

# Análise de Textos Científicos utilizando Python

**Disciplina:** Introdução a Inteligência Artificial  
**Professor:** Professor Doutor Wagner Igarashi  
**Curso:** Informática

**Equipe:**  
Álvaro de Araújo Ferreira Lima Neto  
Karoline Harummy Romero Moriya  
Rafael Prado Torres

# Agenda

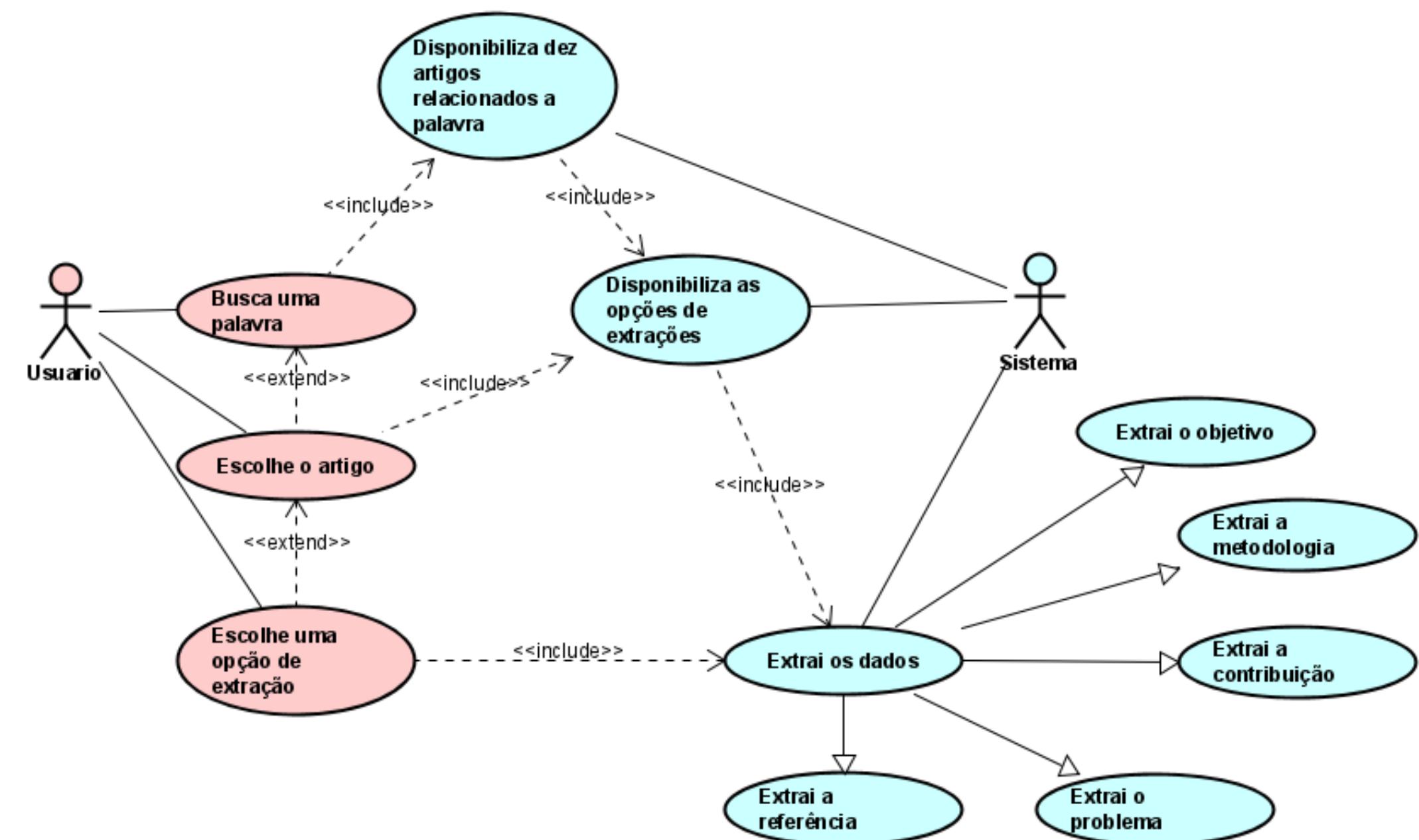
- |   |                       |    |                       |
|---|-----------------------|----|-----------------------|
| 3 | Descrição do Problema | 7  | Instruções para Teste |
| 4 | Casos de Uso          | 8  | Código                |
| 5 | Funcionalidades       | 12 | Simulação             |
| 6 | Plataforma            | 10 | Bibliografia          |

