Detecção de fake news a partir do uso de redes neurais artificiais multilayer perceptron (MLP)

Mathias Artur Schulz[1], **Daniel Gomes Soares**[1]

[1]Instituto Federal Catarinense - Campus Rio do Sul (IFC)
Rio do Sul – SC – Brasil

mathiasschulz34@gmail.com, daniel.soares@ifc.edu.br

Abstract. *This article reports how the Multilayer Perceptron (MLP) network for Fake News detection on the internet was implemented. The model was developed in the Python programming language and used as a basis for training, validation and testing a dataset with over 20,000 news. Data processing was performed from the Gensim and NLTK libraries, for training and validation the Keras library was used. Despite the challenge of converting texts into numerical representations, the network presented excellent metrics, such as R2 of 92.39 % and RMSE of 0.08.*

Resumo. *O presente artigo relata como foi implementado a rede Multilayer Perceptron (MLP) para a detecção de Falsas Notícias (Fake News) na internet. O modelo foi desenvolvido na linguagem de programação Python e utilizou como base para treinamento, validação e testes um dataset com mais de 20.000 notícias. O tratamento dos dados foi realizado a partir das bibliotecas Gensim e NLTK, para o treinamento e validação foi utilizado a biblioteca Keras. Constatou-se que, apesar do desafio da conversão de textos em representações numéricas, a rede apresentou ótimas métricas, como R2 de 92.39% e RMSE de 0.08.*

1. Introdução

As fake news sempre estiveram presentes no cotidiano das pessoas, entretanto o surgimento da internet alavancou as falsas notícias, facilitando a forma como essas notícias são disseminadas entre as pessoas. Fake news são notícias que possuem como principal objetivo espalhar conteúdos falsos com a intenção de obter vantagens, como financeira, eleitoral, entre outros (Carvalho, 2019; Kanffer, 2019).

Segundo o Dicionário de Cambridge o conceito falsas notícias (fake news) indica histórias falsas que possuem a aparência de notícias jornalísticas. A partir da sua aparência e grande semelhança a uma notícia verdadeira, são disseminadas pela internet e as mídias sociais, podendo ser usadas para influenciar as pessoas, por exemplo em posições políticas (Carvalho, 2019; Kanffer, 2019).

O presente artigo possui como objetivo a apresentação de uma rede neural artificial multilayer perceptron para a detecção de fake news, será apresentado o que são as redes neurais artificiais (RNAs) e também dois tipos de RNAs, a perceptron e a multilayer perceptron, será explicado um pouco sobre a disseminação e identificação de fake news, a forma de tratamento dos textos para entrada no modelo, treinamento, validação, testes e métricas de avaliação da rede criada, além disso será apresentado os resultados obtidos e o código documentado.

2. Redes Neurais Artificiais

As Redes Neurais Artificiais (RNA) são técnicas computacionais que apresentam um modelo matemático inspirado na estrutura neural de organismos inteligentes, no qual a obtenção de conhecimento é realizada através da experiência. Uma rede neural artificial complexa, pode conter entre centenas ou milhares de neurônios, entretanto o cérebro de um mamífero, por exemplo, contêm bilhões de neurônios (André, 2019).

As redes neurais artificiais são uma abstração das redes neurais biológicas, seu objetivo não é replicar, contudo servir de modelo para o aprendizado e resoluções de problemas complexos, sendo moldada a partir de características biológicas, como: alto paralelismo, robustez, tolerância a falhas e aprendizado através de padrões (Grübler, 2018).

O sistema nervoso é formado por um conjunto extremamente complexo de células, chamados neurônios. No qual possuem um papel essencial na determinação do funcionamento e comportamento do corpo humano e do raciocínio. Os neurônios se comunicam através de sinapses, sendo a região onde dois neurônios entram em contato e, com isso, permitem o envio de impulsos nervosos entre os neurônios (André, 2019).

As RNA's vem sendo amplamente utilizadas em diversas áreas de pesquisa e permitem auxiliar em diversos tipos de problemas, como: Processamento de linguagem natural; Reconhecimento de fala e imagens; Previsão de valores; Auxílio na tomada de decisões (Grübler, 2018).

2.1. Perceptron

A Rede Neural Artificial Perceptron foi introduzida por Frank Rosenblatt em 1958, sendo um dos modelos mais antigos e trabalha com apenas um neurônio, sendo assim, o resultado é apresentado de forma linear. A rede perceptron recebe recebe os valores de entrada y(n), os valores de entrada são multiplicados pelos pesos da sinapse w, após a multiplicação são somados com um conjunto de entrada, como apresentado abaixo (Grübler, 2018).

$$\varepsilon = \sum w * y(n)$$

Após a soma passam por uma função de ativação e transmitem a saída, caso a soma ultrapasse o valor do limite da função de ativação, o neurônio é ativado e o valor é retornado (Grübler, 2018).

2.2. Multilayer perceptron

Redes de uma camada, como a rede perceptron, não são capazes de solucionar problemas que não sejam linearmente separáveis. Com isso, em 1986 foi desenvolvimento o algoritmo de treinamento backpropagation, comprovando que é possível treinar eficientemente redes com camadas intermediárias, resultando em modelo de Redes Neurais Artificiais amplamente utilizadas, como as redes Perceptron Multicamadas (MLP) (Grübler, 2018).

A rede MLP possui a camada de entrada, que recebe os padrões, as camadas intermediárias, que extraem características e codificam as características em pesos e a camada de saída, que recebe os estímulos da camada intermediária e constrói o padrão que será a resposta, como pode ser observado na figura 1 abaixo, na qual apresenta as camadas do modelo da rede MLP com os seus respectivos neurônios (Grübler, 2018).

