Identificação de desvios de linguagem através de redes neurais artificiais

Gustavo Augusto Pires
Escola de Engenharia
Universidade Federal de Minas Gerais
Belo Horizonte, Brasil
gustavoaugustopires@ufmg.br

Frederico Gualberto Ferreira Coelho
Escola de Engenharia
Universidade Federal de Minas Gerais
Belo Horizonte, Brasil
Email: fredgfc@ufmg.br

***Abstract*—Este artigo propõe um método de identificação de desvios de linguagens em frases escritas na língua portuguesa através de Redes Neurais Artificiais. São abordadas diferentes configurações de neurônios e funções de ativação para obter a melhor acurácia. O modelo desenvolvido apresenta um desempenho suficientemente satisfatório capaz de classificar com sucesso vários exemplares de desvios. Por ter extraído uma generalização das características dos desvios de linguagem, tem como vantagem uma estrutura que não é atrelada a um dicionário de dados fixo.**

I. INTRODUÇÃO

Os desvios de linguagens são erros que envolvem problemas de gramática, ortografia, acentuação ou concordância, tais como "este modelo démodé" (Estrangeirismos), "desça já lá em baixo" (Pleonasmo) e "Não vi ela" (Cacofonia). Eles são comumente cometidos por estudantes da língua portuguesa. Contudo, os corretores de textos tradicionais não são capazes de realizar sua identificação. Algumas ferramentas mais complexas, se propõem a realizar essa tarefa com o uso de técnicas de comparação com dicionários de dados pré-montados [9]. Nesses métodos, as estruturas são fixas e não apresentam possibilidade de evolução autônoma. Para criar um corretor capaz de classificar os desvios de linguagem em frases em português, este artigo aborda um método desenvolvido através de redes neurais artificiais.

Para isso, desenvolvemos um banco de dados com frases contendo desvios de linguagem, a partir do dicionário de dados elaborado em [8]. Buscamos um conjunto de dados grande o suficiente para encontrar um número adequado de exemplares dos desvios. Além disso, para garantir a boa qualidade dos dados, realizamos a remoção das *stopwords* e aplicamos o processo de *stemming*.

A técnica de remoção de *stopwords* [4] consiste em eliminar palavras que não contribuem para a formação de um termo, como "de", "e", "a", entre outras. Sua remoção diminui o tamanho dos textos e garante que apenas palavras com maior relevância para a formação do significado sejam avaliadas pelo algoritmo.

Já o processo de *stemming* visa transformar palavras que representem um mesmo significado em apenas um termo. O algoritmo reduz as variações morfológicas presentes no texto e facilita o processo de aprendizado. Isso se deve a redução da palavra ao item lexial essencial, removendo as variações afixais [2]. Palavras tais como musicalização, musical e músicas, que são representações da palavra música, serão transformadas apenas no termo "music". Desse modo, incluímos esse processo no algoritmo, conforme descrito em [2]. Esses tratamentos constituem a etapa de pre-processamento dos dados.

Na construção do classificador, utilizamos as redes neurais artificiais. Elas são compostas por unidades de processamento chamadas neurônios, que aplicam uma função matemática aos dados de entrada e oferecem uma única resposta. Durante o desenvolvimento do algoritmo, testamos diversas configurações para avaliar a melhor estrutura. Analisamos diferentes números de neurônios e camadas escondidas, para verificar qual obtém a generalização das características, evitando o problema de *overfitting*, quando a rede neural artificial fica ligada os dados de treinamento ao invés de extrair as características gerais.

Ao encontrar a melhor configuração para a rede, buscaremos qual a melhor função de ativação para o problema proposto. Elas são aplicadas sobre o resultado da soma das entradas com o peso atribuído as respectivas conexões. As funções que serão analisadas serão a *ReLU*, a *Leaky ReLU*, a *Sigmoid*, a *Softmax* e a *Tangente Hiperbólica*. Após o processamento, o modelo informa a probabilidade de haver determinado tipo de desvio de linguagem naquela frase.

