Trabalho Prático de Grupo nº 1

Mestrado em Engenharia Biomédica – Informática Médica

Processamento de Linguagem Natural

1º 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 processamento da linguagem natural é o campo da computação (mais concretamente, da inteligência artificial) que tem a função de fornecer aos computadores a capacidade de entender a linguagem falada e escrita da mesma forma que os seres humanos. Atualmente, esta tecnologia atingiu um alto nível graças à aplicação de tecnologias como o machine learning, o big data, e a internet das coisas (IoT) ou as redes neurais. A comunicação diária levada a cabo pelas pessoas varia imenso, quer pelo seu contexto, pela linguagem não verbal, pelo tom de voz ou até quer pela frequência com que se repetem palavras ou expressões. E esta comunicação, desde o aparecimento da Internet, tem sido transportada a uma grande escala para o digital [1].

Algumas das aplicações mais importantes de PLN são:

Na inteligência de negócios (business intelligence), a possibilidade de analisar automaticamente as reações dos clientes através do que publicam na Internet ou das perguntas que são feitas para obterem informações. Os chatbots são outra aplicação que, embora ainda tenha uma grande margem de melhoria, agiliza a interação com os clientes via chats ou secretárias telefónicas, com respostas rápidas e automatizadas em função do processamento de linguagem natural.

A classificação grandes quantidades de documentos conforme seu assunto ou estilo pode ser agilizada. Também permitem encontrar padrões em textos e localizar coincidências entre eles, o que facilita a deteção de plágios e o controle de qualidade.

O tradutor do Google que deteta as nuances entre diferentes palavras conforme o contexto.é um exemplo de uma aplicação de PLN [2].

Uma expressão regular é uma sequência de texto que permite criar padrões que ajudam a corresponder e localizar texto. Estas são utilizadas em PLN para executar o processamento de texto para as mais diversas aplicações. Posto isto, o trabalho realizado tem por base o processamento de diversos textos em pdf, sobretudo dicionários, e a utilização de técnicas de PLN, mais concretamente expressões regulares com o intuito de retirar informação relevante e estruturada desses documentos. Toda a informação retirada dos documentos, ou seja termos portuguêses, as suas traduções e definições foram agrupadas num formato json.

Recolha de Dados

Foram fornecidos aos grupos de trabalho 7 *pdfs* para processar, sendo um de processamento obrigatório e os restantes de processamento opcional. O documento obrigatório apresenta traduções de termos médicos de português para inglês e espanhol. Existe também um dicionário de termos português-inglês, um de termos inglês-espanhol e um de termos com traduções em várias línguas. Existem ainda dois documentos com definições para vários termos médicos (em português) e um documento com expressões populares utilizadas para cada termo (em português).

Após análise aprofundada, foram selecionados 5 dos 6 documentos opcionais. Isto porque a grande maioria tinha traduções ou descrições para palavras portuguesas. Assim sendo, excluímos o dicionário com traduções para várias línguas, pois dificilmente teria a tradução de muitos termos que não estivessem já presentes nos outros 3 dicionários; e porque apesar de a tradução para várias línguas ser um elemento enriquecedor do *json* final, este ficaria com muitos termos apenas com tradução em inglês e/ou espanhol, por não ser encontrada uma correspondência no dicionário.

Assim, e de forma a simplificar a estrutura do ficheiro final decidiu-se obter um *json* com a palavra principal em português, tradução em inglês e espanhol, uma definição e um termo popular, caso existissem.

Arquitetura

De modo a obter-se o produto final desejado, neste caso um ficheiro *json* que contivesse toda a informação recolhida dos diversos documentos, cada documento foi tratado e a sua informação mais relevante foi transformada também num documento *json*.

Pretendia-se que o documento *json* final fosse um dicionário de dicionários, em que as *keys* do dicionário principal seriam as designações, os termos em português, e os *values* os dicionários secundários. Estes dicionários secundários contêm como *keys en, es, definição* e *expressão popular,* cujos *values* correspondem à tradução do termo em inglês, tradução do termo em espanhol, definição do termo, e designação popular do termo. Para tal, os ficheiros *json* de cada documento foram obtidos com o seguinte formato:

- Documento *dicionario_termos_medicos_pt_es_en.pdf*: dicionário de dicionários em que o dicionário principal tem como *keys* as designações em português e como *values* os dicionários secundários. Os dicionários secundários, por sua vez, contêm como *keys es* e *en*, ou seja, as traduções dos termos em espanhol e inglês, respetivamente;
- Documento *RU5HW615037.pdf*: dicionário cujas *keys* são os termos em português e os *values* são as traduções dos termos em inglês;

- Documento *CIH Bilingual Medical Glossary English-Spanish.pdf*: dicionário cujas *keys* são os termos em inglês e os *values* são as traduções dos termos em espanhol;
- Documento *Dicionario_de_termos_medicos_e_de_enfermagem.pdf*: dicionário cujas *keys* são os termos em português e os *values* são as suas definições/descrições;
- Documento *anatomia_geral.pdf*: dicionário cujas *keys* são os termos em português e os *values* são as suas definições;
- Documento *Glossario de Termos Medicos Tecnicos e Populares.pdf*: dicionário cujas *keys* são os termos em português e os *values* são as expressões regulares correspondentes às mesmas.