# **Descrição do Problema**

O presente trabalho apresenta uma ontologia sobre a estrutura de um artigo científico e a implementação de um software que utiliza técnicas de Processamento de Linguagem Natural para extrair dados destes artigos.



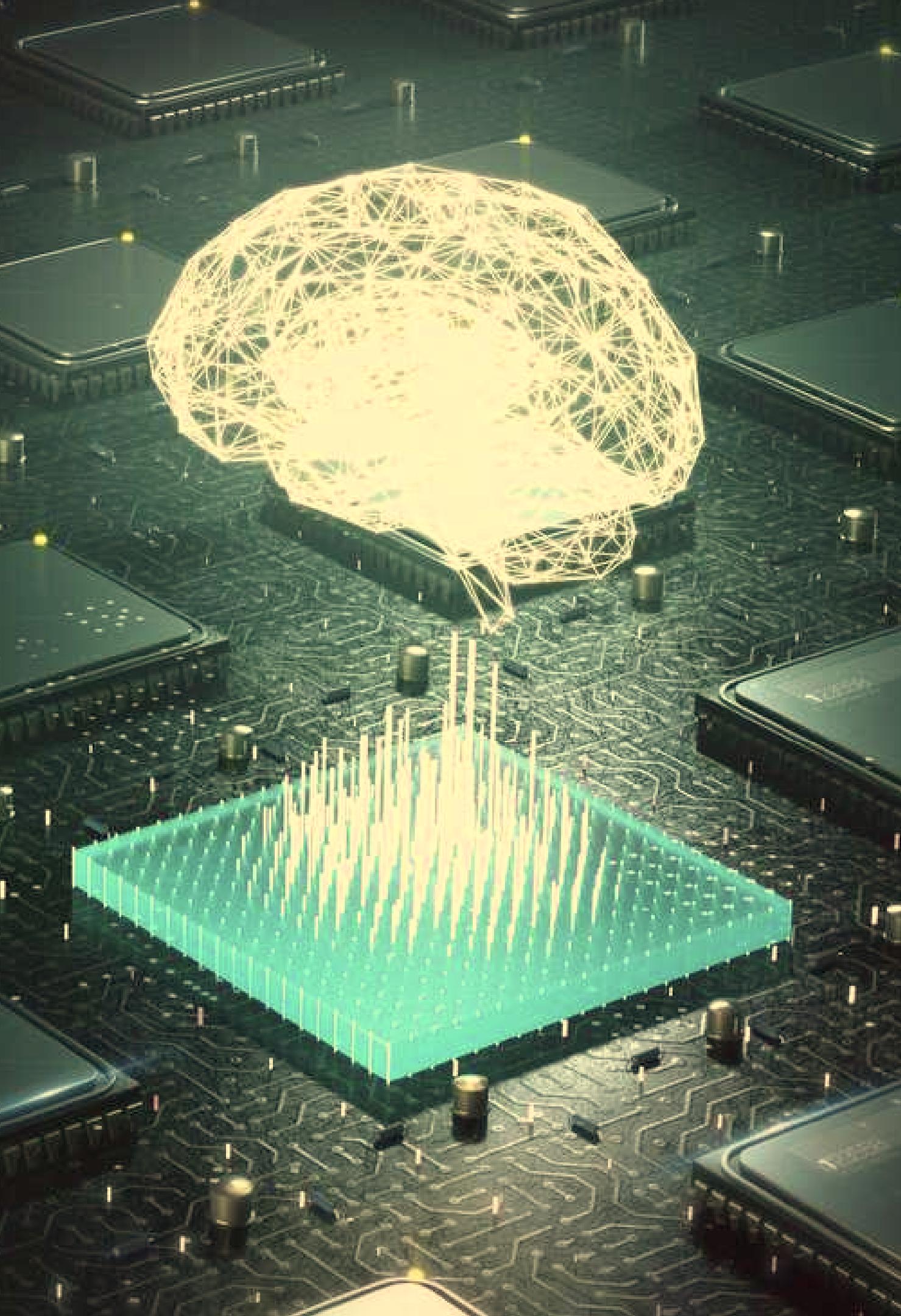
# Casos de Uso



# Funcionalidades

O sistema conta com uma interface, que permite que o usuário busque uma palavra. A partir desta palavra, artigos que contém a mesma em sua composição são mostrados na tela para que o usuário escolha qual ele quer visualizar. Após isso, abre-se uma opção para que o usuário selecione qual extração de dado deseja, ou seja, qual informação ele quer saber sobre o artigo, estando dentro das possíveis informações:

- Objetivo do artigo
- Problema do artigo
- Metodologia do artigo
- Contribuição do artigo
- Referências do artigo



# Plataforma

Esclarecer os principais objetivos e metas gerais do projeto.



Dados do computador

Macbook Air 2017, i5, 4gb

RAM

Acer Aspire 5, 8gb RAM

Dell, i5, 8gb RAM



S.O.

