Thiago Lessa da Costa

Aplicação de Redes Neurais para Classificação de Dados Textuais

Niterói - RJ, Brasil

28 de julho de 2022

Universidade Federal Fluminense

Thiago Lessa da Costa

Aplicação de Redes Neurais para Classificação de Dados Textuais

Trabalho de Conclusão de Curso

Monografia apresentada para obtenção do grau de Bacharel em Estatística pela Universidade Federal Fluminense.

Orientador(a): Prof. Dr. Jessica Kubrusly

Niterói - RJ, Brasil

28 de julho de 2022

Universidade Federal Fluminense

Thiago Lessa da Costa

Aplicação de Redes Neurais para Classificação de Dados Textuais

Monografia de Projeto Final de Graduação sob o título *"Aplicação de Redes Neurais para Classificação de Dados Textuais"*, defendida por Thiago Lessa da Costa e aprovada em 28 de julho de 2022, na cidade de Niterói, no Estado do Rio de Janeiro, pela banca examinadora constituída pelos professores:

Profa. Dra. Jessica Krubusly
Departamento de Estatística – UFF

Prof. Dr. Douglas Rodrigues Pinto
Departamento de Estatística – UFF

Profa. Dra. Karina Yuriko Yaginuma
Departamento de Estatística – UFF

Niterói, 28 de julho de 2022

Ficha catalográfica automática - SDC/BIME
Gerada com informações fornecidas pelo autor

C837a Costa, Thiago Lessa da
Aplicação de redes neurais para classificação de dados textuais / Thiago Lessa da Costa ; Jessica Quintanilha Kubrusly, orientadora. Niterói, 2022.
25 f. : il.

Trabalho de Conclusão de Curso (Graduação em Estatística)-Universidade Federal Fluminense, Instituto de Matemática e Estatística, Niterói, 2022.

1. Redes neurais. 2. Classificação de texto. 3. Aprendizado de máquinas. 4. Deep learning. 5. Produção intelectual. I. Kubrusly, Jessica Quintanilha, orientadora. II. Universidade Federal Fluminense. Instituto de Matemática e Estatística. III. Título.

CDD -

Bibliotecário responsável: Debora do Nascimento - CRB7/6368

Resumo

Este trabalho aplica redes neurais para classificação de texto. Foi usado um banco de textos público e supervisionado onde cada texto é rotulado como discurso de ódio ou não. O banco foi dividido em treino e teste. As redes neurais aplicadas foram a *Perceptron* com uma e duas camadas ocultas. Foi observado um sobreajuste do modelo e o que apresentou melhor resultado foi aquele com 2 camadas ocultas, sendo 1 neurônio na primeira e 2 na segunda. Para esta configuração, conseguiu-se uma acurácia de 87,5% nos dados de treino e 61,7% nos dados de teste.

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

Com o grande avanço da Internet e da tecnologia, o acesso às mídias e à interação com outras pessoas foi muito facilitado. Porém, com isso, criou-se um grande e recorrente problema, os discursos de ódio, consequência também de um errôneo conceito de que a liberdade de expressão não possui limites, muito menos na Internet.

Este trabalho tem como objetivo aplicar técnicas de pré-processamento de texto e criar uma rede neural que torne possível a detecção desses discursos de ódio em textos extraídos de comentários do site G1[1], com o intuito de que sejam rapidamente identificados, apagados e os autores possam sofrer as punições de acordo com seus crimes.

Badjatiya et al. (2017) propuseram utilizar *Deep Learning* para classificar *tweets* como sexista, racista, ou nenhum dos dois. Para tal, foram utilizadas técnicas de mineração de texto como N-gramas, CNN, vetores de *Bag of Words*, etc, além de algoritmos de Aprendizado de Máquinas e *Deep Learning*, como *Random Forest*, *Gradient Boosting*, e Redes Neuras Profundas. De acordo com este trabalho, os algoritmos e técnicas de *Deep Learning* superaram os métodos mais tradicionais de aprendizado de máquinas por cerca de 18 pontos no F1-score.

No estudo de Zhang, Robinson e Tepper (2017) foi utilizada uma rede neural convolucional combinada com uma rede LSTM para detectar discursos de ódio em 7 bancos de dados de textos, e comparada com métodos mais básicos de algoritmos de Aprendizado de Máquinas. Como resultado, o modelo de *Deep Learning* superou os modelos básicos em 6 dos 7 conjunto de dados utilizados, com uma diferença no F1-score variando entre 0.2 e 18 pontos.

O objetivo deste trabalho é aplicar técnicas de mineração de texto e observar a diferença na performance de modelos de Redes Neurais em um banco de dados de texto real supervisionado utilizando unigramas e bigramas, a fim de classificar os textos que contenham discursos de ódio.

[1] https://g1.globo.com/

Este trabalho está dividido da seguinte forma: No Capítulo 2, são apresentadas as metodologias utilizadas no trabalho, como pré-processamento de texto e as redes neurais. No Capítulo 3, são apresentados os resultados obtidos. Por fim, no Capítulo 4, é apresentada a conclusão do trabalho.

