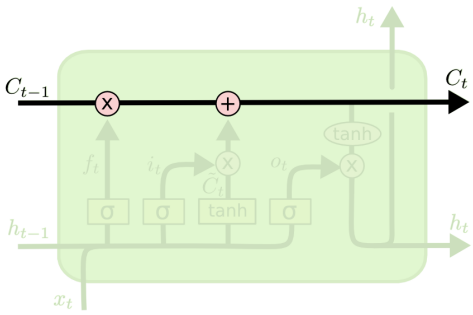
Fonte: Olah (2015)

A célula possui três variáveis de entrada, C_{t-1}, h_{t-1} e x_t, e duas saídas, C_t e h_t, sendo que x_t é geralmente o vetor que representa a palavra da t-ésima iteração.

A variável C_t na Figura 3 representa o estado da célula. Essa informação é passada por todas as iterações sendo apenas modificada por pequenas alterações lineares. A primeira porta, na Figura 4, controla quanto da informação original é mantida, e é chamada de porta de esquecimento, tradução livre de *forget gate*. As variáveis de entrada da célula x_t e o estado oculto h_{t-1} da iteração anterior geram uma saída no intervalo entre zero e um que define o quanto de informação da iteração anterior será mantida e esquecida.

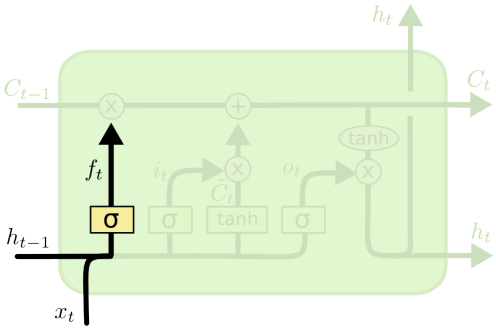
A próxima porta, mostrada na Figura 5, controla o quanto de nova informação será adicionada ao estado C_t. A terceira porta, na Figura 6, controla a informação que vai para o estado oculto h_t, ao combinar as informações de $h_t - 1$, x_t e C_t.

Figura 3 – Estado da Célula C_i



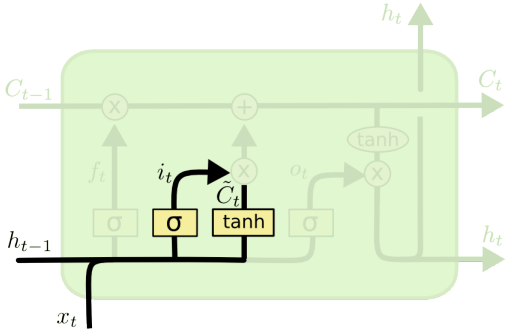
Fonte: Olah (2015)

Figura 4 – Primeira Porta



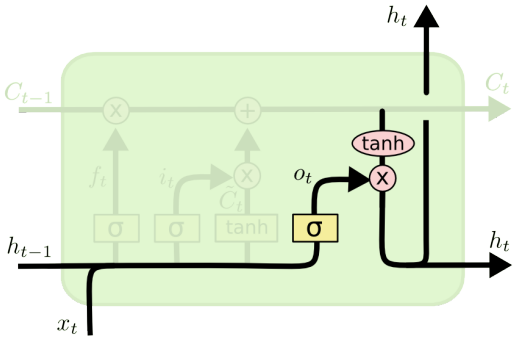
Fonte: Olah (2015)

Figura 5 – Segunda Porta



Fonte: Olah (2015)

Figura 6 – Terceira Porta



Fonte: Olah (2015)

3.4 CNN

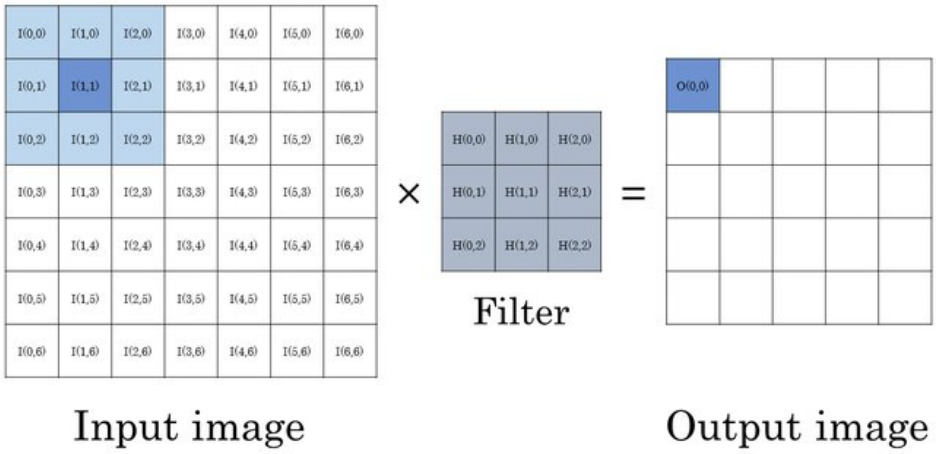
As redes neurais convolucionais (CNN) são arquiteturas de redes profundas que utilizam filtros convolucionais como extratores de característica dos dados. Apesar de terem sido criadas inicialmente para visão computacional, elas têm demonstrado eficácia em tarefas de processamento de linguagem natural (NLP) (KIM, 2014).

No campo de visão computacional, os dados de entrada geralmente são considerados como uma matriz de duas dimensões, ou um tensor (entidade com 3 dimensões ou mais), que representam uma imagem.

Existem diferentes tipos de camadas em uma arquitetura de uma CNN, sendo a principal delas a camada convolucional. A camada convolucional realiza a operação de produto escalar entre duas matrizes, onde uma matriz é a matriz de dados e a outra é um kernel de duas dimensões. O kernel é uma matriz de duas dimensões formado por pesos que serão otimizados pela rede neural. O kernel se sobrepõe à matriz de dados e realiza a operação de produto escalar, como mostrado na Figura 7. A figura mostra a operação sobre um dos elementos da imagem de entrada, porém se o kernel for “deslizado” sobre a

imagem de entrada, ou seja, se a mesma operação for repetida para os outros elementos, uma imagem de saída é gerada. A ideia é que esse kernel extraia alguma característica da imagem apresentada. Porém, geralmente diversos kernels são empilhados em uma camada de forma a extrair diversas características da imagem, todos com as mesmas dimensões, porém com diferentes parâmetros.

Figura 7 – Operação de Convolução 2D



Fonte: Baskin et al. (2018)

Ao aplicar redes convolucionais em dados textuais, é necessário utilizar redes com convolução em apenas uma dimensão, ou CNN-1D. Isto acontece, pois, em dados textuais, o interesse é reconhecer padrões que variam no eixo temporal.

A convolução 1D funciona de maneira similar a operação descrita anteriormente. Considere um documento de tamanho n representado por *word embeddings* de dimensão D, ou seja, uma matriz $n \times D$. Para a convolução 1D, pelo menos uma das dimensões do kernel tem que ser igual a uma das dimensões dos dados, e o movimento do kernel ocorrerá no outro eixo.

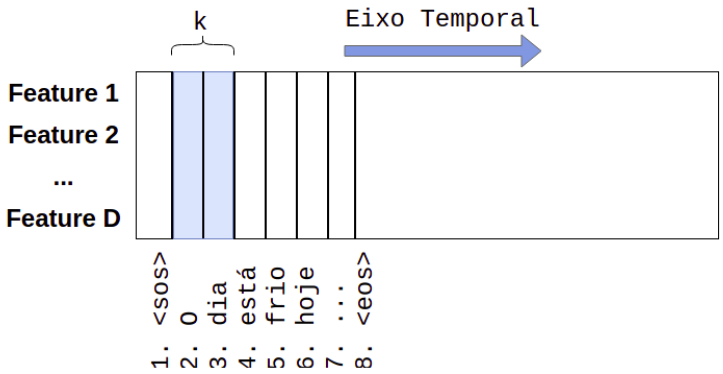
Nesse caso, o kernel será de tamanho $k \times D$, onde k é um hiperparâmetro a ser ajustado, e o kernel se moverá pelo eixo temporal, como mostra a Figura 8.

Redes CNN tendem a ser úteis para o reconhecimento de padrões locais, por exemplo, expressões idiomáticas (MINAEE et al., 2021b).

3.5 Aumento de Dados para Textos

Técnicas de aprendizado profundo demandam uma grande quantidade de dados rotulados. Porém, como explicado no Capítulo 2, a construção de bases de dados rotuladas é um trabalho manual e custoso. Modelos de aprendizado de máquinas complexos tendem a sofrer *overfitting* ao serem treinados com um número reduzido de dados.

Figura 8 – Operação de Convolução 1D sobre texto



Fonte: A autora.

Existem diversas maneiras de tentar amenizar esse problema, uma delas é utilizando técnicas de aumento de dados. Esse tipo de técnica é comumente utilizada na área de visão computacional e consiste em alterar exemplos da base de dados de maneira a obter novas amostras com a mesma classe das imagens originais. Na área de visão computacional pode-se recortar uma imagem ou inverter algum de seus sentidos. Também é possível adicionar ruído às imagens para criar amostras adicionais para o treino dos algoritmos (MA, 2019). Porém, tais alterações não podem ser aplicadas tão facilmente em dados textuais.

Diversas opções de aumento de dados com foco em textos são encontradas na literatura. Algumas das mais simples são a Remoção Aleatória (*Random Deletion*) (RD), Troca Aleatória (*Random Swap*) (RS), Substituição por Sinônimo (*Synonym Replacement*) (SR) e Inserção Aleatória *Random Insertion* (RI). Estas técnicas foram compiladas por (WEI; ZOU, 2019).

A Substituição por Sinônimos utiliza algum tipo de dicionário de sinônimos para substituir palavras da frase original. Um número N de palavras é sorteado e para cada palavra escolhida, o algoritmo gera um texto novo substituindo a palavra por algum de seus sinônimos. Ou seja, um número N de novas amostras são geradas para cada amostra original.

A Inserção Aleatória é similar à Substituição por Sinônimos, porém, ao invés de cada palavra sorteada ser substituída por seu sinônimo, a palavra original permanece no texto e seu sinônimo é inserido em alguma posição aleatória do texto. Esse método também gera N novas amostras para cada amostra original, sendo que N continua sendo a quantidade de palavras sorteadas.

Na Troca Aleatória duas palavras distintas são selecionadas do texto e tem suas posições trocadas. Esse método gera uma nova amostra para cada amostra original.

Na Remoção Aleatória, N posições distintas do texto são sorteadas. O método gera

uma amostra nova para cada palavra sorteada, sendo que o novo texto é o texto original sem a palavra na posição selecionada. Sendo assim, são geradas N novas amostras para cada exemplo original.

Também existem técnicas de aumento de dados que utilizam ferramentas de tradução automática.