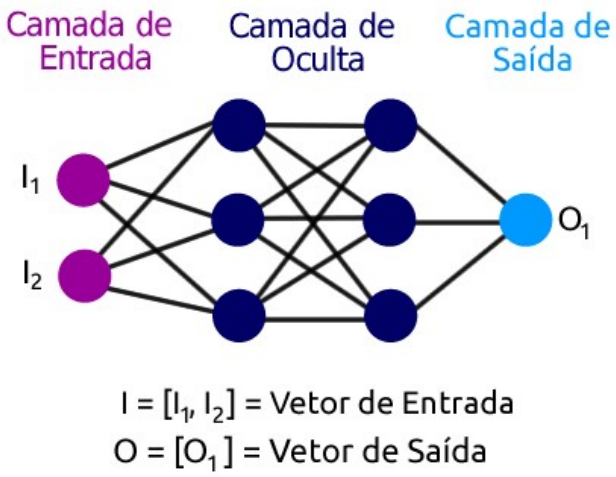

Figura 1. Rede MLP com duas camadas intermediárias (Grübler, 2018)

O treinamento das redes que utilizam o algoritmo backpropagation possui duas etapas. A primeira etapa é a apresentação de um padrão para a camada de entrada da rede, que flui camada por camada até a camada de saída, que apresenta a resposta obtida. A segunda etapa é a comparação da resposta obtida com o resposta desejada, caso não esteja correta, o erro é calculado e propagado a partir da camada de saída até a camada de entrada com o objetivo de ajustar os pesos das das camadas internas (Grübler, 2018).

3. Disseminação e identificação de fake news

Diversos fatores influenciam na disseminação de falsas notícias, como a sua origem, o grau de credibilidade das pessoas que a disseminaram ou as pessoas que a referendaram (como influenciadores digitais), bem como a quantidade de pessoas que estão disseminando a informação (Carvalho, 2019; Kanffer, 2019).

Segundo Komatsu e Sanches (2018) uma pesquisa realizada pela Associação dos Especialistas de Políticas Públicas de São Paulo (AEPPSP) foi identificado oito características em comum que se encaixam no perfil de sites disseminadores de Fake News, essas características são:

1. Foram registrados sem .br no final, o que dificulta a identificação de seus responsáveis
2. Não possuem em sua página qualquer ferramenta que possa identificar quem são seus administradores e jornalistas
3. As notícias não são assinadas
4. Notícias sem imparcialidade, algumas possuindo até mesmo discurso de ódio
5. Intenso fluxo de notícias, a cada poucos minutos ou horas
6. Os sites e páginas possuem nomes semelhantes com os de outros sites jornalísticos bastante conhecidos
7. Layouts cheios de conteúdo dando a falsa impressão de serem um grande site
8. Páginas cheias de anúncios comercias

Ainda segundo Komatsu e Sanches (2018) uma pesquisa realizada na Universidade de Oxford apresenta que grande parte do tráfego da internet é realizado por “bots”,

programas que simulam ações humanas na internet e que possuem a capacidade de tornar um assunto tendência e espelharem falsas notícias, tornando-se uma ferramenta poderosa.

Os bots atuam como pessoas e são guiados, por exemplo, por meio de hashtags (#) que circulam pelas redes sociais, uma maneira fácil de encontrar usuários que estão conversando sobre determinado assunto. Outra forma de criação de falsas notícias pode ser realizada pelos próprios internautas, a partir da criação de sites e perfis falsos para criação de fake news.

4. Rede MLP para detecção de fake news

Nesse capítulo é apresentado o processo de tratamento, treinamento, validação e teste do modelo MLP para detecção de fake news. O dataset utilizado para construção do modelo se encontra disponível no seguinte link: `https://www.kaggle.com/mohit28rawat/fake-news`.

4.1. Tratamento dos dados

Para o tratamento dos dados que serão utilizados no modelo, foram utilizados os dados apresentados no arquivo *train.csv* disponível no dataset de fake news. Inicialmente o arquivo possui 5 colunas, que são: 'id', 'title', 'author', 'text' e 'label', entretanto foram utilizados apenas as colunas 'text', onde se encontra o texto da notícia e 'label' onde se encontra um valor 0 ou 1 para determinar se a notícia é uma fake news ou não.

O dataset *train.csv* possui um total de 20800 registros, entretanto é necessário realizar a retirada de registros que possuam uma ou as duas colunas 'text' e 'label' vazias. Com isso, foi utilizado o comando 'dropna' da biblioteca Pandas para remoção de registros com algum campo vazio, após a utilização do comando restaram 20761 registros.

Para o treinamento de uma rede multilayer perceptron é necessário que os dados de entrada no modelo sejam, obrigatoriamente, valores numéricos, o que torna um desafio para a detecção de fake news, no qual possui como dados para cada registro, o texto da notícia e a determinação se é uma fake news ou não. Com isso, foram utilizados duas bibliotecas disponíveis para a linguagem de programação Python, chamadas Gensim e NLTK, que auxiliaram na conversão do texto para valores numéricos.

4.1.1. Biblioteca Gensim

O Gensim começou como uma coleção de vários scripts Python em 2008, servindo como base para gerar uma pequena lista dos artigos mais semelhantes a um determinado artigo. Atualmente, o Gensim é uma biblioteca robusta do Python para a realização de modelagem semântica não supervisionada a partir de texto sem formatação (Řehůřek, 2019).

