Trabalho Prático de Grupo nº 2

Mestrado em Engenharia Biomédica – Informática Médica

Processamento de Linguagem Natural
2º Semestre 2022/2023

Equipa docente
José Almeida, Luís Cunha

Beatriz Rodrigues Leite, PG50253
Joana Maria Freitas Fernandes, PG50459
Mariana Rodrigues Oliveira, PG50635

ÍNDICE

Introdução

O presente trabalho prático pressupôs a complexão do ficheiro desenvolvido no trabalho anterior. Para fazer esta complexão, recorreu-se a técnicas de *web scrapping,* através da biblioteca *Beautiful Soup.* Destas páginas, foram recolhidas descrições em falta em alguns termos, e categorias dos termos já existentes no *json* do trabalho anterior.

A segunda parte do trabalho pressupôs a utilização da biblioteca *Flask* para *Python*, que permite o rápido desenvolvimento de aplicações *web*, fornecendo uma sintaxe simples de criação de rotas e de execução de pedidos *HTTP*.

O objetivo do projeto desenvolvido foi criar um glossário de anatomia geral que incluísse termos médicos associados a um sistema do corpo humano, assim como as suas informações específicas e traduções. Assim, o intuito foi criar uma aplicação web com utilidade para os estudantes desta temática, por exemplo, em que os mesmos pudessem adicionar também as suas próprias notas aos termos do glossário, tal como alterá-las e removê-las.

Arquitetura

Para o desenvolvimento deste projeto partiu-se do dicionário já construído anteriormente que continha, para cada termo, a sua tradução em inglês e espanhol, a sua definição, e a sua expressão popular. A esta estrutura de dados, juntaram-se outras, obtidas através da extração de informação de páginas web, usando a ferramenta *BeautifulSoup*. Todas estas estruturas, permitiram obter uma estrutura final, em formato de ficheiro *json*, que inclui, para cada termo, a sua tradução em inglês e espanhol, a sua definição, a sua categoria (em que sistemas da anatomia se inserem) e uma nota.

De modo a concretizar o objetivo final do projeto, esta estrutura criada foi processada na criação de páginas web usando as ferramentas *Flask* e *Jinja2,* que possibilitaram a organização da informação pretendida em diversas páginas web, e ainda a inserção de variadas funcionalidades.

Posto isto, por fim obteve-se uma aplicação web com distintas páginas personalizadas onde se apresenta um glossário de anatomia geral devidamente organizado.

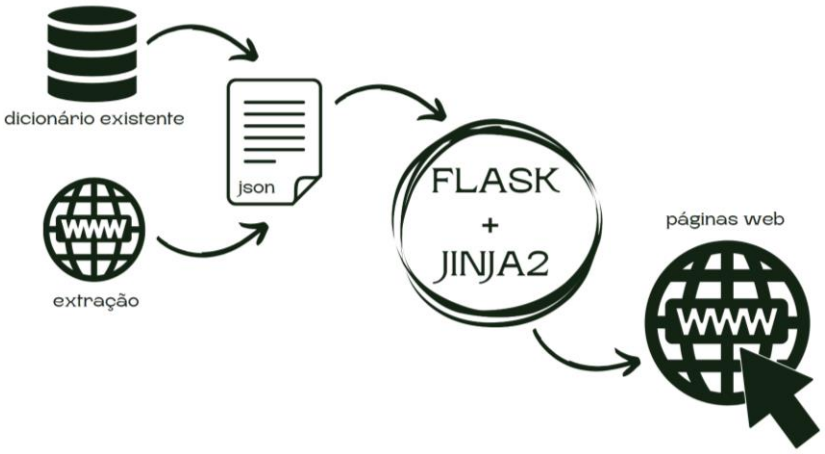

Figura 1. Esquematização da arquitetura.

Extração

Categoria.py

De modo a conseguir obter a informação relativa à categoria de cada termo procedeu-se à extração de informação presente numa página web, usando o *BeautifulSoup*. Para cada categoria (Sistema Digestivo, Sistema Urinário, etc.) foi utilizada uma página distinta. A extração passou então por identificar na página web selecionada para cada categoria a lista de palavras que se inseriam na mesma. Assim, criou-se uma estrutura de dados, um dicionário, com as palavras extraídas e a sua categoria correspondente, para cada página web. A informação pretendida encontrava-se numa secção da página web identificada por *" "* e cada palavra de interesse encontrava-se em *" *". Com esta informação e a partir da ferramenta e funcionalidades do bs4 foi possível realizar a extração.