2 Materiais e Métodos

Neste capítulo, serão apresentados os materiais utilizados nas análises e os métodos que serão aplicados visando alcançar o objetivo final do trabalho, tais como Pré-Processamento de Texto e Redes Neurais.

2.1 Materiais

Neste trabalho, será usado o banco de dados supervisionado "OFFCOMBR-2"[1] com 1.250 comentários do site G1[2], classificados como ofensivos ou não de acordo com a análise de 3 juízes. Neste banco de dados, um comentário foi classificado como ofensivo se 2 dos 3 juízes concordassem.

Para o processo de *Deep Learning*, o banco de dados será divido em 80% para treinar os algoritmos, e 20% para realizar a validação dos modelos, seguindo as proporções mais usuais nestes tipos de trabalho.

2.2 Pré-processamento de texto

Nesta seção serão apresentadas as técnicas de pré-processamento de texto que serão utilizadas neste trabalho. Esta etapa é uma das mais importantes em um projeto de Aprendizado de Máquinas aplicado a um banco de texto supervisionado. Trata-se de um conjunto de métodos de limpeza, estruturação e organização dos dados, a fim de deixá-los com a maior qualidade possível, removendo dados faltantes e inconsistentes, reduzindo as variáveis do banco de dados e transformando-os em dados numéricos e estruturados. Após o processo, os dados estarão em condições de serem utilizados em um modelo de redes neurais, de forma que o modelo possa ter a melhor performance possível nesses dados.

Existem inúmeras técnicas de pré-processamento. Neste trabalho, serão apresentados

[1] Link para as bases http://inf.ufrgs.br/rppelle/hatedetector/
[2] https://g1.globo.com/

alguns métodos como os de *tokenização*, remoção de *stop words*, normalização de palavras, e criação da matriz termo-documento. A seguir, tem-se a explicação de cada uma delas em detalhes.

2.2.1 *Tokenização*

De acordo com Junior (2008),um dos primeiros processos de pré-processamento é a *tokenização*, que consiste em dividir um texto em *tokens*, ou seja, unidades do texto original. Na maioria dos casos, um *token* se refere à uma única palavra, conhecido como uni-grama, porém também podem se referir a duas ou mais palavras (bi-gramas ou n-gramas).

Exemplo 2.1 *Exemplo de tokenização de unigrama para a frase "Vocês são um bando de idiotas":*

[Vocês] [são] [um] [bando] [de] [idiotas].

Exemplo 2.2 *Exemplo de tokenização de bi-grama para a frase "Vocês são um bando de idiotas":*

[Vocês são] [são um] [um bando] [bando de] [de idiotas].

Para realizar a *tokenização*, precisa-se definir um caractere separador. Em geral, usa-se o espaço entre as palavras como esse separador, porém pode ser utilizado uma pontuação, palavra ou letra.

2.2.2 Remoção de *Stop Words*

Stop Words é a nomenclatura dada às palavras que possuem pouco valor semântico e que aparecem com frequência nos textos, como preposições, artigos e conjunções. Para enxutar a análise e melhorar a qualidade da mesma, as *stop words* são removidas do conjunto de dados. Neste estudo, será utilizada uma lista de *stop words* em português sugerida por Virtuati[3], que também pode ser encontrada no Anexo 1 deste trabalho.

[3]Lista de *stop words* em português: https://virtuati.com.br/cliente/knowledgebase/25/Lista-de-StopWords.html (Acessado em 13 de maio de 2022 às 15:47)

2.2.3 Normalização

Esta etapa consiste em "simplificar"os *tokens*, retirando o máximo possível de variações de uma palavra e as agrupando em um único *token*. Para esta etapa, será usado o processo de **stemização**, que consiste em reduzir as palavras até a raiz da mesma.

De acordo com Junior (2008), existem 3 principais métodos de stemização: Método de Porter, *Stemmer S*, e método de Lovins.

- Método de Porter: Reduz palavras até o radical, removendo variações e inflexões.
- *Stemmer S*: Remove apenas algumas variações e sufixos, como "es"e "s".
- Método de Lovins: Remove, no máximo, um sufixo por palavra, se baseando nas regras de Lovins.

Abaixo, temos um exemplo do método de Porter.

Exemplo 2.3 *Andar, Andando, Andamos* $\Rightarrow$ *Anda*

2.2.4 Seleção dos Termos (*Term Frequency*)

Nesta etapa, são removidos os *tokens* que aparecem com muita frequência ou que são raros no banco de dados, pois não são relevantes para as análises. Com isso, consegue-se obter um conjunto de termos mais enxuto e representativo dos dados. (SILVA, 2021)

2.2.5 Matriz termo-documento

A última etapa do pré-processamento consiste em transformar os dados para um formato binário para aplicar os algoritmos de processamento. Isso se dá através de uma matriz, onde a coluna j diz respeito ao token j e a linha i diz respeito ao texto i (SILVA, 2021).

