Universidade do Minho
Escola de Engenharia

Processamento de Linguagem Natural em Engenharia Biomédica - TP1

Ano Letivo 2023/2024

Informática Médica

Grupo x: Diogo Rodrigues PG54450

Lia Madurago PG53633

Inês Faria A95494

Docentes: Luís Filipe Costa Cunha

José João Antunes Guimarães Dias Almeida

Braga, 7 de abril de 2024

Resumo

O presente relatório insere-se no âmbito da Unidade Curricular de Processamento de Linguagem Natural em Engenharia Biomédica. Destina-se à extração de informação dos ficheiros fornecidos, guardando essa informação de forma conveniente para que possa ser utilizada em trabalhos futuros. Este trabalho foi realizado de acordo os conceitos lecionados na aula.

Índice

1 Introdução **1**

2 Desenvolvimento **2**

2.1 Análise do documento glossario_ministerio_saude.pdf 2

2.2 Análise do documento medicina.pdf 6

2.3 Análise do documento anatomia geral.pdf 11

2.4 Análise do documento Glossário de Termos Médicos e Populares.pdf 14

3 Conclusão **17**

4 Referências **17**

Índice de Figuras

Figura 1: Abertura do ficheiro XML. 2
Figura 2: Limpeza do ficheiro XML. 3
Figura 3: Continuação da limpeza do ficheiro XML. 4
Figura 4: Guardar alterações no ficheiro *glossario_ atualizado.xml*. 4
Figura 5: Guardar alterações no ficheiro *glossario_ atualizado.xml*. 4
Figura 6: Armazenamento das informações em listas. 5
Figura 7: *Script* que resulta na lista de tuplos com as informações descritas. . . . 5
Figura 8: Guardar alterações no ficheiro *glossario.json*. 6
Figura 9: Conteúdo do ficheiro *glossario.json*. 6
Figura 10: Abertura e limpeza do ficheiro XML. 7
Figura 11: Continuação da limpeza do ficheiro XML e a marcação do conteúdo. . . 7
Figura 12: Arquivo das alterações no ficheiro *med_ atualizado.xml* 8
Figura 13: Adição dos termos traduzidos às listas. 8
Figura 14: Adição final dos termos ao dicionário geral. 9
Figura 15: Ficheiro final. 9
Figura 16: Resultado do ficheiro json. 10
Figura 17: Abertura do ficheiro XML e função *clean*. 11
Figura 18: Limpeza do ficheiro xml. 11
Figura 19: Formatação padrão. 12
Figura 20: Formatação padrão. 13
Figura 21: Modificação da informação recolhida para formato de dicionário num ficheiro json. 13
Figura 22: Ficheiro anatomiageral.json. 13
Figura 23: Abertura do ficheiro XML e limpeza inicial. 14
Figura 24: Continuação da limpeza do XML e extração dos termos. 14
Figura 25: Continuação da limpeza do XML e colocação de marcação. 15
Figura 26: *Script* para criar e atualizar ficheiro *glossario_ termos_ populares.txt* . . 15
Figura 27: Visualização do conteúdo do ficheiro *glossario_ termos_ populares.txt* . . 15
Figura 28: Extração das designações e criação do dicionário. 16
Figura 29: Guardar resultados no ficheiro *json_ termos_ populares.json* 16
Figura 30: Visualização do conteúdo do ficheiro *json_ termos_ populares.json*. . . . 16

1 Introdução

O Processamento de linguagem natural (PLN) é uma vertente da inteligência artificial que é auxiliar aos computadores a interpretar e manipular a linguagem humana. Resulta de diversas disciplinas, incluindo ciência da computação e linguística computacional, que procuram colmatar a lacuna entre a comunicação humana e o entendimento dos computadores. [1]

Ainda que o processamento de linguagem natural não seja uma ciência atual, esta tecnologia está a avançar rapidamente graças ao interesse cada vez maior na comunicação homem-máquina, paralelamente à disponibilidade de big data, computação mais poderosa e algoritmos aprimorados. [1]

Algumas das aplicações mais importantes concentram-se na inteligência de negócios (business intelligence), que permite analisar automaticamente as reações dos clientes através do que publicam na Internet ou das perguntas que são feitas para obterem informações. Os chatbots são também uma aplicação que, embora ainda tenha uma grande margem de melhoria, agiliza a interação com os clientes via chats ou secretárias telefônicas, oferecendo respostas rápidas e automatizadas em função do processamento de linguagem natural. [2]

Além disso, o PLN é útil na classificação de grandes volumes de documentos de acordo com seu conteúdo ou estilo, o que pode agilizar a identificação de padrões, localização de coincidências entre textos e detecção de plágio, melhorando o controle de qualidade. O tradutor do Google, por exemplo, é uma aplicação notável de PLN, pois consegue capturar as nuances entre palavras diferentes dependendo do contexto. [2]

O trabalho prático 1 tem como objetivo aplicar o conhecimento adquirido na Unidade Curricular de Processamento de Linguagem Natural em Engenharia Biomédica (PLN) para extrair informações relevantes de documentos médicos em PDF e armazená-las para uso posterior. Como resultado, é necessário criar *parsers* que possam identificar e extrair as informações relevantes dos arquivso PDF, que serão posteriormente armazenados em arquivos JSON.