Para construir o ficheiro final, foi usado como base o dicionário obtido pelo Documento *dicionario_termos_medicos_pt_es_en.pdf* e depois foram adicionadas novas palavras ao mesmo e as suas traduções em inglês e em espanhol, assim como as definições e as expressões populares. Isto é, a partir do documento *json* base fez-se uma comparação de *keys* com o documento *json* obtido através do *RU5HW615037.pdf* e foram acrescentadas palavras não existentes no base. No que toca ao ficheiro *CIH Bilingual Medical Glossary English-Spanish.pdf* este serviu para completar o anterior de modo a adicionar a tradução em espanhol e também novas palavras ao dicionário final. Quanto às definições realizou-se novamente uma comparação entre as *keys* dos *json* já analisados e do *json* obtido pelo *Dicionario_de_termos_medicos_e_de_enfermagem.pdf* e foram acrescentadas as definições às palavras já existentes assim como foram adicionadas novas palavras e as suas definições. O mesmo foi feito para o *json* proveniente do *anatomia_geral.pdf*, no entanto, as palavras que já tinham tradução, esta não foi substituída por nova informação. Um processo semelhante foi ainda realizado para o *json* do *Glossario de Termos Medicos Tecnicos e Populares.pdf*.

Tratamento de Dados

Para o tratamento de dados, todos os ficheiros *pdf* selecionados foram convertidos em *xml* utilizando o comando *pdftohtml -c -xml* na *bash*. De seguida, todos os documentos no formato *xml* foram limpos, desenvolvendo um ficheiro python, isto é, retirou-se toda a sua informação desnecessária principalmente usando a função *re.sub* e, por fim, extraiu-se a informação pretendida, utilizando maioritariamente a função *re.findall*.

Documento *dicionario_termos_medicos_pt_es_en.pdf*

O documento obrigatório, *dicionario_termos_medicos_pt_es_en.pdf* , foi utilizado para extrair a informação principal relativa a traduções dos termos português-inglês-espanhol. Após este ser convertido em *xml* passou-se para o desenvolvimento de um ficheiro *python* no qual se fizeram os *imports* e se procedeu à abertura do ficheiro *xml*.

```
import re
import json

ficheiro = open("dicionario_termos_medicos_pt_es_en.xml", encoding="utf-8")
text = ficheiro.read()
ficheiro.close()
```

Figura 1. Imports e abertura do ficheiro xml

Uma vez que as primeiras páginas do dicionário correspondem à tradução de inglês para os outros dois idiomas, e de seguida de espanhol para os outros dois idiomas, mas o que mais se adequa ao objetivo do trabalho é a tradução de português para os outros dois idiomas, de forma a que a *key* do *json* seja a palavra que no *xml* se encontra a negrito (*bold*); o primeiro passo foi o de remover as primeiras 2 secções, já que os termos eram os mesmos. Apesar das tentativas de construir uma expressão regular que removesse todo o conteúdo do documento até essa secção, não foi conseguido desenvolver nenhuma, pelo que apenas se eliminou manualmente a primeira parte do documento, sobrando a tradução de português para inglês e espanhol. De modo a extrair a informação desnecessária do documento *xml*, prosseguiu-se à retirada de alguns elementos como *<page number="252" position="absolute" top="0" left="0" height="807" width="540">,* ou seja, os inícios de página, informações de fonte e a informação *xml* relativa a cada parágrafo (linhas 1,2 e 3). De seguida, foram verificados alguns casos particulares no documento referentes a valores em itálico não relevantes e frases no rodapé (linhas 4, 5 e 6). Ainda no que toca a informação presente nas páginas do documento que não é relevante, foi retirado o cabeçalho, que continha o número de página, e dependendo da página, o número antes ou depois (linha 7). De seguida, foi retirado qualquer caractere individual que se encontrasse a negrito, que correspondia às 24 ocorrências das letras do alfabeto que iniciavam uma nova secção. De modo a ter também em atenção palavras separadas por hífen e a translineação foram executada a linha 9 e 11 e ainda a linha 10 que retirou espaços a mais para facilitar a extração da informação importante. Assim sendo, o resultado deste processamento foi o livro em texto corrido (sem quebras de linha) com a *key* em *bold* e as traduções, sendo que a tradução inglesa era antecipada por um 'U' e a espanhola por um 'E'.

```
text = re.sub(r"</?page.*>", "", text)
text = re.sub(r"</?fontspec.*>", "", text)
text = re.sub(r"</?text.*?>", "", text)
text = re.sub(r"\n<i>.*</i>\n", " ", text)
text = re.sub(r"<b>português – inglês – espanhol</b>", " ", text)
text = re.sub(r"<b>português</b>\n<b>–</b>\n<b>inglês</b>\n<b>–</b>\n<b>espanhol</b>\n", "", text)
text = re.sub(r"(<b>(.*?)</b>\n)?<b>[0-9]+</b>(\n<b>(.*?)</b>)?", "", text)
text = re.sub(r"<b>[A-Z]</b>\n", "", text)
text = re.sub(r"\n+", " ", text)
text = re.sub(r"\s+", " ", text)
text = re.sub(r"- ", "", text)
```

