Mestrado em Informática Médica

Processamento de Linguagem Natural em Engenharia Biomédica

Trabalho Prático 1

Realizado por:
Beatriz Rodrigues, PG53696
Bruno Machado, PG53709
Rúben Ganança, PG54203

Docentes: José Almeida e Luís Cunha

7 de abril de 2024

Índice

1. Introdução

Este relatório tem como principal objetivo acompanhar o primeiro trabalho prático realizado na UC de Processamento de Linguagem Natural em Engenharia Biomédica. De forma sucinta, o projeto consiste na extração de informação relevante de diferentes documentos médicos, fornecidos em formato *PDF* e a sua posterior preservação num ficheiro *JSON*.

Para manipular os textos e identificar padrões, foram utilizadas expressões regulares. Estas permitem que a partir de um determinado padrão de pesquisa constituído por uma sequência de caracteres, possam ser efetuadas diversas operações como pesquisar, substituir e extrair texto a partir de grandes conjuntos de informação. Além disso, permitem também efetuar a limpeza dos dados extraídos e remoção de elementos desnecessários como cabeçalhos, rodapés e números de página.

2. Documentos Processados

Para a realização deste documento foram utilizados quatro documentos onde se realizou a extração de informação, sendos os documentos escolhidos:

- glossario_ministerio_saude.pdf
- Glossário de Termos Médicos Técnicos e Populares.pdf
- WIPOPearl_COVID-19_Glossary.pdf
- anatomia geral.pdf

O primeiro documento referido (documento de análise obrigatória) é constituído por um dicionário técnico-científico de termos médicos. Já o segundo, consiste num conjunto de diversos termos médicos e a sua definição coloquial, isto é, popular. Relativamente ao terceiro documento, alicerça-se em diversos termos médicos, a sua definição e tradução em 9 línguas: árabe, alemão, espanhol, francês, japonês, coreano, português, russo e chinês. Por fim, o último documento contêm um diverso conjunto de termos anatómicos.

3. Arquitetura

O objetivo final deste trabalho seria obter dois grandes ficheiros *json*, um deles o *wipo.json* onde constam designações de termos em inglês, a sua descrição, categoria e diversas traduções, e um outro documento denominado *glossario_geral.json* em que consta designações de termos em português, a categoria desse termo e a sua descrição. Outros ficheiros *json* menores que serão obtidos são o *siglas.json* e o *abreviacoes.json*, sendo que este último foi feito manualmente.

Em relação ao primeiro documento, o *wipo.json*, este documento seria obtido através da análise do *pdf WIPOPearl_COVID-19_Glossary.pdf*.

O segundo documento, *glossario_geral.json*, é obtido através da junção de outros ficheiros *json*:

- **conceitos.json**: Este ficheiro é obtido através da análise do documento obrigatório "glossario_ministerio_saude.pdf";
- **termos_glossario.json**: Este é obtido através da análise do ficheiro "Glossário de Termos Médicos Técnicos e Populares.pdf";
- **glossario_anatomia.json**: Por fim este último é obtido pelo "anatomia geral.pdf".

4. Tratamento de Dados

Para o tratamento de dados, todos os ficheiros *pdf* selecionados foram convertidos em *xml* utilizando o comando `pdftohtml -xml` na linha de comandos. Após isso, procedeu-se à limpeza inicial de todos os documentos no formato *xml*, removendo-se toda a informação desnecessária quer utilizando a função `re.sub`, como também através de alguma limpeza manual.

4.1 glossario_ministerio_saude.pdf

Este documento obrigatório trata-se de um glossário de saúde do qual pode ser extraído informação relativamente a siglas e conceitos médicos. Desta maneira, o objetivo neste documento foi obter 2 ficheiros *json* um com informações de siglas e o seu significado e outro com informação de conceitos médicos, bem como a sua categoria e a sua descrição.

Uma vez que as primeiras páginas não contêm informação relevante para o objetivo, como a capa e informações de edição, estas páginas foram removidas manualmente. Foram também removidas manualmente páginas que estivessem em branco e as últimas páginas do documento, dado que, novamente, não contêm informação relevante.

Após esta remoção de páginas manual, foi necessário realizar um *script* em *Python* para o resto da limpeza do ficheiro e para a extração de informação do documento.