2 Desenvolvimento

Numa primeira fase do desenvolvimento do trabalho prático, para além do documento obrigatório, foram escolhidos três documentos PDF para selecionar a informação relevante de cada um. Os documentos escolhidos foram:

- "medicina.pdf";
- "anatomia geral.pdf";
- "Glossário de Termos Médicos e Populares.pdf".

Assim, de modo a proceder à limpeza e análise dos documentos, os ficheiros PDF foram convertidos para documentos XML através do seguinte comando:

- pdftohtml -c -xml nomeFicheiro.pdf .

Posteriormente, é explicada detalhadamente a limpeza e a seleção da informação em cada um dos documentos escolhidos.

2.1 Análise do documento glossario_ministerio_saude.pdf

O documento glossario_ministerio_saude.pdf foi transformado numa lista de listas onde cada sublista contém três elementos:

- "termo";
- "categoria (se presente)";
- "descrição".

Assim, para iniciar a análise do documento, foi necessário converter o documento PDF em XML através do comando anteriormente citado. De seguida, criou-se um *script* em Python para proceder à limpeza e extração de informação útil do documento em questão.

Inicialmente, foi necessário importar as bibliotecas necessárias para todo este processo, bem como o ficheiro

```
1  import re
2  import json
3
4  ficheiro = open('glossario_ministerio_saude.xml','r',  encoding="utf-8")
5  texto = ficheiro.read()
```

Figura 1: Abertura do ficheiro XML.

Após a análise do documento, iniciaram-se as operações para a remoção de conteúdos que não eram importantes no seguimento deste trabalho. Para isso, removeu-se a parte inicial e final do documento por não possuírem informação relevante. Para além disso, foram removidas as tags de XML de *<page>*, *<text>*, *<image>*, *<pdf2xml>*, *<DOCTYPE*, *<xml>*, *<pdf>* e *<fontspec>* por não serem significativas na exploração do problema. Os números de página foram também retirados pelo mesmo motivo.

De seguida, frases que começassem com * , siglas de 3 maiúsculas consecutivas, sequências de três números consecutivos, foram também excluídas do documento final. As tags de XML ** e ** foram também removidas para permitir uma melhor extração dos conteúdos pretendidos.

```
#Limpeza de texto e marcação
texto = re.sub(r'<page number="107" position="absolute" top="0" left="0" height="1263" width="892">\n[\d\D]+',r'',texto) # Remove parte final
texto = re.sub(r'<page number="15" (.*\n*){11}', r'', texto) # Remover parte inicial
texto = re.sub(r"</?page.*>","", texto)  # Remover <page> e </page>
texto = re.sub (r"</?text.*?>", "", texto) # remover <text> e </text>
texto = re.sub(r"</?fontspec.*?>", "", texto) # Remover <fontspec> e </fontspec>
texto = re.sub(r'</?pdf2xml.*?>', r'', texto)
texto = re.sub(r'<image.+/>', r'',texto)
texto = re.sub(r"<[?]xml(.*)?>", "", texto) # Remover <xml> e </xml>
texto = re.sub(r"<[!]DOCTYPE(.*)?>","", texto) # Remover <DOCTYPE> e </DOCTYPE>
texto = re.sub(r"</?pdf(.*)?>", "", texto) # Remover <pdf> e </pdf>
texto = re.sub(r'<b>\d{2}[15|3|5|7|9]</b>\n<b>.*</b>\n', r'', texto) # Retirar número de páginas ímpares
texto = re.sub(r'<b>.*</b>\n<b>\d{2}[16|2|4|6|8]</b>\n', '', texto) # Retirar número de páginas pares

texto = re.sub(r"<text(.*)>\s*(\*)(.*)</text>", "", texto) # Remover frases inicializadas com *, ** e ***
texto = re.sub(r"\n(\s*)\n", "\n", texto) # Remover espaços
texto = re.sub(r"[A-Z]\n\d{1,3}", "", texto) # Retira letras
texto = re.sub(r"\d{1,3}\n", "", texto) # Retira números
texto = re.sub(r"<b> </b>", "", texto) # Remove os <b> e </b>
```

Figura 2: Limpeza do ficheiro XML.

No seguimento do mesmo raciocínio, para harmonizar todo o conteúdo, todos os titulos foram colocados na mesma linha. O mesmo processo foi aplicado ao conteúdo.