Figura 2. Limpeza do documento xml, usando a função re.sub

Para retirar a informação relevante do texto processado em grupos de captura, foi utilizado o seguinte *findall*, sendo que o primeiro grupo de captura é o termo que se encontra em *bold*, a tradução em inglês é tudo o que se encontra entre U e E, e a tradução em espanhol é o restante antes do próximo *bold*. A informação obtida nestes três grupos de captura foi colocada numa lista de listas que junta cada designação às suas traduções correspondentes, em *nova_lista*.

```
lista = re.findall(r"b>(.*?)</b>\sU\s(.*?)\sE\s(.*?) <", text)
nova_lista = [[limpa(designacao), ingles, espanhol] for designacao, ingles, espanhol in lista]
```

Figura 3. Extração da informação, usando a funçaõ re.findall

No entanto, a designação precisa de um processamento adicional, uma vez que algumas designações apresentam símbolos de *bold* entre as palavras, pois eram palavras que se encontravam em linhas diferentes.

```
('vesícula</b> <b>biliar', 'gall bladder', 'vesícula biliar'),
```

```
def limpa(text):
    text = re.sub(r"</?b>","", text)
    return text
```

```
['vesícula biliar', 'gall bladder', 'vesícula biliar']
```

Figura 4. Processamento adicional

Para terminar a extração de informação relevante, a lista de listas foi convertida num dicionário de dicionários, com uma estrutura correspondente à já descrita (as *keys* são os termos, que englobam vários dicionários, cujas *keys* são *en* ou *es*, e os *values* as traduções para essas respetivas línguas), e este passou para um ficheiro *json*, tal como era pretendido.


```
{
    "à morte (bras.)": {
        "en": "mortally ill, dying",
        "es": "a la muerte"
    },
    "abaulamento": {
        "en": "swell",
        "es": "hinchazón"
    },
    "abcesso": {
        "en": "abscess",
        "es": "absceso"
```


Figura 5. Excerto do ficheiro json obtido

Documento *RU5HW615037.pdf*

O segundo documento, *RU5HW615037.pdf*, foi utilizado para extrair traduções adicionais português-inglês. Após este ser convertido em *xml* passou-se para o desenvolvimento de um ficheiro *python* no qual se fizeram os *imports* e se procedeu à abertura do ficheiro *xml*.

```
import re
import json

ficheiro = open("dicionario_termos_medicos_pt_es_en.xml", encoding="utf-8")
text = ficheiro.read()
ficheiro.close()
```

Figura 6. Imports e abertura do ficheiro xml

De modo a extrair a informação desnecessária do documento *xml*, prosseguiu-se à retirada de alguns elementos como *<page number="252" position="absolute" top="0" left="0" height="807" width="540">,* ou seja, os inícios de página, informações de fonte e a informação *xml* relativa a cada parágrafo (linhas 1,2 e 3). De seguida, foram verificados alguns casos particulares no documento referentes a valores em itálico não relevantes e frases no cabeçalho (linhas 4 e 5).

```
text = re.sub(r"</?page.*>", "", text)
text = re.sub(r"</?fontspec.*>", "", text)
text = re.sub(r"</?text.*?>", "", text)
text = re.sub(r"\n<i>.*</i>\n", "", text)
text = re.sub(r"<b>(.*?)</b>", "", text)
```