Em primeiro lugar, para a extração dos conceitos, começou-se por fazer os *imports* necessários e abrir o documento em modo leitura, como se pode observar na figura 4.1:

```python
import re, json

filename = "glossario_ministerio_saude.xml"

with open(filename, 'r', encoding='utf-8') as file:
    ficheiro = file.read()
```

Figura 4.1: *Imports* e abertura do ficheiro para leitura

De seguida, foi necessário remover linhas recorrentes no *xml* que dificultassem a extração da informação necessária. Algumas das linhas removidas incluem:

- Linhas que indicam o começo ou término de uma página;
- Imagens;
- Linhas de texto, cabeçalhos ou rodapés;
- Caracteres especiais.

A remoção ou substituição por *strings* vazias, destas linhas encontra-se na figura 4.2. Note-se ainda que o início de uma categoria foi marcado com um "*"de modo a ser mais fácil a sua deteção.

```python
#Remocao de linhas não necessárias
ficheiro = re.sub(r"</?page.*?>",r"",ficheiro) #serve para remover a tag page
ficheiro = re.sub(r"</?pdf2xml.*?>",r"",ficheiro)
ficheiro = re.sub(r"</?image.*?>",r"",ficheiro)

padrao = r'<text.*font="(22|15|25|23)".*>.*</text>\n?' #padrao para remover tudo o que for texto com font="15" ou "22" ...
padrao2 = r'<text.*top="238".*>.*</text>\n?' #Remove texto dos cabeçalhos
ficheiro = re.sub(padrao, r"", ficheiro)
ficheiro = re.sub(padrao2, r"", ficheiro)
ficheiro = re.sub (r"</?text.*?>", "", ficheiro) #*? para ele parar no primeiro > e não retirar info importante
ficheiro = re.sub (r"<i>Categoria: </i>", "*", ficheiro)
ficheiro = re.sub (r"- ", "", ficheiro) #usado para tirar quando é quebra de linha
ficheiro = re.sub (r" □ ", "-", ficheiro)
ficheiro = re.sub('^\s*$', r"", ficheiro, flags=re.MULTILINE) #Remove linhas vazias
ficheiro = re.sub (r"<i>", "", ficheiro) #É preciso ser retirado isto porque há termos em ingles em italico como "Western blot"
ficheiro = re.sub (r"</i>", "", ficheiro)
```

Figura 4.2: Remoção de informação não relevante em *Python*

Posteriormente, foi preciso fazer dois processos para a obtenção dos conceitos corretamente. O primeiro passou por juntar linhas consecutivas do nome do conceito, uma vez que o nome de alguns conceitos encontrava-se cortado em duas linhas. O segundo passa pela obtenção da informação relevante através de grupos de captura:

1. O primeiro grupo de captura é referente ao nome do conceito, visto que o mesmo se encontra entre e ;

2. O segundo é referente à categoria, já que o início da mesma está marcada por um "*"e o término por um "\n";

3. Por fim o último grupo de captura é a descrição do conceito que começa no término da categoria e vai até ao início do próximo conceito, marcado pelo <b>.

Estas expressões regulares encontram-se na figura 4.3:

```
ficheiro = re.sub(r'</b>\n<b>(.*?)</b>\n', r'\1</b>\n', ficheiro) #Junta duas linhas consecutivas de <b> ou seja de nome de termos cortados

lista = re.findall(r"<b>(.*?)</b>\n(?:\*\n)?(.*?)\n(.*?)<b>", ficheiro, re.DOTALL)
```

Figura 4.3: Expressões regulares para a captura dos conceitos

Por fim, tendo já obtido cada conceito e a sua informação foi necessário incorporá-la num dicionário para ser escrito num ficheiro *json*.

```
# Processar os termos
novos_conceitos = []
glossario = {}
for termo, categoria, descricao in lista:
    novo_termo = termo.strip()
    novo_termo = re.sub (r"<b>", "", novo_termo)
    novo_termo = re.sub (r"</b>", "", novo_termo)
    novo_termo = re.sub (r"\n", "", novo_termo)
    nova_categoria = categoria.strip()
    nova_categoria = re.sub (r"<b>", "", nova_categoria)
    nova_categoria = re.sub (r"</b>", "", nova_categoria)
    nova_categoria = re.sub (r"\n", "", nova_categoria)
    nova_descricao = descricao.strip()
    nova_descricao = re.sub (r"\n", "", nova_descricao)
    if novo_termo not in glossario:
        if nova_descricao !="":
            glossario[novo_termo] = {"Categoria": nova_categoria, "Descricao": nova_descricao}
        elif nova_descricao == "": #Quando não tem categoria um termo - antes a descrição era atribuida a categoria então foi necessário trocar
            glossario[novo_termo] = {"Categoria": "Sem Categoria", "Descricao": nova_categoria}

file_out = open("conceitos.json","w",encoding= 'utf-8')
json.dump(glossario,file_out,indent=4,ensure_ascii=False)
file_out.close()
```

