



Mestrado em Engenharia Biomédica – Informática Médica  
1º Ano | 2022/2023 | 2º semestre

## **Processamento de linguagem Natural**

Trabalho Prático de Grupo

Ciarán McEvoy A87240  
Gonçalo Carvalho PG50392

## Introdução

Os métodos de PLN podem ser considerados alguns dos mais úteis no que toca a recolha de informação, ainda mais quando a quantidade desta se torna elevada. Não seria difícil retirar informação de uma página, porém retirar toda a informação relevante de centenas de páginas seria um trabalho moroso.

Assim sendo, quando se verifica esta condição torna-se útil recorrer a expressões regulares para trabalhar o texto que pretendemos extrair, recorrendo-se assim a uma combinação de remoções de texto que não tem relevância e retirar os termos que consideramos importantes recorrendo a padrões presentes no texto.

Com o presente trabalho, o objetivo será a remoção e preparação de informação de vários documentos diferentes para criar um documento nosso que contenha a combinação da informação relevante de todos estes documentos.

## Escolha e conversão de documentos

O primeiro passo quando se pretende tratar documentos de qualquer tipo para encontrar a informação relevante, é uma conversão dos ficheiros para um formato que seja mais adequado. Depois de efetuadas conversões tanto para html como xml através do comando *pdftohtml -xml*.

Os livros escolhidos para a nossa análise foram, além do essencial (dicionario\_termos\_medicos\_pt\_es\_en), os livros Dicionario\_de\_termos\_medicos\_e\_de\_enfermagem, CIH\_Bilingual\_Medical\_Glossary\_English\_Spanish e anatomia\_geral também foram escolhidos para análise e criação de um dicionário json.

Pode-se adiantar que a escolha dos documentos adicionais foi, de certo modo, aleatória, sendo que apenas se garantiu que pelo menos um dos outros documentos escolhidos iria conter definições e não traduções.

## Deteção de padrões e criação dos Jsons individuais: dicionario\_termos\_medicos\_pt\_es\_en

De seguida, e em todos os documentos, uma análise geral começou por ser feita, sendo que foi tentado localizar zonas específicas que pudessem ser problemáticas, assim como padrões. Nomeadamente, no caso do documento obrigatório, alguns padrões foram facilmente encontrados. Para começar, este dicionário continha três linguagens diferentes no formato termo, traduções. No entanto, o termo e as traduções rodavam, ou seja, numa secção uma linguagem continha a tradução enquanto nas restantes, este termo estaria representado como uma tradução.

Depois de se ter conseguido observar isto, foi decidido que seria benéfico usar apenas uma das secções em questão, sendo que a escolhida foi a última por conter os termos em português, e as traduções em espanhol e inglês. Esta foi a escolha uma vez que a presença de

outros dicionários que continham a informação em português faria com que a escolha deste termo principal nos ajudasse.

```
76209 <text top="104" left="59" width="28" height="50" font="15"><b>A</b></text>
76210 <text top="162" left="59" width="74" height="11" font="16"><b>à morte (bras.)</b></text>
```

Figura 1: Início da secção que contém os termos em português

Assim sendo, a linha em que esta secção começava foi delimitada (76209) e foi feita uma limpeza de algumas linhas do ficheiro. As linhas do ficheiro começaram a ser percorridas a partir desse ponto, e poderiam ir até ao fim do ficheiro uma vez que esta era a última secção do livro.

A primeira coisa a fazer de seguida foi perceber como iríamos seleccionar os termos. Rapidamente se reparou que estes estavam delimitados por *bold anchors* e assim sendo estas foram usadas como referência, assim como o facto de todos os termos que eram importantes estarem escritos na *font 16*. As *fonts 19 e 20* também são consideradas para ajudar nos casos em que os termos estão separados por serem duas palavras diferentes.

Além disto, adicionamos todos os valores que já foram visitados a uma lista para assegurar que não temos termos repetidos, ou, mais especificamente, que não temos duas entradas para um só termo (como por exemplo, no caso de “zona de inervação” poderíamos acabar com duas entradas, uma para “zona” e outra para “enervação”). Também se verificou se o método *re.search* tem algum conteúdo que possa ser utilizado, verificando se este não está a devolver um *None*, e também nos asseguramos que os valores das páginas, apesar de estarem a *bold* não são considerados, como verificado na figura 2.

```
if re.search(r'font="(16|19|20)"[^>]*>(.*)</text>', ficheiro1[i]):
    list1.append(i)
    search = re.search(r'font="(16)"[^>]*>(.*)</text>', ficheiro1[i])
    if search is not None:
        search = search.group(0)
        #somar todos os valores de search desde que estes não sejam none ou ap
        if re.findall(r"<b>\d+</b>", search):
            ficheiro1[i] = re.sub(r"<b>\d+</b>", "", ficheiro1[i])
            continue
        else:
            search = re.sub(r".*(16|20).*<b>(.*)</b>.*", r"\2", str(search))
            match = match + " " + search
```

Figura 2: Código das expressões utilizadas e que permite a concatenação de termos

Numa segunda fase, verificamos se as nossas traduções estão presentes, ou seja, se estão em *bold* ou *itálico*, como verificado na figura 3.

```
if re.search(r'^(?!.*<(b|i)>).+<', ficheiro1[i]):
```

Figura 3: Expressão regular que encontra as traduções

Se não estiverem, então quer dizer que são as nossas traduções. Caso isto aconteça vamos adicionar o valor da linha a uma lista para assegurar que não repetimos estes valores, e adicionamos os valores retornados a uma lista. Esta iteração é realizada para se conseguir obter toda a informação relevante da linha até a condição referida anteriormente se revelar.