Na técnica de *Back-translation*, as amostras originais são traduzidas para uma segunda língua, por exemplo o inglês, e então traduzidas de volta para o português. Como a tradução é feita de forma automática, algumas inconsistências são esperadas tanto na tradução $L1 \rightarrow L2$ como na tradução inversa $L2 \rightarrow L1$. Sendo assim, o resultado é um texto com conteúdo semântico igual ou similar ao do texto original, porém escrito de forma ligeiramente diferente (MA, 2019).

Outra forma de utilizar a tradução automática para o aumento de dados é traduzindo uma base de dados de outra língua. Se uma base de dados em inglês estiver disponível, por exemplo, é possível utilizar uma ferramenta de tradução automática para traduzir a base para português. Os problemas encontrados com essa técnica é que, devido à qualidade atual da tradução automática, muitos textos são traduzidos de forma literal ou incorreta. Mesmo textos traduzidos de forma correta podem não carregar as mesmas características dos textos escritos originalmente em português, como construção de frase, jargões e gírias. Esse problema foi observado durante a primeira etapa desse trabalho, em que foram feitos experimentos preliminares com a diversas técnicas.

Uma terceira via é a utilização de *word embeddings*, proposta por Wang e Yang (2015). A técnica é similar à Substituição por Sinônimos, porém, ao invés de utilizar um dicionário de sinônimos, a palavra selecionada é substituída por sua vizinha mais próxima dentro do *embedding*. Para isso, utiliza-se o algoritmo *k-nearest neighbours* (kNN) com a métrica de cosseno como medida de similaridade entre os *embeddings*.

A maior desvantagem dessa técnica é o tempo de processamento. Para encontrar vizinho mais próximo de uma palavra é necessário calcular a similaridade desta palavra com todas as outras palavras do vocabulário. O *embedding* para português desenvolvido por Hartmann et al. (2017), por exemplo, contém 929.606 palavras. Caso a versão de 200 dimensões fosse utilizada, para cada palavra substituída seria necessário calcular quase 1 milhão de similaridades entre vetores de 200 dimensões.

4 Bases de Dados e Método Proposto

Neste trabalho é proposto um método para a melhora da identificação de discurso de ódio em bases de dados disponíveis em português utilizando técnicas de aprendizado profundo.

Dada a revisão teórica feita e apresentada no Capítulo 2, observou-se que um dos grandes problemas para o uso de redes profundas na detecção de discurso de ódio em português é a limitação nas bases de dados. Durante a pesquisa de revisão de literatura foram identificadas apenas três bases de dados, sendo que apenas duas estão disponíveis. Além disso, as bases disponíveis possuem poucos exemplos, as duas bases encontradas totalizam 6.918 documentos.

Sendo assim, neste trabalho é proposto o uso de técnicas de aumentos de dados para amenizar o problema de disponibilidade de documentos rotulados. Este trabalho analisa o efeito da utilização de três técnicas de aumento de dados distintas como também a combinação dessas três técnicas.

Além disso, também é proposta a utilização de redes híbridas para a classificação das bases de dados em português. Até o momento dessa pesquisa, a autora não encontrou na literatura o uso de redes híbridas com camadas convolucionais em ramos paralelos para a detecção de discurso de ódio.

Nesse capítulo são expostas as arquiteturas profundas utilizadas, as técnicas de aumentos de dados, o processo de preparação dos documentos para treinamento e teste, e como foram realizados os experimentos.

4.1 Bases de Dados e Preparação dos Dados

4.1.1 Bases de Dados

Neste trabalho foram utilizadas a base de dados de Fortuna et al. (2019), a partir daqui referida como base Fortuna, e a base OffComBR-2 de Pelle e Moreira (2017). Ambas as bases de dados foram escolhidas por serem em português, disponíveis publicamente, e terem como foco o discurso de ódio e linguagem ofensiva.

A base de dados Fortuna é formada por tweets e possui 5.668 documentos, sendo 31,5% dos textos classificados como discurso de ódio. Já a base de dados OffComBr-2 é formada por 1.250 comentários e respostas à comentários feitos no site `g1.globo.com`, sendo 32,5% considerados ofensivos.

Durante a execução do trabalho, percebeu-se uma diferença nos resultados entre as

duas bases. Devido à diferença de tamanho entre as duas bases (a base Fortuna possui mais de 4 vezes a quantidade de exemplos da base OffComBr-2) houve uma suspeita que o tamanho das bases fosse a principal causa. Para testar essa hipótese foi criada uma versão reduzida da base de dados Fortuna. Para a base reduzida, 25% dos dados da base original foram selecionados aleatoriamente, porém, de forma a manter a proporção de classes da base Fortuna original. Sendo assim, a base reduzida possui 31,5% dos exemplos classificados como discurso de ódio.

4.1.2 Pré-processamento das Bases de Dados

Primeiramente, as bases de dados passaram por uma etapa de pré-processamento seguindo as etapas apresentadas a seguir:

1. Tokenização: o texto é dividido em *tokens*, ou termos. Nesse caso, os documentos são convertidos em listas de palavras.

2. Conversão das letras para minúsculo: desconsidera-se a influência de letras maiúsculas e minúscula. Essa normalização é feita para que palavras como “Casa” e “casa” sejam consideradas idênticas.

3. Lematização: O processo de lematização reduz a palavra ao seu lema, ou forma do dicionário. A palavra “brincando”, por exemplo, é substituída pela palavra “brincar”. Esse processo auxilia a diminuir o número de palavras distintas no vocabulário do corpus minimizando a perda de sentido das palavras.

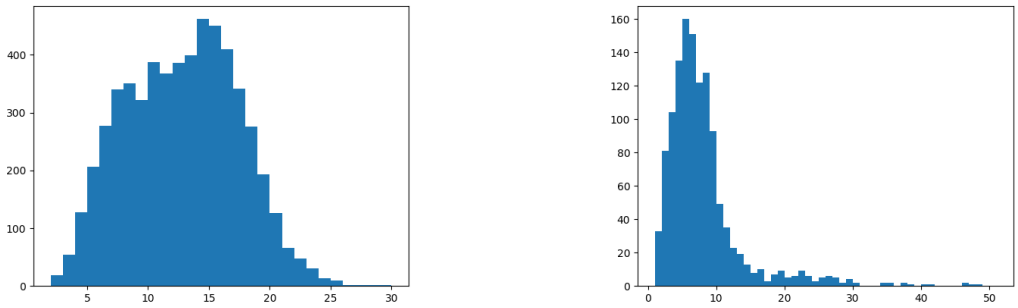
4. Remoção de pontuação: os sinais de pontuação são removidos dos documentos.

5. Remoção de *Stopwords*: *stopwords* são termos com pouca ou nenhuma importância semântica, como artigos, conjunções, entre outros. Por esse motivo elas são removidas dos documentos.

Os processos de tokenização, conversão das letras para minúsculo e lematização foram feitos utilizando a ferramenta NLPyPort para Python (FERREIRA et al., 2019). Apesar de ser uma das ferramentas mais completas entre as analisadas, ela ainda possui alguns problemas. Mesmo após a lematização, alguns verbos podem ser encontrados fora da forma do dicionário. Além disso, a ferramenta pode apenas processar palavras escritas na norma culta, então palavras escritas incorretamente e gírias passam pela ferramenta sem alterações.

A Figura 9 mostra os histogramas do número de palavras por documento após a etapa de pré-processamento para as duas bases de dados. Pode-se observar na figura que a maioria dos textos possui menos do que 25 palavras.

Figura 9 – Histograma do número de palavras por documento da base de dados de (a) Fortuna e (b) OffComBr-2 após o pré-processamento



(a) Fortuna (b) OffComBR-2

Fonte: A autora.

Após o pré-processamento dos textos, cada base de dados foi dividida em 5 partes, procurando respeitar a razão entre documentos classificados como positivo e documentos classificados como negativo. As técnicas de aumentos de dados foram aplicadas aos arquivos particionados, criando novos arquivos aumentados.

Essa forma de separação de arquivos permitiu que fosse aplicada a técnica k-fold com facilidade. A cada iteração do k-fold são selecionados 4 partes com aumento para treino e a parte restante sem aumentos para teste, como mostrado na Figura 10.

Essa estratégia também garante que se, por exemplo, um certo documento d_1 esteja no conjunto de treino, nenhum documento derivado de d_1, como d_1^* em que foi aplicada alguma técnica de aumento, esteja no conjunto de teste.

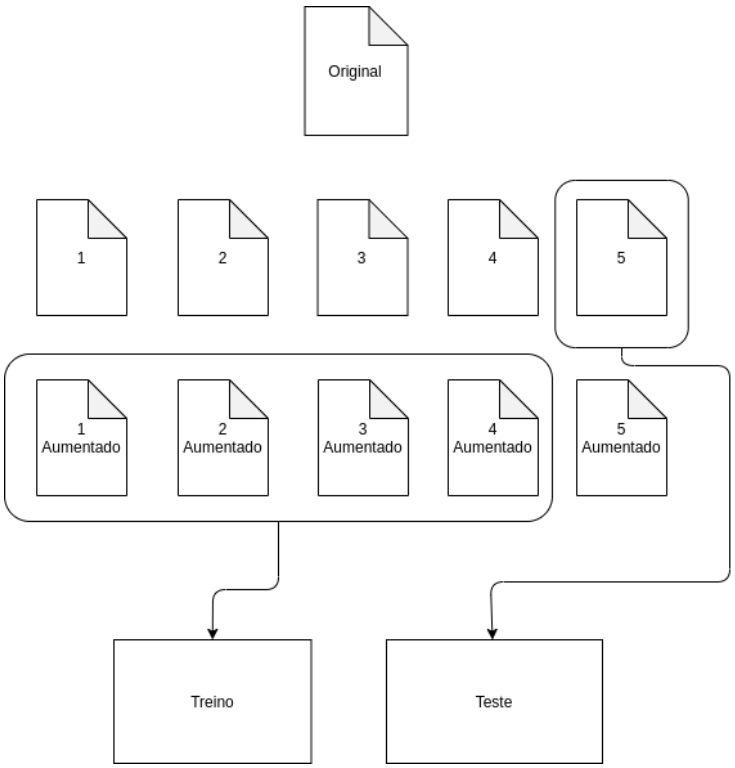
4.1.3 Aumento de Dados

As bases de dados disponíveis possuem uma quantidade pequena de amostras para o treinamento de redes profundas (1.250 e 5.668 amostras). Por esse motivo, foram aplicadas estratégias de aumentos de dados. As estratégias escolhidas foram as técnicas de Substituição por Sinônimos (SR), Remoção Aleatória (RD) e Troca Aleatória (RS), descritas no Capítulo 3.

Tanto a técnica Remoção Aleatória quanto a técnica de Troca Aleatória não exigem recursos externos, apenas são feitas modificações com as palavras já existentes no documento. Porém, a técnica de Substituição por Sinônimos exige um dicionário de sinônimos. Nesse trabalho foi utilizado o dicionário de sinônimos em português compilado por Dias-Da-Silva e Moraes (2003), que pode ser encontrado na plataforma GitHub[1].