Considerando que, depois de todo o processo de pré-processamento, o banco de dados final possui n textos e m *tokens*, teremos uma matriz termo-documento $n \times m$, onde cada elemento a_{ij} é igual a 1, se o texto i contém o *token* j, e igual a 0 caso contrário.

2.3 Redes Neurais Artificiais

Nesta seção, será apresentada a rede neural *Perceptron* e a *Perceptron* multi-camadas (PMC), e suas funções de ativação. (SILVA; SPATTI; FLAUZINO, 2016)

Antes de ser introduzido o conceito da Rede Neural *Perceptron*, é interessante entender a diferença entre Aprendizado de Máquinas, *Deep Learning* e Redes Neurais, como mostra a Figura 1.

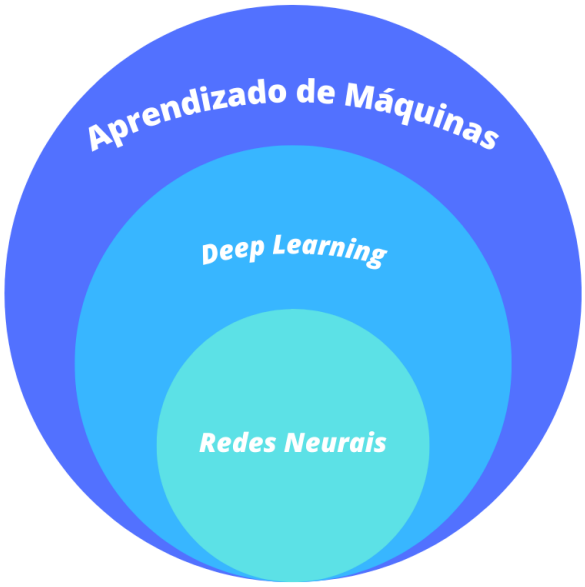

Figura 1: Aprendizado de Máquinas x *Deep Learning* x Redes Neurais

Em resumo, o Aprendizado de Máquinas é o uso de algoritmos e modelos estatísticos para organizar dados, reconhecer padrões e realizar previsões, fazendo com que os computadores aprendam com os modelos e os dados.

Já o *Deep Learning* é uma sub-área do Aprendizado de Máquinas, que utiliza algoritmos de alto nível para simular o comportamento do cérebro humano. Um desses algoritmos é justamente as redes neurais artificiais, que simulam o comportamento dos neurônios humanos.

2.3.1 Rede *Perceptron*

O *Perceptron* é a forma mais simples de uma rede neural artificial. Possui apenas uma camada oculta com apenas um único neurônio, como pode ser observado na Figura 2, que mostra uma rede com quatro variáveis de entrada.

A rede possui uma camada de entrada, que é onde os dados X_j serão fornecidos e

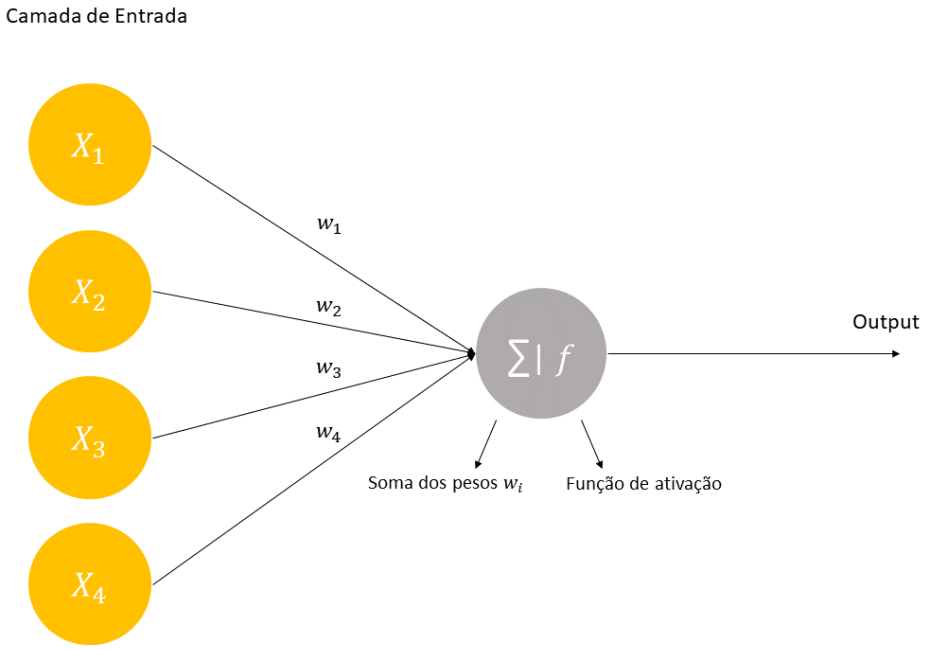

Figura 2: Exemplo de uma rede neural *Perceptron*

ponderados pelos pesos w_j. Já na camada interna, esses dados ponderados serão somados e o valor resultante será usado como argumento para a função de ativação do neurônio, cujo resultado será a saída do modelo, ou seja, a resposta da variável-resposta.