MacOS

Windows 10

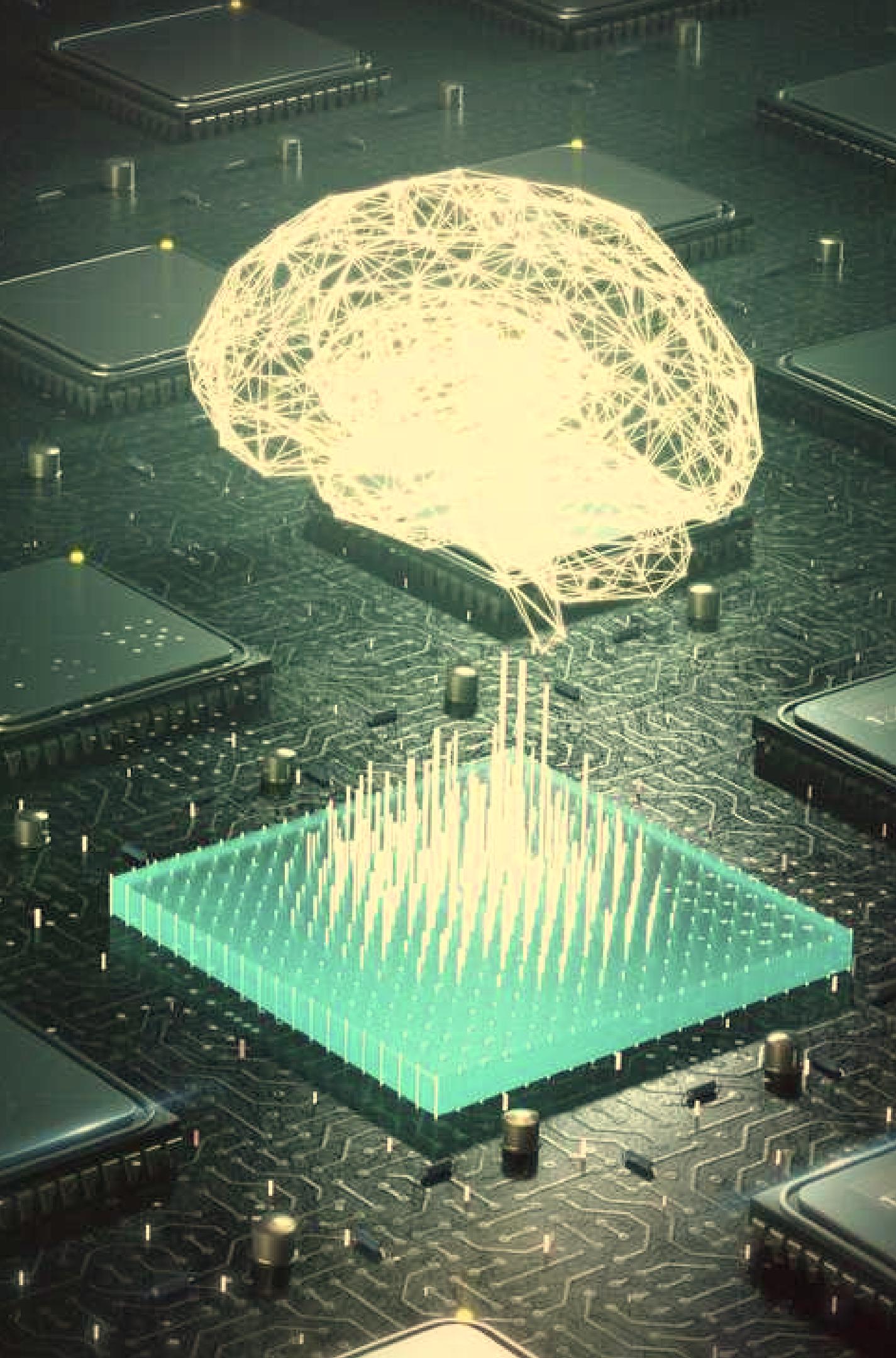


Linguagem

Python

# Instruções para Teste

O programa funciona a partir da interface.  
Abra o diretório da interface, rode o arquivo interface.py (feito com tkinter) e aparecerá para buscar a palavra, escolher o artigo e escolher a extração do dado desejado.



# Trechos de Código

# Manipular PDF

```
# if busca0:  
#     termina = busca0.end()  
#     pagina = pagina[termina:]  
if busca1:  
    termina = busca1.end()  
    pagina = pagina[termina:]  
elif busca2:  
    termina = busca2.end()  
    pagina = pagina[termina:]  
elif busca3:  
    termina = busca3.end()  
    pagina = pagina[termina:]  
  
return pagina  
  
def removerPontuacao(texto :str) -> str:  
    return "".join(caractere for caractere in texto if caractere not in punctuation)  
  
def limparTexto(texto :str) -> str:  
    texto = texto.lower()  
    texto = removerNumeroPagina(texto)  
    texto = removerPontuacao(texto)  
    return texto  
  
# é usado na classe problema. Será que é necessário?  
def removerBarraN(texto :str) -> str:  
    return texto.replace('\n', ' ')  
  
def lerPDF(arquivo :str) -> str:  
    return PyPDF2.PdfReader(arquivo)
```

# Sumário

```
import re
from manipularPDF import limparTexto

class Sumario:
    def __init__(self, pdfLido :object) -> None:
        self.__sumario = self.__extrairSumario(pdfLido)

    def getSumario(self) -> dict:
        return self.__sumario

    def getPaginasTopico(self, topicoRegex: re) -> list:
        posicaoPaginas = []

        for keys, values in self.__sumario.items():
            if re.match(topicoRegex, keys) and len(posicaoPaginas) == 0:
                posicaoPaginas.append(values)
            elif len(posicaoPaginas) == 1:
                posicaoPaginas.append(values)
        return posicaoPaginas

    def __extrairTextoSumario(self, pdfLido :object) -> list:
        textoFinal = ''
        reInicioTopico = r'sum\s*\á\s*rio'
        reFimTopico = r'referências\s*\.*\b'
        achouTopico = False
        posicaoSumario = None
```

# Referências

```
from sumario import Sumario
from extrairTopico import ExtrairTopico

class Referencia():

    def __init__(self, pdfLido: object, sumario :Sumario) -> None:
        self.__topico = ExtrairTopico(sumario, self.__getPadroes())
        self.__referencia = self.__extrairReferencia(pdfLido)

    def getReferencia(self) -> list:
        return self.__referencia

    def __getPadroes(self):
        dictPadroes = {'topico': r'referências\b',
                      'reComecoTopico': r'referências\b',
                      'reFimTopico': r'\s*(apêndice(|s)|anexo)\b'}

        return dictPadroes

    def __extrairReferencia(self, pdfLido: object) -> list:
        textoTopico = self.__topico._getTopico(pdfLido)
        textoTopico = textoTopico.split('.\n')

        return textoTopico
```

# Simulação

# Bibliografia

- <https://www.insightlab.ufc.br/pln-processamento-de-linguagem-natural-para-iniciantes/>