REDE NEURAL PARA IDENTIFICAÇÃO DE ERROS DE DIGITAÇÃO

André Ananias Barreto[1]

Fahme Alves Salim Junior[1]

James Clauton[2]

RESUMO

O trabalho descreve a implementação de uma Rede Neural utilizando o algoritmo de *backpropagation* para a resolução de um problema de "tema livre", cujo mesmo se diz respeito à identificação de erros de digitação recebidos por um *chatbot*. O objetivo é descobrir se uma pequena Rede Neural pode oferecer respostas coerente sobre a possibilidade de duas palavras de igual tamanho, com diferença de apenas uma letra, são resultado ou não de um erro de digitação. Após a implementação de uma Rede Neural de três neurônios com o uso de *backpropagation* foram obtido resultados positivos mostrado que a premissa é válida.

Palavras-Chave: Redes neurais; Retropropagação.

[1] Acadêmicos do 9º termo do curso de Engenharia da Computação do Centro Universitário Católico Salesiano Auxilium – UniSALESIANO de Araçatuba.

[2] Orientador

ABSTRACT

The work describes the implementation of a Neural Network using the backpropagation algorithm to solve a "free theme" problem, which the chosen one refers to the identification of mistype received by the chatbot. The objective is to find out if a small Neural Network can offer coherent answers about the possibility of two words of equal length, with a difference of only one letter, are or are not a typo. After the implementation of a Neural Network of three neurons with the use of backpropagation, positive results were obtained showing that the premise is valid.

Keywords: *Neural network; Backpropagation.*

Introdução

Ao decorrer da matéria de Sistemas Inteligentes, foi proposto pelo professor James o desenvolvimento de um projeto de inteligência artificial que resolveria um problema (sendo esse problema hipotético ou não) com escolha livre de linguagem, para a resolução do mesmo foi decretado o uso do algoritmo de *Backpropagation* como Núcleo da mesma.

Como bem sabemos, *Backpropagation* é um modelo de Redes neurais que visa como resultado a diminuição do Erro Proposto.

Há quem diga que o *Backpropagation* é o algoritmo mais importante da história das Redes Neurais pois sem o mesmo seria impossível treinar redes de aprendizagem profunda como vemos atualmente.

A mesma é composta de duas fases:

- O *Forward pass* é onde as entradas são passadas através da rede por seus neurônios e também são anunciadas as previsões de saída.
- O *Backward pass* é onde são calculados os gradientes da função perda (chamada de previsão), esse gradiente é usado recursivamente na *chain rule*, atualizando assim os pesos por toda a rede.

O problema em questão foi escolhido tendo como base o uso de *chatbots*,

estes são, em suma, robôs programados para responder mensagens de um usuário humano, um dos maiores problemas neste procedimento é a detecção de equívocos ortográficos, se a palavra escrita pelo usuário foi propositalmente escrita daquela forma, ou se o mesmo trocou algum caractere durante a sua digitação.

Este trabalho busca descobrir se uma pequena Rede Neural pode oferecer respostas coerente sobre a possibilidade de duas palavras de igual tamanho, com diferença de apenas uma letra, são resultado ou não de um erro de digitação.

A linguagem escolhida para o projeto foi *phyton* 3 devido a experiência prévia nessa linguagem e por conta da facilidade para implementação de Redes Neurais.

No caso, o algoritmo programado deve fazer uma validação de erro tendo em base a distância de cada tecla no teclado, quanto mais próximo, maior a chance do usuário ter errado, e quanto mais longe, menor as chances de erro, para fazer isso possível, utilizamos de técnicas evidenciadas nos tópicos abaixo.

Metodologia

Foi desenvolvida uma Rede Neural utilizando *Backpropagation* com dois neurônios na camada escondida e um na camada final.

A rede deve identificar, a partir de duas palavras com diferença de apenas uma letra, se a diferença se trata de um erro de digitação, retornando um número entre 0 e 1, representando a chance de o erro ser um erro cometido na hora de editar.

Para realização do cálculo, duas palavras são submetidas à lógica. Suas entradas serão então calculadas a partir das palavras. A equação 1 determina a primeira entrada, resultando em um número maior, quanto maior “tamanho” que é o tamanho da palavra em caracteres. Essa função foi escolhida pois quanto mais caracteres uma palavra tem, maiores as chances de um erro de digitação.

A segunda entrada é baseada na posição das palavras no teclado e será 1 se as letras que diferem nas palavras inseridas então lado a lado no teclado, e será zero caso contrário.