Para além da categoria associada a cada palavra, pretendia-se também extrair a informação mais detalhada de cada parte do copo extraída na fase anterior. Para tal, foi necessário aceder à página web presente numa hiperligação em cada expressão. Para tal, foi criada uma função que recebe um *url* e procede à extração de informação incluída numa parte da página html identificada por *" *". O *url* recebido por esta função é construído pela junção do *url* base da página web inicial e do *url* presente na hiperligação. Uma vez acedida a página web que contém informação detalhada, extrai-se o primeiro parágrafo, de modo a obter uma definição. Mais uma vez, é construída uma estrutura de dados que contém o termo e a sua definição correspondente.

Ao longo desta extração foram identificadas duas exceções correspondentes a duas chaves em que não foi permitido extrair a sua informação detalhada pois a estrutura da página html apresentava-se numa estrutura diferente das restantes.

Para terminar esta fase, foi criado um dicionário que junta a informação da categoria ao dicionário já existente.

```
sites={"Sistema digestivo": "https://pt.wikipedia.org/wiki/Categoria:Sistema_digestivo",
       "Sistema urinário":"https://pt.wikipedia.org/wiki/Categoria:Sistema_urin%C3%A1rio",
       "Sistema reprodutor":"https://pt.wikipedia.org/wiki/Categoria:Sistema_reprodutor",
       "Sistema sensorial": "https://pt.wikipedia.org/wiki/Categoria:Sistema_sensorial",
       "Sistema endócrino": "https://pt.wikipedia.org/wiki/Categoria:Sistema_end%C3%B3crino",
       "Sistema nervoso" : "https://pt.wikipedia.org/wiki/Categoria:Sistema_nervoso",
       "Sistema tegumentar": "https://pt.wikipedia.org/wiki/Categoria:Sistema_tegumentar",
       "Sistema muscular": "https://pt.wikipedia.org/wiki/Categoria:Sistema_muscular",
       "Sistema circulatório": "https://pt.wikipedia.org/wiki/Categoria:Sistema_circulat%C3%B3rio",
       "Sistema respiratório": "https://pt.wikipedia.org/wiki/Categoria:Sistema_respirat%C3%B3rio",
       "Sistema esquelético": "https://pt.wikipedia.org/wiki/Categoria:Sistema_esquel%C3%A9tico"
       }

url_principal = "https://pt.wikipedia.org/"
```

Figura 2. URLs utilizados

```
for divs in subsection:
    divs = divs.find_all("div", class_="mw-category-group")
    for div in divs:
        termos = div.find_all("li")
        for termo in termos:

            #print (termo.a.text)
            chave = termo.text.strip().lower()
            if chave == "urinotórax" or chave == "predefinição:sistema nervoso":
                continue
            valor = categoria.strip().lower()
            dicionario[chave] = valor
            page_url = url_principal + termo.a["href"]
            page_info = extractInfo(page_url)
            dicio_desc[chave] = page_info


def extractInfo(url):
    page_html = requests.get(url).text
    page_soup=BeautifulSoup(page_html, "html.parser")
    page_div = page_soup.find("div", class_="mw-parser-output")
    res = page_div.p.text
    return str(res)
```

Figura 3. Extração

Final.py

O objetivo deste ficheiro é apenas o de filtrar para o json final apenas os termos que apresentam descrições.

Descrição.py

Foi criado um ficheiro *Python* que abre ambos os ficheiros *json* já existentes, e que compara as *keys* de ambos. Assim, para cada termo que não possua ainda uma descrição, a do segundo ficheiro, ou seja, a que foi retirada da página web, é adicionada ao dicionário. São retiradas destas descrições as referências. Por fim, a estrutura de dados final é adicionada a um ficheiro *json*.

Aplicação Web

Para o desenvolvimento da aplicação web foi utilizada a libraria Flask, que permite desenvolvê-las facilmente, e utiliza templates Jinja2.