Figura 4.4: Tratamento dos conceitos e inserção no dicionário e no *json*

Por outro lado, para a extração das siglas o processo efetuado foi muito semelhante ao previamente explicado, sendo necessário fazer os *imports*, a leitura do ficheiro e a limpeza de linhas ou páginas indesejadas no *xml*. Após isto, procedeu-se então à expressão regular que conseguisse obter o nome da sigla e o seu significado. Por fim, foi necessário meter esta informação noutro documento *json*. Este processo encontra-se na figura 4.5.

```python
siglas = re.findall(r"<b>(.+?)</b>\n([^<]+)",ficheiro) #separa a sigla da sua descricao

# Processar as siglas
novos_conceitos = []
for designacao,descricao in siglas:
    nova_desig = designacao.strip()
    nova_desig = re.sub (r"\n", "", nova_desig)
    nova_descri = descricao.strip()
    nova_descri = re.sub (r"\n", "", nova_descri)
    novos_conceitos.append((nova_desig,nova_descri))


conceitos_dict = dict(novos_conceitos)

file_out = open("siglas.json","w",encoding= 'utf-8')
json.dump(conceitos_dict,file_out,indent=4,ensure_ascii=False)
file_out.close()
```

Figura 4.5: Procedimento para a obtenção das siglas e do seu significado

Os documentos *json* obtidos encontram-se nas figuras 4.6 e 4.7:


```
{
    "Abordagem médica tradicional do adulto hospitalizado": {
        "Categoria": "Atenção à Saúde",
        "Descricao": "Focada em uma queixa principal e o hábito médico de tentar explicar todas as queixas e os sinais por um único
        diagnóstico, que é adequada no adulto jovem-não se aplica em relação ao idoso."
    },
    "Abuso incestuoso": {
        "Categoria": "Acidentes e Violência",
        "Descricao": "Consiste no abuso sexual envolvendo pais ou outro parente próximo, os quais se encon-tram em uma posição de maior
        poder em re-lação à vítima."
    },
    "Abuso sexual na infância": {
        "Categoria": "Acidentes e Violência",
        "Descricao": "É todo ato ou jogo sexual, relação heterosse-xual ou homossexual, cujo agressor está em estágio de desenvolvimento
        psicossexual mais adiantado que a criança ou adolescente. Tem por intenção estimulá-la sexualmente ou uti-lizá-la para obter
        satisfação sexual. Essas prá-ticas eróticas e sexuais são impostas à criança ou adolescente pela violência física, por amea-ças
        ou pela indução de sua vontade."
```


Figura 4.6: *Json* com a informação dos conceitos


```
{
    "AB": "Atenção Básica",
    "ABEn": "Associação Brasileira de Enfermagem",
    "ADT": "Assistência Domiciliar Terapêutica",
    "AFE": "Autorização de Funcionamento de Empresa",
    "AIDPI": "Atenção Integrada às Doenças Prevalentes na Infância",
    "AIDS": "Síndrome da Imunodefi ciência Adquirida",
    "AIH": "Autorização de Internação Hospitalar",
    "AIS": "Ações Integradas de Saúde",
    "ANCED": "Associação Nacional de Centros de Defesa",
    "ANS": "Agência Nacional de Saúde",
```


Figura 4.7: *Json* com a informação das siglas

4.2 Glossário de Termos Médicos Técnicos e Populares.pdf

À semelhança dos restantes documentos abordados, também este documento foi convertido para o formato *xml*, de modo a facilitar o tratamento de texto. O código para extrair os campos pretendidos foi desenvolvido num *script Python*.

Inicialmente, recorrendo-se à função `re.sub` foram eliminados alguns parâmetros prescindíveis. Começou por se retirar os elementos relativos à configuração estética, isto é, as especificações próprias de um ficheiro *xml*. Além disso, removeu-se também as *tags* de itálico presentes no texto correspondente às descrições dos termos, bem como, linhas com letras maiúsculas que se referem a transição de secções. Pode-se verificar o código criado para tal efeito na figura 4.8

```
texto = re.sub(r"</?page.*>", "", texto)  # remoção dos pages
texto = re.sub(r"</?text.*?>", "", texto)  # remoção da tag texto e
texto = re.sub(r"</?fontspec.*?>", "", texto) # remoção da linha com
texto = re.sub (r"<i>", "",texto) # remoção da tag itálico, onde se
texto = re.sub (r"</i>", "",texto) # remoção da parte final da tag
texto = re.sub(r"<b>[A-Z]</b>", "", texto) # remoção das linhas onde
```