Sua função de ativação, de modo geral, é definida por:

$$\widehat{Y} = f\left(\sum_{j=1}^{m} w_j X_j\right). \tag{2.1}$$

2.3.2 Rede *Perceptron* multi-camadas (PMC)

O PMC segue a mesma lógica do *Perceptron*, porém com mais camadas internas e mais neurônios nessas camadas.

Nesta rede, a primeira camada oculta funciona da mesma maneira que o algoritmo anterior, porém com mais de um neurônio, e as saídas dos neurônios são usadas como entrada para a próxima camada oculta, até que chega-se na camada de saída, onde cada neurônio $\boldsymbol{n_i}$ retorna a probabilidade dos dados de entrada serem de uma classe i. Na Figura 3, tem-se um exemplo de uma PMC com quatro variáveis de entrada, duas camadas ocultas e dois neurônios na camada de saída, porém essas quantidades podem variar em cada situação.

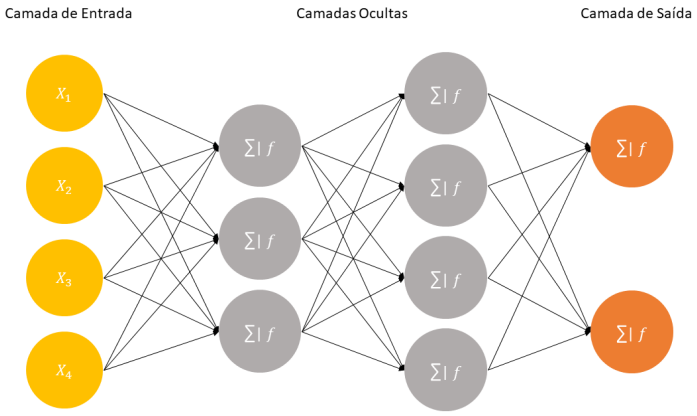

Figura 3: Exemplo de uma rede neural *Perceptron* multi-camadas

Em ambas as redes *Perceptron*, os parâmetros a serem otimizados são os pesos que cada neurônio atribui às suas entradas, e a otimização é feita através de *backpropagation*, que consiste em atribuir valores iniciais aos pesos $\boldsymbol{w}$ e realizar iterações para chegar a valores que minimizam os erros da camada de saída da rede.

2.3.3 Função de ativação

As funções de ativação são responsáveis por realizar o processamento das informações dentro dos neurônios. Elas recebem como argumento a soma dos pesos $\boldsymbol{w}$ multiplicados pelas entradas $\boldsymbol{X}$ do neurônio, e retornam uma saída de acordo com cada tipo de função de ativação. Cada camada da rede neural pode possuir uma função de ativação diferente das outras. Nessa seção, serão apresentadas as duas funções de ativação mais usuais.

- **Função de Ativação ReLU**

  É a função mais usada nas camadas ocultas do algoritmo, por não ativar todos os neurônios ao mesmo tempo e, com isso, ser computacionalmente mais leve.

  É uma função muito simples, pois apenas retorna valores positivos. Se a entrada da função for negativa, ela retornará zero e não ativará o neurônio. Tem como equação:

$$f(x) = máx(0, x). \tag{2.2}$$

- **Função de Ativação Sigmoid**

  De modo geral, é utilizada nas camadas de saída das redes neurais quando se trata de um problema de classificação, pois retorna valores entre 0 e 1, que podem ser

interpretados como a probabilidade dos dados pertencerem a uma classe definida.

Sua equação é dada por:

$$f(x) = \frac{1}{1 + e^{-x}}. \tag{2.3}$$

2.3.4 Métricas de Avaliação do Modelo

Após ser criada a rede neural, ou qualquer outro método de classificação, precisa-se de medidas de avaliação de performance do modelo. A construção dessas medidas serão feitas a partir da matriz de confusão, que é definida pelo pelo número de acertos e erros em cada classe. A matriz de confusão é uma forma simples de visualizar a saída do modelo e ela indica a quantidade de Falsos Positivos, Falsos Negativos, Verdadeiros Positivos e Verdadeiros Negativos. Veja a sua representação na Tabela 1 a seguir.