Este artigo está organizado da seguinte maneira: a revisão da bibliografia e de conceitos importantes foram abordados na sessão 2. Na sessão 3, foi apresentado o método desenvolvido para a resolução do problema. Os experimentos foram descritos na sessão 4. Na sessão 5 foi apresentado as conclusões encontradas nesse estudo e na sessão 6 os possíveis trabalhos futuros.

II. REVISÃO BIBLIOGRÁFICA

A. Desvios de linguagem

De acordo com a pesquisa realizada em [8], os desvios de linguagens são palavras ou construções que ferem as normais gramaticais, por descuido ou falta de conhecimento. Para a construção do algoritmo, selecionamos alguns dos desvios para que sejam utilizados com as classes do conjunto de dados. Nesse trabalho, serão considerados os seguintes desvios de linguagem:

- Arcaísmos: expressões que caíram em desuso.
- Cacófatos: junção de palavras que resultam em som desagradável ou obsceno.
- Clichês: utilização de frases prontas ou provenientes do senso comum.
- Estrangeirismos: emprego desnecessário de palavras estrangeiras.
- Marcas de Oralidade: são traços da fala cotidiana no texto escrito.
- Plebeísmos: expressões populares que caracterizam falta de instrução.
- Pleonasmos: repetição desnecessária da palavra ou da ideia contida nela.

B. Ferramentas de correção ortográfica

Os editores de textos tradicionais possuem ferramentas para realizar a correção de erros ortográficos, como acentuação, concordância e alguns erros estilísticos. No entanto, esses corretores não são capazes de identificar erros mais complexos como os desvios de linguagem. Uma ferramenta mais robustas que permite uma correção de mais tipos de erros é o *LanguageTool* [9].

O *LanguageTool* é um software de código aberto que realiza a correção de erros gramaticais e de estilo. A ferramenta possui um dicionário que pode ser modificado, afim de adicionar novos erros ao catálogo, de modo que possam ser identificados. Contudo, conforme analisado em [8], ela consegue detectar poucos tipos de desvios de linguagens. Sendo assim, sua atuação é melhor aplicada a correção de erros gramaticais.

C. Aprendizado de Máquina Supervisionado

O aprendizado de máquina supervisionado consiste no fornecimento de um conjunto de dados para um algoritmo capaz de extrair características gerais dele através do processo de treinamento. Esse conjunto deve ser rotulado, ou seja, possuir parâmetros e um rótulo indicando a qual classe pertence. Durante o processamentos, os dados são divididos em duas bases: uma de treinamento e outra de teste, que será utilizado para avaliar o desempenho final do método.

Esse tipo de algoritmo pode ser utilizado na construção de classificadores. Esses tem o objetivo de alcançar uma generalização dos dados utilizados em seu conjunto de treinamento, possibilitando que ele seja capaz de prever corretamente a classe de novos dados [6]. Um exemplo de algoritmo de aprendizado de máquina supervisionado, que pode ser utilizado para construção de classificadores, é a Rede Neural Artificial.

D. Redes Neurais Artificiais

De acordo com [3], uma rede neural artificial é um modelo computacional baseado no funcionamento do cérebro humano. Ela é composta de unidades de processamento chamadas neurônios, que aplicam uma função matemática aos dados de entrada e oferecem uma única resposta. Os neurônios são dispostos em camadas e interligados por conexões as quais são atribuídas pesos. O treinamento de uma rede neural consiste no ajuste dos valores desses pesos e é responsável por extrair as características dos dados e armazenar o conhecimento.

As redes neurais artificiais são utilizadas em problemas de classificação. Nessas aplicações, o algoritmo trabalha com um conjunto de parâmetros fornecido pela base de dados, que estão atrelados a rótulos. Esses representam as classes as quais o registro pertence. Assim, podem ser definidas como uma forma de aprendizado de máquina supervisionado.

III. MÉTODO PROPOSTO

Este artigo propõe a criação de um classificador de desvios de linguagem em frases na língua portuguesa e realiza uma análise de como as redes neurais artificiais são capazes de atingir a generalização para armazenamento de conhecimento e substituir técnicas baseadas em estruturas fixas e regras desenvolvidas manualmente.