Figura 7. Remoção de informação, usando a função re.sub

Para retirar a informação relevante do texto processado em grupos de captura, foi utilizado o seguinte *findall*, sendo que o primeiro grupo de captura é o termo e o segundo a tradução. A informação obtida nestes dois grupos de captura foi colocada numa lista de listas que junta cada designação à sua tradução correspondente, em *nova_lista*. Todas as *keys* levaram um processamento adicional e foram passadas a minúscula, para coincidirem com as *keys* do documento principal quando for feita essa comparação.

```
lista = re.findall(r"(.*) - (.*)", text)
nova_lista = [[(pt.strip()).lower(), en] for en, pt in lista]

dicionario = dict(nova_lista)
```

Figura 8. Extração da informação relevante, usando a função re.findall

Para terminar a extração de informação relevante, a lista de listas foi convertida num dicionário com uma estrutura correspondente à já descrita (as *keys* são os termos, e os *values* as traduções para inglês), e este passou para um ficheiro *json*, tal como era pretendido.


```
{
    "cavidade abdominal": "Abdominal Cavity",
    "ecotomografia abdominal": "Abdominal Echotomography",
    "tuberculose abdominal": "Abdominal Tuberculosis",
    "ecografia abdominal": "Abdominal Ultrasonography",
    "dor abdominal": "Abdominalgia; Celialgia; Abdominal Pain",
    "aborto": "Abortion",
    "abcesso": "Abscess",
    "corpo cetónico": "Acetone Bodies",
    "acidose": "Acidosis",
    "imunidade adquirida": "Acquired Immunity",
```


Figura 9. Excerto do ficheiro json obtido

Documento *CIH Bilingual Medical Glossary English-Spanish.pdf*

À semelhança dos documentos abordados anteriormente, 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 no ficheiro *dicio_en_es.py.*

```
text = re.sub(r"</?pdf2xml.*?>", "", text)
text = re.sub(r"<!DOCTYPE.*?>", "", text)
text = re.sub(r"<\?xml.*?>", "", text)
text = re.sub(r"</?fontspec.*?>", "", text)
text = re.sub(r"</?page.*>", "", text)
text = re.sub(r">\d+\s</text>", "", text)
text = re.sub(r"<text top=\"1066\" left=\"802\" width=\"11\" height=\"16\" font=\"1\"", "", text)
text = re.sub(r"<text top=\"1066\" left=\"795\" width=\"19\" height=\"16\" font=\"1\"", "", text)
text = re.sub(r"</?text.*?>", "", text)
text = re.sub(r"<b>(.*)</b>", "", text)
text = re.sub(r"Copyright © 2004 by the Center for Immigrant Health", "", text)
text = re.sub(r"<i>Language Initiatives </i>", "", text)
text = re.sub(r"<i>Center for Immigrant Health \(New York University School of Medicine\) </i>", "", text)
text = re.sub(r"<i>\s</i>", "", text)
text = re.sub(r"<i></i>", "", text)
text = re.sub (r"<i>", "",text)
text = re.sub (r".</i>", "",text)
text = re.sub (r"</i>", "",text)
text = re.sub(r"^\s*$\n?", "", text, flags=re.MULTILINE)
```

Figura 10. Limpeza do texto.

Inicialmente, recorrendo-se à função *re.sub* foram eliminados alguns parâmetros prescindíveis, por exemplo campos relativos à configuração estética, informações do rodapé, cabeçalho, espaços, entre outros elementos (Figura 11). Após este tratamento do texto, foi criado um ficheiro no formato *.txt* cujas linhas ímpares correspondiam às designações em inglês e as linhas pares diziam respeito aos termos espanhóis. Assim, foram criadas duas listas: *linhas_impares* para os termos em inglês e *linhas_pares* para os termos em espanhol (Figura 11).

```
result_f=open("en_es.txt", "w", encoding='UTF-8')
result_f.write(text)
result_f.close()

with open('en_es.txt', 'r', encoding='UTF-8') as f:
    linhas = f.readlines()

#ingles:
linhas_impares = []

#Espanhol:
linhas_pares = []


for i, linha in enumerate(linhas):
    if i % 2 == 0:
        linhas_pares.append(limpa(linha.lower()))
    else:
        linhas_impares.append(limpa(linha.lower()))
```

Figura 11. Criação do dicionário dicio_en_es.json.