O Gensim foi desenvolvido principalmente para modelagem de tópicos. No entanto, atualmente ele suporta uma variedade de outras tarefas do processamento de linguagem natural (PNL), como converter palavras em vetores (Word2Vec), documento em vetores (Doc2Vec), encontrar semelhança e resumo de texto. O público-alvo do Gensim é a comunidade de processamento de linguagem natural (PNL) e recuperação de informações (IR) (Malik, 2019; Penkov, 2019).

O Word2Vec é um modelo disponibilizado pela biblioteca Gensim e é usado para gerar vetores de representação a partir de palavras, ele fornece uma representação numérica para cada palavra, capaz de capturar relações, fazendo parte de um conceito mais amplo de aprendizado de máquina, os vetores de recursos. As representações encapsulam diferentes relações entre palavras, como sinônimos, antônimos ou analogias (Shperber, 2017).

A suposição implícita do Word2Vec é que duas palavras que compartilham contextos semelhantes também compartilham um significado semelhante e, sendo assim, uma representação vetorial similar do modelo (Kanoki, 2019).

Ao utilizar uma simples codificação de palavras, elas perdem o significado. Por exemplo, se codificarmos 'Paris' como id_4, 'França' como id_6 e 'poder' como id_8, todas as palavras possuirão a mesma relação. Entretanto, preferimos uma representação em que a 'França' e 'Paris' estejam mais próximas do que a 'França' e o 'poder' (Shperber, 2017).

Outro exemplo seriam as palavras 'banco', 'dinheiro' e 'contas', elas são frequentemente usadas em situações semelhantes, e também são utilizadas com outras palavras similares, como 'real', 'empréstimo' ou 'crédito'. Com isso, de acordo com o Word2Vec, eles compartilham uma representação vetorial semelhante (Kanoki, 2019).

A partir da suposição realizada pelo Word2Vec, ela pode ser utilizada para descobrir as relações entre palavras em um conjunto de dados, calcular a semelhança entre elas ou usar a representação vetorial dessas palavras como entrada para outros aplicativos, como classificação de texto (Kanoki, 2019).

A representação numérica de textos é uma tarefa desafiadora na inteligência artificial e no aprendizado de máquina. Com isso, o método Doc2Vec é um conceito apresentado em 2014 por Quoc V. Le e Tomas Mikolov, é uma técnica muito boa, fácil de usar, oferece bons resultados e é baseado no Word2Vec. O Doc2Vec também é um modelo da biblioteca Gensim, no qual representa cada documento como um vetor, independente do seu tamanho (Kanoki, 2019; Řehůřek, 2019; Shperber, 2017).

O modelo Doc2Vec foi utilizado para realizar o tratamento das notícias sobre fake news, o modelo é treinado e utilizado para criar uma representação numérica de sentenças, parágrafos e documentos. Diferentemente do Word2Vec que calcula um vetor de recurso para cada palavra, o Doc2Vec calcula um vetor de recurso para cada documento. Os vetores gerados pelo doc2vec podem ser usado para tarefas como encontrar semelhança entre frases, parágrafos e documentos (Kanoki, 2019; Řehůřek, 2019; Shperber, 2017).

4.1.2. Biblioteca NLTK

Outra biblioteca utilizada foi a Natural Language Toolkit (NLTK), a NLTK é uma biblioteca utilizada para a criação de programas Python para trabalhar com dados da linguagem humana. A partir dela é possível realizar o processamento de texto para classificação, tokenização, stemming, marcação, análise e raciocínio semântico (NLTK, 2019).

A partir da biblioteca NLTK, foi utilizado o pacote Stopwords, para a retirada

de stopwords dos textos antes de serem tratados no modelo Doc2Vec. Stopwords são palavras que possuem apenas significado sintático dentro de uma sentença, entretanto não acrescentam informações importantes sobre o seu sentido, não agregam muito valor ao significado do documento (Lima, 2017; Singh, 2019).

As stopwords devem ser retiradas pois estão em grande quantidade nos textos, independente do idioma do texto, caso não sejam retiradas o modelo Doc2Vec avaliará também palavras como: 'the', 'is', 'in', 'for', 'where', 'when', 'to', 'at', entre outras, prejudicando o desempenho do modelo (Lima, 2017; Singh, 2019).

Por exemplo a frase: 'There is a book on the table', as palavras 'is', 'a', 'on' e 'the' não acrescentam significado à frase enquanto é analisada, entretanto as palavras 'there', 'book' e 'table' são consideradas palavras-chave na frase (Singh, 2019).

4.2. Treinamento, Validação e Teste

Após os dados já tratados e representados numericamente podem ser utilizados na rede MLP. Primeiramente é realizado a divisão dos dados para treino, validação e depois para teste da rede. Os dados de treinamento correspondem a 70% de todo o conjunto, os dados de validação correspondem a 20% de todo o conjunto e os dados de teste da rede correspondem a 10% de todo o conjunto.

No final do tratamento dos dados, o resultado é um array, no qual cada posição do array representa um texto, o texto também é representado por um array, um array de palavras, no qual cada posição é uma palavra tratada do texto. O array de palavras do texto possui 300 posições, no qual é pego as 300 primeiras palavras do texto, sem contar as stopwords.

Como cada texto é um array com 300 posições, isso representa que os dados de entrada da rede MLP possuirão uma dimensão igual a 300.

