



UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE  
CENTRO DE TECNOLOGIA  
DEPARTAMENTO DE ENGENHARIA ELÉTRICA  
**ELE0606** - TÓPICOS ESPECIAIS EM INTELIGÊNCIA ARTIFICIAL

**Docente:**

José Alfredo Ferreira Costa

**Autores:**

Diego Maia Marques

Vinícius Venceslau Venancio da Penha

**Introdução à Processamento de Linguagem Natural (NLP)**

Natal - RN  
20 de dezembro de 2023

## RESUMO

Este relatório abrange os principais aspectos do Processamento de Linguagem Natural (NLP). Explora-se o uso de Redes Neurais Recorrentes (RNNs) e Redes Neurais Convolucionais (CNNs) no processamento de texto, com detalhes matemáticos sobre seu funcionamento. Apresenta-se exemplos de aplicação, como assistentes virtuais e análise de sentimentos, e destaca bibliotecas essenciais, como NLTK e spaCy.

Além disso, o relatório inclui um algoritmo em Python, desenvolvido pelos discentes, para ilustrar conceitos básicos de processamento de texto.

# Sumário

<b>1</b>	<b>Introdução</b>	<b>2</b>
<b>2</b>	<b>Desenvolvimento</b>	<b>3</b>
2.1	Definição de NLP . . . . .	3
2.2	Aprendizado de Máquina em NLP . . . . .	3
2.2.1	Redes Neurais Recorrentes (RNNs) . . . . .	3
2.2.2	Redes Neurais Convolucionais (CNNs) . . . . .	4
2.3	Exemplos Práticos e Aplicações . . . . .	5
2.4	Bibliotecas importantes . . . . .	5
2.5	Algoritmo em Python - Detecção de Fake News . . . . .	5
<b>3</b>	<b>Conclusão</b>	<b>12</b>
<b>4</b>	<b>Referências</b>	<b>13</b>

## Lista de Figuras

1	Ilustração de NLP. . . . .	2
2	Reprodução do comportamento de RNNs. . . . .	4
3	Representação didática do funcionamento de CNNs em NLP. . . . .	4
4	Visualização do Head do Dataframe - <b>Saída</b> . . . . .	6
5	Exibição do método .info() - <b>Saída</b> . . . . .	7
6	Primeiros textos processados - <b>Saída</b> . . . . .	8
7	Visualização de 10 amostras - <b>Saída</b> . . . . .	8
8	Avaliação do Modelo MLP - <b>Saída</b> . . . . .	9
9	Matriz confusão 01 - <b>Saída</b> . . . . .	10
10	Matriz confusão 02 - <b>Saída</b> . . . . .	11

# 1 Introdução

O Processamento de Linguagem Natural (NLP) é um campo da inteligência artificial (IA) dedicado à interação entre computadores e linguagem humana. Esta área foca na capacidade das máquinas de compreender, interpretar e gerar linguagem humana de forma natural.

Este relatório busca oferecer uma visão abrangente sobre NLP, desde suas definições fundamentais até suas aplicações práticas e bibliotecas essenciais.



Figura 1: Ilustração de NLP.

## 2 Desenvolvimento

### 2.1 Definição de NLP

NLP refere-se à capacidade computacional de compreender, interpretar e gerar linguagem natural. O objetivo principal é permitir que os computadores compreendam textos e fala, assimilando seu significado e contexto semântico. Isso envolve várias tarefas, incluindo análise de sentimento, tradução automática, sumarização de texto, reconhecimento de entidades nomeadas, entre outros.

### 2.2 Aprendizado de Máquina em NLP

O aprendizado de máquina é essencial no desenvolvimento de modelos de NLP. Em particular, técnicas como redes neurais profundas têm sido amplamente empregadas. A abordagem mais comum é o uso de redes neurais recorrentes (RNNs) e redes neurais convolucionais (CNNs), assim como o desenvolvimento de modelos de transformadores, como o BERT (Bidirectional Encoder Representations from Transformers).

Esses modelos são treinados com grandes conjuntos de dados textuais anotados, utilizando técnicas de aprendizado supervisionado, onde os algoritmos aprendem padrões e relações entre palavras e frases.

#### 2.2.1 Redes Neurais Recorrentes (RNNs)

As RNNs são projetadas para processar dados sequenciais, mantendo uma memória interna que lhes permite capturar dependências de longo alcance nos dados. Sua estrutura permite a retroalimentação de informações para estados anteriores, sendo especialmente úteis em tarefas em que a ordem dos dados importa, como na compreensão de sentenças ou na previsão de palavras seguintes em um texto.