Já com uma visão na extração de caraterísticas importantes, alterada a forma como visualizamos o campo *Categoria*, removendo algumas diferenças que existiam, tais como espaços, para que assim fique uniforme a representação de todos os campos *Categoria* ao longo do documento, como podemos observar através da figura abaixo apresentada. Os caracteres especiais que apareciam no texto foram removidos também por uma questão de congruência, como é o caso do e @.

```
texto = re.sub(r'<text .+?>',"",texto) # retira <text top="605" left="176" width="289" height="16" font="14"> de todo o texto
texto = re.sub(r'<position="absolute" top="0" left="0" height="918" width="663">[\d\D]+',r'',texto)
texto = re.sub(r'</text>', r'', texto) # Remove os </text>
texto = re.sub(r'</b>\n<b>','',texto) # Colocar os títulos todos na mesma linha
texto = re.sub(r'</i>\n<i>','',texto) # Colocar as categorias todas na mesma linha


texto = re.sub(r'<i>([^Categoria])(.*)</i>',r'\1\2', texto)
texto = re.sub(r"<i>Categoria:?\s?</i>\n(.*)\n(.*)\s\n\s", r"<i>Categoria: \1\2</i>\n", texto)
texto = re.sub(r"<i>Categoria:?\s?</i>\n(.*)\n", r"<i>Categoria: \1</i>\n", texto)
texto = re.sub(r"em </i>\nSaúde", r" em Saúde</i>\n", texto)
texto = re.sub(r"</i>\n", r"</i>\n@", texto)
texto = re.sub(r"\n<b>", r"@\n<b>", texto)
texto = re.sub(r"\\n", "", texto)
texto = re.sub(r"\\n@", "", texto)
```

Figura 3: Continuação da limpeza do ficheiro XML.

Nesta fase, guardaram-se as alterações no ficheiro *glossario_ atualizado.xml* para observar as alterações realizadas.

```
46  file = open('glossario_atualizado.xml', 'w',encoding = 'utf-8')
47  file.write(texto)
48  file.close
49
```

Figura 4: Guardar alterações no ficheiro *glossario_ atualizado.xml*.

Nesta etapa, foi inciada a extração de caraterísticas. Extrai tudo o que se encontra entre $$ e $$, ou seja, os termos. Também faz a extração do que se encontra entre @ (descrições), bem como as categorias atraves do $<i>$. Todos estes dados são armazenados em variáveis distintas como seria espectável.

```
52  #Extração de Informação
53
54  dicionario = {}
55  #ver termos que n tem descrições
56
57  all_titles = re.findall(r'<b>(.+)</b>',texto) # Extrai o título
58  all_descriptions = re.findall(r'@([^@<]+)@', texto) # Extrai tudo o que está a frente do arroba
59  all_categories = re.findall(r'<i>.+\n.+',texto) # Extrai Categoria: categoria
60
```

Figura 5: Guardar alterações no ficheiro *glossario_ atualizado.xml*.

Após isso armazenaram-se os termos, descricoes e categorias em listas distintas.

```
61    titulos = []
62    descricoes = []
63    categorias = []
64
65    for titulo in all_titles:
66        titulos.append(titulo)
67
68    for categoria in all_categories:
69        categorias.append(categoria)
70
71    for descricao in all_descriptions:
72        descricoes.append(descricao)
```

Figura 6: Armazenamento das informações em listas.

Foi posteriormente necessário criar tuplos, onde cada tuplo contém informações de *título*, e *descrição*, ou seja, os conteúdos extraídos anteriormente.

É também garantido que o item *categoria* apenas está presente nos casos corretos, visto que não é uniforme ao longo do documento.

O tuplo tem como primeiro elemento o *título*, o segundo elemento é a *categoria* se existir, caso contrário, é None, e o terceiro elemento é a *descrição*.

No final, a obtemos uma lista de tuplos, onde cada um possui as informações anteriormente descritas.

```
95  # Verificar o comprimento das listas e usando o menor comprimento para iterar
96  lista = []
97  comprimento_minimo = min(len(titulos), len(descricoes), len(categorias))
98
99  for i in range(comprimento_minimo):
100     tuplo = (titulos[i], categorias[i] if i < len(categorias) else None, descricoes[i])
101     lista.append(tuplo)
```

Figura 7: *Script* que resulta na lista de tuplos com as informações descritas.

Finalmente, toda a informação relevante é exportada no formato json para o ficheiro *glossario.json*

```
103    output = open("glossario.json", "w", encoding="utf-8")
104    json.dump(lista, output, ensure_ascii=False, indent=4)
105
106    output.close()
107
108    ficheiro.close()
```

Figura 8: Guardar alterações no ficheiro *glossario.json*.

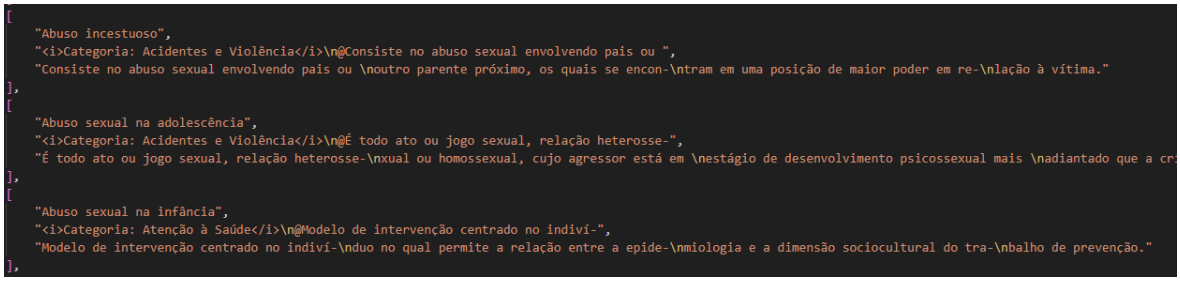

Figura 9: Conteúdo do ficheiro *glossario.json*.

