Universidade Estadual de Maringá

Processamento de linguagem natural

CTC

Photo: wallpapercave.com

9918 - 31 - Introdução a Inteligência Artificial

Informática

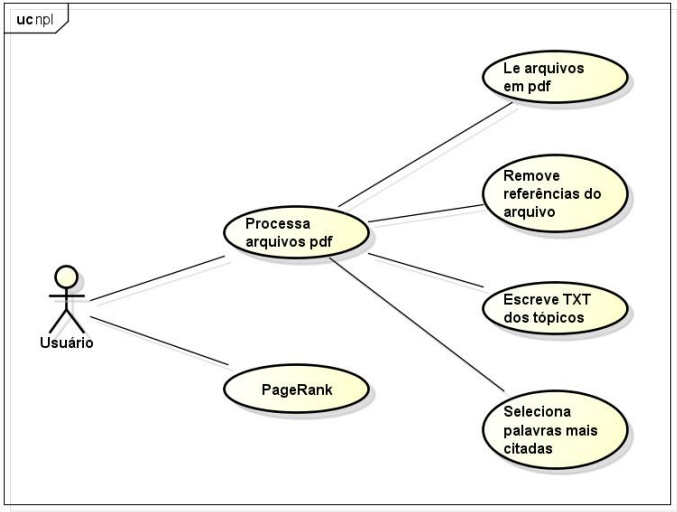
Prof. Dr. Wagner Igarashi

Acadêmico: Henrique Yoshiharu Kajihara RA: 78607

Descrição

O problema a ser resolvido é do tipo de Processamento de Linguagem Natural (NPL - Natural Language Process). Através do programa, iremos realizar a identificação das palavras mais citadas excluindo pronomes e preposições, a extração das informações: objetivo, problema, método e contribuição do artigo armazenando estes dados já filtrados em outro arquivo. Para isso, será utilizado a biblioteca spaCy do Python que irá realizar a busca destes tópicos por palavras-chave. O programa também dispõe de um PageRank, funcionalidade que irá avaliar pelo número de ocorrências da palavra buscada qual o artigo que mais tem esta palavra e irá mostra-los em ordem para o usuário.

Diagrama de Casos de Uso



Funcionalidades - Processa arquivos

Esta funcionalidade irá ler os arquivos do diretório onde o programa está sendo executado + ../arquivos/+(image, ia, machine_learning) e realizará o processamento destes conteúdos lendo o conteúdo de seus dados contabilizando quais as palavras mais citadas, removendo as referências do arquivo e após isso extraindo os tópicos para escrita nos arquivos txt.

Funcionalidades - PageRank

Esta função pega as dez palavras mais citadas dentro de cada artigo (estas palavras mais citadas estão armazenadas dentro do arquivo .txt salvo) e mostra ordenadamente os arquivos com a maior quantidade da palavra procurada.

Plataforma

- Computador: Processador Intel(R) Core(TM) i7-5500U CPU @ 2.40GHz, 8GB Ram, 512GB SSD
- Sistema operacional: Windows 10 Home Single Language
- Linguagem: Python
- Bibliotecas: PyPDF2, spaCy, os

Teste do programa

Para executar o programa é necessário:

- Python 3.10
- spaCy (pip install spacy)
- PyPDF2 (pip install PyPDF2)

Após as instalações é necessário colocar os artigos na pasta do mesmo diretório ../arquivos/+(image, ia, machine_learning) e executar o programa pela linha de comando utilizando o comando "python main.py"

Código

```
for numero_pagina in range(total_paginas):

    texto : str = arquivo_lido.pages[numero_pagina].extract_text()
    lista_texto_arquivo.append(texto.splitlines())
```

Função: processa_transforma_arquivos_pdf

Código

```
texto_inteiro: Literal[''] = ''
texto_referencias: Literal[''] = ''
texto_introducao: Literal[''] = ''
is_referencia = False
is_introducao = False
for pagina in lista_texto_arquivo:
    for linha in pagina:

        if  is_possivel_referencia(linha):
            is_referencia = True
```