Matematicamente, a estrutura de uma RNN é definida por uma sequência de operações que atualizam seu estado interno a cada passo de tempo. A fórmula básica para uma célula de RNN é:

$$h_n = \sigma(W_{hx}x_n + W_{hh}h_{n-1} + b_h) \quad (1)$$

Onde:

- $h_n$  é o estado oculto na etapa  $n$ ;
- $x_n$  é a entrada na etapa  $n$ ;
- $W_{hx}$  e  $W_{hh}$  são matrizes de pesos;
- $b_h$  é o viés (um número ou vetor) introduzido para ajustar a saída de uma unidade;
- $\sigma$  é uma função de ativação, como a função sigmóide ou tangente hiperbólica.

No entanto, as RNNs tradicionais enfrentam o problema do desaparecimento ou explosão do gradiente, limitando sua capacidade de capturar dependências de longo prazo. Isso levou ao desenvolvimento de variantes mais sofisticadas, como as Long Short-Term Memory Networks (LSTMs) e Gated Recurrent Units (GRUs), que melhoram o fluxo de informação ao longo do tempo.

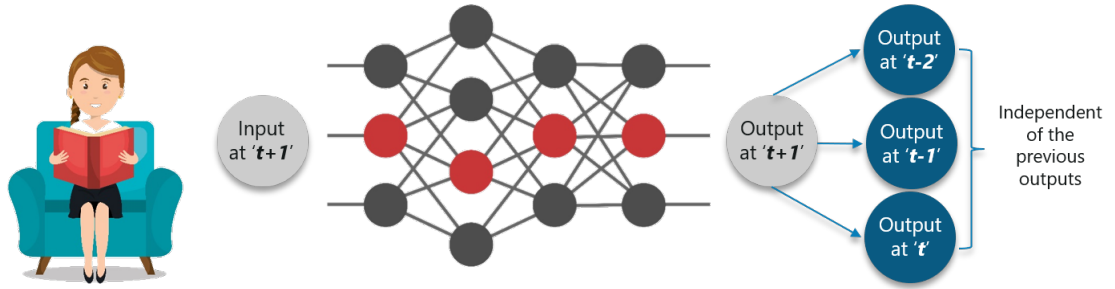


Figura 2: Reprodução do comportamento de RNNs.

### 2.2.2 Redes Neurais Convolucionais (CNNs)

As CNNs, inicialmente usadas para processamento de imagens, foram adaptadas para tarefas de NLP, aproveitando-se de suas capacidades de extrair e aprender características relevantes de dados sequenciais. Essas redes aplicam operações de convolução sobre sequências de texto, identificando padrões e relações locais entre palavras.

Uma operação de convolução em NLP envolve a aplicação de filtros (kernels) sobre as representações vetoriais das palavras, permitindo a detecção de características relevantes, como n-gramas ou relações contextuais.

A fórmula para a operação de convolução em NLP pode ser expressa como:

$$f_i = g(W \cdot x_{i:i+k-1} + b) \quad (2)$$

Onde:

- $f_i$  é a característica (feature) na posição  $i$ ;
- $x_{i:i+k-1}$  é a janela de palavras no intervalo  $i$  a  $i + k - 1$ ;
- $W$  é o kernel (filtro);
- $b$  é o viés;
- $g$  é uma função de ativação.

Posteriormente, a aplicação de camadas de pooling (como MaxPooling) reduz a dimensionalidade das características extraídas, preservando as mais relevantes para a tarefa em questão.

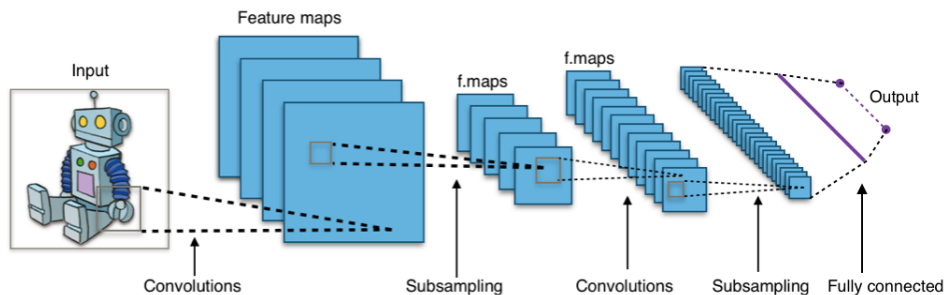


Figura 3: Representação didática do funcionamento de CNNs em NLP.