Tabela 1: Exemplo de Matriz de Confusão

		Predito	
		Positivo	Negativo
Observado	Positivo	Verdadeiro Positivo	Falso Negativo
	Negativo	Falso Positivo	Verdadeiro Negativo

Baseado na matriz de confusão, pode-se criar algumas métricas para avaliar modelos de classificação binária. São elas:

- **Acurácia**

  A acurácia indica a quantidade de observações que foram classificadas corretamente, independente da classe.

$$Acurácia = \frac{VP + VN}{VP + VN + FP + FN}. \tag{2.4}$$

- **Sensibilidade**

  A Sensibilidade, também conhecido como *Recall*, é definida pela razão entre a quantidade de observações que foram classificadas como Positivo e a quantidade de observações que realmente são Positivo.

$$Sensibilidade = \frac{VP}{VP + FN}. \tag{2.5}$$

- **Especificidade**

  É definida pela razão entre a quantidade de observações que foram classificadas corretamente como Negativos e a quantidade total de observações que foram classificadas como Negativos pelo modelo.

$$Especificidade = \frac{VN}{VN + FN}. \tag{2.6}$$

- **F1-Score**

  É interpretada como uma média harmônica entre a Sensibilidade e a Precisão, que é uma métrica definida pela quantidade de verdadeiros positivos sobre o total de observações que o modelo classificou como positivo. Abaixo tem-se a fórmula da Precisão e do F1-Score.

$$Precisão = \frac{VP}{VP + FP}. \tag{2.7}$$

$$F_1 = 2 \times \frac{Precisão \times Sensibilidade}{Precisão + Sensibilidade}. \tag{2.8}$$

  Percebe-se que a expressão definida acima pode ser escrita como:

$$F_1 = 2 \times \frac{\frac{\cancel{Precisão} \times Sensibilidade}{\cancel{Precisão}}}{\frac{\cancel{Precisão} + Sensibilidade}{\cancel{Precisão}}}$$

$$F_1 = 2 \times \frac{\frac{\cancel{Sensibilidade}}{\cancel{Sensibilidade}}}{(1 + \frac{Sensibilidade}{Precisão}) \times \frac{1}{Sensibilidade}}$$

$$F_1 = \frac{2}{\frac{1}{Sensibilidade} + \frac{1}{Precisão}}. \tag{2.9}$$

  A Equação 2.9 mostra que, de fato, a medida F1-Score trata-se da média harmônica entre a Sensibilidade e a Precisão

3 Análise dos Resultados

Neste capítulo serão apresentados os resultados e as análises obtidas. Todas as etapas deste estudo foram feitas usando o *software* R versão 3.6.3 e a interface R *Studio*.

Seguindo o princípio FAIR (sigla para *Findable*, *Accessible*, *Interoperable* e *Reusable*) sugerido por Mondelli, Peterson e Gadelha (2019),os *scripts* produzidos neste trabalho, assim como o banco de dados, se encontram em um repositório no Github, que pode ser acessado através do link: https://github.com/lessathiago1/.

3.1 Amostras de Treino e de Teste

Inicialmente, o banco de dados possuía 1.250 comentários, sendo 419 classificados como discurso de ódio e 831 como não sendo discurdo de ódio, como pode ser observado na Figura 4.

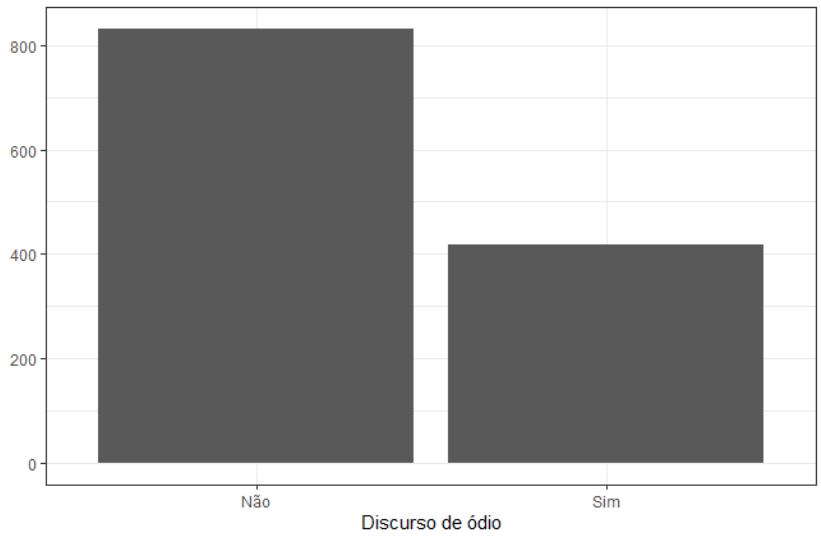

Figura 4: Distribuição da variável alvo no banco de dados.

Com o intuito de balancear o número de elementos de cada categoria da variável alvo

nos bancos de treino e teste e seguindo a proporção de divisão de 80% dos dados para treino e 20% para teste, foram escolhidos, aleatoriamente e sem reposição, 336 elementos de cada classe para compor o banco de treino, enquanto no banco de teste, foram selecionados 83 textos classificados como discurso de ódio e 85 classificados como não sendo discurso de ódio. A distribuição pode ser observada na Figura 5.