Finalmente, foi então originado um dicionário com as keys que termos médicos em inglês e os values a tradução espanhola (Figura X). Este dicionário encontra-se no ficheiro *dicio_en_es.json*.

```
lista_de_listas = [[x, y] for x, y in zip(linhas_pares, linhas_impares)]


dicionario={}
dicionario = dict(lista_de_listas)

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

Figura 12. Criação de um ficheiro texto. Criação de duas listas, uma para termos ingleses e outra para as traduções espanholas.

Documento *Dicionario_de_termos_medicos_e_de_enfermagem.pdf*

O documento utilizado para extrair a informação principal relativa às definições dos termos foi o documento *Dicionario_de_termos_medicos_e_de_enfermagem.pdf*. Após este ser convertido em *xml* passou-se para o desenvolvimento de um ficheiro *python* em que se fizeram os *imports* e se procedeu à abertura do ficheiro *xml*. Além disso, foi ainda retirada manualmente a parte introdutória do *pdf* tendo em conta que a mesma não era útil para a extração de informação. No ficheiro *python*

inicialmente foi acrescentada uma função limpa que pretende limpar algum conteúdo desnecessário no resultado final, como *\n*.

```
1  import re
2  import json
3
4  ficheiro = open("Dicionario_de_termos_medicos_e_de_enfermagem.xml", encoding="utf-8")
5  text = ficheiro.read()
6
7  def limpa(text):
8      text = re.sub(r"\s+", " ", text)
9      return text.strip()
```

Figura 13. Imports e abertura do documento xml

De modo a extrair a informação desnecessária do documento *xml*, começou-se por retirar alguns elementos automaticamente colocados pelo *xml* (linhas 11, 15 e 22), como *<page number="15" position="absolute" top="0" left="0" height="785" width="573">*. De seguida, foram verificados alguns casos particulares no documento referentes a números de página e letras do alfabeto presentes no início de algumas páginas, utilizando o valor *height* destes carateres do *xml* (linhas 12, 13 e 14), e ainda outros casos excecionais (linha 25). Ainda no que toca a informação presente nas páginas do documento que não é relevante, foi retirada a frase do rodapé (linha 18 e 21). Tendo em conta o objetivo da limpeza do documento, a formatação de itálico não tem importância pelo que também foram retirados os símbolos do *xml* referentes a esta formatação (linhas 16 e 17). De modo a ter também em atenção palavras separadas por hífen e a translineação foram executadas as linhas 19 e 20 e ainda as linhas 26 e 27 para a translineação de palavras a negrito. Para facilitar a extração da informação importante foram retirados espaços a mais e foi também colocado tudo em texto corrido (linhas 23 e 24).

```
11 text = re.sub (r"</?page.*>", "", text)
12 text = re.sub (r"<text top=\"[0-9]+\" left=\"[0-9]+\" width=\"[0-9]+\" height=\"11\" font=\"[0-9]+\">.*</text>", "", text)
13 text = re.sub (r"<text top=\"[0-9]+\" left=\"[0-9]+\" width=\"[0-9]+\" height=\"111\" font=\"[0-9]+\"><i>.*</i></text>", "", text)
14 text = re.sub (r"<text top=\"[0-9]+\" left=\"[0-9]+\" width=\"[0-9]+\" height=\"185\" font=\"[0-9]+\"><i>.*</i></text>", "", text)
15 text = re.sub (r"</?text.*?>", "", text) #*? para ele parar no primeiro > e não retirar info importante
16 text = re.sub (r"<i>", "", text)
17 text = re.sub (r"</i>", "", text)
18 text = re.sub (r"Sou Enfermagem - Cadastre-se grátis", "", text)
19 text = re.sub (r"\s-\s", "", text)
20 text = re.sub (r"-\s", "", text)
21 text = re.sub (r"<a href=\"https://souenfermagem.com.br\">em: https://souenfermagem.com.br</a>", "", text)
22 text = re.sub (r"<fontspec.*/>", "", text)
23 text = re.sub (r"\n\s", "\n", text)
24 text = re.sub (r"\n", "", text)
25 text = re.sub (r"ooooooooooooooooooooooooooooooo", "", text)
26 text = re.sub (r"-</b><b>", "", text)
27 text = re.sub (r"</b><b>", "", text)
```

Figura 14. Limpeza do xml usando a função re.sub

As informações essenciais a retirar deste documento *xml* foram as palavras em *bold*, os termos em português, e as suas definições, seguidas das palavras a *bold*. Para tal, foi utilizada uma expressão regular com dois grupos de captura em que o primeiro é relativo aos termos e o segundo às descrições.

A informação obtida nestes dois grupos de captura foi colocada numa lista de listas que junta cada designação e a sua descrição correspondente.

```
32   lista = re.findall(r"<b>(.*?)</b>([^<]*)", text)
33   lista = [[designacao.lower(), limpa(descricao)] for designacao, descricao in lista]
```

Figura 15. Extração da informação relevante, usando a função re.findall

Para terminar a extração de informação relevante, a lista de listas foi convertida num dicionário com uma estrutura correspondente á já descrita (as *keys* são os termos e os *values* as definições) e este passou para um ficheiro *json*, tal como era pretendido.