[1] https://github.com/stavarengo/portuguese-brazilian-synonyms

Figura 10 – Sistema de arquivos para treino/teste



Fonte: A autora.

Alguns cuidados foram tomados na implementação dessas técnicas. As técnicas Remoção Aleatória e Substituição por Sinônimos possuem um parâmetro N, que é a quantidade de palavras sorteadas no documento, seja para a substituição ou para a remoção.

O valor de N não pode ser maior que o número de palavras no texto. No caso da substituição por sinônimos, também há o risco da palavra sorteada não ser encontrada no dicionário de sinônimos. Para lidar com isso, ao implementar o SR, em vez de sortear N palavras, foi gerada uma lista com os índices das palavras da frase em ordem aleatória. O algoritmo de SR vai de palavra em palavra na lista aleatória procurando sinônimos e substituindo na frase original. Caso o algoritmo não encontre sinônimos para a palavra selecionada, ele passa para o próximo índice da lista aleatória. O algoritmo para quando não há mais palavras na lista ou quando o número de palavras substituídas por sinônimos chega a N.

Já no caso da Troca Aleatória, o único cuidado a ser tomado é que o número de palavras no documento seja maior ou igual a dois.

Também foram feitos experimentos mesclando as técnicas. Para esses experimentos, as bases de dados foram aumentadas utilizando uma técnica de cada vez.

A ordem utilizada foi: $RD \rightarrow SR \rightarrow RS$. Ou seja, primeiramente as bases foram

aumentadas utilizando RD, a base resultante desta operação foi aumentada utilizando SR, e a base resultante dessas duas operações foi aumentada utilizando RS.

Dadas todas as possíveis permutações da ordem dos 3 aumentos, foi escolhida a ordem $RD \rightarrow SR \rightarrow RS$ baseada no seguinte raciocínio:

1. É imprudente posicionar o RD após o SR, pois é possível que a palavra substituída por seu sinônimo seja deletada, invalidando a operação SR.
2. É mais prudente apenas realizar a operação de RS após definir quais palavras irão compor a sentença, deixando essa operação para o final.
3. Não faz diferença intercalar SR ou RS, pois nenhuma das operações exclui ou adiciona palavras à lista.

Foi feita apenas uma versão com a mescla de aumentos para cada base de dados. Os valores de N para os aumentos de Remoção Aleatória e Substituição por Sinônimos foram definidos utilizando os experimentos com os aumentos simples. Foram escolhidos os valores de N que obtiveram os melhores resultados para a métrica F1 em cada base. Para as bases Fortuna e OffComBr-2, foi encontrado o valor $N = 1$ tanto para RD quanto para SR. No caso da base Fortuna Reduzida, foi encontrado o valor $N = 3$ também para as duas técnicas. A técnica RS não possui um parâmetro N para ser ajustado.

4.2 Arquiteturas

Para a realização da tarefa de detecção de discurso de ódio, foram avaliadas cinco diferentes arquiteturas profundas como classificadores. Entretanto, devido ao conjunto limitado de dados e, para evitar *overfitting*, as redes possuem um número reduzido de camadas, se comparadas com modelos atuais de redes profundas. Além dos classificadores mais simples CNN e LSTM, foram utilizadas como classificadores a rede CNN paralela e as redes híbridas CNN-LSTM e LSTM-CNN.

4.2.1 Camada de *Embedding*

Em todos os experimentos, foi utilizada uma camada de *embedding* para codificar os textos em vetores numéricos. Foram escolhidos vetores do tipo GloVe pré-treinados. Essa escolha foi feita pois os *embeddings* pré-treinados por HARTMANN et al. foram treinados em um volume de dados muito superior ao das bases de dados disponíveis.

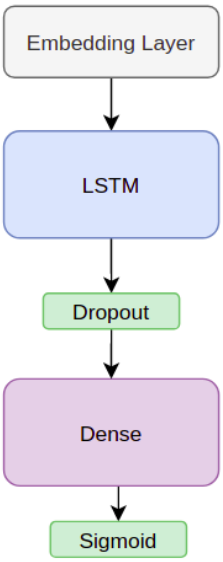
Os *embeddings* utilizados eram todos de dimensão 50 e os pesos da camada não foram fixados, significando que esses valores também foram ajustados durante o treinamento.

Essas configurações foram mantidas para todos os casos, evitando criar mais variáveis.

4.2.2 LSTM

A primeira arquitetura é uma rede LSTM simples. A sua configuração é mostrada na Figura 11. A primeira camada da rede é a camada de *embedding*, seguida por uma camada LSTM simples. Esta camada está configurada para retornar apenas a saída correspondente à última palavra, porém, esta saída contém informações sobre as palavras anteriores. O vetor de saída passa por uma operação de *dropout* e por fim passa por uma camada completamente conectada (também chamada de camada densa), sendo que essa camada final possui a função de ativação sigmoidal.

Figura 11 – Configuração da rede LSTM



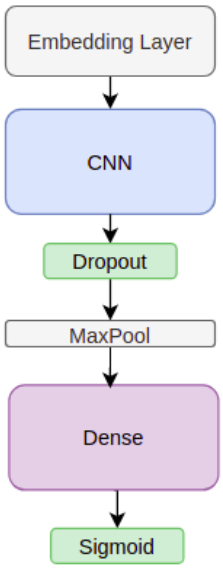
Fonte: A autora.

4.2.3 CNN

A primeira rede CNN utilizada também possui uma configuração simples, e é constituída por uma camada de *embedding*, seguida por uma camada CNN 1D com função de ativação *ReLU*. Não é utilizado *padding*, considerando que não há necessidade de manter o formato dos dados e evitando criar valores inexistentes nos dados.

A saída da rede convolucional passa por uma camada de *dropout* e por uma camada de *max pooling* com tamanho 3. Por fim, os vetores passam por uma camada completamente conectada, com função de ativação sigmoide. Essa configuração é mostrada na Figura 12

Figura 12 – Configuração da rede CNN



Fonte: A autora.

4.2.4 PCNN

A segunda configuração utilizada para a rede convolucional é uma adaptação da rede utilizada por Badjatiya et al. (2017). Este trabalho se refere a ela como CNN Paralela, ou simplesmente PCNN.

Nesta configuração são utilizadas duas camadas CNN em ramos paralelos, como mostrado na Figura 14. Nessa rede, as dimensões dos kernels nas camadas convolucionais em paralelo foram fixadas em 3 e 4, conforme ilustrados na figura.

As saídas passam por uma operação de *global max pooling* e depois são concatenadas. Por fim, os vetores passam por uma camada de *dropout* antes de irem para a camada completamente conectada com função de ativação sigmoide.

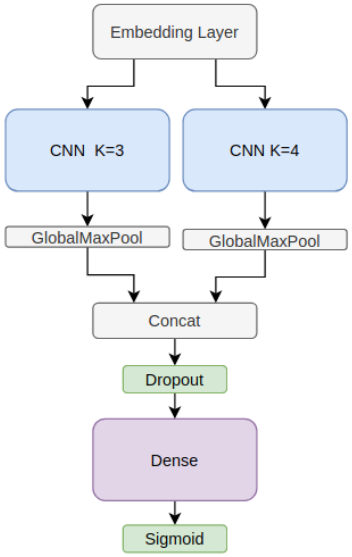
4.2.5 PCNN-LSTM

A rede CNN-LSTM é a uma das redes híbridas utilizada nesse trabalho. A rede é formada por dois blocos principais. O primeiro é um bloco com camadas CNN similar à rede PCNN exposta anteriormente, e o segundo bloco é uma camada LSTM.

Os dados codificados pela camada de *embedding* são processados por duas camadas CNN em ramos paralelos, com kernels de tamanho 3 e 4, conforme ilustrados na figura.

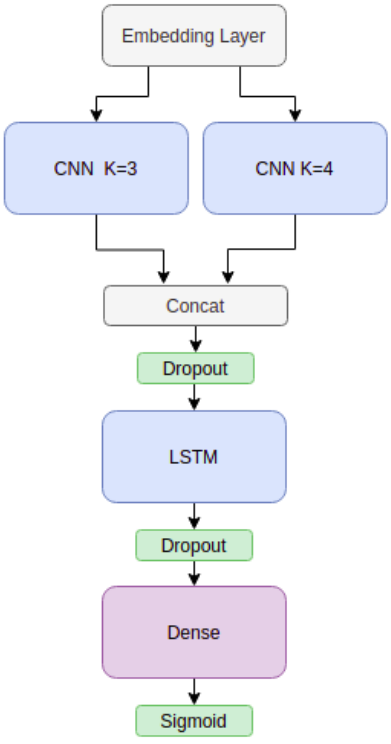
As saídas das camadas CNN são conectadas sem passar por uma operação de *global max pooling*. O vetor resultante serve como entrada para a camada recursiva LSTM. Assim como na rede LSTM simples, apenas é utilizado o estado de saída da rede relativo à última palavra. O vetor de saída é encaminhado para a camada completamente conectada

Figura 13 – Configuração da rede PCNN



Fonte: A autora.

Figura 14 – Configuração da rede PCNN-LSTM



Fonte: A autora.

4.2.6 LSTM-PCNN

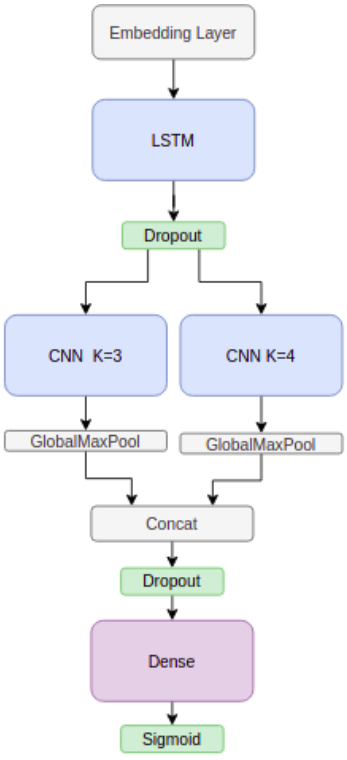
A Figura 15 mostra a configuração da rede LSTM-PCNN. Nessa configuração, inicialmente os dados passam pela rede recursiva, onde espera-se que as informações temporais sejam reconhecidas. A camada recursiva é seguida das camadas convolucionais em paralelo, onde espera-se que sejam reconhecidos padrões locais.

No caso da rede LSTM-PCNN, a saída da rede LSTM inclui a saída de todos os passos da dimensão temporal, e não apenas do último passo. Nesse caso, a saída da rede LSTM tem formato $n \times L$, onde L é o tamanho da saída da camada LSTM e n a quantidade de termos no documento.

Os kernels da rede CNN serão de tamanho $k \times L$, pois a dimensão L substitui o D relativo à dimensão do *embedding*, explicado no Capítulo 3.