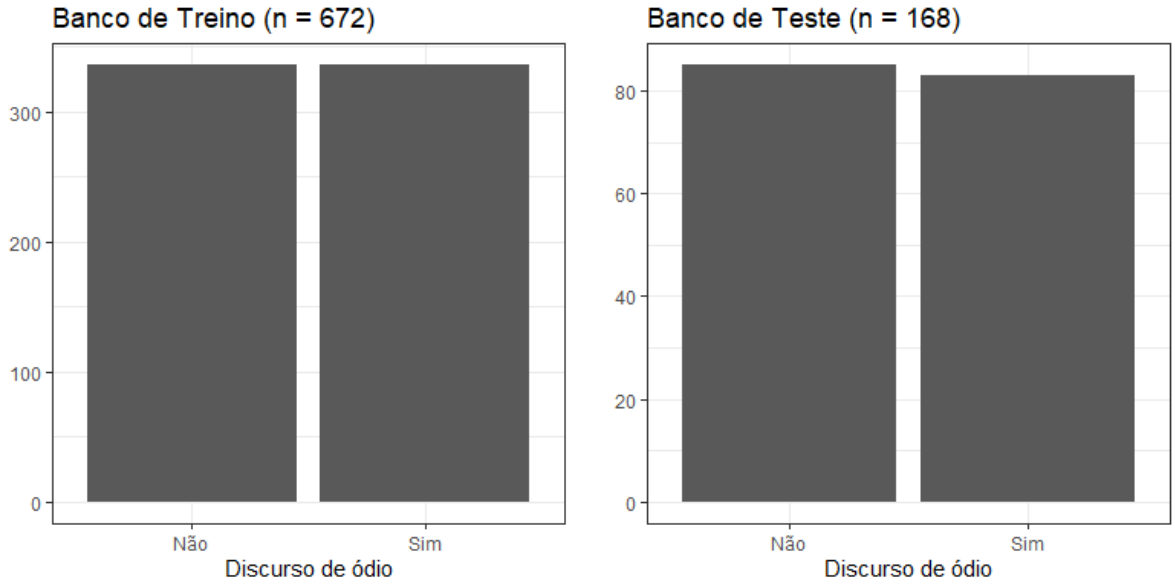

Figura 5: Distribuição da variável alvo nos bancos de treino e de teste.

3.2 Pré-processamento de Texto

Para a etapa de pré-processamento de texto, foram utilizados os métodos citados na Seção 2.2. São eles: *tokenização*, remoção de *stop words*, normalização, seleção dos termos e criação da matriz termo-documento. Abaixo estão descritos como foi realizada cada etapa do pré-processamento.

- **Tokenização**

  Para esta etapa, foi utilizada a função *unnest_tokens* do pacote *tidytext* (SILGE; ROBINSON), que permite dividir os textos dos bancos de dados em *tokens* de quantas palavras for necessário. Para este trabalho, os textos foram divididos em unigrama (com uma única palavra em cada *token*) e bigrama (*tokens* de duas palavras). Com isso, ficou-se com dois bancos de treino e de teste, um para os unigramas e outro para os bigramas.

- **Remoção de *stop words***

Nesta etapa, foram removidas todas as palavras consideradas *stop words* e que não apresentam informação relevante.[1] Se pelo menos uma palavra de um bigrama era uma *stop word*, o bigrama foi removido dos bancos de texto.

- **Normalização**

  Os *tokens* foram reduzidos até um radical. Palavras como "achar", "acham" e "achei" foram reduzidas e agrupadas no *token ach*, por exemplo.

- **Seleção dos Termos**

  Nesta etapa, foram retirados os termos mais frequentes e que apareciam na mesma proporção entre os textos rotulados como discurso de ódio e como não sendo discurso de ódio. Para cada termo, foi contabilizada sua frequência relativa em cada classe. Aqueles cuja frequência relativa ficava entre 40% e 60% foram descartados por entender que não apresentavam muita informação a respeito de cada classe. Após essa operação, foram selecionados os 100 termos mais frequentes.

- **Matriz termo-documento**

  Após todas as etapas de pré-processamento de texto, os bancos de dados ficaram com um total de 100 termos cada. Os textos que ficaram sem termos após o processo de Seleção dos Termos foram descartados. Foram montadas as matrizes termo-documento, com as linhas e colunas especificadas na Tabela 2 abaixo:

Tabela 2: Dimensões das Matrizes Termo-Documento

Matriz Termo-Documento	Nº de linhas (documentos)	Nº de colunas (termos)
Matriz de treino (unigrama)	606	100
Matriz de teste (unigrama)	168	100
Matriz de treino (bigrama)	207	100
Matriz de teste (bigrama)	147	100

Na Figura 6, tem-se os termos mais frequentes no banco de treino de unigramas, enquanto na Figura 7, tem-se os do banco de bigramas.

Percebe-se que, tanto na Figura 6 quanto na Figura 7, os *tokens* que mais se repetem são os que possuem contexto de negação. Ao observar a Figura 6, nota-se que a nuvem de palavras está com bem menos *tokens*, indicando que, com bigramas, é mais difícil que hajam *tokens* iguais.