```
35   dicionario = dict(lista)
36
37   out = open ("definicoes.json", "w", encoding="utf-8")
38   json.dump(dicionario, out, ensure_ascii=False, indent=4)
39   out.close()
```

Figura 16. Conversão em dicionário e em documento json

Documento *anatomia_geral.pdf*

O documento *anatomia_geral.pdf* contém também definições e serviu para atribuir essa informação a palavras não presentes no outro documento de descrições. Uma vez mais, este foi convertido em *xml* e passou-se para o desenvolvimento de um ficheiro *python* em que se fizeram os *imports* e se procedeu à abertura do ficheiro *xml*. Inicialmente foi acrescentada novamente a função *limpa* que pretende limpar algum conteúdo desnecessário no resultado final, como *\n*.

```
1   import re
2   import json
3
4   ficheiro = open("anatomia_geral.xml", encoding="utf-8")
5   text = ficheiro.read()
6
7   def limpa(text):
8       text = re.sub(r"\s+", " ", text)
9       return text.strip()
```

Figura 17. Imports e abertura do ficheiro xml

De modo a extrair a informação desnecessária do documento *xml*, começou-se por retirar alguns elementos automaticamente colocados pelo *xml* (linhas 13, 14, 15 e 17), como *<page number="15" position="absolute" top="0" left="0" height="785" width="573">*. De seguida, foram eliminados os números de página e letras do alfabeto presentes junto de algumas palavras (linhas 16 e 18). Ainda no que toca a informação presente nas páginas do documento que não é relevante, foram retiradas

palavras-chave do documento que estavam no formato itálico e que não tinham definição (linha 19). A maior parte das palavras relevantes estavam em *bold*, à exceção de algumas que estavam em itálico então, para ser mais fácil fazer a sua extração, estas últimas foram todas passadas a bold (linhas 20 e 21). Para facilitar a extração da informação importante foi colocado tudo em texto corrido (linha 22). Quase a finalizar a limpeza do documento, teve-se atenção a uma situação particular referente ao facto de existirem palavras sem definição também em formatação bold, palavras estas que não se pretendia extrair. Então, colocou-se o texto de forma mais adequada a não incluir estas palavras na extração da informação (linha 23, 24 e 25) e apenas as que se pretendiam, tendo em conta que as palavras em *bold* sem definição encontravam-se imediatamente antes de uma outra palavra em *bold*.

```
13   text = re.sub (r"</?page.*>", "", text)
14   text = re.sub (r"</?text.*?>", "", text)
15   text = re.sub (r"<fontspec.*/>", "", text)
16   text = re.sub (r"[0-9]+", "", text)
17   text = re.sub (r"<image.*/>", "", text)
18   text = re.sub (r"\s[A-Z](<|,|\n)", "", text)
19   text = re.sub (r"<i><b>(.*)</b></i>", "", text)
20   text = re.sub (r"<i>", "<b>", text)
21   text = re.sub (r"</i>", "</b>", text)
22   text = re.sub (r"(\n)+", "", text)
23   text = re.sub (r"</b>(\s)+<b>", "</b><b>", text)
24   text = re.sub (r"</b><b>", "\n<b>", text)
25   text = re.sub (r"<b>", "@<b>", text)
```

Figura 18. Limpeza do documento xml, usando a função re.sub

As informações essenciais a retirar deste documento *xml* são, à semelhança do anterior, as palavras em *bold*, os termos em português, e as suas definições, seguidas das palavras a *bold*. Para tal, foi utilizada uma expressão regular com dois grupos de captura em que o primeiro é relativo aos termos e o segundo às descrições. Esta expressão regular permite diferenciar as palavras a *bold* com descrição e as que estão com a mesma formatação, mas não têm descrição. A informação obtida nestes dois grupos de captura foi colocada numa lista de listas que junta cada designação e a sua descrição correspondente.

```
29   lista = re.findall(r"<b>\s(.*?)\.</b>(.*?)\s+@", text)
30   lista = [[designacao.lower(), limpa(descricao)] for designacao, descricao in lista]
```

Figura 19. Extração da informação relevante, usando a função re.findall

Para terminar a extração de informação relevante, a lista de listas foi convertida num dicionário com uma estrutura correspondente á já descrita (as *keys* são os termos e os *values* as definições) e este passou para um ficheiro *json*, tal como era pretendido.