As saídas das camadas convolucionais passam por uma operação de *global max pooling* e são concatenadas. Por fim, as informações seguem para a camada completamente conectada com saída sigmoide, que retornará como saída a classe do documento.

Figura 15 – Configuração da rede LSTM-CNN



Fonte: A autora.

4.2.7 Configurações Gerais

Os hiperparâmetros das redes foram ajustados utilizando a otimização bayesiana (SNOEK et al., 2012). Para isso, foi aplicada a função `BayesianOptimization` do módulo

`bayes_opt` para Python. A otimização foi feita para cada base de dados separadamente, e para cada arquitetura. Para esse processo, foram utilizadas as bases de dados sem aumento, e foi feita uma divisão simples 70/30 para os conjuntos de treino e teste. Não foi utilizado o procedimento de k-fold para a otimização. Os espaços de busca para cada hiperparâmetro foram os seguintes:

- Dropout: $0 - 0,5$
- Taxa de Aprendizado: $0 - 0,1$
- Tamanho LSTM: $5 - 50$
- Número de kernels: $5 - 50$
- Tamanho dos kernels: $2 - 6$

Nas Tabelas 5 a 9 estão expostos os valores finais dos hiperparâmetros para cada arquitetura e para cada base de dados. Esses valores foram utilizados em todos os experimentos. Nas tabelas abaixo, o termo `t.a.` se refere à taxa de aprendizado.

Tabela 5 – Configurações para LSTM

LSTM	t.a.	Dropout	Tamanho LSTM
Fortuna	0,025	0,4	45
Fortuna Reduzido	0,001	0,125	35
OffCom	0,03	0,1	25

Tabela 6 – Configurações para CNN

CNN	t.a.	Dropout	Nº de kernels	Tamanho dos kernels
Fortuna	0,02	0,15	20	2
Fortuna Reduzido	0,015	0	15	2
OffCom	0,03	0,15	30	5

Tabela 7 – Configurações para PCNN

PCNN	t.a.	Dropout	Nº kernels
Fortuna	0,025	0,3	35
Fortuna Reduzido	0,0075	0,2	15
OffCom	0,015	0,0	45

Alguns hiperparâmetros foram adotados para todos os experimentos:

- Otimizador: Adam
- Tamanho do batch: 64

Tabela 8 – Configurações para PCNN - LSTM

PCNN - LSTM	t.a.	Dropout	Nº de kernels	Tamanho LSTM
Fortuna	0,035	0,5	8	50
Fortuna Reduzido	0,002	0,15	40	40
OffCom	0,02	0,5	30	15

Tabela 9 – Configurações para LSTM - PCNN

LSTM - PCNN	t.a.	Dropout	Nº de kernels	Tamanho LSTM
Fortuna	0,07	0,3	10	16
Fortuna Reduzido	0,01	0,2	12	12
OffCom	0,02	0,1	10	20

Tabela 10 – Resultados de testes com função de perda ponderada e função de perda simples

	Aurácia	Precisão	Recall	F1
Ponderada	0.699	0.729	0.699	0.706
Simples	0.724	0.721	0.724	0.717

- Função de perda: entropia cruzada binária
- Máximo de palavras por documento: 40

Considerou-se utilizar a função de perda entropia cruzada binária ponderada, considerando que as duas bases de dados utilizadas são desbalanceadas. Porém, após testes preliminares com a rede LSTM, demonstrados na Tabela 10, observou-se que a função de perda entropia cruzada simples apresentava melhores resultados, além de ser mais fácil de configurar.

O máximo de palavras se deu pela análise dos histogramas das bases de dados, mostrados na Figura 9. Caso o documento possua mais do que 40 palavras, as últimas palavras serão excluídas. Não foi utilizado um valor maior, pois como a maior parte dos documentos possui um número de palavras muito inferior a 40, haveria processamento de muitos vetores vazios nas redes neurais.

5 Experimentos e Resultados

5.1 Métricas

As métricas de avaliação dos modelos foram estabelecidas considerando que o problema lidado é um problema de classificação binária.

Considere um conjunto de n exemplos $X = \{x_1, x_2, ..., x_n\}$, onde cada elemento x_i possui um rótulo y_i. Cada amostra classificada pode ser encaixada em uma das quatro categorias expostas na matriz de confusão da Tabela 11, onde Verdadeiros Positivos (*True Positive* - TP) e Verdadeiros Negativos (*True Negative* - TN) são amostras corretamente classificadas como positivas e negativas, respectivamente. Por outro lado, Falsos Positivos (*False Positive* - FP) e Falsos Negativos (*False Negative* - FN) são amostras erroneamente classificadas como positivas e negativas, respectivamente.

É possível aglomerar as amostras nessas quatro categorias e contar quantas amostras se encaixam em cada. As métricas utilizadas para avaliação nesse trabalho são baseadas na quantidade total de amostras enquadradas em cada uma das categorias.

5.1.1 Acurácia

A acurácia (ACC) é uma das métricas mais simples e também uma das mais utilizadas na literatura. Ela é calculada conforme a Equação 5.1, e seu valor está no intervalo entre 0% e 100%, sendo quanto maior o valor, melhor.

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \times 100\% \tag{5.1}$$

A acurácia, porém, não é uma métrica adequada para avaliar o desempenho sobre bases de dados desbalanceadas. Se uma base de dados, por exemplo, possui 95% das amostras da classe positiva e 5% das amostras da classe negativa, um classificador que

Tabela 11 – Exemplo de matriz de confusão para classificação binária

		Predito	
		Positivos 1	Negativos 0
Real	Positivos 1	Verdadeiros Positivos TP	Falsos Negativos FN
	Negativos 0	Falsos Positivos FP	Verdadeiros Negativos TN

identifique todas as amostras como positivas terá uma acurácia de 95%, o que supostamente é um bom desempenho de classificação.

5.1.2 Precisão

A precisão (Prec) mede, dentre todas as amostras classificadas como positivas pelo modelo, quantas são realmente positivas. Para o caso de detecção de discurso de ódio, a precisão é importante para avaliar a confiabilidade de classificação do modelo quando ele classifica um texto como discurso de ódio. Uma taxa baixa de precisão pode levar a muitos textos serem censurados sem motivo. A Equação 5.2 mostra o cálculo da métrica, e seu valor está no intervalo entre 0% e 100%, sendo quanto maior o valor, melhor.

$$Prec = \frac{TP}{TP + FP} \times 100\% \tag{5.2}$$

5.1.3 Sensibilidade (*Recall*)

A sensibilidade, ou *recall* (Rec), avalia quantas das amostras verdadeiramente positivas estão sendo classificadas como positivas pelo modelo. Ela é uma métrica relevante, pois informa a capacidade do modelo de conseguir detectar todos os discursos de ódio existentes na base de dados. Uma sensibilidade baixa indica que uma quantidade alta de amostras positivas não estão sendo identificadas. No caso da detecção de discurso de ódio, isso significa que textos com conteúdo danoso ou ofensivo continuariam circulando sem ser detectados. A Equação 5.2 mostra o cálculo da métrica, e seu valor está no intervalo entre 0% e 100%, sendo quanto maior o valor, melhor.

$$Rec = \frac{TP}{TP + FN} \times 100\% \tag{5.3}$$

5.1.4 *F1-score*

A métrica F1 é utilizada para balancear a importância entre precisão e sensibilidade. Ela é calculada conforme a Equação 5.4. Conforme visto no Capítulo 2, essa métrica é frequentemente utilizada em trabalhos para avaliar o desempenho de classificadores para a tarefa de detecção de discurso de ódio ou linguagem ofensiva. O valor de F1 está no intervalo entre 0% e 100%, sendo quanto maior o valor, melhor.

$$F1 = \frac{2 \times Prec \times Rec}{Prec + Rec} \times 100\% \tag{5.4}$$

5.2 Procedimentos Experimentais

Para os experimentos foi utilizado o método de validação cruzada *k-fold*. A técnica consiste em dividir a base de dados em k partes e realizar k iterações de treino/teste. A cada iteração, uma diferente parte da base de dados é selecionada para teste e as outras $k-1$ são utilizadas para o treino. Desta forma, todas as partes da base de dados podem ser utilizadas, tanto para teste quanto para treino. Neste trabalho foi utilizado $k = 5$.

Durante o treinamento é selecionado o modelo que apresenta o melhor resultado com respeito a métrica F1. Os modelos são treinados extensivamente por uma grande quantidade de épocas. A cada época, são salvos os pesos das camadas, e é feita a avaliação sobre o conjunto de validação. Por fim, os pesos que obtiveram melhor resultado no conjunto de validação são escolhidos para o teste. Durante os experimentos foi verificado que 20 épocas é um número suficiente para treinar os modelos até a saturação.

5.3 Recursos

Para os experimentos realizados nesse trabalho, foram utilizados os computadores disponíveis no laboratório Cisne, da Ufes. Mais especificamente, foi utilizada uma máquina com as seguintes configurações:

- Ubuntu 18.04 LTS
- CPU Intel(R) Core(TM) i9-9900KF 3.60GHz
- 64 Gb de memória RAM
- GPU Nvidia TITAN V com 12 Gb de memória dedicada

Toda a implementação foi feita em linguagem Python, incluindo as etapas de pré-processamento, os algoritmos de aumentos de dados e os modelos de redes neurais profundas. Para os modelos de redes neurais foi utilizado o *framework* Keras, com base em TensorFlow.

5.4 Resultados

Foram feitos experimentos com as bases aumentadas e com as bases sem aumento. Os resultados das métricas estão expostos nas Tabelas 12 a 15, para a base de dados Fortuna, e nas Tabelas 16 a 19 para a base de dados OffComBr-2. Os aumentos estão indicados como `del`, para a remoção aleatória, `swap`, para a troca aleatória, `syn`, para a substituição por sinônimos e `mix` para as bases alteradas pelos 3 tipos de aumento. O termo `original` se refere ao uso da base de dados sem aumento. O número após `del`

e `syn` indica o parâmetro N utilizado para cada um dos algoritmos. Dessa forma, `del2` indica os resultados para experimentos executados sobre a base aumentada pelo método de remoção aleatória com $N = 2$, ou seja, dois documentos novos criados para cada documento original. Além disto, o melhor resultado para cada modelo neural está em negrito e o melhor resultado em cada métrica está sublinhado.