Por fim, percorremos a lista e verificamos se a nossa lista está vazia. Se não estiver, procedemos a unir as linhas que anteriormente encontramos e por fim, unimos o nosso termo às traduções que tínhamos encontrado.

Por fim, para evitar perdas de informação tiveram de ser realizadas duas mudanças manuais no documento. A primeira passou pelos bloqueadores beta, e consistiu da eliminação de uma linha isolada com a letra  $\beta$  e inserir a mesma na linha anterior que continha o *bold*. A outra, foi semelhante e envolveu a eliminação de uma linha que continha apenas o fecho de parênteses e a sua colocação na linha anterior que continha o *bold*.

Por fim, foi verificado se as traduções continham “U” ou “E”, sendo estes passados para “en” ou “es”, e removidos das traduções, passando a ajudar a identificar a linguagem das traduções, e facilitando o trabalho futuro.

Depois de efetuadas todas estas transformações, os pares termo/tradução são adicionados ao um dicionário que por último é escrito num ficheiro em formato json.

#### Deteção de padrões e criação dos Jsons individuais: Dicionario\_de\_termos\_medicos\_e\_de\_enfermagem

Já no documento `Dicionario_de_termos_medicos_e_de_enfermagem`, foi adotada uma estratégia diferente. Primeiro, foi criado um programa denominado *enfermage\_setup* para se proceder à remoção de grande parte da informação não relevante, através dos valores das *fonts* uma vez que grande parte da informação irrelevante poderia ser facilmente detectada assim, como visto na figura 4. Depois de feita esta limpeza foi escrito no ficheiro `Dicionario_de_termos_medicos_e_de_enfermagem_setup.xml` que foi usado de seguida.

```
ficheiro1 = re.sub(r'<b>(.)</b>', r'\1', ficheiro1)
ficheiro1 = re.sub(r'<i>(.)</i>', r'\1', ficheiro1)
ficheiro1 = re.sub(r'.+font="20".+', '', ficheiro1)
ficheiro1 = re.sub(r'.+font="19".+', '', ficheiro1)
ficheiro1 = re.sub(r'.+font="6".+', '', ficheiro1)
ficheiro1 = re.sub(r'.+font="0".+', '', ficheiro1)
ficheiro1 = re.sub(r'.+font="27".+', '', ficheiro1)
ficheiro1 = re.sub(r'<page.+>', '', ficheiro1)
ficheiro1 = re.sub(r'</page>', '', ficheiro1)
ficheiro1 = re.sub(r'\n+', r'\n', ficheiro1)
```

Figura 4: Remoção de caracteres não importantes

De uma forma geral, a segunda etapa para remover a informação relevante foi idêntica à do documento anterior, salvo pequenas variações como o uso de valor de *fonts* diferentes, entre outras pequenas mudanças, mas o funcionamento geral do programa mantém-se. Uma das coisas a referir será que em vez de se utilizar as *anchors* do *bold* foram apenas utilizados os valores das *fonts* uma vez que alguns elementos a *bold* eram apenas para chamar a atenção do leitor e não eram os termos que procurávamos.

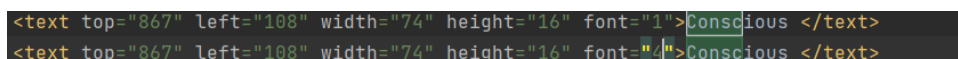
As expressões regulares utilizadas são bastante semelhantes às da Figura 2, contendo apenas variações menores.

### Deteção de padrões e criação dos Jsons individuais: CIH

No documento `CIH_Bilingual_Medical_Glossary_English_Spanish`, começou por se proceder, tal como no anterior, a uma limpeza do ficheiro. O primeiro passo passou pela remoção de todas as linhas que não contivessem uma *font* igual a 4 ou a 5, uma vez que eram estes termos que continham a informação relevante.

De seguida procedeu-se à remoção de todas as linhas que não contivessem informação dentro dos símbolos “>” e “<”, assim como as linhas que não contém informação dentro de marcas para *bold*.