4.2.1. Métricas de avaliação

Para cada configuração testada na rede MLP é necessário possuir métricas de avaliação da qualidade dos resultados obtidos, como indicadores de erro e indicadores de desempenho geral, as métricas utilizadas são apresentadas abaixo.

- **Erro Médio Quadrático da Raiz**

O Erro Médio Quadrático da Raiz (RMSE - Root Mean Square Error) é uma medida utilizada a partir da diferença entre os valores previstos por um modelo e os valores realmente observados no ambiente que está sendo modelado, essa diferença entre os valores reais e os valores previstos são chamados de resíduos, a figura 2 abaixo apresenta a fórmula de cálculo do RMSE (Holmes, 2000).

$$RMSErrors = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}}$$

Figura 2. Fórmula RMSE (Holmes, 2000)

Os resíduos podem ser positivos ou negativos, pois o valor previsto pode estar acima ou abaixo do valor estimado. Com isso, como pode ser visualizado na figura 2 acima, cada resíduo é elevado ao quadrado, após é calculado a média dos quadrados e obtido a raiz quadrada, com isso é obtido o RMSE. Em seguida, o RMSE pode ser utilizado como uma medida da dispersão dos valores y sobre o valor y previsto (Holmes, 2000).

- **Erro Médio Percentual Absoluto**

O Erro Médio Percentual Absoluto (MAPE) é uma medida para precisão da previsão, é a média dos erros percentuais absolutos (APE). A figura 3 abaixo apresenta a fórmula do MAPE (S. Kim, 2016; H. Kim, 2016).

$$\mathrm{MAPE} = \frac{1}{N} \sum_{t=1}^{N} \left| \frac{A_t - F_t}{A_t} \right|$$

Figura 3. Fórmula MAPE (S. Kim, 2016; H. Kim, 2016)

De acordo com a figura 3 acima, At e Ft denotam os valores reais e os valores previstos no dado atual t, N é o número de dados. Por fim, toda a equação deve ser multiplicada por 100, para ser mais rigoroso. O MAPE é independente de escala e fácil de interpretar, entretanto possui a chance de produzir valores infinitos ou indefinidos quando os valores reais são zero ou próximos a zero, o que é comum em algumas áreas (S. Kim, 2016; H. Kim, 2016).

- **Coeficiente de Determinação**

O Coeficiente de Determinação (R2) é um valor entre 0 e 1. Um valor de R2 perto de 1 indica que a maior parte da variação dos dados de resposta é explicado pelos diferentes valores de entrada, enquanto que um valor de R2 perto de 0 indica que pouca variação é explicada pelos diferentes valores de entrada (Ross, 2010).

Por exemplo, dados que relacionam a altura de um filho à altura de seu pai, o cálculo de R2 resultou em 0.96, ou seja, 96% da variação das alturas de todos os indivíduos é explicada pelas alturas de seus pais. Entretanto, os outros 4% restantes da variação não são inexplicáveis a partir da altura de seus pais (Ross, 2010).

O valor de R2 é utilizado como um indicador de quão bem o modelo ajusta aos dados, um valor próximo de 1 indica um bom ajuste e um valor perto de 0 indica um ajuste ruim (Ross, 2010).

4.3. Resultados

Neste capitulo é apresentado o resultado dos testes realizados com a rede MLP a partir de diferentes configurações montadas na rede. Para cada teste da tabela abaixo, a rede foi treinada, validada e o resultado dos testes podem ser visualizados abaixo.

Cada coluna da tabela significa: Teste, número do teste; Loss, representa a perda, quanto menor a perda, mais próximas as previsões são dos rótulos verdadeiros; R2, R2 Detecção, MAPE, RMSE, são métricas de avaliação da qualidade da rede; Épocas, representa a quantidade de épocas utilizadas na rede MLP; N.E., representa a quantidade

de neurônios utilizadas na camada de entrada; N.I., representa a quantidade de neurônios utilizados nas camadas intermediárias; C.I., representa a quantidade de camadas intermediárias; Tempo, apresenta o tempo necessário para a rede realizar o treinamento, validação e teste, sem contar o tempo da tratamento dos dados.

Tabela 1. Resultados de cada teste da rede após o treinamento e validação

Teste	Loss	R2 (%)	R2 Detecção (%)	MAPE	RMSE	Épocas	N.E.	N.I.	C.I.	Tempo (m)
1	0.60	90.52	95.61	49428800.00	0.10	150	12	8	1	3.90
2	0.35	91.38	95.73	38085868.00	0.09	150	12	8	3	4.62
3	1.33	90.23	96.72	51110684.00	0.10	150	12	120	3	5.66
4	3.89	91.86	97.62	42114624.00	0.08	150	120	120	3	7.00
5	9.00	91.24	97.50	50734172.00	0.09	150	120	300	3	14.15
6	1.11	89.41	95.90	44423496.00	0.11	300	12	8	1	8.58
7	1.21	92.39	97.59	36165672.00	0.08	150	151	128	1	6.54
8	5.18	92.30	97.77	37236172.00	0.08	150	151	128	3	7.82

De acordo com a tabela acima, percebe-se que todos os testes obtiveram resultados semelhantes. Entretanto, o teste 7 quando comparado aos outros 7 testes, aparenta possuir um desempenho melhor, possui o 4º melhor Loss, o melhor R2, o 3º melhor R2 na detecção, o melhor MAPE, o melhor RMSE e possui um bom tempo de execução.

A figura 4 abaixo apresenta o RMSE do treino e da validação do teste 7.