5.4.1 Resultados na Base Fortuna

Tabela 12 – Valores de Acurácia obtidos para Base Fortuna

ACC	Original	del1	del2	del3	swap	syn1	syn2	syn3	mix
CNN	69,89	**70,61**	68,26	68,15	68,33	68,79	68,70	69,09	68,03
LSTM	**74,18**	69,86	69,58	68,90	69,51	69,63	69,31	69,47	67,23
PCNN	**73,28**	69,84	69,87	68,34	71,82	70,66	70,30	68,98	67,77
PCNN-LSTM	70,35	70,20	69,46	70,12	70,15	**71,10**	69,90	69,62	69,59
LSTM-PCNN	69,67	69,73	69,38	68,55	68,62	69,67	**69,84**	68,73	68,26

Tabela 13 – Valores F1 obtidos para Base Fortuna

F1	Original	del1	del2	del3	swap	syn1	syn2	syn3	mix
CNN	68,50	**70,26**	68,00	67,69	68,17	67,94	67,98	68,16	66,63
LSTM	**72,89**	69,37	69,42	68,84	69,03	69,03	68,64	68,73	66,96
PCNN	**71,28**	69,10	69,12	67,63	70,83	70,34	69,48	67,98	66,78
PCNN-LSTM	67,55	66,69	65,09	66,09	66,8	**68,73**	66,39	65,54	64,88
LSTM-PCNN	66,64	**68,11**	67,38	66,36	67,01	67,54	67,58	67,04	67,05

Tabela 14 – Valores de Precisão obtidos para Base Fortuna

Prec	Original	del1	del2	del3	swap	syn1	syn2	syn3	mix
CNN	69,07	**70,62**	68,06	67,71	68,13	68,05	68,07	68,09	66,50
LSTM	**73,14**	69,22	69,57	68,94	69,11	69,68	68,48	68,50	67,21
PCNN	**72,77**	69,34	68,93	67,45	70,77	70,48	69,34	68,10	66,95
PCNN-LSTM	68,11	68,17	66,88	67,93	67,79	**69,18**	67,33	67,52	66,66
LSTM-PCNN	66,05	**68,24**	67,52	66,14	67,18	67,94	67,73	66,92	66,70

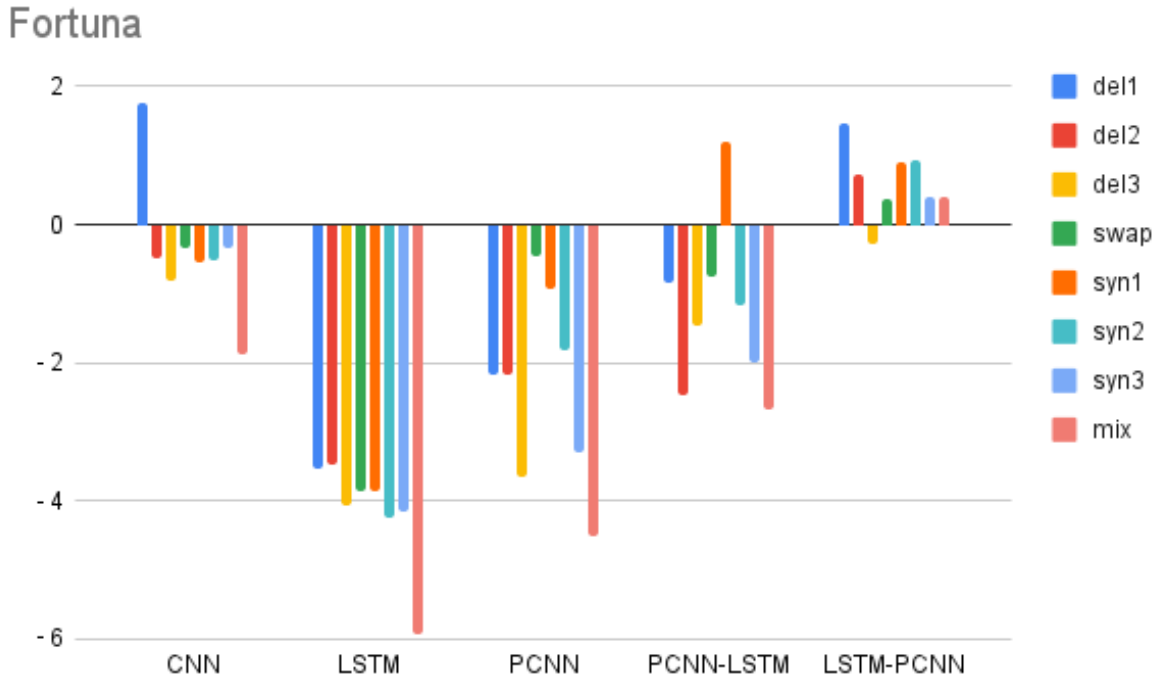
Tabela 15 – Valores de Sensibilidade (*Recall*) obtidos para Base Fortuna

Rec	Original	del1	del2	del3	swap	syn1	syn2	syn3	mix
CNN	69,89	**70,61**	68,26	68,15	68,33	68,79	68,7	69,09	68,03
LSTM	**74,18**	69,86	69,58	68,90	69,51	69,63	69,31	69,47	67,23
PCNN	**73,28**	69,84	69,87	68,34	71,82	70,66	70,3	68,98	67,77
PCNN-LSTM	70,35	70,20	69,46	70,12	70,15	**71,10**	69,90	69,62	69,59
LSTM-PCNN	69,67	69,73	69,38	68,55	68,62	69,67	**69,84**	68,73	68,26

A Figura 16 mostra o ganho na métrica F1 para a base de dados Fortuna para cada uma das técnicas de aumento aplicadas, onde no eixo da abcissa estão os modelos e os

tipos de aumento de dados e no eixo da ordenada está o ganho em porcentagem em relação ao resultado obtido com a base de dados original. É possível perceber que para a maioria dos casos o ganho foi negativo, ou seja, houve uma piora nos resultados, principalmente para a rede LSTM.

Figura 16 – Aumento da métrica F1 para Base Fortuna



Fonte: A autora.

5.4.2 Resultados na Base OffComBr-2

Tabela 16 – Valores de Acurácia obtidos para Base OffComBr-2

ACC	Original	del1	del2	del3	swap	syn1	syn2	syn3	mix
CNN	**76,80**	74,40	74,76	74,13	76,40	74,22	73,16	73,91	74,00
LSTM	78,04	77,56	78,71	78,09	**79,51**	78,04	77,96	77,47	77,64
PCNN	78,53	79,02	79,73	78,53	79,42	**80,04**	79,20	**80,04**	77,91
PCNN-LSTM	76,09	75,56	75,87	74,80	76,76	75,64	76,04	**76,93**	72,76
LSTM-PCNN	77,16	78,67	78,18	77,73	78,44	**78,76**	77,42	**78,76**	77,07

A Figura 17 mostra os ganhos para cada técnica de aumento aplicada à base OffComBr-2. Diferentemente dos experimentos com a base Fortuna, é possível perceber que para a maioria dos casos há algum ganho. Os ganhos são mais expressivos para a rede PCNN, em que todos os aumentos aplicados resultaram em um ganho por volta de 2%, com exceção da versão com mescla de aumentos, que resultou em um ganho irrisório.

Tabela 17 – Valores F1 obtidos para Base OffComBr-2

F1	Original	del1	del2	del3	swap	syn1	syn2	syn3	mix
CNN	**75,11**	74,63	74,40	73,33	73,41	74,25	73,57	**75,18**	73,53
LSTM	77,48	78,12	77,83	77,50	**78,61**	77,68	77,46	77,60	77,30
PCNN	77,36	79,83	79,18	79,17	78,72	79,10	**79,99**	79,79	77,60
PCNN-LSTM	72,14	72,31	**74,09**	71,81	73,34	71,15	72,28	72,80	72,24
LSTM-PCNN	77,83	78,62	78,54	77,43	78,48	**78,85**	77,34	77,35	76,68

Tabela 18 – Valores de Precisão obtidos para Base OffComBr-2

Prec	Original	del1	del2	del3	swap	syn1	syn2	syn3	mix
CNN	**76,61**	74,26	74,4	73,86	76,27	73,94	72,76	73,52	73,75
LSTM	78,01	77,33	**78,69**	78,09	79,32	78,18	78,05	77,33	77,67
PCNN	78,65	79,03	79,69	78,5	79,3	**79,88**	78,96	79,87	77,70
PCNN-LSTM	75,96	75,24	75,6	74,62	76,91	75,54	75,72	**76,78**	72,55
LSTM-PCNN	77,01	78,50	78,28	77,67	78,36	78,59	77,25	**78,91**	76,82

Tabela 19 – Valores de Sensibilidade (*Recall*) obtidos para Base OffComBr-2

Rec	Original	del1	del2	del3	swap	syn1	syn2	syn3	mix
CNN	**76,80**	74,40	74,76	74,13	76,40	74,22	73,16	73,91	74,00
LSTM	78,04	77,56	**78,71**	78,09	79,51	78,04	77,96	77,47	77,64
PCNN	78,53	79,02	79,73	78,53	79,42	**80,04**	79,20	**80,04**	77,91
PCNN-LSTM	76,09	75,56	75,87	74,80	76,76	75,64	76,04	**76,93**	72,76
LSTM-PCNN	77,16	78,67	78,18	77,73	78,44	**78,76**	77,42	**78,76**	77,07

Apesar dos aumentos não obterem um resultado positivo para todos os casos com a base de dados OffComBr-2, é possível notar uma diferença entre os resultados da aplicação dos aumentos para as duas bases de dados. Um dos possíveis motivos para essa discrepância pode ser o tamanho original das bases. A base OffComBr-2 possui apenas 1.250 documentos, enquanto a base Fortuna possui mais de 5.000 documentos.

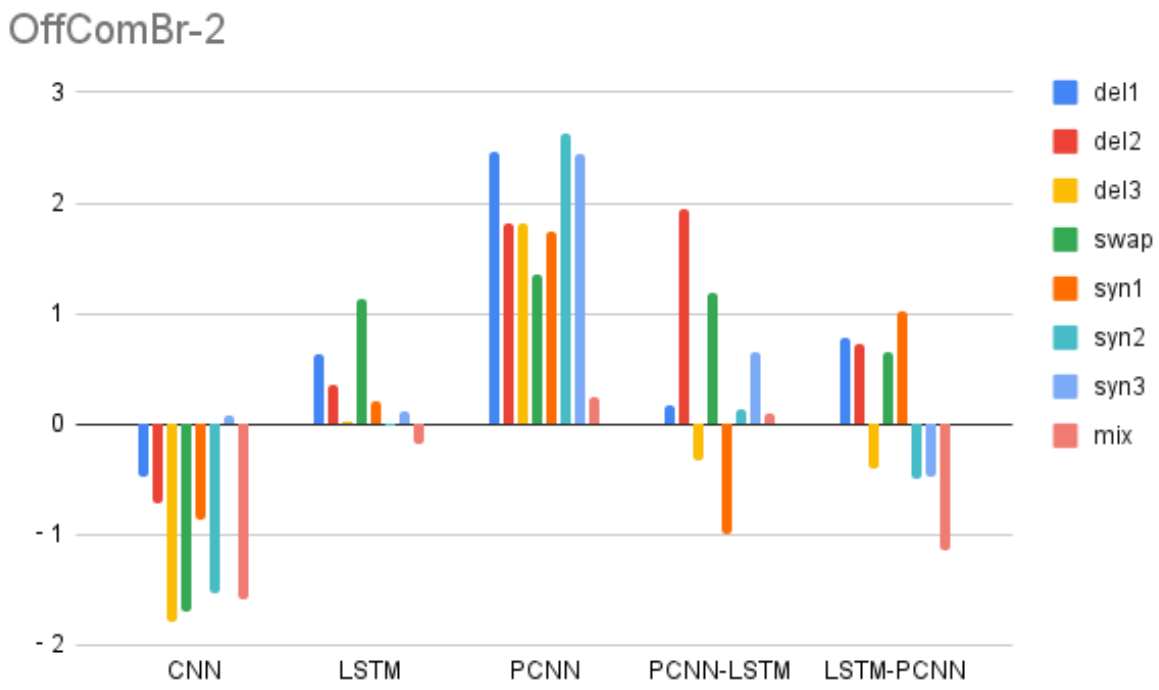
5.4.3 Resultados na Base Fortuna Reduzida

Para averiguar a possibilidade citada anteriormente, foi criada a base Fortuna Reduzida, como explicada no Capítulo 4. A base Fortuna Reduzida possui 25% dos documentos originais da base Fortuna, selecionados aleatoriamente, porém de maneira a manter a proporção entre classes. Sendo assim, a base Fortuna Reduzida possui 1.417 documentos, uma quantidade mais próxima da base OffComBr-2 com 1.250 documentos. As Tabelas 20 a 23 expõem os resultados obtidos nos experimentos com a base Fortuna Reduzida para as métricas escolhidas.