2.2 Análise do documento medicina.pdf

O documento medicina.pdf foi transformado num dicionário no qual as traduções dos termos do dicionário foram organizados pelas línguas de traduções. Deste modo, cada língua está organizado por um dicionário e, posteriormente, por uma lista dos termos traduzidos.

Assim, como anteriormente, o documento PDF foi convertido para um documento XML, sendo desenvolvido um *script* Python para lidar com os respetivos dados. Desta forma, procedeu-se à limpeza do ficheiro xml. Assim, as expressões regulares apresentadas na figura são utilizadas para remoção de determinados padrões de texto para, posteriormente, ser mais fácil a manipulação da informação.

Primeiramente, importou-se novamente as bibliotecas necessárias e procedeu-se à remoção da parte inicial e final do documento por não possuírem informação relevante, nomeadamente a informação não referente às traduções dos termos. Para além disso, , foram removidas as *tags* de XML de <page>, <text>, <image>, <pdf2xml>, <DOCTYPE, <xml>, <pdf> e <fontspec> por não serem significativas na exploração do problema, assim como informações relativas ao cabeçalho.

Por fim, procedeu-se à remoção de espaços entre palavras, bem como aos espaços entre letras.

```
import re
import json
ficheiro = open('medicina.xml','r')
texto =ficheiro.read()

#Limpar o texto
texto = re.sub(r'^[\d\D]+<text\stop="128"\sleft="121"\swidth="3"\sheight="14"\sfont="17">\s</text>',r'',texto)  # Remover parte inicial
texto = re.sub(r'<page number="544" position="absolute" top="0" left="0" height="918" width="663">\n[\d\D]+',r'',texto) # Remover parte final
texto = re.sub(r"</?page.*>","", texto)  # Remover <page> e </page>
texto = re.sub (r"</?text.*?>", "", texto) # remover <text> e </text>
texto = re.sub(r"</?fontspec.*?>", "", texto) # Remover <fontspec> e </fontspec>
texto = re.sub(r'</?pdf2xml.*?>', r'', texto)
texto = re.sub(r"<[?]xml(.*)?>", "", texto) # Remover <xml> e </xml>
texto = re.sub(r"<[!]DOCTYPE(.*)?>","", texto) # Remover <DOCTYPE> e </DOCTYPE>
texto = re.sub(r"</?pdf(.*)?>", "", texto) # Remover <pdf> e </pdf>
texto = re.sub(r"\n(\s*)\n", "", texto) # Remover espaços
texto = re.sub(r'[ ]{2,}', r'', texto) # Remover espaços nos meios das palavras
texto = re.sub(r'V\nocabulario\n\d+',r'',texto) # Apagar informações sobre o cabeçalho
texto = re.sub(r'<text[^>]*><b>[A-Z]</b>\s*</text>\n', '', texto) # Remover letras isoladas s/ espaços
```

Figura 10: Abertura e limpeza do ficheiro XML.

No que toca à reformulação estética do documento, procedeu-se à colocação das categorias dos termos em linhas diferentes paea a extração mais eficiente do que era pretendido neste documento. Para além disso, removeu-se *tags* sem importância para este trabalho.

Posteriormente, realizou-se a marcação dos diferentes tipos de conteúdo presentes no documento, nomeadamente os sinónimos, as variantes, as notas e as devidas traduções - espanhol, inglês, latim e português. Esta marcação teve como objetivo novamente facilitar a extração da informação e também para tornar a identificação da mesma mais intuitiva.

```
texto = re.sub(r"</b>+", r"</b>\n", texto) # Colocar as categorias em linhas diferentes
texto = re.sub(r'<b></b>', r'', texto)
texto = re.sub(r'<b> </b>', r'', texto)
texto = re.sub(r'<i> </i>', r'', texto)


#Marcação do texto
texto = re.sub(r'(SIN\.-)', r'\n\1@', texto) #Sinónimos
texto = re.sub(r'(VAR\.-)', r'\n\1§', texto) #Variantes
texto = re.sub(r'(Nota\.-)',r'\n\1£',texto) #Notas


#Traduções
texto = re.sub(r'(\bes\b)', r'\n\1#', texto) #Espanhol
texto = re.sub(r'(\ben\b)', r'\n\1$', texto) #Inglês
texto = re.sub(r'(\bla\b)', r'\n\1%', texto) #Latim
texto = re.sub(r'(\bpt\b)', r'\n\1&', texto) #Português
```

Figura 11: Continuação da limpeza do ficheiro XML e a marcação do conteúdo.