A. Banco de dados

A construção do banco de dados é uma etapa importante do desenvolvimento, pois o algoritmo irá buscar a generalização a partir dos exemplares encontrados nele. Se não houver exemplares adequados e em quantidade suficiente, a rede neural artificial não será capaz de extrair o conhecimento do conjunto e não irá obter uma performance adequada.

Como fonte para o banco de dados, utilizamos as frases do ”O corpus do português” desenvolvido em [1]. Neste banco de dados, foi possível encontrar mais de 300 mil frases em português. Apesar disso, as frases encontradas nem sempre possuíam desvios de linguagem e não estavam rotuladas.

Para resolver esse problema, utilizamos o catálogo desenvolvido em [8] e criamos uma tabela de expressões características de cada desvios de linguagens. Desenvolvemos uma consulta SQL para encontrar frases que possuíssem essas expressões. Com isso, foi possível encontrar mais de 50 mil frases contendo desvios de linguagem. Um problema apresentado, foi a diferença entre o número de exemplares em cada tipo de classe, onde algumas possuíam muitas correspondências e outras poucas.

Afim de sanar esse problema, limitamos a quantidade de correspondências em 5 mil por classes, para as que possuíam muitas correspondências. Já para as classes que não tiveram exemplares suficientes, após tentarmos obter mais, duplicamos as frases dessas classes. Essa técnica é chamada de *oversampling* [11], que consiste em aumentar aleatoriamente o conjunto de dados. Isso pode ser problemática porque nem sempre aumenta o ganho de informação. Todavia, entre as possibilidades avaliadas como o *undersampling*, que pode causar perda de informação útil [11], optamos por avaliar se poderíamos obter resultados melhores usando a sobreamostragem.

B. Pré-processamento de dados

O processamento de linguagem natural utiliza abordagens computacionais e estatísticas para construir um modelo capaz de obter características gerais a partir dos textos analisados e ser capaz de utilizar o conhecimento adquirido na realização

de tarefas, tal como a classificação. Para facilitar a análise das características do textos, é necessário que eles sejam tratados por técnicas que tornem mais evidentes as características, eliminando ruídos desnecessários na compreensão do texto. Duas dessas técnicas são a remoção de *Stopwords* e a técnica de *Stemming*. A aplicação dessas técnicas nos conjuntos de dados é denominada etapa de pré-processamento.

As *Stopwords* são palavras, geralmente funcionais, que não devem ser consideras para a formação do texto [4]. O processo de remoção consiste em identifica-las nos textos, algo que pode ser feito através de abordagem estatística, e removê-las. De acordo com [4], essa técnica beneficia a construção de modelos, porque reduz o número de entradas nas redes neurais artificiais. Algo que facilita o processo de aprendizagem de máquina.

A técnica de *Stemming* é um método que reduz o tamanho do vocabulário nos textos de modo que possa ser encontrado similaridades morfológicas entre as palavras. Isso diminui os atributos do texto, como é descrito em [2]. Esse processo é necessário, porque muitas vezes palavras morfologicamente semelhantes possuem um mesmo conceito. Desse modo, ao agrupar palavras com mesmo sentido em uma úncia expressão, o processo de aprendizado se torna mais eficiente.

Para o método proposto, ambas as técnicas descritas foram aplicadas sobre o conjunto de dados. Para a remoção de *stopwords*, encontramos as palavras com maior taxa de incidência e as removemos. Palavras como ”e”, ”é” e ”de” foram removidas. O processo de *stemming* retirou os radicais das palavras. Assim, palavras como computador e computação, se tornaram ”comput”, uma expressão que agrupa as duas palavras com similaridade morfológica em uma.

Antes de exportar os dados para um formato que pudesse ser lido pelo algoritmo, os dados foram dispostos de forma aleatória para que dados de mesmas classes não ficassem próximos demais uns dos outros. Com isso, é possível encontrar exemplares de todas as classes tanto no conjunto de teste quando no de treinamento.