## 2.3 Exemplos Práticos e Aplicações

1. **Assistentes Virtuais:** Como a Siri da Apple, Google Assistant e Amazon Alexa, que processam comandos de voz e respondem a consultas.
2. **Análise de Sentimento:** Avaliação automatizada de opiniões em mídias sociais, reviews de produtos, etc.
3. **Tradução Automática:** Google Translate e outros sistemas que traduzem texto entre idiomas.
4. **Sumarização de Texto:** Resumo automático de documentos extensos.
5. **Processamento de Documentos:** Extração de informações de documentos para facilitar buscas e análises.

## 2.4 Bibliotecas importantes

As bibliotecas essenciais para a criação de soluções em Processamento de Linguagem Natural (NLP) desempenham um papel crucial no desenvolvimento e implementação de modelos. O Natural Language Toolkit (NLTK) é uma dessas ferramentas, oferecendo uma gama de funcionalidades para o processamento de texto em Python, desde tokenização até análise sintática.

Ademais, a biblioteca spaCy é amplamente utilizada devido à sua eficiência no processamento avançado de texto, fornecendo suporte para diversas tarefas, como identificação de entidades nomeadas e análise morfológica. Ela se destaca pela velocidade e facilidade de uso em aplicações de NLP.

Essas ferramentas, como o NLTK, spaCy e outros modelos existentes, são fundamentais para os desenvolvedores na construção e implantação de soluções eficazes em Processamento de Linguagem Natural, oferecendo uma variedade de recursos e modelos para diferentes necessidades de aplicação.

## 2.5 Algoritmo em Python - Detecção de Fake News

### Requisitos do Projeto:

1. Utilizar corpus com textos de notícias rotuladas (falsas e verdadeiras)
2. Comparar resultado com regressão logística e MLP na classificação

→ Tendo em vista que nenhuma das bases de dados fornecidas pelo docente possui uma distinção explícita entre notícias verdadeiras e falsas, foi necessário buscar uma base de dados que detém estes requisitos.

→ Assim, vale ressaltar que analisou-se uma base de dados que rotula **notícias americanas**.

Nesse viés, foi utilizado a base de dados proveniente da **FakeNewsNet**, onde:

- `politifact_fake.csv` - Amostras relacionadas a notícias falsas coletadas pelo PolitiFact
- `politifact_real.csv` - Amostras relacionadas a notícias reais coletadas pelo PolitiFact

Cada um dos arquivos CSV mencionados é um arquivo separado por vírgulas e possui as seguintes colunas:

- **id** - Identificador único para cada notícia
- **url** - URL do artigo da web que publicou essa notícia
- **title** - Título do artigo de notícia
- **tweet\_ids** - IDs de tweets que compartilham a notícia. Este campo é uma lista de IDs de tweets separados por tabulação.
- **label** - Rotulação da notícia em verdadeira ou falsa (0 ou 1)

## 1. Preparação do Ambiente e Dados:

```
1 import pandas as pd
2
3 # Carregar os dados
4 fake_news = pd.read_csv('politifact_fake.csv')
5 real_news = pd.read_csv('politifact_real.csv')
6
7 # Adicionar rotulos aos dados
8 fake_news['label'] = 1 # Noticias falsas tem rotulo 1
9 real_news['label'] = 0 # Noticias verdadeiras tem rotulo 0
10
11 # Combinar os dados em um unico DataFrame
12 data = pd.concat([fake_news, real_news], ignore_index=True)
13
14 # Embaralhar os dados
15 data = data.sample(frac=1).reset_index(drop=True)
16
17 # Visualizar as primeiras linhas dos dados
18 data.head()
```

Listing 1: Importação dos dados.

	id	news_url	title	tweet_ids	label
0	politifact12751	https://www.census.gov/foreign-trade/statistic...	US Trade in Goods	998217739683680256\t998220389787553792\t998234...	0
1	politifact14305	http://politicot.com/paul-ryan-22-million-amer...	Paul Ryan: "22 Million Americans Choose To Be ...	NaN	1
2	politifact14306	https://web.archive.org/web/20170806200608/htt...	Obama Goes To G20 Summit	1436642283\t1436642464\t1436643954\t1437071990...	1
3	politifact4372	http://www.pbs.org/wnet/need-to-know/economy/t...	This Labor Day, we need protests	NaN	0
4	politifact15224	https://www.viraltruthwire.com/obama-secretly-...	Obama Secretly Flees US - Leaves Stunning Evid...	977934019273699328\t978722696711335942\t979262...	1