Após todo o tratamento anterior, guardaram-se as alterações no ficheiro *med_ atualizado.xml*.

```
file = open('med_atualizado.xml', 'w',encoding = 'utf-8')
file.write(texto)
file.close
```

Figura 12: Arquivo das alterações no ficheiro *med_ atualizado.xml*

Após guardar as alterações, efetuou-se, finalmente, a extração de informação relativa ao vocabulário, especialmente das suas traduções. Deste modo, criaram-se 5 listas diferentes, cada uma referente às línguas mencionadas no documento inicial, fazendo, posteriormente, a adição de todas as traduções às respetivas listas.

```
portugues = []
espanhol = []
ingles = []
latim = []
portugues = []

all_ES = re.findall(r'\#\s*<i>(\w+)</i>', texto)
for palavra in all_ES:
    espanhol.append(palavra)

all_IN = re.findall(r'\$\s*<i>(\w+)</i>', texto)
for palavra in all_IN:
    ingles.append(palavra)

all_LA = re.findall(r'\%\s*<i>(\w+)</i>', texto)
for palavra in all_LA:
    latim.append(palavra)

all_PT = re.findall(r'\&\s*<i>(\w+)</i>', texto)
for palavra in all_PT:
    portugues.append(palavra)
```

Figura 13: Adição dos termos traduzidos às listas.

Foram, depois, armazenados nos respetivos dicionários, sendo esses dicionários depois colocados num dicionário geral.

```
dicE = {'Termos': espanhol}
dicI = {'Termos': ingles}
dicL = {'Termos':  latim}
dicP = {'Termos': portugues}

dicionario = {}
dicionario['Espanhol'] = dicE
dicionario['Inglês'] = dicI
dicionario['Latim'] = dicL
dicionario['Português'] = dicP
```

Figura 14: Adição final dos termos ao dicionário geral.

Por fim, toda a extração efetuada foi armazenada em um novo ficheiro, nomeadamente um ficheiro json.

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

output.close()

ficheiro.close()
```

Figura 15: Ficheiro final.

```
{
    "Espanhol": {
        "Termos": [
            "ala",
            "abdomen",
            "ablación",
            "abordaje",
            "absceso",
            "abulia",
            "acalasia",
            "acantolisis",
            "ácaro",
            "aceptación",
            "acetábulo",
            "acetilcolina",
            "acidemia",
            "acidificación",
            "acidificante",
            "acidosis",
            "aciduria",
            "acinesia",
            "acino",
            "acolia",
            "acomodación",
            "acomodación",
            "acondroplasia",
            "acrocianosis",
            "acromegalia",
    },
    "Inglês": {
        "Termos": [
            "wing",
            "abdomen",
            "abduction",
            "ablation",
            "approach",
            "abscess",
            "aboulia",
            "achalasia",
            "acantholysis",
            "acarus",
            "acceptance",
            "acetabulum",
            "acetylcholine",
            "acidemia",
            "acidification",
            "acidifier",
            "glutamate",
            "acidosis",
            "aciduria",
            "akinesia",
            "acinus",
            "achlorhydria",
            "acne",
            "acholia",
            "accommodation",
            "accommodation",
            "achondroplasia",
            "event",
            "acrocyanosis",
            "acrocinesia",
```


Figura 16: Resultado do ficheiro json.

2.3 Análise do documento anatomia geral.pdf

O documento anatomia_geral.pdf foi convertido num dicionário cujas *keys* são os termos em português e os values são as suas definições.

Mais uma vez, o PDF foi convertido em XML e, em seguida, desenvolveu-se um arquivo Python para lidar com esses dados. Inicialmente, foi introduzida a função *clean*, que tem como objetivo remover qualquer conteúdo desnecessário no resultado final, como '\n'.

```
import re
import json

file = open("C:\\Users\\liama\\OneDrive\\Ambiente de Trabalho\\anatomiageral\\Anatomia_GeralORIGINAL.xml", "r", encoding="utf-8")
text = file.read()

def clean(text):
    text = re.sub(r"\s+", " ", text)   # substitui todos os espaços e \n por apenas 1 espaço
    return text.strip() # Remove o \n no início e fim da frase
```

Figura 17: Abertura do ficheiro XML e função *clean*.

De seguida, procedeu-se à limpeza do ficheiro xml. As expressões regulares, representadas na seguinte Figura X, são usadas para remover determinados padrões de texto de uma *string*. Procuram por padrões específicos, como tags HTML/XML e respetivos conteúdos, substituindo-os por uma *string* vazia, removendo assim esses elementos do texto. A sua utilidade consiste maioritariamente na limpeza e pré-processamento do texto a tratar, removendo partes indesejadas ou desnecessárias antes de realizar outras operações de processamento de texto.

Na seguinte figura encontra-se em comentário a função sucinta de cada expressão regular em causa.