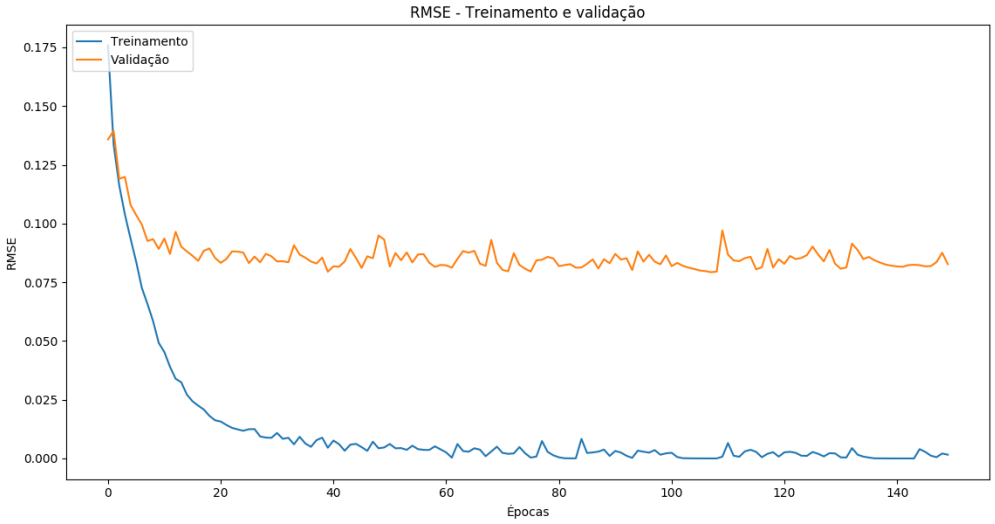

Figura 4. RMSE da rede do teste 7 durante treino e validação

A figura 5 abaixo apresenta o MAPE do treino e da validação do teste 7.

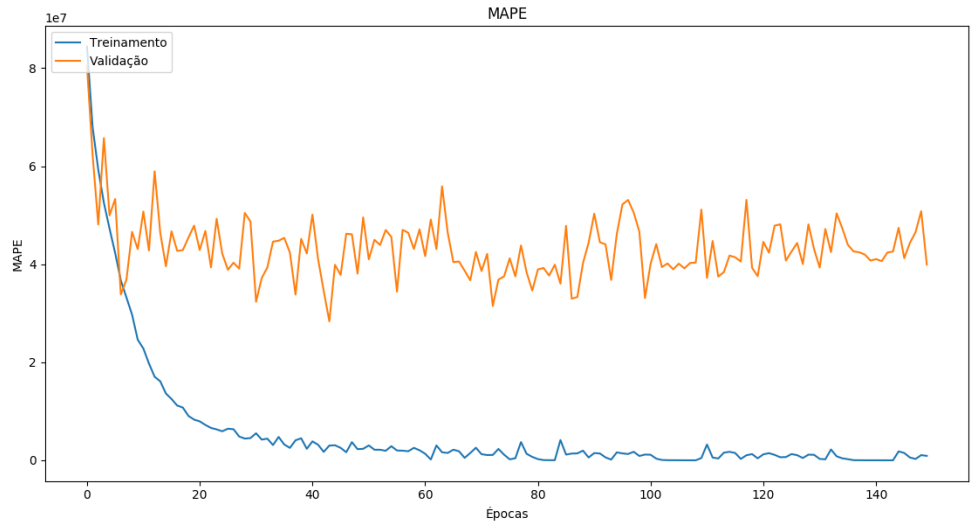

Figura 5. MAPE da rede do teste 7 durante treino e validação

A figura 6 abaixo apresenta o R2 do treino e da validação do teste 7.

Figura 6. R2 da rede do teste 7 durante treino e validação

Como pode ser observado nos testes, a detecção para fake news está com ótimos resultados, entretanto também percebe-se que com o passar das épocas, todas as métricas passaram a se estabilizar em um valor, isso ocorreu com todos os testes realizados, em alguns casos as métricas chegaram a estabilizar em um determinado valor e não evoluíram mais, como é o caso do RMSE para o teste 8 apresentado na figura 7 abaixo.

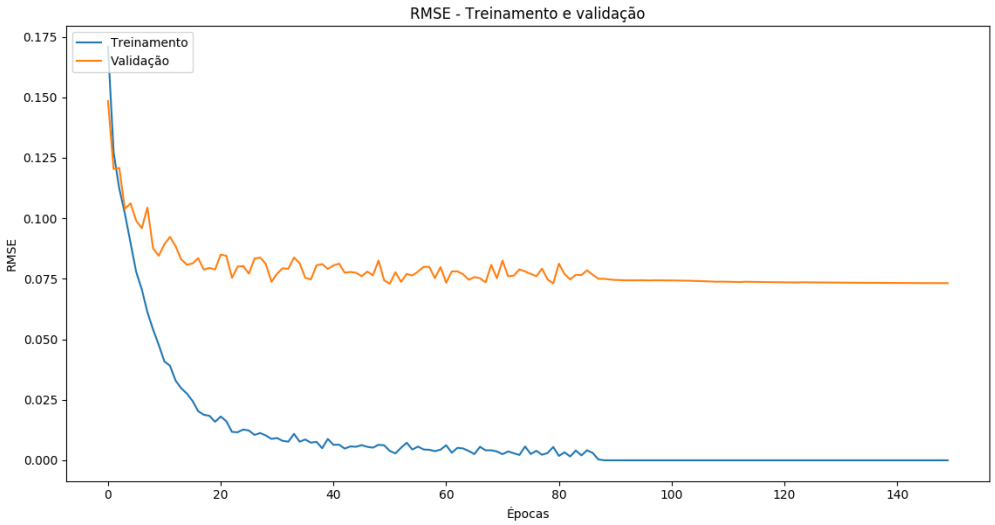

Figura 7. RMSE da rede do teste 8 durante treino e validação

Como pode ser observado na figura 7 acima, o RMSE se estabilizou em 0, entretanto durante a validação o RMSE subiu para aproximadamente 0,08 e durante os testes o RMSE estabilizou em 0,08.