A Figura 18 mostra os ganhos para cada técnica de aumento aplicada à base Fortuna Reduzida.

Para o caso da base Fortuna Reduzida, todas as técnicas de aumento resultaram em

Figura 17 – Aumento da métrica F1 para Base OffComBr-2



Fonte: A autora.

Tabela 20 – Valores de Acurácia obtidos para Base Fortuna Reduzida

ACC	Original	del1	del2	del3	swap	syn1	syn2	syn3	mix
CNN	68,21	67,97	69,46	68,80	**70,48**	69,34	68,91	69,5	67,86
LSTM	65,39	68,72	69,19	**70,99**	68,56	67,89	69,03	70,05	68,01
PCNN	69,73	70,40	71,22	**71,85**	70,40	70,72	70,87	71,66	69,54
PCNN-LSTM	68,09	68,68	70,64	70,91	69,03	69,54	**71,34**	70,13	68,44
LSTM-PCNN	68,76	70,13	69,03	68,60	69,46	**70,40**	69,19	69,07	67,93

Tabela 21 – Valores F1 obtidos para Base Fortuna Reduzida

F1	Original	del1	del2	del3	swap	syn1	syn2	syn3	mix
CNN	61,23	65,54	**68,51**	67,90	68,09	66,69	67,81	68,27	67,61
LSTM	52,72	62,30	64,33	**67,54**	61,38	61,00	64,96	65,92	67,18
PCNN	65,90	67,87	68,81	**70,46**	67,41	68,14	69,25	70,09	67,50
PCNN-LSTM	60,49	61,91	67,77	69,09	65,47	65,44	**69,39**	68,62	67,20
LSTM-PCNN	61,56	**68,52**	67,88	67,70	67,94	68,34	67,62	67,83	66,45

um ganho de F1, em alguns casos chegando à um ganho de quase 15%. Porém é importante notar que os resultados com a base original são baixos se comparados às outras bases. Isso pode ser um indicativo que a base reduzida é muito pequena para treinar os modelos propostos, sendo assim o aumento contribui significativamente.

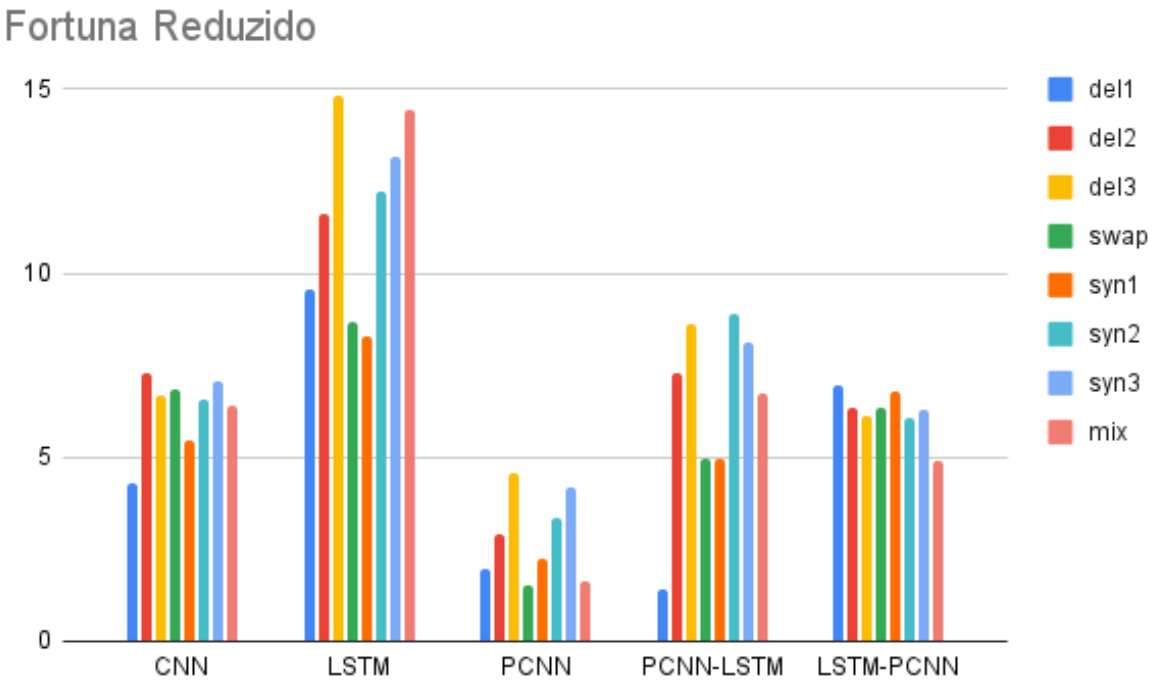
Tabela 22 – Valores de Precisão obtidos para Base Fortuna Reduzida

Prec	Original	del1	del2	del3	swap	syn1	syn2	syn3	mix
CNN	68,78	66,34	69,01	67,74	**69,68**	67,91	67,74	68,42	67,52
LSTM	60,34	69,01	68,71	**70,31**	67,33	67,55	68,08	70,17	67,06
PCNN	69,34	69,39	70,65	**70,95**	69,53	69,94	69,74	70,62	68,32
PCNN-LSTM	68,94	69,21	69,87	70,06	68,91	69,07	**70,45**	68,96	67,33
LSTM-PCNN	67,44	69,01	67,95	67,62	68,52	**69,58**	67,91	68,05	66,60

Tabela 23 – Valores de Sensibilidade (*Recall*) obtidos para Base Fortuna Reduzida

Rec	Original	del1	del2	del3	swap	syn1	syn2	syn3	mix
CNN	68,21	67,97	69,46	68,80	**70,48**	69,34	68,91	69,50	67,86
LSTM	65,39	68,72	69,19	**70,99**	68,56	67,89	69,03	70,05	68,01
PCNN	69,73	70,40	71,22	**71,85**	70,40	70,72	70,87	71,66	69,54
PCNN-LSTM	68,09	68,68	70,64	70,91	69,03	69,54	**71,34**	70,13	68,44
LSTM-PCNN	68,76	70,13	69,03	68,60	69,46	**70,40**	69,19	69,07	67,93

Figura 18 – Aumento da métrica F1 para Base Fortuna Reduzida



Fonte: A autora.

5.5 Análises Gerais

Uma primeira análise é feita em relação às arquiteturas utilizadas: quais alcançaram melhor desempenho na tarefa de detecção de discurso de ódio e sob quais circunstâncias elas apresentaram esses resultados.

A primeira observação a ser feita é de que a rede CNN simples tem resultados inferiores à rede PCNN em quase todos os casos. É possível afirmar com certa segurança que a rede com ramos paralelos possui melhor desempenho do que a rede simples para a tarefa abordada. Apesar da complexidade da rede não ser muito mais elevada do que a rede CNN simples, o aumento nos resultados é consistente.

Outro ponto a ser notado é que a rede PCNN apresentou os melhores resultados, tanto para a base OffComBr-2 quanto para a base Fortuna Reduzida, e que para a base Fortuna original a rede PCNN apresentou resultados apenas levemente piores do que a rede LSTM, que obteve os melhores resultados. Até onde se tem conhecimento, a rede PCNN não havia sido testada anteriormente em nenhuma das bases de dados para discurso de ódio ou linguagem ofensiva em português.

A rede LSTM obteve o desempenho esperado para a base Fortuna, se igualando ao desempenho apresentado pela rede em outros trabalhos que utilizaram a mesma base, mencionados no Capítulo 2. Já as redes híbridas PCNN-LSTM e LSTM-PCNN não obtiveram um resultado superior às redes simples.

A próxima análise é feita em relação ao efeito das bases aumentadas para cada arquitetura.

É possível observar que praticamente não houve ganho para a base Fortuna e que as técnicas de aumento aplicadas levaram à uma piora no desempenho de todas as redes, com exceção da rede mista LSTM-PCNN, que teve um ganho, porém pouco expressivo.

Já para a base OffComBr-2, houve um acréscimo no desempenho da rede PCNN com todos os aumentos e, com exceção da rede CNN, os aumentos levaram a resultados melhores do que na rede Fortuna. Considerou-se que talvez essa diferença fosse causada pela discrepância no tamanho de cada base, dado que a base Fortuna é mais de quatro vezes maior do que a base OffComBr-2, e isto pode ter afetado no treinamento adequado dos parâmetros dos modelos.

Sendo assim, foi criada a base Fortuna Reduzida, com o tamanho mais próximo ao da base OffComBr-2 e mantendo a proporção entre classes da base Fortuna original. Os resultados sem aumento dessa base foram inferiores ao da base Fortuna, porém, a utilização das técnicas de aumento trouxe resultados positivos para todas as arquiteturas e com todas as técnicas aplicadas.

Inclusive, com a utilização das técnicas de aumento, o melhor resultado da base reduzida aproximou-se do melhor resultado da base original: para a métrica F1 a base Fortuna original alcançou um resultado de 72,89% com a rede LSTM sem aumento, enquanto a base reduzida alcançou o resultado de 71,85% para a rede PCNN com aumento de Remoção Aleatória e $N = 3$.

As técnicas de aumento utilizadas parecem apresentar melhores resultados para

bases de tamanho reduzido, enquanto pioram o desempenho de bases maiores.