Figura 4.8: Expressões regulares correspondentes à limpeza do ficheiro

De seguida, foi criada uma lista, com o auxílio da função `re.findall()`, para se reunirem todas as designações. A expressão regular utilizada encontra-se na figura 4.9.

```
lista_designacoes = re.findall(r"<b>(.*)</b>", texto)
```

Figura 4.9: Expressão regular para a captação das designações

Após a captura das designações, estas foram removidas de modo a restarem apenas as descrições populares. Note-se que as descrições antecedem o elemento (pop), assim, de modo a ser mais eficaz a captura destas expressões substitui-se este constituinte pelo marcador "@@". Existiam algumas incoerências, daí a adição da terceira linha do código presente na Figura 4.10.

Adicionalmente, retiraram-se as quebras de página, espaços e vírgulas desnecessárias.

```
texto2 = re.sub(r"<b>(.*)</b>", "", texto)

texto2 = re.sub(r"\(pop\)", "@@", texto2)
texto2 = re.sub(r"@@  @@", "@@" , texto2)

texto2= re.sub(r"\n+", "", texto2)
texto2 = re.sub(r"\s,", "", texto2)
```

Figura 4.10: Remoção das designações, substituição de "(pop)"por "@@"e resolução dos casos indesejados.

Após este tratamento foi possível a criação de uma expressão regular funcional, sendo assim capturadas as descrições. A lógica de captura é a extração de todos os carateres que se encontram entre os marcadores adicionados. Na `lista2`, encontram-se todos os campos pretendidos à exceção da primeira descrição, uma vez que, esta expressão não é antecedida pelo marcador. Neste sentido, a `lista3` inclui a primeira expressão e a `lista_descricoes` é a concatenação das listas mencionadas anteriormente, reunindo toda a informação desejada. Este processo pode ser verificado no exerto de código presente na figura 4.11

```
lista2 = re.findall(r"@(?:\s|,|x)(.*?)@", texto2)
lista3= re.findall(r"(^.*?)\s@", texto2) # devido
lista_descricoes = lista3 + lista2
```

Figura 4.11: Criação da lista contendo as descrições de todos os termos.

Após a criação das listas contendo as designações e descrições de cada termo, notou-se uma irregularidade, nomeadamente, o comprimento das mesmas, sendo que a lista de designações continha 4 elementos a mais. Após a verificação desta irregularidade, tentou-se perceber onde poderia estar a mesma, surgindo dois casos: nem todas as descrições continham a expressão (pop) utilizada como

marcador ou haveria designações a ser captadas de forma errada. Após análise do ficheiro, verificou-se a existência de algumas *tags* de término de *bold* seguidas por *tags* de início de bold, algo que não é necessariamente uma irregularidade visto que, as designações poderiam estar tanto no início da linha como no fim. No entanto, após uma análise mais detalhada, verificou-se que algumas descrições era separadas por uma quebra de linha, separando uma designação em duas. Posteriormente à identificação destas quatro situações e alteração direta no ficheiro *xml*, estas irregularidades encontram-se resolvidas.

Finalmente, foi criada a lista `termos`, que une as designações e as respetivas descrições. Assim, foi possível dar origem a um dicionário em que as *keys* são as designações e os *values* as descrições (figura 4.12). Este dicionário foi guardado num ficheiro *json*, tal como possível verificar na figura 4.13

```
termos = [[x, y] for x, y in zip(lista_designacoes, lista_descricoes)]

dicionario={}
dicionario = dict(termos)


out = open ("termos_glossario.json", "w", encoding="utf-8")
json.dump(dicionario, out, ensure_ascii=False, indent=4)
out.close()
```

Figura 4.12: Criação da lista termos contendo o par designação- descrição e posterior armazenamento no dicionario.


```
{
    "micrograma": " a milionésima parte de um grama ",
    "perioral": " à volta da boca ",
    "periorbital": " à volta da órbita ",
    "perivascular": " à volta dos vasos sanguíneos ",
    "depressão": "  abaixamento, abatimento, prostração ",
    "abcesso": "abcesso, tumor ",
    "empiema": " abcesso; acumulação de pus ",
    "abdómen": " barriga, ventre ",
    "abdominal": "ventral ",
    "aberrante": "anormal ",
    "perfuração": " abertura; orifício ",
    "extracção": " ablação ",
    "castração": "  ablação dos órgãos sexuais, capação, eviração, emasculação ",
    "anastomose": " comunicação natural ou artificial entre dois vasos ou nervos ",
    "aborto": "  abortamento, desmancho ",
    "abrupto": " repentino, brusco ",
    "absorção": " absorvimento, absorvência ",
```


Figura 4.13: Ficheiro *json* com a informação presente no dicionário.