Figura 4: Visualização do Head do Dataframe - **Saída**.

```
1 # Verificar informacoes sobre os dados
2 data.info()
```

Listing 2: Método para obter informações dos dados.



```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1056 entries, 0 to 1055
Data columns (total 5 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   id          1056 non-null   object
1   news_url    995 non-null    object
2   title       1056 non-null   object
3   tweet_ids   801 non-null    object
4   label       1056 non-null   int64
dtypes: int64(1), object(4)
memory usage: 41.4+ KB

```

Figura 5: Exibição do método .info() - **Saída.**

```

1 import nltk
2 from nltk.corpus import stopwords # Importa a lista de stopwords da
   biblioteca NLTK
3 from nltk.tokenize import word_tokenize # Importa a funcao word_tokenize
   do NLTK
4 import string
5
6 nltk.download('stopwords')
7 nltk.download('punkt') # Um modelo para dividir um texto em tokens (como
   palavras ou frases)

```

Listing 3: Importação e Download de ferramentas de pré-processamento.

```

1 [nltk_data] Downloading package stopwords to /root/nltk_data...
2 [nltk_data]   Package stopwords is already up-to-date!
3 [nltk_data] Downloading package punkt to /root/nltk_data...
4 [nltk_data]   Package punkt is already up-to-date!
5 True

```

Listing 4: Confirmação de Download - **Saída.**

```

1 # Funcao para pre-processar o texto
2 def preprocess_text(text):
3     # Tokenizacao
4     tokens = word_tokenize(text.lower()) # Converte para minusculas e
   tokeniza
5
6     # Remocao de stopwords e pontuacoes
7     stop_words = set(stopwords.words('english')) # Usa-se 'english', uma
   vez que a base de dados e americana
8     tokens = [token for token in tokens if token not in stop_words and
   token not in string.punctuation]
9
10    # Reconstruir o texto apos o pre-processamento
11    processed_text = ' '.join(tokens)
12    return processed_text
13
14 # Aplicar pre-processamento ao DataFrame
15 data['processed_text'] = data['title'].apply(preprocess_text)
16

```

```

17 # Visualizar os primeiros textos processados
18 data['processed_text'].head()

```

Listing 5: Função para pré-processar o texto.

```

0          us trade goods
1  paul ryan " 22 million americans choose poor '...
2          obama goes g20 summit
3          labor day need protests
4  obama secretly flees us - leaves stunning evid...
Name: processed_text, dtype: object

```

Figura 6: Primeiros textos processados - **Saída**.

```

1 # Exibir uma amostra dos dados
2 sample_data = data.sample(n=10, random_state=42) # Mostrar 10 entradas
   aleatorias
3 print(sample_data[['processed_text', 'label']])

```

Listing 6: Permitir a visualização de 10 amostras aleatórias pós processamento.

	processed_text	label
260	kasich " woman intelligent enough keep legs cl...	1
832	fed fight save america washington rick perry	0
846	poverty 2005 highlights	0
1007	nra president jim porter falsely accused sayin...	1
88	video special preview jersey shore miami	0
457	busted obama holding secret meetings overtake ...	1
184	ohio student suspended staying class national ...	1
988	un refugee agency welcomes arrival 10,000th sy...	0
367	trump ' top scientist pick " scientists dumb r...	1
558	breaking laura ingraham fired	1

Figura 7: Visualização de 10 amostras - **Saída**.