C. Construção do Algoritmo

Para desenvolver o algoritmo, foi utilizado a linguagem de programação *Python* como auxílio do *TensorFlow*. Esse é um *framework open-source* desenvolvido pelo *Google*, que oferece suporte para o desenvolvimento de algoritmos de aprendizado de máquina. Ele pode ser utilizado nas linguagens *Python*, *C/C++*, *JavaScript* e *GO* [5]. Na própria página do TensorFlow foi disponibilizado um exemplo de classificador, que serviu e base para o desenvolvimento deste.

Na construção do algoritmo, foi necessário encontrar uma forma de transformar os dados textuais em um tipo que pudesse ser aplicado as entradas da rede neural. Esse processo é conhecido como *embedding*. De acordo com [10], o processo de *embbeding* é a criação de um vetor de números reais que representam as palavras de um dado texto e que contém algum conhecimento de posicionamento entre as palavras.

No algoritmo, utilizamos uma abordagem onde cada palavra do conjunto de dados recebe um índice. Desse modo, transformamos as frases em vetores de no máximo 50 posições, onde cada uma delas recebia um índice que corresponde a palavra que estava naquela mesma posição na frase. Se a frase possuir menos do que 50 palavras, as posições em branco receberam o índice 0, de modo que ao serem aplicadas nos neurônios de entrada, irão resultar em um valor nulo. Assim, influenciando menos no processo de treinamento.

Com os dados tratados, configuramos a rede neural artificial e elaboramos a função de treinamento. Durante o desenvolvimento, testamos várias configurações de camadas e número de neurônios. Além disso, fizemos verificações com diferentes tipos de funções de ativação, para verificar com qual era possível obter o melhor desempenho.

Por fim, o algoritmo segue os seguintes passos:

1) Carregamento do banco de dados
2) Remoção das *stopwords*
3) Processo de *steamming*
4) *Embedding* dos dados
5) Divisão entre os dados de treinamento e teste
6) Configuração da rede neural artificial, definindo função de ativação e neurônios
7) Execução do treinamento
8) Avaliação dos resultados

IV. EXPERIMENTOS E RESULTADOS

A. Metodologia dos testes

O objetivo desse trabalho é construir um classificador de desvios de linguagem, analisando qual melhor configuração de rede neural artificial pode ser aplicado para esse fim. Para isso, redes neurais com diferentes números de neurônios e funções de ativação foram testadas na mesma base de dados.

Como métricas de avaliação, será utilizada a acurácia, medida por:

$$\frac{TP + TN}{TP + FN + Tn + FP}$$

Onde TP é um verdadeiro positivo, TN é um verdadeiro negativo, FN é um falso negativo e FP é um falso positivo. Os valores são indicados de acordo com a média das repetições de cada um dos modelos de redes neurais propostos.

B. Resultados dos testes

Após a implementação do modelo proposto, coletamos testes considerando diferentes configurações camadas e números de neurônios nas redes neurais, conforme descrito pela tabela a seguir:

Camadas	Neurônios	Acurácia
2	250	66,87%
2	50	75,54%
1	250	80,06%
1	150	80,45%
1	100	83,77%
1	50	83,11%

TABLE I
RESULTADOS POR CAMADAS CONFIGURAÇÃO DE NEURÔNIOS

A coluna camadas contém número de camadas escondidas. Elas ficam entre a entrada e a saída do algoritmo. São nelas que as conexões que contém os pesos se formam. A coluna neurônios, define o número de neurônios em cada camada construída. Na coluna acurácia é apresentado o percentual de acerto de cada configuração.

Com os resultados dos testes, pode-se verificar que o modelo não melhora a acurácia com o aumento do número de camadas e neurônios. Observa-se que com uma camada de 100 o modelo atingiu o melhor caso. Conforme explicado em [12], quando se configura um número alto de neurônios, pode ocorrer o problema de *overfitting*. Quando há um número baixo de neurônios, a rede neural pode ser forçada a gastar um tempo excessivo para encontrar uma solução ótima. Com isso, podemos explicar porque 100 neurônios obtiveram o melhor desempenho, ao invés das demais configurações. Para realizar os testes, foi utilizado a função de ativação linear retificada (ReLU).