```
32  dicionario = dict(lista)
33
34  out = open ("definicoes2.json", "w", encoding="utf-8")
35  json.dump(dicionario, out, ensure_ascii=False, indent=4)
36  out.close()
```

Figura 20. Conversão da informação para formato de dicionário num ficheiro json

Documento *Glossario de Termos Medicos Tecnicos e Populares.pdf*

À semelhança dos documentos abordados anteriormente, 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 no ficheiro *pop.py.*

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 (Figura 21).*

```
text = re.sub(r"</?page.*>", "", text)
text = re.sub(r"</?text.*?>", "", text)
text = re.sub(r"</?fontspec.*?>", "", text)
text = re.sub (r"<i>", "",text)
text = re.sub (r"</i>", "",text)
```

Figura 21. Remoção de elementos próprios do xml.

A linha de código da Figura 22 foi implementada com o intuito de se retirarem letras maiúsculas a negrito que no ficheiro marcam a transição de secções.

```
text = re.sub(r"<b>[A-Z]</b>", "", text)
```

Figura 22. Remoção das letras maiúsculas a negrito.

Foi criada uma lista, com o auxílio da função *re.findall(),* para se reunirem todos os termos principais *(Figura 23).*

```
lista1 = re.findall(r"<b>(.*)</b>", text)
```

Figura 23. Captura das designações.

Após a captura das designações, estas foram removidas de modo a restarem apenas as expressões populares. Adicionalmente, retiraram-se as quebras de página, espaços e vírgulas desnecessárias (Figura 24).

```
text2 = re.sub(r"<b>(.*)</b>", "", text)
text2= re.sub(r"\n+", "", text2)
text2 = re.sub(r"\s,", "", text2)
```

Figura 24. Excerto do ficheiro pop.py.

Note-se que as expressões populares antecedem o elemento *(pop)*, assim, de modo a ser mais eficaz a captura destas expressões substitui-se este constituinte pelo símbolo "@@". Existiam algumas incoerências, daí a adição da segunda linha de código da Figura 25.

```
text2 = re.sub(r"\(pop\)", "@@", text2)
text2 = re.sub(r"@@  @@", "@@" , text2)
```

Figura 25. Substituição de "(pop)" por "@@" e resolução dos casos indesejados em que os símbolos apareciam seguidos e separados por espaço.

Após este tratamento foi possível a criação de uma expressão regular funcional e, tal como demonstra a Figura 26, foram capturadas as expressões populares. A lógica de captura é a extração de todos os carateres que se encontram entre os símbolos adicionados. Na *lista2*, encontram-se todos os campos pretendidos à exceção da primeira expressão popular, isto porque, esta expressão não é antecedida pelo carater "@". Neste sentido, a *lista3* inclui a primeira expressão e a *lista4* é a concatenação da *lista3* e *lista2*, reunindo toda a informação desejada.

```
lista2 = re.findall(r"@(?:\s|,|X)(.*?)@", text2)
lista3= re.findall(r"(^.*?)\s@", text2)
lista4= lista3 + lista2
```

Figura 26. Captura das expressões populares.

Reparou-se que algumas designações principais não tinham expressões populares associadas. Não encontrado qualquer padrão para o seu tratamento, foram eliminadas diretamente no código (Figura 27).

```
del lista1[239]
del lista1[245]
del lista1[381]
del lista1[395]
del lista1[561]
del lista1[2550]
```

Figura 27. Remoção de exceções.

Finalmente, foi criada a *lista_de_listas*, que une as designações e as respetivas expressões. Assim, foi possível dar origem a um dicionário em que as *keys* são os termos e os *values* as expressões. Este dicionário foi guardado num ficheiro *pop.json*.

```
lista_de_listas = [[x, y] for x, y in zip(lista1, lista4)]

dicionario={}
dicionario = dict(lista_de_listas)

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

Figura 28. Criação do dicionário e ficheiro pop.json.

Documento Final

Para a obtenção do dicionário final, foi implementado código no ficheiro *juntar_tudo.py (Figura 29).* Definiu-se que o ficheiro *principal.json* possuía o maior grau de importância. Deste modo, foi o dicionário desse ficheiro que foi utilizado como base, tendo esse as traduções dos termos em inglês e espanhol. Foi utilizado o ficheiro *dicio_pt_en.json* para serem adicionados novos termos e as respetivas traduções inglesas.

Como referido anteriormente, o ficheiro *dicio_en_es.json* tem um dicionário com *keys* que são termos ingleses e *values* que são designações em espanhol. Não possuindo termos portugueses, recorreu-se à tradução das palavras inglesas para português (Figura 30). Assim, criou-se um ficheiro *traduzir.py* (Figura 30) que dá origem a um novo dicionário em que as *keys* são portuguesas e os *values* espanhóis. Este dicionário encontra-se no ficheiro *traduzido.json.* Posto isto, os termos que se encontram no ficheiro *traduzido.json* e não constam no ficheiro *principal.json*, foram adicionados ao dicionário base, juntamente com a sua tradução em espanhol.