Uma terceira análise é feita sobre a utilização do aumento acumulado, que une as três técnicas de aumento básicas exploradas: Remoção Aleatória, Substituição por Sinônimos e Troca Aleatória. A técnica com aumentos acumulados, indicada como `mix`, parece não trazer benefícios em praticamente nenhum dos casos, quando comparada às outras técnicas de aumento de dados aplicadas. Nas bases Fortuna e OffComBr-2, a técnica com aumentos acumulados não trás nenhum ganho conclusivo, e na base Fortuna Reduzida, apesar de trazer ganhos quando comparada a base de dados sem aumentos, não possui nenhum destaque.

Com os dados obtidos é possível observar que as técnicas de aumento de dados podem trazer ganhos no desempenho das redes, principalmente quando aplicadas a bases de dados de tamanho restrito. Porém, não foi possível concluir qual a técnica mais eficaz dentre as aplicadas.

6 Conclusão e Trabalhos Futuros

Esse trabalho teve como objetivo propor novas técnicas para a detectar de discurso de ódio em textos de redes sociais em português utilizando para isso redes neurais profundas. Para isso, inicialmente foram analisados os problemas envolvendo essa tarefa, e foram propostas possíveis soluções.

Primeiramente foi limitado o escopo de redes profundas utilizadas, e ficou decidido que o foco seria em redes convolucionais, CNN, e redes recursivas, no caso foi escolhida a LSTM.

Foi proposta a utilização de duas redes híbridas, a PCNN-LSTM e a LSTM-PCNN. Apesar de redes híbridas do tipo CNN-LSTM já terem sido utilizadas para essa tarefa, não foi encontrado outros trabalhos com redes híbridas em que a etapa convolucional trabalhe com ramos paralelos. Também não foram encontrados trabalhos para a detecção de discurso de ódio que usassem arquiteturas em que a etapa recursiva antecedesse a convolucional.

Durante a etapa inicial do trabalho foi identificado que um dos problemas para a detecção automática de discurso de ódio em português é a escassez de bases de dados rotuladas e o pequeno tamanho das bases existentes. As únicas bases encontradas no início do trabalho foram as bases de Fortuna et al. (2019), com 5.668 documentos extraídos da rede social Twitter, e de Pelle e Moreira (2017), com 1.250 documentos retirados da seção de comentários da página online do jornal G1. Sendo assim, foi proposta a utilização de técnicas de aumento de dados para textos, com o intuito de amenizar esse problema.

Foram selecionadas 3 técnicas inicialmente apresentadas por Wei e Zou (2019). Utilizando essas três técnicas, foram criadas versões aumentadas das bases de dados encontradas. As bases resultantes eram de tamanho 2, 3 ou 4 vezes o tamanho das bases originais.

As técnicas de aumento foram testadas para as cinco arquiteturas escolhidas: CNN, PCNN, LSTM, PCNN-LSTM e LSTM-PCNN. Após uma primeira análise dos resultados, percebeu-se que os aumentos não trouxeram ganhos para a base Fortuna. Pelo contrário, quase todos os resultados pioraram com o uso das técnicas de aumento. Os melhores resultados para a base Fortuna foram obtidos pelas rede LSTM e PCNN, sendo que a rede LSTM obteve o melhor resultado com a acurácia de $74,18\%$ e a métrica F1 chegando à $72,89\%$.

Porém, para a base OffComBr-2 foram obtidos resultados diferentes. Para a rede PCNN, por exemplo, todas as técnicas de aumento de dados aplicadas resultaram em

algum ganho, sendo que as técnicas de Remoção Aleatória e Substituição por Sinônimos resultaram em ganhos de mais de $1,5\%$ para a métrica F1, chegando a um ganho de $2,63\%$ para o aumento de Substituição por Sinônimos com $N = 2$. Também foram observados ganhos em desempenho em outras arquiteturas.

Devido à discrepância nos resultados das duas bases, foi cogitada a possibilidade de que as técnicas de aumentos seriam mais efetivas para bases de dados de tamanho menor. Para sanar essa dúvida, foi criada uma versão reduzida da base de dados Fortuna, com apenas 25% dos documentos originais.

As técnicas de aumento apresentaram resultado positivos com a base Fortuna Reduzida. A maioria das arquiteturas apresentaram ganhos de mais de 5% de F1 com a aplicação dos aumentos. Porém é relevante observar que o desempenho sem aumentos para essa base de dados foi relativamente baixo.

Por fim, foi aplicada uma técnica que acumula os 3 aumentos simples, porém em praticamente nenhum caso essa técnica mista obteve resultados superiores aos aumentos simples, com exceção da aplicação na rede LSTM para a base Fortuna Reduzida.

É possível concluir que modelos de arquitetura profunda treinados com bases de dados de tamanho muito reduzido podem ser beneficiados pelas técnicas de aumento de dados. Porém não foi possível concluir de forma consistente qual das técnicas trás melhores ganhos no desempenho.

É importante observar que as técnicas de aumento utilizadas nesse trabalho não garantem que o sentido da sentença resultante permaneça exatamente o mesmo. As técnica de Remoção Aleatória e Troca Aleatória podem, potencialmente, causar mudanças que alterem o sentido do texto. Apesar do risco ser menor com a técnica de Substituição por Sinônimos, quando é considerado que os dicionários de sinônimos não são formados por sinônimos perfeitos, essa técnica pode também modificar o sentido do texto, mesmo que de forma mais sutil.

Sendo assim, o uso dessas técnicas se baseia na premissa que o ganho causado pelos documentos alterados de maneira positiva compense as perdas causadas pelos documentos que tiveram seu sentido modificado. Um dos intuitos dos experimentos foi verificar se essa compensação existe.

Outra conclusão que pode ser retirada dos resultados é de que a rede convolucional com ramos paralelos (PCNN) apresentou melhor desempenho do que a rede convolucional simples (CNN). Já as redes híbridas não apresentaram resultados superiores à rede PCNN, que apresentou bom desempenho para as 3 bases.

Foi observado que os modelos propostos não alcançaram desempenhos elevados na detecção de discurso de ódio. Embora os tamanhos reduzidos das bases de dados tenham colaborado para isto, outros fatores, como erros de escrita, uso de gírias e abreviações,

conteúdo semântico do texto e falta de informação de contexto podem ter contribuído para os resultados obtidos. Posteriormente, também foi observado que o método de pré-processamento utilizado para o treinamento dos *word embeddings* por Hartmann et al. (2017) se difere do utilizado no pré-processamento deste trabalho, o que pode ter contribuído para a piora nos resultados.

Este trabalho focou no tamanho das bases, mostrando alguns resultados promissores, mas outros trabalhos podem ser realizados na busca de amenizar os demais fatores.

6.1 Trabalhos Futuros

Algumas questões foram deixadas sem respostas e podem ser abordadas em trabalhos futuros, além disso, algumas alternativas não foram testadas devido à limitação de escopo.

A primeira questão a ser respondida é se as redes híbridas produziriam melhores resultados se aplicadas à bases de dados naturalmente maiores.

Também é possível experimentar com variações da rede PCNN. Neste trabalho, a arquitetura foi fixada em 2 ramos e o tamanho dos kernels foi fixado em $\{3, 4\}$. Porém seria interessante testar se variações dessa arquitetura trariam melhores resultados.

Outra questão é se técnicas de aumento de dados mais sofisticadas, que busquem conservar o sentido do texto, trariam resultados melhores. Como foi explicado anteriormente, as técnicas aplicadas nesse trabalho podem potencialmente alterar o sentido dos documentos, o que pode limitar o ganho proporcionado.

Devido à limitação de tempo para o projeto não foi possível testar variações das técnicas de aumento aplicadas. Algumas ideias surgiram durante a execução do trabalho:

- Troca Aleatória: delimitar a distância máxima de troca de duas palavras. Por exemplo, só será trocada a posição de palavras que estejam dentro de uma janela de N palavras.
- Remoção aleatória: experimentar critérios para a remoção de palavras, evitando retirar palavras que possam alterar drasticamente o sentido da frase.

Outra possibilidade seria utilizar técnicas de *transfer learning* para contornar o problema da escassez de dados. Existem disponíveis modelos BERT (DEVLIN et al., 2018) pré-treinados para português que poderiam ser utilizados (SOUZA et al., 2020).

Além disso, não foi possível experimentar com a base de dados criada por Alves Vargas et al. (2021), pois a base de dados foi divulgada apenas no ano de 2021 e até a data final deste projeto não foi encontrada disponível online. Teria sido interessante realizar

experimentos com essa base de dados pois ela é uma base de tamanho significativamente maior do que as duas outras bases utilizadas, contendo 15.000 comentários retirados da rede social Instagram.

Referências

Alves Vargas, F. et al. Building an Expert Annotated Corpus of Brazilian Instagram Comments for Hate Speech and Offensive Language Detection. arXiv e-prints, p. arXiv:2103.14972, mar. 2021. Citado 4 vezes nas páginas 19, 20, 21 e 55.

BADJATIYA, P. et al. Deep learning for hate speech detection in tweets. In: INTERNATIONAL WORLD WIDE WEB CONFERENCES STEERING COMMITTEE. Proceedings of the 26th International Conference on World Wide Web Companion. [S.l.], 2017. p. 759–760. Citado 2 vezes nas páginas 18 e 38.

BANKS, J. Regulating hate speech online. International Review of Law, Computers & Technology, Taylor & Francis, v. 24, n. 3, p. 233–239, 2010. Citado na página 15.

BASKIN, C. et al. Streaming architecture for large-scale quantized neural networks on an fpga-based dataflow platform. In: IEEE. 2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW). [S.l.], 2018. p. 162–169. Citado na página 29.

BBC. Facebook, twitter and youtube face action over hate speech. BBC, Maio 2016. Acessado em 14/08/2021. Disponível em: <https://www.bbc.com/news/technology-36301772>. Citado 2 vezes nas páginas 12 e 15.

BUARQUE, B.; CRETTON, M. Mapa do ódio no Brasil 2019: Percepções e recomendações para políticas públicas. 2021. Citado na página 12.

CARCARá, T. A. Discurso do ódio no brasil: elementos de ódio na sociedade e sua compreensão jurídica. Rio de Janeiro: Lumen Juris, 2014. Citado na página 12.

CHETTY, N.; ALATHUR, S. Hate speech review in the context of online social networks. Aggression and violent behavior, Elsevier, v. 40, p. 108–118, 2018. Citado na página 12.

DALAL, M. K.; ZAVERI, M. A. Automatic text classification: a technical review. International Journal of Computer Applications, International Journal of Computer Applications, 244 5 th Avenue,# 1526, New . . . , v. 28, n. 2, p. 37–40, 2011. Citado na página 24.

DAVIDSON, T. et al. Automated hate speech detection and the problem of offensive language. In: Eleventh International AAAI Conference on Web and Social Media. [S.l.: s.n.], 2017. Citado na página 22.

DENG, L.; LIU, Y. Deep learning in natural language processing. [S.l.]: Springer, 2018. Citado na página 23.