A função de ativação escolhida é muito importante para o desenvolvimento da rede neural, uma vez que seu funcionamento implica diretamente na eficiência da rede neural artificial.

A função mais utilizada, é a linear retificada, descrita pela seguinte fórmula:

$$ReLU(x) = max\{0, x\}$$

$$ReLU'(x) = \begin{cases} 1, & \text{se } x \geq 0 \\ 0, & \text{se } x < 0 \end{cases}$$

Está função é amplamente utilizada em problemas de classificação. Contudo, um problema recorrente em sua utilização é quando ocorrem pesos com valores negativos. Nesses casos, como a saída do neurônio é resultado do produto entre os pesos atribuídos a suas conexões e suas entradas, a função de ativação pode receber um parâmetro negativo. Como em seu funcionamento a função ReLU atribui 0 a valores negativos, ela acaba descartando esses casos.

Frente a esse problema, um método elaborado por [7] aprimorou a função, desenvolvendo a *Leaky Relu*, que atribui uma pequena inclinação para os valores negativos. É definida pela seguinte fórmula:

$$LeakyReLU(x, \alpha) = max\{\alpha x, x\}$$

$$LeakyReLU'(x, \alpha) = \begin{cases} 1, & \text{se } x \geq 0 \\ \alpha, & \text{se } x < 0 \end{cases}$$

Além da função *ReLU*, outra função amplamente utilizada é a função *Sigmoid*, que assume valores entre 0 e 1, apenas. É definida por:

$$\sigma(x) = \frac{1}{1 + e^x} \qquad \sigma'(x) = \sigma(x)(1 - \sigma(x))$$

A função Tangente Hiperbólica (TanH), varia entre -1 e 1, tornando-se mais atrativa do que a *Sigmoid*. É definida por:

$$tanh(x) = 2\sigma(2x) - 1 \qquad tanh'(x) = 1 - tanh^2(x)$$

Além disso, existe a função *Softmax*, que força a saída da rede neural a representar a probabilidade dos dados serem de uma das classes definidas. Ela é definida pela fórmula:

$$\phi_i = \frac{e^{z_i}}{\sum_{j \in group} e^{z_j}}$$

Para buscar a melhor abordagem, os testes foram realizados com cada um dos tipos de funções de ativação. Os resultados encontrados foram:

Função	**Acurácia**
Leaky ReLU	84,08%
ReLU	83,77%
Sigmoid	75,64%
Softmax	72,44%
TanH	82,17

TABLE II
RESULTADOS POR FUNÇÃO DE ATIVAÇÃO

Os testes foram realizados com a rede neural de 100 neurônios. Verificamos que a função *Leaky ReLU* obteve um desempenho melhor que as demais na classificação dos desvios de linguagem. Isso se dá por alguns dos valores obtidos serem negativos. Assim, como ela não os descarta, tal como a *ReLU*, esse valores não deixam de ser considerados para aprimorar os valores dos pesos da rede neural.

A saída da rede neural, então, é a probabilidade de existir cada um dos desvios de linguagem na frase de entrada. Essa pode ser dita com erro, quando uma das classes apresenta probabilidade alta de existir naquele exemplar. Como o resultado apresentando o percentual de todas as classes, é possível verificar se ocorreu mais de um erro.

V. CONCLUSÃO

Conforme o experimento e os seus resultados, conseguimos analisar como diferentes configurações de redes neurais artificiais influenciam no resultado proposto, testando diferentes modos para a resolução de um mesmo problema. Assim, foi possível desenvolver uma fundamentação teórica e prática que elenca diversos tipos de problemas encontrados e possíveis soluções.

Com os testes, verificamos que o excesso de neurônios nas camadas da rede neural artificial, pode levar ao desenvolvimento dos problemas de *overfitting*. O que mostra a importância de obter um modelo com o tamanho adequado para a rede. Além disso, para esse problema de classificação, a função de ativação *Leaky ReLU* demonstrou um melhor desempenho, uma vez que não atribui apenas 0 a valores negativos.