```
text = re.sub(r"</?page.*>","", text)  # Remover <page> e </page>
text = re.sub(r"</?image.*?>", "", text) # Remover <image> e </image>
text = re.sub(r"<[!]DOCTYPE(.*)?>","", text) # Remover <DOCTYPE> e </DOCTYPE>
text = re.sub(r"</?pdf(.*)?>", "", text) # Remover <pdf> e </pdf>
text = re.sub(r"</?fontspec.*?>", "", text) # Remover <fontspec> e </fontspec>
text = re.sub(r"<[?]xml(.*)?>", "", text) # Remover <xml> e </xml>
text = re.sub(r"<i><b>(.*)</b></i>", "", text) # Remover <i><b> e </b></i>
text = re.sub(r"</?text.*?>", "", text) # Remover <text> e </text>
```

Figura 18: Limpeza do ficheiro xml.

Na Figura 19, encontram-se representadas as expressões regulares , utilizadas para modificar o texto de acordo com determinados padrões pré definidos.

A primeira remove todos os números presentes no texto. A segunda elimina espaços seguidos por uma letra maiúscula e por determinados caracteres especiais. A terceira exclui

todo o conteúdo entre as tags e . A quarta substitui todas as ocorrências de por . A quinta substitui todas as ocorrências de por . A sexta remove todas as quebras de linha repetidas. A sétima substitui seguido de um ou mais espaços e por . A oitava substitui por uma quebra de linha seguida de . A nona adiciona @ antes de cada . A décima remove os delimitadores [[e]], mantendo o conteúdo. A décima primeira remove seguido opcionalmente de @ e novamente . A décima segunda une expressões que estão separadas pelas regras de translineação. A décima terceira elimina espaços em branco. A décima quarta remove frases que começam com *, ** e ***.

Importante denotar que, teve-se em consideração uma situação específica em que existem palavras sem definição também em formatação *bold* a serem mantidas. O texto foi ajustado para evitar incluir essas palavras na extração de informações, selecionando apenas as desejadas. Note-se que as palavras em formato *bold* sem definição estão imediatamente antes das restantes a *bold*.

```
text = re.sub(r"</?page.*>","", text)  # Remover <page> e </page>
text = re.sub(r"</?image.*?>", "", text) # Remover <image> e </image>
text = re.sub(r"<[!]DOCTYPE(.*)?>","", text) # Remover <DOCTYPE> e </DOCTYPE>
text = re.sub(r"</?pdf(.*)?>", "", text) # Remover <pdf> e </pdf>
text = re.sub(r"</?fontspec.*?>", "", text) # Remover <fontspec> e </fontspec>
text = re.sub(r"<[?]xml(.*)?>", "", text) # Remover <xml> e </xml>
text = re.sub(r"<i><b>(.*)</b></i>", "", text) # Remover <i><b> e </b></i>
text = re.sub(r"</?text.*?>", "", text) # Remover <text> e </text>
```

Figura 19: Formatação padrão.

Dado que as informações a retirar deste documento xml são as palavras em formato *bold*, os termos em português e as suas definições, seguidas das palavras a *bold*, foi formulado a expressão regular caracterizada na Figura 20, com dois grupos de captura, onde o primeiro se refere aos termos e o segundo às descrições, permitindo diferenciar as palavras com formato *bold* e descrição e as que estão com a mesma formatação, mas não têm descrição.

Os grupos de captura descritos anteriormente são depois incluídos numa lista de listas, que reúne cada designação com a respetiva descrição.

```python
# Pesquisar a expressão
list = re.findall(r"<b>\s(.*?)\.</b>(.*?)\s+@", text)
nova_lista = [(designacao, re.sub(r"@<b>|</b>"," ", descricao)) for designacao, descricao in list] #remover @<b> de alguns termos que originava erros

# Conversão
new_list = []
for designacao, descricao in nova_lista:
    new_designacao = designacao.lower()
    new_descricao = clean(descricao)
    new_list.append([new_designacao, new_descricao])

list = new_list
```

Figura 20: Formatação padrão.

Após a extração das informações relevantes e sua organização, o passo seguinte foi converter essa estrutura num dicionário, onde as chaves representam os termos e os valores as definições. Esse dicionário foi então convertido para o formato json e armazenado num arquivo (Figura 21), conforme o objetivo estabelecido previamente. Esse processo permite a fácil manipulação e armazenamento dos dados extraídos.

```python
dicionario = dict(list)

output = open("C:\\Users\\liama\\OneDrive\\Ambiente de Trabalho\\anatomiageral\\anatomiageral.json", "w", encoding="utf-8")
json.dump(dicionario, output, ensure_ascii=False, indent=4)

output.close()

file.close()
```

Figura 21: Modificação da informação recolhida para formato de dicionário num ficheiro json.

Por fim, na seguinte Figura X, pode visualizar-se o conteúdo que resultou do processo descrito anteriormente do ficheiro anatomiageral.json.