Assim, foi desenvolvido o script Python que instancia a aplicação Flask, as diferentes routes e ainda corre a aplicação, indicando o host e a porta na qual esta vai correr.

As routes desenvolvidas foram as seguintes:

- @app.route("/") , que apenas apresenta o *template "home.html"* e lhe fornece uma variável title.
- @app.route("/main_info"), que apresenta o *template "main_info.html*" e lhe fornece o dicionário de termos.

- @app.route("/specific_info/<s>"), que apresenta o *template "specific_info/<s>.html"* sendo s o termo em questão. As informações passadas já variam dependendo do termo, sendo o próprio termo e os seus valores no dicionário.
- @app.route("/main_info/search"), que acedendo ao componente texto do formulário preenchido no *frontend* através do código *text = request.args.get("text")*, procura a ocorrência desse mesmo texto nas *keys* do dicionário, ou na descrição de cada termo, e adiciona a uma lista a *key* e os seus valores caso alguma destas condições se verifique. Por fim chama o *template "search.html"* e dá-lhe a lista para ser apresentada na página html.
- @app.route("/specific_info/<s>", methods=["POST","DELETE"]). Neste caso, temos 2 ações diferentes dentro da mesma rota. No caso do método *DELETE*, a rota confirma a existência desse termo s no ficheiro *json*. Caso isto se verifique, a rota elimina a nota e atualiza o *json*. No caso do método *POST*, a rota acede ao componente "nota" preenchido no formulário e adiciona ao termo essa nota, ou atualiza uma já existente. Em ambos os casos, o *template* atual é recarregado com novas informações fornecidas, ou seja, a designação e os seus valores associados.

Templates HTML e Jinja2

Como referido anteriormente, para o *frontend*, foram implementados *templates* que fazem uso da ferramenta *Jinja2*, facilitando a utilização dinâmica de algumas variáveis e agilizando o processo de criação das páginas HTML, uma vez que deste modo foi possível utilizarem-se estruturas de controlo, como ciclos, entre outras.

O ficheiro layout.html fornece uma estética base, ou seja, que é comum a todas as páginas. Mais concretamente, este arquivo, usando a sintaxe do *Jinja2*, define o cabeçalho, rodapé e título que surgem em todos os *templates*. O conteúdo específico de cada página será adicionado aos blocos definidos no layout base, como o bloco *{% block body %}*. É também neste ficheiro que se encontra implementado o arquivo *script.js*, um documento *javascript* que contribui para o funcionamento da tabela principal.

O ficheiro home.html representa a página inicial do site, definindo elementos os elementos que se podem visualizar na *Figura X*. Tal como as restantes páginas, estende o ficheiro *layout.html*.

Figura 4. Página inicial, implementada pelo ficheiro home.html.

O arquivo *main_info.html* gera uma página para exibição de uma tabela dinâmica com informações sobre as várias partes do corpo humano (*Figura X*).