A generalização obtida com o treinamento, possibilita que o algoritmo seja utilizado com sucesso para encontrar desvios de linguagem em textos na língua portuguesa. Com isso, demonstramos uma aplicação de inteligência computacional em contraste com técnicas estáticas, onde os algoritmos se

baseiam em conjuntos de regras pré-definidas. Assim, o desenvolvimento de novas soluções, se torna mais eficiente, uma vez que não depende apenas do desenvolvedor prever o maior número de regras possíveis, mas sim, de obter um banco de dados completo suficiente para que um modelo possa ser treinado e obtenha uma generalização adequada.

VI. TRABALHOS FUTUROS

Para trabalhos futuros, sugerimos a aplicação de uma técnica de *emmbeding* melhorada. A aplicada ao método proposto não coloca termos com significado semelhante próximos um dos outros, algo que influencia negativamente no processo de generalização. Uma outra abordagem, seria a combinação de duas funções de ativação, como aplicar a função *Leaky ReLU* nas camadas escondidas e a *Softmax* na camada de saída, o que pode potencializar os resultados, mesclando as características positivas de ambas. Aumentar o tamanho do banco de dados também, pode melhorar os resultados obtidos. Além disso, os mesmo métodos que empregamos para a classificação de desvios de linguagens em frases na língua portuguesa, podem ser aplicados a outros idiomas e não só para desvios de linguagem, mas também para outras aplicações com estruturas similares.

REFERENCES

[1] A. Aggarwal. O corpus do português. https://www.corpusdoportugues.org, 12 2020.

[2] R. V. Alvares, A. C. B. Garcia, and I. Ferraz. Stembr: A stemming algorithm for the brazilian portuguese language. In *Portuguese conference on artificial intelligence*, pages 693–701. Springer, 2005.

[3] A. d. P. Braga, T. B. Ludermir, and A. C. P. d. L. F. Carvalho. *Redes neurais artificiais: teoria e aplicações*. LTC, 2007.

[4] I. A. Braga. Evaluation of stopwords removal on the statistical approach for automatic term extraction. In *2009 Seventh Brazilian Symposium in Information and Human Language Technology*, pages 142–149, 2009.

[5] A. J. G. Busson, L. C. Figueiredo, G. P. dos Santos, A. L. d. B. Damasceno, S. Colcher, and R. L. Milidiú. Desenvolvendo modelos de deep learning para aplicações multimídia no tensorflow. *Sociedade Brasileira de Computação*, 2018.

[6] A. C. Lorena and A. de Carvalho. Uma introdução às support vector machines. *Revista de Informática Teórica e Aplicada*, 14(2):43–67, 2007.

[7] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3. Citeseer, 2013.

[8] J. Nau, A. H. Filho, G. Passero, and V. Cavaco. Uma ferramenta para identificar desvios de linguagem na língua portuguesa (a tool to identify the linguistic deviations in the Portuguese language)[in Portuguese]. In *Proceedings of the 11th Brazilian Symposium in Information and Human Language Technology*, pages 12–16, Uberlândia, Brazil, Oct. 2017. Sociedade Brasileira de Computação.

[9] N. Oco and A. Borra. A grammar checker for Tagalog using LanguageTool. In *Proceedings of the 9th Workshop on Asian Language Resources*, pages 2–9, Chiang Mai, Thailand, Nov. 2011. Asian Federation of Natural Language Processing.

[10] C. Pari, G. Nunes, and J. Gomes. Avaliação de técnicas de word embedding na tarefa de detecção de discurso de ódio. In *Anais do XVI Encontro Nacional de Inteligência Artificial e Computacional*, pages 1020–1031. SBC, 2019.

[11] L. Paterno, C. Ferreira, M. Dosciatti, and E. Paraiso. Estudo do impacto de um corpus desbalanceado na identificação de emoções em textos. In *Estudo do Impacto de um Corpus Desbalanceado na Identificação de Emoções em Textos*, 10 2014.

[12] E. Silva and A. C. d. OLIVEIRA. Dicas para a configuração de redes neurais. *Rio de Janeiro*, 2001.