4.3 WIPOPearl_COVID-19_Glossary.pdf

Este documento é composto por termos em inglês e cada termo tem a sua descrição, categoria e traduções. O objetivo neste documento seria extrair a informação corretamente de cada termo e separar cada uma das línguas.

À semelhança do que foi feito em documentos anteriores, foi removido em primeiro lugar informação não relevante, manualmente e também através de expressões regulares. Foi retirado do *xml* o seguinte:

- Da página 1 à 5 e a 38 (manualmente);
- Linhas no *xml* começadas por `<fontspec` (manualmente);
- Cabeçalhos e rodapés (*Expressões Regulares*);
- Texto não relevante (*Expressões Regulares*);

As alterações foram feitas no *script* em *Python* através de expressões regulares que podem ser observadas na figura 4.14:

```
#Remocao de linhas não necessárias
ficheiro = re.sub(r"</?page.*?>",r"",ficheiro) #serve para remover a tag page

padrao = r'<text.*font="(1|15|22)".*>.*</text>\n?'
padrao2 = r'<text.*top="(65|1131|1158|1185)".*>.*</text>\n?' #Remove texto dos cabeçalhos e rodape (nº paginas ...)
ficheiro = re.sub(padrao, r"", ficheiro)
ficheiro = re.sub(padrao2, r"", ficheiro)
ficheiro = re.sub(r'<text.*font="8".*><b>(.*)</b></text>\n?',r"*\1 *\n",ficheiro) #poe o nome do termo entre *
ficheiro = re.sub(r'<text.*font="11".*>(.*)</text>\n?', r"@\1\n", ficheiro) #poe a categoria a começar com @
ficheiro = re.sub (r"</?text.*?>", "", ficheiro)
ficheiro = re.sub (r"- ", "", ficheiro) #usado para tirar quando é quebra de linha
ficheiro = re.sub('^\s*$', r"", ficheiro, flags=re.MULTILINE) #Remove linhas vazias
ficheiro = re.sub (r"<i>", "", ficheiro)
ficheiro = re.sub (r"</i>", "", ficheiro)
```

Figura 4.14: Expressões regulares para limpeza do ficheiro

Novamente, houve a necessidade de criar marcas para marcar o nome do termo entre "*". Isto é importante pois marca o início do nome de um termo e o término do seu nome. Assim sendo, como o nome de um termo pode aparecer cortado em duas linhas, se houverem duas linhas consecutivas a começar por um "*", essas linhas são juntadas numa só, de forma a facilitar o resto da procura dos termos. Para encontrar as informações específicas de cada termo foi feita a expressão regular da figura 4.15:

```
# Ajuste para unir linhas de fonte "8" (nome do termo) consecutivas
ficheiro = re.sub(r'\*\n\*(.*?)\*\n', r'\1*\n', ficheiro)

padrao3 = r'\*(.*?)\s+\*\n(.*?)\n@(.*?)\n((?:(?!\*|@).)*)'
correspondencias = re.findall(padrao3, ficheiro, re.DOTALL)
```

Figura 4.15: Expressões regulares para captar a informação do termo

Por último, após a captação de cada termo e da sua respetiva informação, foi necessário haver um breve tratamento da mesma para a sua inserção no *json*, como a remoção de alguns caracteres especiais e remoção de algumas *tags*. O tratamento feito e a inserção no *json* podem ser observados na figura 4.16:

```
glossario = {}
for correspondencia in correspondencias:
    termo = correspondencia[0].strip()
    descricao = correspondencia[1].strip()
    categoria = correspondencia[2].strip()
    traducoes_raw = correspondencia[3].strip()
    traducoes = re.split(r'<b>\s*(.*?)\s*</b>', traducoes_raw)[1:] #divide quando encontra uma nova lingua

    #limpeza antes de meter no documento
    termo = re.sub (r"\*", "", termo)
    termo = re.sub (r"\n", " ", termo)
    descricao = re.sub (r"<b>", "", descricao)
    descricao = re.sub (r"</b>", "", descricao)
    descricao = re.sub (r"\n", "", descricao)
    traducoes = [re.sub(r"\n", "", traducao) for traducao in traducoes]

    glossario[termo] = {
        "Descricao": descricao,
        "Categoria": categoria,
        "Traducoes": {traducoes[i].strip(): traducoes[i+1].strip() for i in range(0, len(traducoes), 2)}
    }

file_out = open("wipo.json","w",encoding= 'utf-8')
json.dump(glossario,file_out,indent=4,ensure_ascii=False)
file_out.close()
```