## 2. Separação dos Dados em Treinamento e Teste:

```

1 from sklearn.model_selection import train_test_split
2 from sklearn.feature_extraction.text import TfidfVectorizer
3 from sklearn.neural_network import MLPClassifier
4 from sklearn.metrics import accuracy_score, classification_report,
   confusion_matrix
5
6 # Separar os dados em treino e teste
7 X = data['processed_text']
8 y = data['label']
9 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
   random_state=42)
10
11 # Vetorizacao dos textos usando TF-IDF
12 vectorizer = TfidfVectorizer()
13 X_train_tfidf = vectorizer.fit_transform(X_train)
14 X_test_tfidf = vectorizer.transform(X_test)

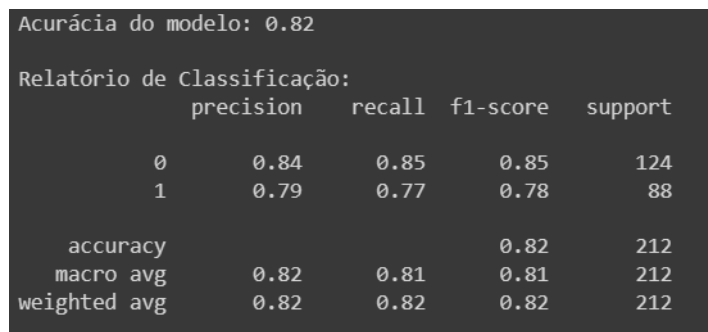
```

Listing 7: Separação dos Dados.

### 3. Criação do Modelo de Classificação (MLP) e Visualização dos Resultados:

```
1 import warnings
2 from sklearn.exceptions import ConvergenceWarning # Para remover a
   mensagem de aviso de nao convergencia
3
4 # Desativar os warnings de convergencia temporariamente
5 warnings.filterwarnings("ignore", category=ConvergenceWarning)
6
7 # Criar e treinar o modelo MLP
8 mlp = MLPClassifier(hidden_layer_sizes=(32, 16), max_iter=100,
   random_state=42)
9 mlp.fit(X_train_tfidf, y_train)
10
11 # Fazer previsoes
12 predictions1 = mlp.predict(X_test_tfidf)
13
14 # Avaliar o modelo
15 accuracy = accuracy_score(y_test, predictions1)
16 print(f'Acuracia do modelo: {accuracy:.2f}\n')
17
18 print('Relatorio de Classificacao:')
19 print(classification_report(y_test, predictions1))
```

Listing 8: Criação de uma MLP.



A terminal window showing the output of the code. The first line is 'Acurácia do modelo: 0.82'. The second line is 'Relatório de Classificação:'. Below this is a table with 5 columns: precision, recall, f1-score, and support. The rows are for classes 0 and 1, and then summary rows for accuracy, macro avg, and weighted avg.

	precision	recall	f1-score	support
0	0.84	0.85	0.85	124
1	0.79	0.77	0.78	88
accuracy			0.82	212
macro avg	0.82	0.81	0.81	212
weighted avg	0.82	0.82	0.82	212

Figura 8: Avaliação do Modelo MLP - Saída.

```
1 import matplotlib.pyplot as plt
2 import seaborn as sns
3
4 # Criacao e visualizacao da matriz confusao
5 rotulos = ['verdadeiras', 'falsas']
6 cm1 = confusion_matrix(y_test, predictions1)
7
8 plt.figure(figsize=(8, 6))
9 sns.heatmap(cm1, annot=True, fmt='d', cmap='Blues', annot_kws={"size":
   14}, xticklabels=rotulos, yticklabels=rotulos)
10 plt.xlabel('Previsoes')
11 plt.ylabel('Valores Verdadeiros')
12 plt.title('Matriz de Confusao')
13 plt.show()
```

Listing 9: Criação da matriz confusão.