Função: processa_transforma_arquivos_pdf

Código

```
if not is_referencia:
    #RETIRANDO A INTRODUCAO DO TEXTO
    if is_possivel_introducao(linha):
        #print("INICIO INTRO: " + linha)
        is_introducao = True

    if is_introducao and is_possivel_final_introducao(linha):
        #print("FINAL INTRO: " + linha)
        is_introducao = False
    if is_introducao:
        texto_introducao += linha + '\n'
#ESCREVE TODO O TEXTO NESTE PONTO
    texto_inteiro += linha+ '\n'
    adiciona_lista_palavras(palavras_recorrentes, linha)
    adiciona_lista_palavras(palavras_recorrentes_arquivo, linha)
else:
#ADICIONA O TEXTO DE REFERÊNCIAS SÓ PARA TESTAR DEPOIS
    texto_referencias += linha + '\n'
```

Função: processa_transforma_arquivos_pdf

Código

```python
#*****************************************************************************
def retorna_informacoes(texto_introducao, lista_palavras_referencia) -> list:

    nlp: Language = spacy.load("en_core_web_sm")
    documento = nlp(texto_introducao)
    matcher = PhraseMatcher(nlp.vocab, attr="LOWER")

    lista_patterns: list = []
    for palavra_referencia in lista_palavras_referencia:
        lista_patterns.append(nlp(palavra_referencia))

    for palavra_referencia in lista_palavras_referencia:
        matcher.add(palavra_referencia, None, *lista_patterns)

    sentencas: list = []
    for sentenca in documento.sents:
        if matcher(nlp(sentenca.text)):
            sentencas.append(sentenca.text)
            if len(sentencas) > 0:
                break

    return sentencas
```

Função: retorna_informacoes

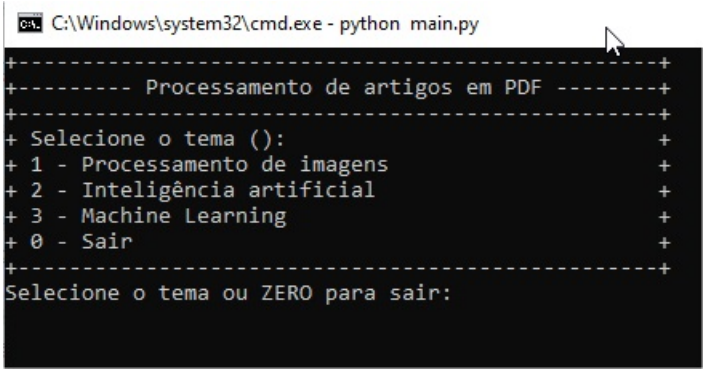
Código

```
#*************************************************************************#
def retorna_informacoes_problemas(texto_introducao, texto_inteiro) -> list:

    lista_problemas: list[str] = ["limitation", "issue", "problem", "challenge", "di
    problema_sentencas: list = retorna_informacoes(texto_introducao, lista_problemas
    if lista_vazia(problema_sentencas):
        problema_sentencas: list = retorna_informacoes(texto_inteiro ,lista_problema
    return problema_sentencas
```

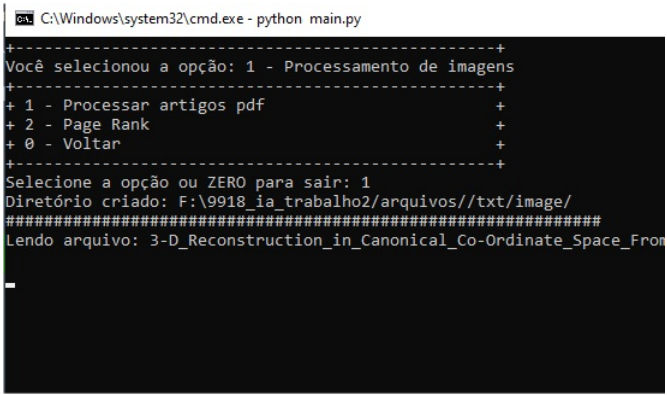
Função: retorna_informacoes_problemas

Programa



"Tela" inicial

Programa



Seleção 1

Programa

```
############################################################
Lendo arquivo: Super-Resolution_Axial_Localization_of_Ultrasound_Scatter

Rank de palavras mais encontradas:
['image', 661]
['images', 336]
['method', 286]
['reconstruction', 284]
['data', 279]
['methods', 214]
['using', 213]
['used', 195]
['prior', 180]
9 arquivos lidos com sucesso
Pressione uma tecla para continuar...
```

Resultados

Referências

- Notas de aulas
- https://pypdf2.readthedocs.io/
- https://spacy.io/

Link repositório

https://github.com/henriqueykajihara/9918-IA-Processamento_de_linguagem_natural