[1] Lista de *stop words* em português: https://virtuati.com.br/cliente/knowledgebase/25/Lista-de-StopWords.html

Figura 6: Nuvem de palavras do banco de treino de unigramas.

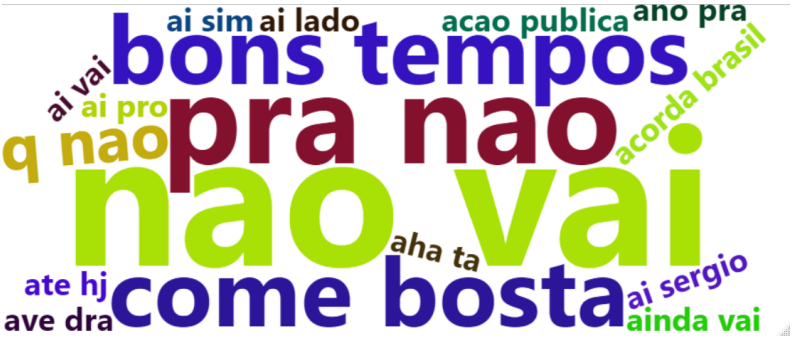

Figura 7: Nuvem de palavras do banco de treino de bigramas.

3.3 Redes Neurais

Terminado todo o processo de pré-processamento textual, antes de começar a construção dos modelos de redes neurais, foi realizado um filtro para retirar todos os textos que não possuíam nenhum *token* entre os 100 mais frequentes. Após a realização do filtro, observou-se que as matrizes de treino e teste de bigramas ficaram com poucas linhas (< 100 nos dados de treino e < 10 nos dados de teste). Além disso, a matriz de teste ficou sem nenhum texto rotulado como discurso de ódio, o que inviabilizou a construção das métricas de avaliação dos modelos. Portanto, para a construção dos modelos, serão considerados apenas as matrizes termo-documento de unigramas.

Para este trabalho, foram construídos 9 modelos de redes neurais, utilizando diferentes combinações de camadas ocultas e de neurônios em cada uma delas. A função de ativação utilizada em todos os modelos foi a sigmoid. Todos os modelos foram construídos com o pacote *neuralnet* (FRITSCH; GUENTHER; WRIGHT). Na Tabela 3, tem-se as combinações de camadas e neurônios e suas respectivas métricas.

Ao analisar a Tabela 3, percebe-se que a grande maioria dos modelos testados sofreram de sobreajuste, fenômeno no qual os modelos se adequam demais aos dados de treino e

Tabela 3: Métricas dos modelos de redes neurais

		Métricas							
		Acurácia		Sensibilidade		Especificidade		F1-score	
Modelo	Rede Neural	Treino	Teste	Treino	Teste	Treino	Teste	Treino	Teste
1	Nenhuma camada oculta	79,8%	63,9%	76,3%	57,4%	83,3%	69,4%	0,78	0,59
2	1 camada oculta com 3 neurônios	92,4%	58,6%	89,6%	54,1%	95,1%	62,5%	0,92	0,54
3	1 camada oculta com 4 neurônios	93,8%	60,1%	91,4%	50,8%	96,2%	68,0%	0,93	0,53
4	1 camada oculta com 5 neurônios	94,5%	57,9%	96,4%	65,6%	92,7%	51,9%	0,94	0,58
5	2 camadas ocultas com 2 e 1 neurônios	90,8%	65,4%	95,3%	77,1%	86,5%	55,6%	0,91	0,67
6	2 camadas ocultas com 1 e 2 neurônios	87,5%	61,7%	89,6%	60,7%	85,4%	62,5%	0,87	0,59
7	2 camadas ocultas com 2 e 2 neurônios	88,3%	64,7%	81,6%	54,1%	94,8%	73,6%	0,87	0,58
8	2 camadas ocultas com 2 e 3 neurônios	90,1%	60,1%	95,3%	72,1%	85,1%	50,0%	0,90	0,62
9	2 camadas ocultas com 3 e 2 neurônios	92,2%	60,1%	94,2%	59,0%	90,3%	61,1%	0,92	0,57

possuem baixa performance em dados novos, como os de teste. O Modelo 6 foi o que obteve a performance mais equilibrada ao considerar a Sensibilidade e a Especificidade, com ambos acima de 60%, enquanto o Modelo 5 foi o que apresentou o melhor F1-score, tendo boa Sensibilidade, porém Especificidade baixa.

4 Conclusão

O objetivo original deste trabalho era analisar a performance de modelos de redes neurais para classificação de texto utilizando unigramas e bigramas.

Com os dados coletados, não foi possível analisar o desempenho dos modelos com bigramas, vistos que os dados resultantes das etapas de pré-processamento possuíam pouca informação e acabaram sendo desconsiderados.

Com os unigramas, como elucidado na Tabela 3, os modelos em sua maioria apresentaram sobreajuste, o que prejudica a confiança na sua possível utilização em um ambiente real de classificação de textos. Pode-se observar que os modelos apresentaram uma melhor performance quando possuíam mais camadas ocultas e menos neurônios em cada uma delas, como é o caso dos Modelos 5 e 6. No geral, os modelos com 2 camadas ocultas performaram melhor do que os com apenas uma.