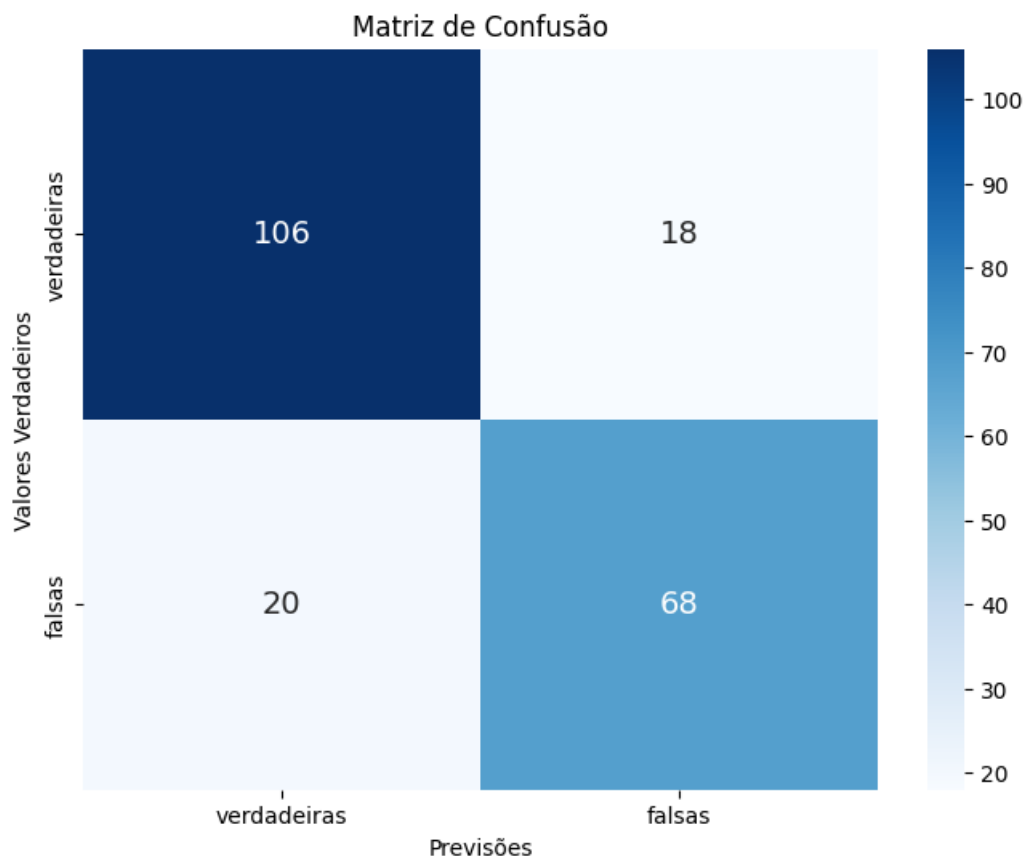


Figura 9: Matriz confusão 01 - Saída.

#### 4. Criação do Modelo de Classificação (Regressão Logística) e Visualização dos Resultados:

```

1 from sklearn.linear_model import LogisticRegression
2
3 # Criar o classificador de Regressao Logistica -> "max_iter=10000"
  representa o numero de iteracoes na tentativa de encontrar uma
  solucao aceitavel.
4 clf = LogisticRegression(max_iter=10000)
5
6 # Treinar o modelo.
7 clf.fit(X_train_tfidf, y_train)
8
9 # Fazer previsoes no conjunto de teste.
10 predictions2 = clf.predict(X_test_tfidf)
11
12 # Calcular e exibir a acuracia.
13 accuracy = accuracy_score(y_test, predictions2)
14 print(f'Acuracia da regressao logistica: {accuracy:.2f}')
```

Listing 10: Criação do Classificador por Regressão Logística.

```

1 Acuracia da regressao logistica: 0.76
```

Listing 11: Acurácia do Segundo Modelo - Saída.

```

1 # Criacao e visualizacao da matriz confusao
2 rotulos = ['verdadeiras', 'falsas']
3 cm2 = confusion_matrix(y_test, predictions2)
4
5 plt.figure(figsize=(8, 6))
6 sns.heatmap(cm2, annot=True, fmt='d', cmap='Blues', annot_kws={"size":
7     14}, xticklabels=rotulos, yticklabels=rotulos)
8 plt.xlabel('Previsoes')
9 plt.ylabel('Valores Verdadeiros')
10 plt.title('Matriz de Confusao')
11 plt.show()

```

Listing 12: Criação da matriz confusão.