Figura 4.16: Tratamento dos termos e inserção no *json*

Depois de correr este código, é obtido o seguinte *json*, com a informação exatamente como era pretendida:

```
{
    "acute respiratory distress syndrome": {
        "Descricao": "(syn.) ARDS  Respiratory disease characterized by the rapid onset of widespread inflammation in the lungs.",
        "Categoria": "MEDI, Pathology",
        "Traducoes": {
            "AR": "ةمزلاتمةقئاضلاةيسفنتلاةدا حلا",
            "DE": "akutes Atemnotsyndrom des Erwachsenen, (syn.) ARDS",
            "ES": "síndrome de dificultad respiratoria aguda, (syn.) SDRA",
            "FR": "syndrome de détresse respiratoire aiguë, (syn.) SDRA",
            "JA": "急性呼吸窮迫症候群, (syn.) 急性呼吸促迫症候群, ARDS",
            "KO": "급성  호흡곤란  증후군, (syn.) ARDS",
            "PT": "síndrome do desconforto respiratório agudo, (syn.) SDRA",
            "RU": "острый респираторный дистресс-синдром, (syn.) ОРДС",
            "ZH": "急性呼吸窘迫综合征, (syn.) ARDS"
        }
    },
    "adaptive immunity": {
        "Descricao": "(syn.) acquired immunity  Immunity acquired during a person's lifetime upon exposure to an infectious agent
        either by previous infection or immunization.",
        "Categoria": "MEDI, Anatomy &amp;amp; physiology",
        "Traducoes": {
            "AR": "ةعانمةبسكتم،(syn.) ةعانمةفيكتم ،ةعانمةبُيكت",
            "DE": "erworbene Immunität, (syn.) adaptive Immunität",
            "ES": "inmunidad adquirida, (syn.) inmunidad adaptativa",
            "FR": "immunité acquise, (syn.) immunité adaptative",
            "JA": "獲得免疫, (syn.) 適応免疫",
            "KO": "후천성  면역",
            "PT": "imunidade adquirida, (syn.) imunidade adaptativa",
            "RU": "приобретенный иммунитет, (syn.) адаптивный иммунитет",
            "ZH": "获得性免疫, (syn.) 适应性免疫"
        }
    },
```


Figura 4.17: *Json* com os termos e a sua respetiva informação

4.4 anatomia geral.pdf

O documento anatomia_geral.pdf contém definições de termos sobre anatomia do corpo humano e foi utilizado para enriquecer o ficheiro *Json* com um maior número de descrições. Tal como os outros documentos, este foi convertido em *xml* e passou-se para o desenvolvimento de um script *python* onde se realizaram os *imports* e se procedeu à abertura do ficheiro *xml*.

Inicialmente, procedeu-se ao *data cleaning*, de modo a extrair a informação desnecessária do documento *xml*, nomeadamente:

- Remoção das *pages*, *pdf2xml*, *fontspec* e *image*;
- remoção de informação contendo apenas letras maiúsculas que consistem em legenda das imagens presentes no documento;
- remoção de informação que contem dígitos correspondentes a legenda da imagem;

- remoção de informação não relevante, através da *font* utilizada, tal como, títulos, sub-títulos, dígitos presentes nas imagens, rodapés, notas, entre outras;
- substituição de mais do que um \n por apenas um, bem como remoção de espaçamentos a mais de forma a manter o texto organizado;
- Por fim remoção de informação irrelevante, presente no texto correspondente às descrições de cada termo.

Todo o tratamento inicial realizado no documento pode ser verificado na figura 4.18.

```
texto = re.sub(r"</?page.*>", "", texto)  # remoção dos pages
texto = re.sub(r"</pdf2xml>","", texto)
texto = re.sub(r"</?fontspec.*?>", "", texto)  # remoção da linha com a informação fontspec
texto = re.sub(r'<image[^>]*>', '', texto) # remoção das imagens
texto = re.sub(r"<text[^>]*\s*>\s*[A-Z](( [A-Z])+)?<\/text>", "", texto) # remoção de linhas cont
texto = re.sub(r"<text[^>]*>\s*(\d+)\s*</text>", "", texto) # remoção de linhas com digitos corres
texto = re.sub(r'<text.*font="(1|4|8|9|10|11|12|14|15|16|17|18|19)".*>.*</text>\n',"",texto) # rem
texto = re.sub(r'\n{2,}', '\n', texto)
texto = re.sub(r'^\s+|\s+$', '', texto, flags=re.MULTILINE)
texto= re.sub(r'<text[^>]*\sfont="3"[^>]*>\s*(.*?)\s*</text>', r'\1', texto)
```