A fim de encontrar melhores resultados, uma das propostas para trabalhos futuros aumentar o número de textos no banco de dados original, visto que modelos de redes neurais são feitos para serem utilizados com grandes volumes de dados, ou utilizar algoritmos mais simples de aprendizado de máquinas.

Referências

BADJATIYA, P. et al. Deep learning for hate speech detection in tweets. 2017. Disponível em: ⟨https://arxiv.org/abs/1706.00188⟩.

FRITSCH, S.; GUENTHER, F.; WRIGHT, M. N. *neuralnet: Training of Neural Networks*. [S.l.], 2019. R package version 1.44.2. Disponível em: ⟨https://CRAN.R-project.org/package=neuralnet⟩.

JUNIOR, J. R. C. Desenvolvimento de uma metodologia para mineração de textos. 2008. Disponível em: ⟨https://www.maxwell.vrac.puc-rio.br/colecao.php?strSecao=resultado&nrSeq=11675@1⟩.

MONDELLI, M. L.; PETERSON, A.; GADELHA, L. Exploring reproducibility and fair principles in data science using ecological niche modeling as a case study. *Advances in Conceptual Modeling*, p. 23–33, 10 2019.

SILGE, J.; ROBINSON, D. tidytext: Text mining and analysis using tidy data principles in r. *JOSS*, The Open Journal, v. 1, n. 3, 2016. Disponível em: ⟨http://dx.doi.org/10.21105/joss.00037⟩.

SILVA, D. R. P. da. Técnicas de mineração de texto e de análise de conglomerados aplicadas em banco de dados de automóveis. 2021. Disponível em: ⟨http://estatistica.uff.br/wp-content/uploads/sites/33/2021/09/tcc_20211_DanielleRibeiroPereiraDaSilva_218054079.pdf⟩.

SILVA, I. N. da; SPATTI, D. H.; FLAUZINO, R. A. Redes neurais artificiais para engenharia e ciências aplicadas: Fundamentos teóricos e aspectos práticos. 2016.

ZHANG, Z.; ROBINSON, D.; TEPPER, J. Hate speech detection using a convolution-lstm based deep neural network. 2017. Disponível em: ⟨https://www.semanticscholar.org/paper/Hate-Speech-Detection-Using-a-Convolution-LSTM-Deep-Zhang/c3e8e67bfb0675449e5ffc4b63ab8d6c343338c3⟩.

APÊNDICE 1 – Lista de *stop words*

A lista de *Stop Words* em português, retirada do site ⟨https://virtuati.com.br/cliente/knowledgebase/25/Lista-de-StopWords.html⟩, possui as seguintes palavras:

a, agora, ainda, alguém, algum, alguma, algumas, alguns, ampla, amplas, amplo, amplos, ante, antes, ao, aos, após, aquela, aquelas, aquele, aqueles, aquilo, as, até, através, cada, coisa, coisas, com, como, contra, contudo, da, daquele, daqueles, das, de, dela, delas, dele, deles, depois, dessa, dessas, desse, desses, desta, destas, deste, deste, destes, deve, devem, devendo, dever, deverá, deverão, deveria, deveriam, devia, deviam, disse, disso, disto, dito, diz, dizem, do, dos, e, é, ela, elas, ele, eles, em, enquanto, entre, era, essa, essas, esse, esses, esta, está, estamos, estão, estas, estava, estavam, estávamos, este, estes, estou, eu, fazendo, fazer, feita, feitas, feito, feitos, foi, for, foram, fosse, fossem, grande, grandes, há, isso, isto, já, la, lá, lhe, lhes, lo, mas, me, mesma, mesmas, mesmo, mesmos, meu, meus, minha, minhas, muita, muitas, muito, muitos, na, nas, nem, nenhum, nessa, nessas, nesta, nestas, ninguém, no, nos, nós, nossa, nossas, nosso, nossos, num, numa, nunca, o, os, ou, outra, outras, outro, outros, para, pela, pelas, pelo, pelos, pequena, pequenas, pequeno, pequenos, per, perante, pode, pude, podendo, poder, poderia, poderiam, podia, podiam, pois, por, porém, porque, posso, pouca, poucas, pouco, poucos, primeiro, primeiros, própria, próprias, próprio, próprios, quais, qual, quando, quanto, quantos, que, quem, são, se, seja, sejam, sem, sempre, sendo, será, serão, seu, seus, si, sido, só, sob, sobre, sua, suas, talvez, também, tampouco, te, tem, tendo, tenha, ter, teu, teus, ti, tido, tinha, tinham, toda, todas, todavia, todo, todos, tu, tua, tuas, tudo, última, últimas, último, últimos, um, uma, umas, uns, vendo, ver, vez, vindo, vir, vos, vós.