4.4. Código Documentado

Neste capítulo é apresentado o código para criação da rede MLP desenvolvido na linguagem de programação Python, com explicação presente no código.

O primeiro arquivo .py criado é o 'mlp-data-processing.py', responsável por ler o dataset, realizar o tratamento dos dados e salvar os dados tratados no arquivo json 'processed-data-train.json', o arquivo py pode ser visualizado abaixo.

```
1  import re, string, time, codecs, json
2  import numpy as np
3  import pandas as pd
4  from gensim.models import Doc2Vec
5  from gensim.models.doc2vec import TaggedDocument
6  from gensim import utils
7  from nltk.corpus import stopwords
8  import warnings
9  warnings.filterwarnings("ignore")
10
11
12 # Metodo responsavel por realizar a limpeza do texto
13 def textClean(text):
14     # Realiza a substituicao de todos os caracteres diferentes
       do regex abaixo
15     text = re.sub(r"[^A-Za-z0-9^,!.\/'+-=]", " ", text)
16     # Coloca o texto em lowercase e realiza um slip no espaco
```

```
    text = text.lower().split()
    # Remove stopwords
    stops = set(stopwords.words("english"))
    text = [w for w in text if not w in stops]
    text = " ".join(text)
    # Atualiza a string sem os caracteres de pontuacao
    text = text.translate(str.maketrans("", "", string.
       punctuation))
    return text


# Realiza a construcao do array de sentencas que sera utilizado
   no gensim doc2vec
def constructSentences(data):
    sentences = []
    for index, row in data.iteritems():
        # Converte para um formato legivel para o computador
        sentences.append(TaggedDocument(utils.to_unicode(row).
           split(), [str(index)]))
    return sentences


# Metodo responsavel por realizar o processamento dos textos
# Convertendo o texto para um formato numerico
def dataProcessing(data, vector_dimension=300):
    # Realiza a limpeza de cada registro
    for i in range(len(data)):
        data.loc[i, 'text'] = textClean(data.loc[i,'text'])

    # Realiza a construcao das sentencas
    x = constructSentences(data['text'])
    y = data['label'].values

    # Modelo Doc2Vec
    model = Doc2Vec (
        min_count=1,
        window=5,
        vector_size=vector_dimension,
        sample=1e-4,
        negative=5,
        workers=7,
        epochs=10,
        seed=1
    )
    model.build_vocab(x)
    model.train(x, total_examples=model.corpus_count, epochs=
       model.iter)

    # Converte os dados numericos para um array numpy
```

```
62      x = np.zeros((len(model.docvecs), vector_dimension), dtype=
            float)
63      for i in range(len(model.docvecs)):
64          x[i] = model.docvecs[str(i)]
65
66      return x, y
67
68
69  # Realiza a leitura do dataset e ja realiza a remocao dos
        registros com campos vazios
70  def getDataset(path):
71      data = pd.read_csv(path)
72      # Pega as colunas 'text' e 'label'
73      data = data.iloc[:, [3, 4]]
74      # Dropa todas os registros que possuem campos vazios
75      data.dropna(inplace = True)
76      # Atualiza os index com as linhas removidas
77      data = data.reset_index(drop = True)
78      return data
79
80
81  # Gravacao dos dados tratados em um json
82  def writeJson(x, y, file_path):
83      data = {
84          'y': y.tolist(),
85          'x': x.tolist()
86      }
87      json.dump(
88          data, codecs.open(file_path, 'w', encoding='utf-8'),
                separators=(',', ':'), sort_keys=True, indent=4
89      )
90
91
92
93
94  PATH = './dataset/train.csv'
95  JSON_NAME = './dataset/processed-data-train.json'
96  VECTOR_DIMENSION = 300
97
98  execucaoInicio = time.time()
99  print('### Fake News detection - Tratamento dos dados')
100
101 print('Leitura do dataset... ')
102 data = getDataset(PATH)
103
104 print('Tratamento dos dados... ')
105 x, y = dataProcessing(data, VECTOR_DIMENSION)
106
107 print('Gravando os dados tratatos no arquivo JSON... ')
```

```
writeJson(x, y, JSON_NAME)

# Calculo do tempo de execucao
execucaoFim = time.time()
tempoExecucao = (execucaoFim - execucaoInicio) / 60

print('Dados tratados e salvo no JSON com sucesso! ')
print('Tempo de execucao do tratamento dos dados: %.2f minutos'
    % tempoExecucao)
```

O segundo arquivo .py criado é o 'mlp-fake-news.py', responsável por ler o arquivo json 'processed-data-train.json', realizar o treinamento, validação e teste da rede MLP, além disso também é responsável por apresentar os resultados, o arquivo py pode ser visualizado abaixo.

```
import codecs, json, os.path, sys, time
import numpy as np
from keras.models import Sequential
from keras.layers import Dense
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings("ignore")
import matplotlib.pyplot as plt
from keras import backend


# Realiza a leitura do arquivo json com o dados tratados
def readJson(file_path):
    if (not os.path.isfile(file_path)):
        return None, None

    json_string = codecs.open(file_path, 'r', encoding='utf-8').
        read()
    json_data = json.loads(json_string)

    # Transforma os dados json em um array numpy
    x = np.array(json_data['x'])
    y = np.array(json_data['y'])

    return x, y


# Metodo responsavel por realizar o calculo do RMSE
def rmse(y_true, y_pred):
    return backend.sqrt(backend.mean(backend.square(y_pred -
        y_true), axis = -1))