Figura 4.18: Expressões regulares correspondentes à limpeza do ficheiro

Analogamente a outros documentos, houve a necessidade de recorrer a marcações, nomeadamente, das designações dos termos presentes no ficheiro, utilizando a marca "@@". Neste documento, houve a necessidade de ter um cuidado especial no momento da marcação, dado que, as designações dos termos não seguiam todas a mesma estrutura. Após análise do mesmo, é possível verificar quatro representações diferentes: negrito, itálico, negrito-itálico e uma fonte de letra diferente. Para identificar corretamente todas as designações recorreu-se às *font* de cada um dos casos. Na figura 4.19 pode-se verificar o excerto do *script* responsável pela marcação.

```
texto = re.sub(r'<text[^>]*\sfont="([567])"[^>]*>(.*?)</text>', r'@@\2', texto)
texto = re.sub(r'(<text[^>]*\sfont="13"[^>]*>)(.*?)(</text>)', r'@@\2', texto)
texto = re.sub(r'@@<([bi])>(.*?)</\1>', r'@@\2', texto) # remoção das tags bold e italic após a marcação
```

Figura 4.19: Expressões regulares correspondentes à marcação das designações

Posteriormente à marcação, é necessário retirar toda a informação necessária do texto. Por isso, recorre-se a uma expressão regular de forma a captar essa informação, utilizando os marcadores colocados previamente de forma, a localizar e separar a informação.

Após a captação da informação relevante é criado um dicionário para guardar o par chave-valor, isto é, designação-descrição. Antes da adição da informação no dicionário é necessário realizar algum tratamento nomeadamente:

- criação de uma lista `linhas` para cada `match` proveniente da expressão regular, através do `strip` e `split('\n')`.
- atribuição do `linhas[0]` à variável designação.
- atribuição dos restantes elementos da lista à variável `descrição`. Caso este termo não seja composto por descrição, é acrescentada a *string* "Sem Descrição"no seu lugar.
- Por fim é realizada alguma limpeza às descrições e o termo é adicionado ao dicionário.

Os passos referidos em cima, podem ser verificados na figura 4.20.

```
# Capturar designações e descrições
matches = re.findall(r'@@\s*(.+?)(?=\n@@|$)', texto, re.DOTALL)

# Criar dicionário com designações e descrições
dicionario = {}
for match in matches:
    linhas = match.strip().split('\n')
    designacao = linhas[0]
    descricao = '\n'.join(linhas[1:]) if len(linhas) > 1 else "Sem Descrição"
    descricao = descricao.replace('\n', '')
    descricao = re.sub(r"\s[A-Z]((, [A-Z])+)?$", "",descricao)
    dicionario[designacao] = descricao
```

Figura 4.20: Expressão regular para captação dos termos e posterior adição ao dicionário

Por fim, e tal como nos outros ficheiros, a informação final foi guardada num ficheiro *json*. De seguida, apresenta-se um excerto do ficheiro resultante na figura 4.21.

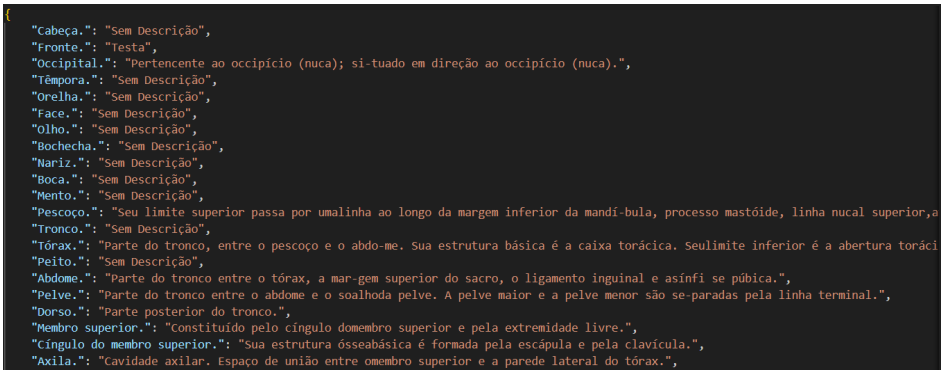

Figura 4.21: Fciheiro *json* obtido.