A primeira e a segunda entrada são então submetidas à Rede Neural Artificial.

$$Primeira\ entrada = \frac{tamanho - 1}{tamanho} \quad (1)$$

Como exemplo, diga-se que ao tentar escrever a palavra “soja” o indivíduo escreveu “soka”, a primeira entrada será o resultado da equação 1 com tamanho sendo 4(caracteres), resultando em 0,75. A segunda entrada será 1 já que o caractere “k” está ao lado do “j” no teclado.

A linguagem de desenvolvimento utilizada foi *python* 3, como não se podia utilizar bibliotecas complexas a única utilizada foi a “*math*” para realizar a exponenciação e nada mais.

Cada neurônio - e consequentemente a rede - pode receber duas entradas e retorna um valor entre 0 e 1, para isso foi usada a função 2 como sigmóide(função de ativação), sendo λ a inclinação da função, *s* a saída intermediária do neurônio e e o número de Neper. A saída intermediária é a somatória dos pesos multiplicados pelas entradas mais um peso multiplicado pelo bias.

$$y = \frac{1}{1 + e^{(-\lambda s)}} \quad (2)$$

Essa lógica foi implementada em uma classe chamada Neuron em uma função chamada “run”. Também foi implementada a função “update” que calcula o erro e atualiza os pesos.

Para que o erro se propague pela rede foi criada uma classe Network, que guarda cada Neuron e chama a função update para cada um de trás para frente, se repetindo até que o erro alcance o limite máximo pré estabelecido.

As lógicas que não fazem parte da Rede Neural foral colocadas no arquivo principal(“*main.py*”) que também chama a classe Network realizando treino e teste.

Os dados de treino foram simples sendo mostrados na Figura 1. Os dois primeiros valores são as duas entradas e o último a saída esperada. Observa-se que eles foram preparados de forma a dar mais peso para a segunda entrada, visto que após múltiplos testes, esta forma mostrou melhores resultados.

Figura 2 - Dados de treinamento

```
training_data = [[1, 1, 1.00],
                 [1, 0, 0.25],
                 [0, 0, 0.00],
                 [0, 1, 0.75]]
```

Fonte: elaboração própria, 2020.

Resultados e Discussão

Após treinamento com 863 iterações, foram submetidos quatro testes ao modelo.

O primeiro entre as palavras “soja” e “soka” mostra um resultado alto de 0.65 de chances de ser um erro de digitação, condizente com a realidade. Ver Figura 2.

Figura 2 - Saída do Teste 1, entre as palavras “soja” e “soka”

```
Teste: soja vs soka
Input 1: 0.75, Input 2: 1 = Output: 0.6537397105903469
```

Fonte: elaboração própria, 2020.

O segundo teste, entre as palavras “soja” e “soda” mostra um resultado baixo de 0.34 de chances de ser um erro de digitação, também condizente com a realidade visto que a palavra “soda” existe. Ver Figura 3.

Figura 3 - Saída do Teste 2, entre as palavras “soja” e “soda”

Fonte: elaboração própria, 2020.

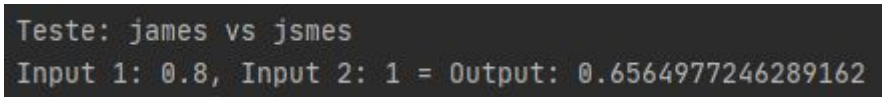
O terceiro teste, com a palavra propositalmente grande “pneumoultramicroscopicossilicovulcanoconiótico” mostra um resultado alto de 0.66 de chances de ser um erro de digitação, também condizente com a realidade. Ver Figura 4.

Figura 4 - Saída do Teste 3, a palavra “pneumoultramicroscopicossilicovulcanoconiótico” e a mesma com erro no último caractere.

Fonte: elaboração própria, 2020.

Por fim, o quarto teste, entre as palavras “james” e “jsmes” é semelhante ao primeiro e mostra um resultado alto de 0.65 de chances de ser um erro de digitação, mostrando a consistência do modelo. Ver Figura 5.

Figura 5 - Saída do Teste 4, entre as palavras “james” e “jsmes”



Fonte: elaboração própria, 2020.

Conclusão

Foi possível criar um modelo de Rede Neural consistente que ajuda a definir se duas palavras com diferença de um caractere sofreram erro de digitação ou são realmente palavras diferentes.

Futuros trabalhos podem realizar a inclusão deste modelo no fluxo de *Chatbots* para avaliar sua relevância prática.

Referências Bibliográficas

Conteúdo dado em aula pelo Professor James. Unisalesiano.

<Capítulo 15 - Algoritmo Backpropagation Parte 2 - Treinamento de Redes Neurais>
Acesso em 12 de maio de 2020.