```

```
# Criacao do modelo MLP
def createModelMLP(vector_dimension = 300):
    # Variaveis auxiliares para serem apresentadas no final da
       execucao
    # Quantidade de neuronio da camada de entrada
    input_layer_quantity_neuron = 12
    # Quantidade de neuronio das camadas intermediarias
    hidden_layer_quantity_neuron = 8
    # Quantidade de camadas intermediarias
    hidden_layer_quantity = 1

    # Criacao do modelo

    # Camada de Entrada
    model = Sequential()
    model.add(Dense(
        input_layer_quantity_neuron,
        input_dim = vector_dimension,
        kernel_initializer = 'uniform',
        activation = 'relu'
    ))

    # Camadas intermediarias
    model.add(Dense(
        hidden_layer_quantity_neuron,
        kernel_initializer = 'uniform',
        activation = 'relu'
    ))

    # Camada de saida
    model.add(Dense(
        1,
        kernel_initializer = 'uniform',
        activation = 'sigmoid'
    ))

    # Compilacao do modelo com as metricas: R2, RMSE e MAPE
    model.compile(loss = 'binary_crossentropy', optimizer = '
       adam', metrics = ['accuracy', rmse, 'mape'])
    return model, input_layer_quantity_neuron,
       hidden_layer_quantity_neuron, hidden_layer_quantity


# Metodo responsavel por realizar o treinamento e validacao da
   MLP
def train(model, x_train, y_train, x_val, y_val, epochs = 150):
    # Treinamento do modelo
    history = model.fit(x_train, y_train, epochs = epochs,
       batch_size = 10, validation_data = (x_val, y_val))
```

```
76      return model, history
77
78
79  # Metodo responsavel por realizar o teste da MLP
80  def test(model, x_test, y_test, x_data, y_data):
81      # Avalia o modelo com os dados de teste
82      loss, accuracy_model, rmse, mape = model.evaluate(x_test,
            y_test)
83
84      # Gera as deteccoes se cada noticia e fake ou nao
85      detections = model.predict(x_data)
86
87      # Ajusta as deteccoes
88      rounded = [round(x[0]) for x in detections]
89      accuracy_detection = np.mean(rounded == y_data)
90
91      return loss, accuracy_model, rmse, mape, accuracy_detection
92
93
94
95
96  JSON_NAME = './dataset/processed-data-train.json'
97  VECTOR_DIMENSION = 300
98  EPOCHS = 150
99
100 execucaoInicio = time.time()
101 print('### Fake News detection - Treinamento, validacao e teste
        da MLP\n')
102
103 x, y = readJson(JSON_NAME)
104 if ((x is None) or (y is None)):
105     print('Fim de Execucao! ')
106     print('Antes de treinar e testar a rede neural realize o
            tratamento dos dados! ')
107     sys.exit()
108
109 # Divisao dos dados para: Treinamento -> 70%, validacao -> 20% e
        teste -> 10%
110 x_train, x_val, y_train, y_val = train_test_split(x, y,
        test_size = 0.3)
111 x_val, x_test, y_val, y_test = train_test_split(x_val, y_val,
        test_size = 0.333333)
112
113
114 # Criacao do modelo MLP
115 model, input_layer_neuron, hidden_layer_neuron,
        hidden_layer_quantity = createModelMLP(VECTOR_DIMENSION)
116 # Treinamento e validacao do modelo
117 model, history = train(model, x_train, y_train, x_val, y_val,
```

```
    EPOCHS)
# Teste do modelo
loss, accuracy_model, rmse, mape, accuracy_detection = test(
    model, x_test, y_test, x, y)

# Calculo do tempo de execucao
execucaoFim = time.time()
tempoExecucao = (execucaoFim - execucaoInicio) / 60


print("\n\n### Resultados do teste da rede: ")
# Quanto menor a perda, mais proximas nossas previsoes sao dos
    rotulos verdadeiros.
print("Loss: %.2f" % loss)
print("R2: %.2f%%" % (accuracy_model * 100))
print("R2 Deteccoes: %.2f%%" % (accuracy_detection * 100))
print("MAPE: %.2f" % mape)
print("RMSE: %.2f" % rmse)
print("###")
print("QTD registros: %i " % len(x))
print("QTD registros treino: %i " % len(x_train))
print("QTD registros validacao: %i " % len(x_val))
print("QTD registros teste: %i " % len(x_test))
print("QTD Epocas: %i" % EPOCHS)
print("QTD neuronios camada de entrada: %i" % input_layer_neuron
    )
print("QTD neuronios camadas intermediarias: %i" %
    hidden_layer_neuron)
print("QTD de camadas intermediarias: %i" %
    hidden_layer_quantity)
print("Tempo de Execucao: %.2f minutos" % tempoExecucao)


# Apresentacao dos graficos de treinamento e validacao da rede
plt.plot(history.history['rmse'])
plt.plot(history.history['val_rmse'])
plt.title('RMSE - Treinamento e validacao')
plt.xlabel('Epocas')
plt.ylabel('RMSE')
plt.legend(['Treinamento', 'Validacao'], loc='upper left')
plt.show()

plt.plot(history.history['mape'])
plt.plot(history.history['val_mape'])
plt.title('MAPE')
plt.xlabel('Epocas')
plt.ylabel('MAPE')
plt.legend(['Treinamento', 'Validacao'], loc='upper left')
plt.show()
```

```
161
162 plt.plot(history.history['accuracy'])
163 plt.plot(history.history['val_accuracy'])
164 plt.title('R2 - Treinamento e validacao')
165 plt.xlabel('Epocas')
166 plt.ylabel('R2')
167 plt.legend(['Treinamento', 'Validacao'], loc='upper left')
168 plt.show()
```