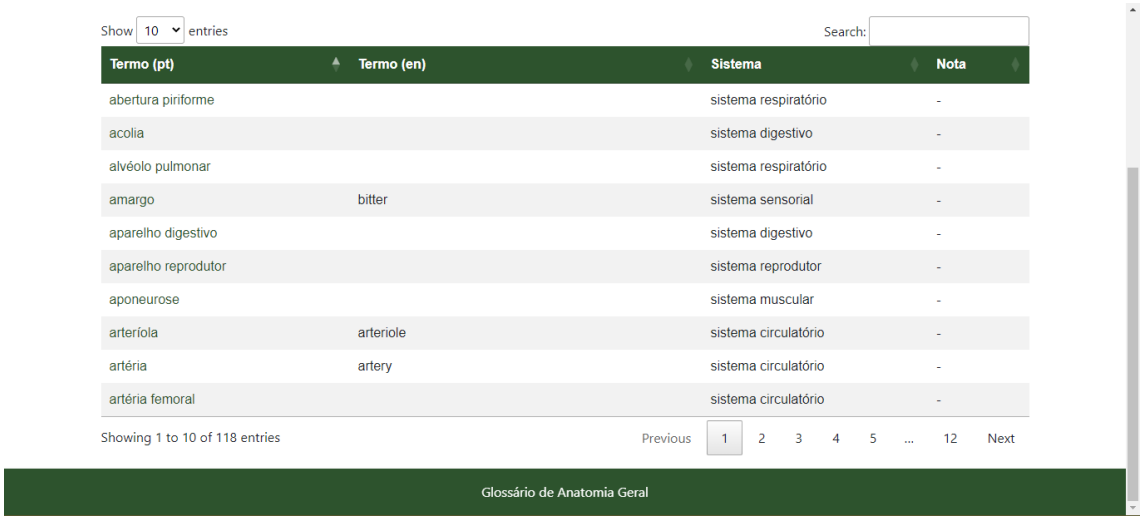

Figura 5. Excerto da página gerada pelo ficheiro main_info.html.

O ficheiro *search.html* define uma página de pesquisa, definindo um formulário com um campo de texto para o utilizador inserir o que quer pesquisar (*Figura X*).

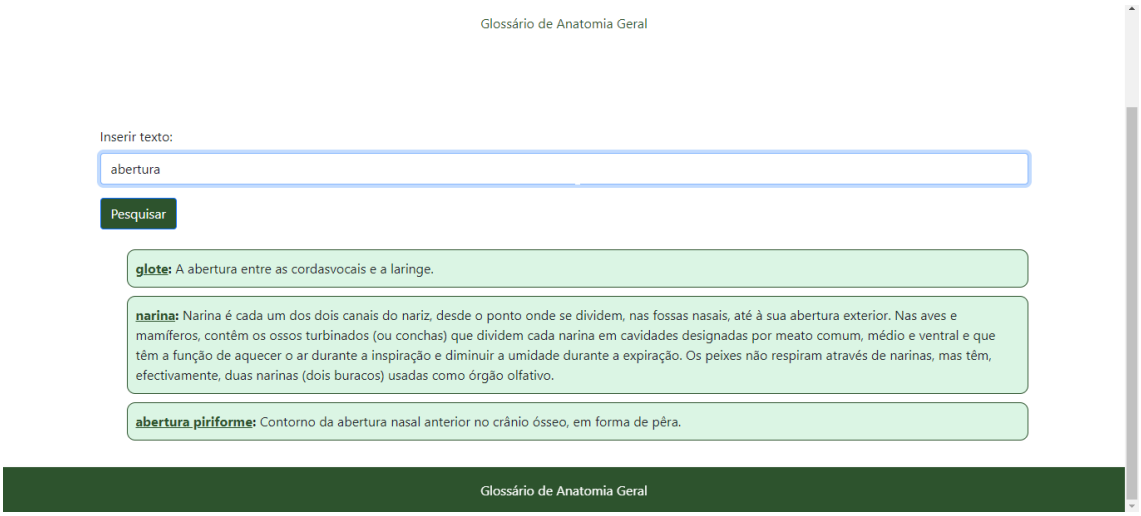

Figura 6. Excerto da página HTML gerada pelo ficheiro search.html após se pesquisar pelo termo 'abertura'.

Por fim, o ficheiro *specific_info.html* representa uma página que exibe informações detalhadas sobre um termo específico. Nesta página também é permitido adicionar e excluir notas associadas a um termo, ou seja, neste código HTML são manipuladas ações de envio, usando métodos POST e DELETE (*Figura X*).

Figura 7. Exemplo de página implementada pelo ficheiro specific_info.html.

Conclusão

Resumindo, o projeto concebido resultou num website interativo que fornece informações detalhadas sobre vários componentes do corpo humano, permitindo pesquisa, consulta, acréscimo e exclusão de informações.

Importante mencionar que existem alguns aspetos a melhorar, destacando-se o facto de que seria mais prática a junção do código desenvolvido em vários ficheiros num único arquivo. Também a utilização de uma tabela mais complexa poderia ser útil, aumentando a interatividade e organização da informação. Como visto anteriormente, alguns termos não têm associadas traduções, neste sentido a implementação de um mecanismo que preenchesse estes domínios seria vantajoso. Poderia ainda ter sido efetuada a extração de mais informações relevantes para cada designação, utilizando-se mais recursos e fontes mais credíveis.