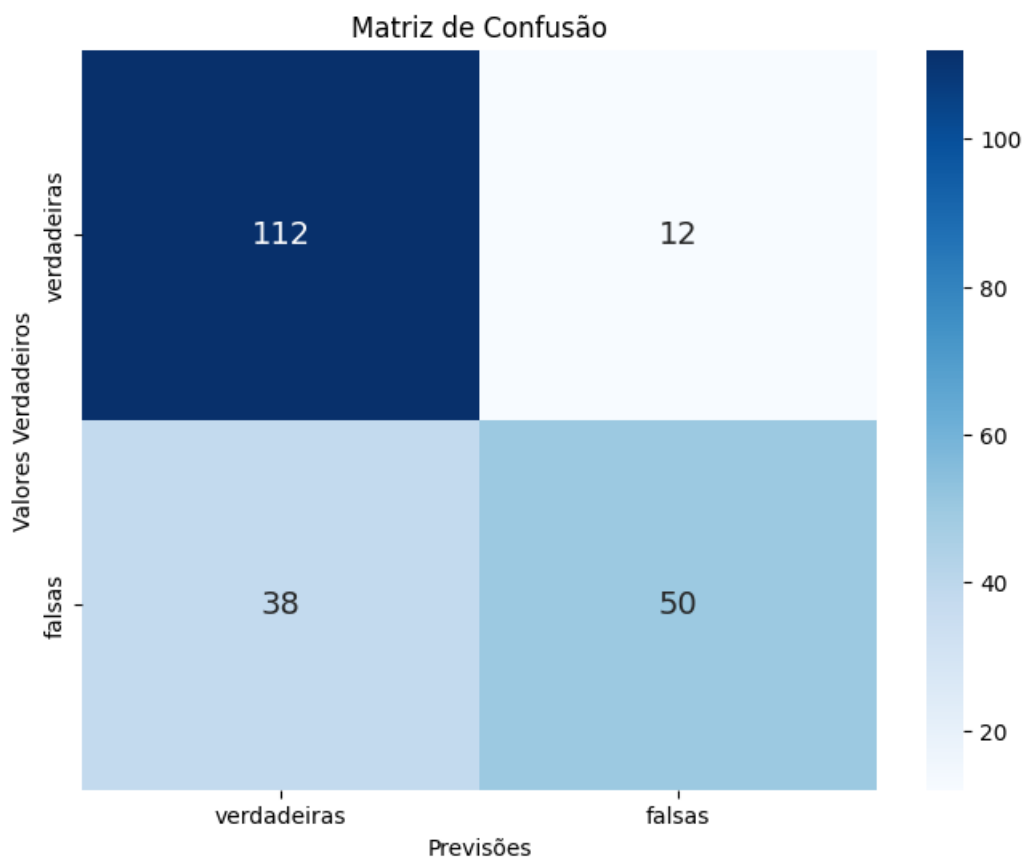


Figura 10: Matriz confusão 02 - **Saída**.

Em suma, percebe-se que o primeiro método, isto é, o qual utiliza uma rede neural de múltiplas camadas, foi mais eficiente para distinguir notícias verdadeiras e falsas, uma vez que teve uma acurácia de 82%.

Outrossim, infere-se que, o segundo método também se comportou bem, embora tenha tido uma acurácia inferior de 76%.

### **3 Conclusão**

O Processamento de Linguagem Natural é um campo em constante evolução, impulsionado pelo aprendizado de máquina e pela disponibilidade de grandes conjuntos de dados. Suas aplicações são vastas e têm impacto em diversas áreas, desde assistentes virtuais até análises complexas de texto. As bibliotecas e ferramentas disponíveis estão facilitando cada vez mais o desenvolvimento e a implementação de soluções baseadas em NLP.

## 4 Referências

- [1] **O que é NLP (Processamento de Linguagem Natural)**, seus usos e como utilizar. Disponível em: <https://rockcontent.com/br/blog/o-que-e-nlp/>. Acesso em: 14 de dezembro de 2023.
- [2] TECHLABS, M. **Top 5 Benefits of NLP in Leading Business Domains**. Disponível em: <https://medium.com/mlearning-ai/top-5-benefits-of-nlp-in-leading-business-domains-d86d0d988cf7>. Acesso em 15 de dezembro de 2023.
- [3] **Mathematical understanding of RNN and its variants**. Disponível em: <https://www.tutorialspoint.com/mathematical-understanding-of-rnn-and-its-variant-s#:~:text=The%20RNN%20modifies%20its%20undisclosed%20state%20using%20the>. Acesso em: 17 de dezembro de 2023.
- [4] **Chapter 5 Convolutional neural networks and their applications in NLP — Modern Approaches in Natural Language Processing**. [s.l: s.n.]. Disponível em: [https://slds-lmu.github.io/seminar\\_nlp\\_ss20/convolutional-neural-networks-and-their-applications-in-nlp.html](https://slds-lmu.github.io/seminar_nlp_ss20/convolutional-neural-networks-and-their-applications-in-nlp.html). Acesso em: 18 de dezembro de 2023.
- [5] OPENAI. **ChatGPT**. 2023. Disponível em: <https://openai.com/>. Acesso em: 18 de dezembro de 2023.
- [6] SHU, K. **KaiDMML/FakeNewsNet**. Disponível em: <https://github.com/KaiDMML/FakeNewsNet>. Acesso em: 19 de dezembro de 2023.