5. Conclusão

As fake news sempre estiveram presentes na vida das pessoas e são muito perigosas, pois possuem o poder de manipular as pessoas. Entretanto com o surgimento da internet e as mídias sociais, grande parte das pessoas estão conectadas, o tráfego de informações é extremamente grande, o que acaba dificultando a identificação de falsas notícias.

A rede desenvolvida obteve um ótimo resultado e consegue detectar falsas notícias com uma boa qualidade e ótimas métricas, como R2 de 92.39% e RMSE de 0.08. Entretanto, as notícias sofrem modificações e evoluem com o tempo, assim como as fake news tendem a ficar mais complexas e mais semelhantes a notícias verídicas, por isso é necessário a constante atualização do dataset de treinamento, para que a inteligência artificial não fique defasada.

6. Referências

CARVALHO, Gustavo A.C.L.; KANFFER, Gustavo G.B.; **O Tratamento Jurídico das Notícias Falsas (fake news)**; Disponível em: https://www.conjur.com.br/dl/tratamento-juridico-noticias-falsas.pdf; Acesso em: 21/11/2019 às 08:19;

KOMATSU, Juliana P.; SANCHEZ, Cláudio J.P.; **NOTÍCIAS FALSAS E SEU IMPACTO NO MUNDO POLÍTICO**; Disponível em: http://intertemas.toledoprudente.edu.br/index.php/ETIC/article/view/7128/67647229; Acesso em: 21/11/2019 às 08:50;

CARVALHO, André P.L.F.; **Redes Neurais Artificiais**; Disponível em: http://conteudo.icmc.usp.br/pessoas/andre/research/neural/; Acesso em: 20/11/2019 às 11:09;

GRÜBLER, Murillo; **Entendendo o funcionamento de uma Rede Neural Artificial**; Disponível em: https://medium.com/brasil-ai/entendendo-o-funcionamento-de-uma-rede-neural-artificial-4\tolerance9999\emergencystretch3em\hfuzz.5\p@\vfuzz\hfuzz463fcf44dd0; Acesso em: 20/11/2019 às 21:32;

REHUREK, Radim; **Gensim = "Generate Similar"**; Disponível em: https://radimrehurek.com/gensim/about.html; Acesso em: 20/11/2019 às 22:31;

PENKOV, Michael; **gensim 3.8.1**; Disponível em: https://pypi.org/project/gensim/; Acesso em: 20/11/2019 às 22:35;

MALIK, Usman; **Python for NLP: Working with the Gensim Library (Part 1)**; Disponível em: https://stackabuse.com/python-for-nlp-working-with-the-gensim-library-part-1/; Acesso em: 20/11/2019 às 22:38;

SHPERBER, Gidi; **A gentle introduction to Doc2Vec**; Disponível em: https://medium.com/wisio/a-gentle-introduction-to-doc2vec-db3e8c0cce5e; Acesso em: 21/11/2019 09:05;

KANOKI; **Sentence Similarity in Python using Doc2Vec**; Disponível em: https://kanoki.org/2019/03/07/sentence-similarity-in-python-using-doc2vec/; Acesso em: 21/11/2019 09:33;

NLTK; **Natural Language Toolkit**; Disponível em: http://www.nltk.org/; Acesso em: 23/11/2019 14:44;

LIMA, Vinícius R.; **Utilizando processamento de linguagem natural para criar um sumarização automática de textos**; Disponível em: https://medium.com/@viniljf/utilizando-processamento-de-linguagem-natural-para-criar-u\tolerance9999\emergencystretch3em\hfuzz.5\p@\vfuzz\hfuzzm-sumariza%C3%A7%C3%A3o-autom%C3%A1tica-de-textos-775cb428c84e; Acesso em: 23/11/2019 14:50;

SINGH, Shubham; **NLP Essentials: Removing Stopwords and Performing Text Normalization using NLTK and spaCy in Python**; Disponível em: https://www.analyticsvidhya.com/blog/2019/08/how-to-remove-stopwords-text-normalization-nltk-spacy-gens\tolerance9999\emergencystretch3em\hfuzz.5\p@\vfuzz\hfuzzim-python/; Acesso em: 23/11/2019 15:00;

HOLMES, Susan; **Erro RMS**; Disponível em: http://statweb.stanford.edu/~susan/courses/s60/split/node60.html; Acesso em: 23/11/2019 15:30;

KIM, Sungil; KIM, Heeyoung ; **A new metric of absolute percentage error for intermittent demand forecasts**; Disponível em: https://www.sciencedirect.com/science/article/pii/S0169207016000121; Acesso em: 23/11/2019 16:11;

ROSS, Sheldon M.; **Coefficient of Determination (R2)**; Disponível em: https://www.sciencedirect.com/topics/mathematics/coefficient-of-determination-r2; Acesso em: 23/11/2019 16:25;