```
{
    "fronte": "Testa",
    "occipital": "Pertencente ao occipício (nuca); si-tuado em direção ao occipício (nuca).",
    "pescoço": "Seu limite superior passa por uma linha ao longo da margem inferior da mandí-bula, processo mastóide, linha nucal superior, até a protuberância occipital externa; seu limi-te inferior estend
    "tórax": "Parte do tronco, entre o pescoço e o abdo-me. Sua estrutura básica é a caixa torácica. Seu limite inferior é a abertura torácica inferior e o diafragma.",
    "abdome": "Parte do tronco entre o tórax, a mar-gem superior do sacro, o ligamento inguinal e a sínfi se púbica.",
    "pelve": "Parte do tronco entre o abdome e o soalho da pelve. A pelve maior e a pelve menor são se-paradas pela linha terminal.",
    "dorso": "Parte posterior do tronco.",
    "membro superior": "Constituído pelo cíngulo do membro superior e pela extremidade livre.",
    " cíngulo do membro superior": "Sua estrutura óssea básica é formada pela escápula e pela clavícula.",
    "axila": "Cavidade axilar. Espaço de união entre o membro superior e a parede lateral do tórax.",
    "membro inferior": "Constituído pelo cíngulo do membro inferior e pela extremidade livre.",
    " cíngulo do membro inferior": "Sua estrutura óssea básica é formada pelo osso do quadril.",
    "quadril": "Região de união da pelve com a extre-midade livre do membro inferior.",
    "região poplítea": "Fossa superfi cial do joelho.",
    "sura": "Panturrilha.",
    "cavidade abdominopélvica": "Cavidade conjunta do abdome e da pelve.",
    "mediano": "Situado no plano mediano.",
    "coronal": "Situado no plano da sutura coronal.",
    "sagital": "Situado no plano da sutura sagital, ou situado em direção dorsal-ventral.",
    "intermédio": "Situado entre duas estruturas.*",
    "medial": "Situado próximo do plano mediano.",
    "lateral": "Situado afastado do plano mediano.",
    "anterior": "Situado à frente de.",
    "posterior": "Situado atrás de.",
    "ventral": "Situado em direção ao ventre.",
    "dorsal": "Situado em direção ao dorso.",
    "frontal": "Pertencente à fronte (testa); situado paralelamente à fronte (testa).",
    "superior": "Situado em direção à cabeça.",
    "inferior": "Situado em direção à cauda (cóccix).",
    "cranial": "Pertencente ao crânio; situado em dire-ção do crânio.",
    "caudal": "Situado em direção à cauda.",
    "rostral": "Situado em direção ao rostro do corpo caloso.",
    "apical": "Pertencente ao ápice; em direção ao ápice.",
```


Figura 22: Ficheiro anatomiageral.json.

2.4 Análise do documento Glossário de Termos Médicos e Populares.pdf

O documento Glossário de Termos Médicos e Populares.pdf foi transformado num dicionário no qual as palavras em português são usadas como chaves e suas respectivas definições como valores.

Inicialmente o ficheiro PDF foi convertido em XML através do comando anteriormente descrito. Para a limpeza e respetiva extração do conteúdo pretendido, foi desenvolvido um *script* em Python para essa finalidade.

Já no *script* Python, foram importados os módulos necessários. Foi também aberto o ficheiro XML anteriormente citado. Após isso foi feita uma limpeza inicial de caraterísticas intrínsecas ao XML, mais propriamente as *tags* XML <fontspec>, <page> e <text>.

```
1  import re
2  import json
3
4  ficheirotxt = open("Glossario_de_termos_Medicos_Tecnicos_Populares.xml", encoding="utf-8")
5  text = ficheirotxt.read()
6
7  text = re.sub(r"</?fontspec.*?>", "", text)
8  text = re.sub(r"</?page.*>", "", text)
9  text = re.sub(r"</?text.*?>", "", text)
```

Figura 23: Abertura do ficheiro XML e limpeza inicial.

De seguida, ainda na fase de limpeza do ficheiro, todos os termos que correspondem a apenas uma maiúscula são eliminados, tendo em conta que letras maiúsculas apenas aparecem nessas situações. Após isso, são retirados do XML todas as *tags* e para facilitar a recolha das descrições dos termos.

Posteriormente, são recolhidos todos os termos para a variável *termos* como podemos ver através da figura seguinte.

```
12 text = re.sub(r"<b>[A-Z]</b>", "", text) #termos que correspondem a apenas uma maiuscula sao eliminados, especie de blacklist
13 #maiusculas apenas aparecem nesses casos especificos
14
15 text = re.sub (r"<i>", "", text)
16 text = re.sub (r"</i>", "", text)
17 termos = re.findall(r"<b>(.*)</b>", text) #recolha dos termos
```

Figura 24: Continuação da limpeza do XML e extração dos termos.

Nesta fase, foi criada uma nova variavel (*text2*), que não possui os termos (extraídos anteriormente). A ideia deste passo era facilitar a extração das descrições. Pelo mesmo motivo foram também removidos os parágrafos, bem como os espaços seguidos de vírgula, presentes após o *(pop)*.

Após isso foi feita uma marcação através da substituição de (*pop*) que nesta etapa estava presente no início e no fim das descrições por dois @ para facilitar a deteção e consequente extração das descrições.

```
20  text2 = re.sub(r"<b>(.*)</b>", "", text) #remocão dos termos na variavel text2
21  text2= re.sub(r"\n+", "", text2) #remoção dos paragrafos
22  text2 = re.sub(r"\s,", "", text2) #remover espaços seguidos de virgula, presente apos o (pop)
23
24
25
26  text2 = re.sub(r"\(pop\)", "@@", text2) #substituir o (pop) por @@ para marcação
27  #text2 = re.sub(r"@@  @@", "@@" , text2)
```

Figura 25: Continuação da limpeza do XML e colocação de marcação.

Nesta fase, foi criado um ficheiro *.text* para visualizar o conteúdo obtido até ao momento para verificar se a extração das designações foi executada sem qualquer falha.