Definiu-se que o ficheiro *definicoes1.json* possui o dicionário de definições mais relevante. Assim, às *keys* comuns entre os dois dicionários, adicionou-se esta informação. Os termos que constam apenas no dicionário de definições, e a sua respetiva descrição, foram também acrescentados. Posteriormente, o mesmo raciocínio foi aplicado ao ficheiro *definicoes2.json e pop.json,* sendo que este último serviu para acrescentar as expressões regulares.

Finalmente, obteve-se o ficheiro *final.json* que possui o dicionário resultante, ou seja, o dicionário que agrega toda a informação extraída dos documentos (Figura 31).

```python
import json

with open('principal.json', 'r', encoding="UTF-8") as f:
    principal = json.load(f)


with open('dicio_pt_en.json', 'r', encoding="UTF-8") as f:
    dicio_pt_en = json.load(f)


for key in dicio_pt_en:
    if key not in principal:
        principal[key] = {'en': dicio_pt_en[key]}


with open('traduzido.json', 'r', encoding="UTF-8") as f:
    pt_es = json.load(f)

for key in pt_es:
    if key not in principal:
        principal[key] = {'es': pt_es[key]}


with open('definicoes1.json', 'r', encoding="UTF-8") as f:
    definicoes1 = json.load(f)

for key in definicoes1:
    if key not in principal:
        principal[key] = {'definição': definicoes1[key]}
    else:
        principal[key]['definição'] = definicoes1[key]
```

```python
with open('definicoes2.json', 'r', encoding="UTF-8") as f:
    definicoes2 = json.load(f)

for key in definicoes2:
    if key not in principal:
        principal[key] = {'definição': definicoes2[key]}



with open('pop.json', 'r', encoding="UTF-8") as f:
    pop = json.load(f)

for key in pop:
    if key not in principal:
        principal[key] = {'expressões popular': pop[key]}
    else:
        principal[key]['expressões popular'] = pop[key]


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

Figura 29. Ficheiro juntar_tudo.py.

```python
from deep_translator import GoogleTranslator
translator = GoogleTranslator(source='en', target='pt')

import json

file_in = open("dicio_en_es.json", encoding='UTF-8')
dici = json.load(file_in)

new_dici = {}

for designation, description in dici.items():
    en_translation = translator.translate(designation)
    print(en_translation)
    new_dici[en_translation]= description



with open("traduzido.json", "w", encoding='UTF-8') as fp:
    json.dump(new_dici, fp, ensure_ascii=False, indent=4)
```

Figura 30. Ficheiro traduzir.py.


```
},
"hemocultura": {
    "en": "hemoculture, blood culture",
    "es": "hemocultura",
    "definição": "Técnica destinada a evidenciar micróbios exist
},
"hemodiálise": {
    "en": "hemodialysis",
    "es": "hemodiálisis",
    "definição": "Procedimento utilizado em medicina nos casos d
    "expressões popular": " filtragem da circulação do sangue "
},
"hemodiluição": {
    "en": "hemodilution",
    "es": "hemodilución"
```


Figura 31. Excerto do dicionário final - ficheiro final.json.

Conclusão

Através da elaboração deste trabalho prático foi possível efetuar o tratamento de vários documentos, com diferentes formatações e organização de informação. Após o tratamento dos mesmos, a informação recolhida foi organizada num único documento de formato *json*. Os documentos inicialmente em *pdf* apresentaram diferentes graus de dificuldade no que toca à limpeza e extração da informação relevante pelo que tiveram de ser utilizadas diversas expressões regulares e métodos de remoção de informação. Face a essa dificuldade alguns processos de remoção tiveram de ser feitos manualmente principalmente na remoção de partes muito extensas de informação não relevante presente no documento *xml*. Apesar de estas situações terem sido mínimas e a grande

maioria do processamento dos documentos ter sido feita com base nos ficheiros *python*, esta é uma melhoria evidente a realizar.

De qualquer forma, pode dizer-se que os objetivos inicialmente propostos foram cumpridos e que, apesar das melhorias a realizar, o documento final inclui informação relevante e de forma organizada.

Referências

[1] Molina, J. (2018) O que é e qual a utilidade do processamento de linguagem natural?, Impacting Digital. Available at: https://impacting.digital/pt-pt/2018/11/processamento-de-linguagem-natural/ (Accessed: April 19, 2023).

[2] Iberdrola (2021) O que é o processamento de linguagem natural e quais são suas aplicações?, Iberdrola. Iberdrola. Available at: https://www.iberdrola.com/inovacao/pnl-processamento-linguagem-natural (Accessed: April 19, 2023).