DEVLIN, J. et al. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018. Citado na página 55.

DIAS-DA-SILVA, B. C.; MORAES, H. R. d. A construção de um thesaurus eletrônico para o português do brasil. ALFA: Revista de Linguística, Universidade Estadual Paulista (UNESP), 2003. Citado na página 34.

DROZD, A.; GLADKOVA, A.; MATSUOKA, S. Word embeddings, analogies, and machine learning: Beyond king-man+ woman= queen. In: Proceedings of coling 2016, the 26th international conference on computational linguistics: Technical papers. [S.l.: s.n.], 2016. p. 3519–3530. Citado na página 25.

FACEBOOK. Padrões da Comunidade: Conteúdo questionável. 2021. <https://www.facebook.com/communitystandards/objectionable_content>. Acessado em 14/09/2021. Citado 3 vezes nas páginas 12, 15 e 16.

FERREIRA, J.; Gonçalo Oliveira, H.; RODRIGUES, R. Improving NLTK for processing Portuguese. In: Symposium on Languages, Applications and Technologies (SLATE 2019). [S.l.: s.n.], 2019. In press. Citado na página 33.

FORTUNA, P. et al. A hierarchically-labeled portuguese hate speech dataset. In: Proceedings of the Third Workshop on Abusive Language Online. [S.l.: s.n.], 2019. p. 94–104. Citado 7 vezes nas páginas 17, 19, 20, 21, 22, 32 e 53.

GOLDBERG, Y. Neural network methods for natural language processing. Synthesis lectures on human language technologies, Morgan & Claypool Publishers, v. 10, n. 1, p. 1–309, 2017. Citado na página 26.

GOOGLE. Ajuda do YouTube: Política contra discurso de ódio. 2021. <https://support.google.com/youtube/answer/2801939?hl=pt-BR>. Acessado em 14/09/2021. Citado 2 vezes nas páginas 12 e 15.

GRAVES, A. Long short-term memory. In: Supervised sequence labelling with recurrent neural networks. [S.l.]: Springer, 2012. p. 37–45. Citado na página 27.

HARTMANN, N. et al. Portuguese word embeddings: Evaluating on word analogies and natural language tasks. arXiv preprint arXiv:1708.06025, 2017. Citado 3 vezes nas páginas 31, 36 e 55.

HOCHREITER, S.; SCHMIDHUBER, J. Long Short-Term Memory. Neural Computation, v. 9, n. 8, p. 1735–1780, 11 1997. ISSN 0899-7667. Disponível em: <https://doi.org/10.1162/neco.1997.9.8.1735>. Citado na página 26.

KIM, Y. Convolutional neural networks for sentence classification. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Doha, Qatar: Association for Computational Linguistics, 2014. p. 1746–1751. Disponível em: <https://www.aclweb.org/anthology/D14-1181>. Citado na página 28.

KUMAR, E. Natural language processing. [S.l.]: IK International Pvt Ltd, 2011. Citado na página 23.

LORENA, A. C.; CARVALHO, A. C. D.; GAMA, J. M. A review on the combination of binary classifiers in multiclass problems. Artificial Intelligence Review, Springer, v. 30, n. 1-4, p. 19, 2008. Citado na página 23.

MA, E. Data Augmentation in NLP: Introduction to Text Augmentation. 2019. Acessado em 10/07/2021. Disponível em: <https://towardsdatascience.com/data-augmentation-in-nlp-2801a34dfc28>. Citado 2 vezes nas páginas 30 e 31.

MACAVANEY, S. et al. Hate speech detection: Challenges and solutions. PloS one, Public Library of Science San Francisco, CA USA, v. 14, n. 8, p. e0221152, 2019. Citado 3 vezes nas páginas 15, 16 e 17.

MALMASI, S.; ZAMPIERI, M. Detecting hate speech in social media. In: Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017. Varna, Bulgaria: INCOMA Ltd., 2017. p. 467–472. Disponível em: <https://doi.org/10.26615/978-954-452-049-6_062>. Citado na página 19.

MIKOLOV, T. et al. Efficient estimation of word representations in vector space. In: BENGIO, Y.; LECUN, Y. (Ed.). 1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings. [s.n.], 2013. Disponível em: <http://arxiv.org/abs/1301.3781>. Citado na página 25.

MINAEE, S. et al. Deep learning–based text classification: A comprehensive review. ACM Comput. Surv., Association for Computing Machinery, New York, NY, USA, v. 54, n. 3, abr. 2021. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/3439726>. Citado na página 24.

MINAEE, S. et al. Deep learning–based text classification: A comprehensive review. ACM Comput. Surv., Association for Computing Machinery, New York, NY, USA, v. 54, n. 3, abr. 2021. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/3439726>. Citado na página 29.

OLAH, C. Understanding LSTM Networks. 2015. Acessado em 10/07/2021. Disponível em: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>. Citado 2 vezes nas páginas 27 e 28.

OMNICORE. Omnicore. 2021. <https://www.omnicoreagency.com/twitter-statistics/>. Acessado em 15/07/2021. Citado na página 12.

ONU. Declaração Universal dos Direitos Humanos. 1948. <https://www.unicef.org/brazil/declaracao-universal-dos-direitos-humanos>. Acessado em 08/08/2021. Citado na página 12.

PAIVA, P.; SILVA, V. da; MOURA, R. Detecção automática de discurso de ódio em comentários online. In: Anais da VII Escola Regional de Computação Aplicada à Saúde. Porto Alegre, RS, Brasil: SBC, 2019. p. 157–162. Disponível em: <https://sol.sbc.org.br/index.php/ercas/article/view/9052>. Citado na página 21.

PASCANU, R.; MIKOLOV, T.; BENGIO, Y. On the difficulty of training recurrent neural networks. In: PMLR. International conference on machine learning. [S.l.], 2013. p. 1310–1318. Citado na página 26.

PELLE, R. P. de; MOREIRA, V. P. Offensive comments in the brazilian web: a dataset and baseline results. In: SBC. Anais do VI Brazilian Workshop on Social Network Analysis and Mining. [S.l.], 2017. Citado 5 vezes nas páginas 19, 20, 21, 32 e 53.

PENNINGTON, J.; SOCHER, R.; MANNING, C. D. Glove: Global vectors for word representation. In: Empirical Methods in Natural Language Processing (EMNLP). [s.n.], 2014. p. 1532–1543. Disponível em: <http://www.aclweb.org/anthology/D14-1162>. Citado 2 vezes nas páginas 19 e 25.

PERRY, B.; OLSSON, P. Cyberhate: the globalization of hate. Information & Communications Technology Law, Routledge, v. 18, n. 2, p. 185–199, 2009. Disponível em: <https://doi.org/10.1080/13600830902814984>. Citado na página 12.

Ross, B. et al. Measuring the Reliability of Hate Speech Annotations: The Case of the European Refugee Crisis. arXiv e-prints, p. arXiv:1701.08118, jan. 2017. Citado na página 16.

SCOTT, S.; MATWIN, S. Feature engineering for text classification. In: CITESEER. ICML. [S.l.], 1999. v. 99, p. 379–388. Citado na página 23.

SILVA, A.; ROMAN, N. Hate speech detection in portuguese with naïve bayes, svm, mlp and logistic regression. In: Anais do XVII Encontro Nacional de Inteligência Artificial e Computacional. Porto Alegre, RS, Brasil: SBC, 2020. p. 1–12. ISSN 0000-0000. Disponível em: <https://sol.sbc.org.br/index.php/eniac/article/view/12112>. Citado na página 22.

SNOEK, J.; LAROCHELLE, H.; ADAMS, R. P. Practical bayesian optimization of machine learning algorithms. arXiv preprint arXiv:1206.2944, 2012. Citado na página 40.

SOUZA, F.; NOGUEIRA, R.; LOTUFO, R. Bertimbau: Pretrained bert models for brazilian portuguese. In: CERRI, R.; PRATI, R. C. (Ed.). Intelligent Systems. Cham: Springer International Publishing, 2020. p. 403–417. ISBN 978-3-030-61377-8. Citado na página 55.

SUTSKEVER, I. Training recurrent neural networks. [S.l.]: University of Toronto Toronto, Canada, 2013. Citado na página 26.

TWITTER. Política contra propagação de ódio. 2021. <https://help.twitter.com/pt/rules-and-policies/hateful-conduct-policy>. Acessado em 14/09/2021. Citado 2 vezes nas páginas 12 e 15.

WANG, W. Y.; YANG, D. That's so annoying!!!: A lexical and frame-semantic embedding based data augmentation approach to automatic categorization of annoying behaviors using #petpeeve tweets. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. Lisbon, Portugal: Association for Computational Linguistics, 2015. p. 2557–2563. Disponível em: <https://aclanthology.org/D15-1306>. Citado na página 31.

WASEEM, Z.; HOVY, D. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In: Proceedings of the NAACL Student Research Workshop. San Diego, California: Association for Computational Linguistics, 2016. p. 88–93. Disponível em: <http://www.aclweb.org/anthology/N16-2013>. Citado na página 18.

WEI, J.; ZOU, K. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. arXiv preprint arXiv:1901.11196, 2019. Citado 2 vezes nas páginas 30 e 53.

YIN, W. et al. Comparative study of cnn and rnn for natural language processing. arXiv preprint arXiv:1702.01923, 2017. Citado na página 23.

YOUNG, T. et al. Recent trends in deep learning based natural language processing. ieee Computational intelligenCe magazine, IEEE, v. 13, n. 3, p. 55–75, 2018. Citado na página 23.

A Trabalhos Publicados

Dois artigos derivados deste trabalho foram publicados em eventos:

- Venturott LI, Ciarelli PM. Application of Data Augmentation Techniques for Hate Speech Detection with Deep Learning. In: EPIA Conference on Artificial Intelligence 2021 Sep 7 (pp. 778-787). Springer, Cham.
- Venturott LI, Ciarelli PM. Data Augmentation for improving Hate Speech Detection on Social Networks. In: Proceedings of the Brazilian Symposium on Multimedia and the Web 2020 Nov 30 (pp. 249-252).

Ficha catalográfica disponibilizada pelo Sistema Integrado de Bibliotecas - SIBI/UFES e elaborada pelo autor

---

Venturott, Lígia Iunes, 1996-
V468d Detecção de discurso de ódio em redes sociais utilizando deep learning / Lígia Iunes Venturott. - 2021.
63 f. : il.

Orientador: Patrick Marques Ciarelli.
Dissertação (Mestrado em Engenharia Elétrica) - Universidade Federal do Espírito Santo, Centro Tecnológico.

1. Inteligência artificial. 2. Discurso de ódio na Internet. 3. Processamento de linguagem natural (Computação). 4. Aprendizado do computador. I. Ciarelli, Patrick Marques. II. Universidade Federal do Espírito Santo. Centro Tecnológico. III. Título.

CDU: 621.3