```
30  resultado=open("glossario_termos_populares.txt", "w", encoding='UTF-8')
31  resultado.write(text2)
32  resultado.close()
```

Figura 26: *Script* para criar e atualizar ficheiro *glossario_ termos_ populares.txt*

```
a milionésima parte de um grama @@  à volta da boca @@  à volta da órbita @@  à volta dos vasos sanguíneos @@  abaixamento, abatimento, prostração @@  abcesso, tumor @@ abcesso, tumor @@
abcesso; acumulação de pus @@  barriga, ventre @@  ventral @@  anormal @@ abertura; orifício @@  ablação @@  ablação dos órgãos sexuais, capação, eviração, emasculação @@  abocamento @@
abortamento, desmancho @@  abortamento, desmancho @@  repentino, brusco @@  absorvimento, absorvência @@ absorção de água e de solutos por células vivas @@  absorvimento, absorvência @@
jejum @@ abstracção, suposição @@  acaríase sarcóptica, sarna @@  incapacidade em permanecer sentado @@ acção de certos corpos sobre outros @@  acção de enferrujar @@  acção de inalar,
extracção de líquidos ou gases @@  acção de urinar, urinação @@  aceleração da frequência cardíaca @@  acessório, anexo, indesejável @@  acidade, azedume @@  por acaso, sem importância @@
acidade, azedume @@ ácido aninado @@  alteração do equilíbrio ácido básico do sangue e líquidos teciduais @@  ausência de movimento, acinese @@  espinha @@  adaptação @@ acomodação @@
acompanhante @@  acostumado @@  acrescentar um solvente a um medicamento em pó; tipo de regeneração de uma superfícielesionada; retorno à forma líquida do soro ou plasma sanguíneo previamente
dessecado@@  cor azulada das extremidades (mãos-pés) @@  hormônio adreno-córticotrófico, corticotrofina @@ activação de uma droga por outra @@  activação de uma droga por outra @@  activado
```

Figura 27: Visualização do conteúdo do ficheiro *glossario_ termos_ populares.txt*

Nesta etapa, foi possível extrair todas as designações. Para isso foi necessário criar duas expressões regulares visto que a primeira não captava a totalidade das mesmas (a primeira neste caso específico pois nao tinha @ inicial). Todas as designações foram guardadas na variável *designacoes*.

De seguida, foi criada uma lista de listas, onde cada sublista contém um par de elementos, sendo o primeiro elemento *termos* e o segundo *designacoes*. Por outras palavras, cada termos foi associado à sua designação e o resultado guardado num dicionário

```
designacoes1 = re.findall(r"@(?:\s|,|X)(.*?)@", text2)  #encontra todas as que começam em
#@ seguido por um espaço em branco, vírgula ou "X", e terminam com "@".
# As partes entre "@" e "@" são capturadas pelo grupo (.*?) e retornadas como uma lista.
#(?: ) identifica o conteudo como necessario mas nao o captura
designacoes2= re.findall(r"(^.*?)@", text2) #encontra o que começa no início de uma linha e termina com @

designacoes= designacoes2 + designacoes1  #conjunto total de designacoes

lista_de_listas = [[x, y] for x, y in zip(termos, designacoes)]

dicti={}
dicti = dict(lista_de_listas)
```

Figura 28: Extração das designações e criação do dicionário.

Por fim, os resultados obtidos foram guardados num ficheiro json.

```
output = open ("json_termos_populares.json", "w", encoding="utf-8")
json.dump(dicti, output, ensure_ascii=False, indent=4)
output.close()
```

Figura 29: Guardar resultados no ficheiro *json_ termos_ populares.json*


```
{
    "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 ",
    "ressorção": " absorção de água e de solutos por células vivas ",
    "abstinência": "jejum ",
```


Figura 30: Visualização do conteúdo do ficheiro *json_ termos_ populares.json*.

3 Conclusão

Durante a execução deste trabalho prático, foi executado com êxito o processamento de uma variedade de documentos com diferentes formatos e estruturas de informação. Os documentos em PDF inicialmente, apresentaram desafios diversos, especialmente no que diz respeito à limpeza e extração de dados relevantes. Foram, por isso, utilizadas inúmeras combinações de expressões regulares e métodos de remoção de informações. Em certos casos, quando foram encontradas partes extensas de dados irrelevantes, foram realizadas limpezas manuais.

Apesar dos obstáculos encontrados, foram atingidos, na generalidade, os objetivos estabelecidos para o projeto. Os conhecimentos obtidos permitiram extrair informações valiosas dos documentos biomédicos fornecidos. Esta conquista abre portas para a sua aplicação em diversas áreas da Engenharia Biomédica, como análise de dados clínicos, detecção de padrões de doenças e criação de sistemas de diagnóstico assistido por computador.

Em suma, este projeto forneceu ferrramentas fundamentais para enfrentar desafios no campo da saúde, melhorando a capacidade de analisar informações de documentos médicos, otimizando a procura de dados para tomada de decisões.

4 Referências

[1] https://www.sas.com/pt_br/insights/analytics/processamento-de-linguagem-natural.html

[2] https://www.iberdrola.com/inovacao/pnl-processamento-linguagem-natural