4.5 Junção dos ficheiros *Json*

De forma a concretizar o objetivo de obter um ficheiro *json* que fosse a compilação de outros 3 ficheiros menores, foi necessário fazer um *script* em *Python*.

Numa primeira parte, começou-se por fazer o *import* necessário e a leitura dos ficheiros, como consta na figura 4.22.

```
import json

json1 = 'conceitos.json'
json2 = 'termos_glossario.json'
json3 = 'termos_anatomia.json'

with open(json1, 'r', encoding='utf-8') as file:
    conceitos = json.load(file)

with open(json2, 'r', encoding='utf-8') as file:
    termos = json.load(file)

with open(json3, 'r', encoding='utf-8') as file:
    anatomia = json.load(file)
```

Figura 4.22: *Import* e leitura dos ficheiros *json*

De seguida, foi atribuida à variável `glossario` os valores que estavam no json "conceitos.json"e foi verificado se nos outros *json* existiam chaves iguais às já existentes, de forma a que se já existisse juntava a informação, caso contrário adicionava uma nova chave com a informação em questão. Este processo

encontra-se na imagem 4.23.

```
glossario = conceitos

for termo, informacao in termos.items():
    if termo in conceitos:
        # Se o termo já existir em conceitos.json, adicione uma nova chave "Descricao 2" com a nova informação
        glossario[termo]["Descricao 2"] = informacao
    else:
        # Se o termo não existir em conceitos.json, adicione-o ao dicionário conceitos com todas as suas informações
        glossario[termo] = {"Categoria": "Sem Categoria","Descricao": informacao,}

for termo, informacao in anatomia.items():
    if termo in conceitos:
        # Se o termo já existir em conceitos.json, adicione uma nova chave "Descricao Anatomia" com a nova informação
        glossario[termo]["Descricao Anatomica"] = informacao
    else:
        # Se o termo não existir em conceitos.json, adicione-o ao dicionário conceitos com todas as suas informações
        glossario[termo] = {"Categoria": "Sem Categoria","Descricao": informacao,}

file_out = open("glossario_geral.json","w",encoding= 'utf-8')
json.dump(glossario,file_out,indent=4,ensure_ascii=False)
file_out.close()
```

Figura 4.23: *Json* com os termos e a sua respetiva informação

Por fim, pode-se visualizar na figura 4.24 o *json* com a informação total. Note-se que neste *json* não existem termos repetidos e também não houve termos que se juntassem a outros.


```
    "Recesso esfenoetmoidal.": {
        "Categoria": "Sem Categoria",
        "Descricao": "Espaço, em forma defenda, acima da concha nasal superior."
    },
    "Meato nasofaríngeo.": {
        "Categoria": "Sem Categoria",
        "Descricao": "Parte da cavidade nasal quese estende da mar gem posterior das conchas na-sais até os cóanos."
    },
    "Cóano.": {
        "Categoria": "Sem Categoria",
        "Descricao": "Abertura nasal posterior. As duas aber-turas entre a cavidade nasal e a parte nasal dafaringe."
    },
    "Forame esfenopalatino.": {
        "Categoria": "Sem Categoria",
        "Descricao": "Orifício superior na fossapterigopalatina que conduz à cavidade nasal. Amaior parte é formada pelo palatino, e a
        menor,pelo esfenóide."
    }
}
```


Figura 4.24: *Json* com os termos e a sua respetiva informação

5. Conclusão

O trabalho prático realizado permitiu a extração de informação relevante e limpeza de elementos desnecessários de diferentes documentos médicos, com a utilização de expressões regulares e manipulação de textos. O objetivo proposto que consistia na construção de um ficheiro *json* com diversos termos médicos e as suas respetivas descrições, categorias e traduções, foi alcançado. Contudo, ao longo do trabalho foram surgindo algumas exceções durante o processamento dos documentos, o que gerou alguma dificuldade no manuseamento dos mesmos.

Para trabalho futuro, salienta-se a simplificação do código e consequente aumento da eficiência, bem como a possibilidade de trabalhar com mais documentos. Desta forma, o ficheiro final *json* seria composto com mais termos, e as suas respetivas categorias, designações e traduções. Em complementaridade, a utilização do *Google tradutor* seria uma mais valia usar no ficheiro *glossario_geral.json*, uma vez que permitiria a tradução dos termos e uma posterior junção ao ficheiro *wipo.json*.

Por fim, destaca-se a importância do processamento de linguagem natural na área da engenharia biomédica, devido ao acesso mais rápido e preciso a informações relevantes para a tomada de decisões clínicas, contribuindo assim para a melhoria da qualidade do serviço médico.