Na fase seguinte, percorre-se o ficheiro e retira-se todas as linhas e parágrafos que não contenham informação relevante. Também foi nesta fase que alguns casos específicos foram observados e corrigidos, sendo por vezes necessária a remoção de pequenas porções de informação (nomeadamente a remoção de uma sigla num caso, mantendo a nomenclatura completa), e a mudança de *fonts* em alguns casos, acabando por se escrever a informação restante neste ponto para outro ficheiro, como verificado na Figura 5 abaixo.



```
<text top="867" left="108" width="74" height="16" font="1">Conscious </text>
<text top="867" left="108" width="74" height="16" font="4">Conscious </text>
```


Figura 5: Alteração manual de valores no XML

Depois disto ser feito, prossegue-se a percorrer este novo ficheiro, e limpá-lo de marcas de *bold* e espaços adicionais.

De seguida, procede-se a retirar quaisquer parágrafos adicionais que ainda sobrem, e a conectar as palavras à sua tradução correta iterativamente, tendo em conta alguns casos particulares para assegurar o bom funcionamento do loop, sendo estes, a verificação de possíveis frases que pertencem ao elemento anterior ou seguinte.

No `match1` é observado o primeiro caractere ou primeira palavra da linha seguinte à que está a ser iterada por possíveis caracteres que a determinem como fazendo parte da linha anterior. Uma primeira letra minúscula no início da frase é um identificador muito comum.

No `match2`, é verificado o elemento anterior ao que está a ser iterado. Neste caso se a linha anterior terminar com um (:) quer dizer que pertence à linha seguinte.



```
match = re.search(r"^(?![a-z])([a-z])^(Humana)|(Salud)|(Paciente)|(Estreptococcica)|(Adquirida)|(Papanicolaou)", lista[k+count])
match2 = re.search(r"^(^.*\s$", lista[k + count - 1])
```

Figura 6: Casos particulares a considerar na expressão regular

Por fim, antes de se escrever no ficheiro JSON adiciona-se ao dicionário um dos termos que escapou ao ciclo criado, assegurando assim a presença de todos os termos relevantes e das suas respectivas traduções.

### Deteção de padrões e criação dos Jsons individuais: Anatomia geral

No caso do último documento analisado, começou-se por excluir todas as linhas que continham *fonts* que não tinham informação relevante, procedendo-se de seguida a seleccionar todos os termos que estavam em *bold* ou itálico.

De seguida, transformamos a lista de tuplos obtidos anteriormente numa lista normal, procedendo de seguida a eliminar as marcas de *bold* e itálico nos termos guardados na nossa lista.

Depois disto, retiramos os espaços e pontos(.) a mais dos termos. Uma vez aberto este ficheiro, vai ser necessário verificar se os elementos da nossa lista acabam com “-”, ou seja, se as palavras ocupam mais que uma linha. Caso isto se verifique, o hífen será eliminado, e este elemento será junto ao próximo elemento a ser iterado.

Por fim, vamos iterar por todas as linhas no nosso *xml*, ficando com as descrições que pretendemos e filtrando caracteres desnecessários, guardando o resultado final num outro ficheiro.

Este ficheiro é aberto de seguida usando-o como ponto de partida e procedendo a remover ou substituir os caracteres indesejados nesta fase, entrando agora na fase final para a criação deste ficheiro JSON.

```
re.sub('f', 'fi', i)
re.sub("fl", "fl", c)
```

Figura 7 - Exemplo de termos

Um loop vai percorrer todos os termos, frase a frase. Se um termo relevante for encontrado, vai procurar a sua definição nas próximas linhas, juntando termo à sua descrição, e tendo como condição de paragem encontrar um número. Na execução do loop reparou-se que existam vários termos problemáticos como por exemplo os que estão a ser substituídos na figura 7.

Será de referir, por fim, que uma porção dos termos contidos no ficheiro Anatomia\_Geral, não continham uma designação correspondente, sendo que nesses casos o termo principal foi associado a uma designação vazia.

### Criação do ficheiro Json final

A última fase na criação do nosso ficheiro JSON era necessário iterar pelos diversos ficheiros JSON criados até agora, e reunir toda a informação relevante num só.

Para isto, foram abertos os quatro JSON e foi atribuído a um dicionário um destes ficheiros.

De seguida foi criado um ciclo que verificava se o termo chave de cada um dos JSONs coincide com o termo chave do dicionário inicial. Se isto se verificar, adiciona-se a descrição que está presente no nosso JSON ao nosso dicionário.

Caso contrário, o ficheiro vai adicionar tanto o termo como uma entrada (podendo ser ou uma tradução ou uma descrição dependendo do JSON que está a ser utilizado).

Um exemplo deste código está presente na Figura 8.

```
new_dict = dtm_obj

for termo in dtme_obj:
    desc = dtme_obj[termo]
    termo = termo.lower()
    if termo in new_dict:
        new_dict[termo]["dtme_desc"] = [desc]
    if termo not in dtm_obj:
        new_dict[termo] = {"dtme_desc": [desc]}
```

Figura 8 - Exemplo de um bloco da união de JSONs

### Criação do HTML

Por último, foi realizada a criação de um servidor com recurso ao *Flask*, que nos permitisse pesquisar os termos que pretendemos com recurso a este. Esta aplicação vai permitir a utilização dos termos chave do ficheiro JSON final, e com recurso a estes será então possível procurar na nossa página pelos termos que nos sejam de interesse.

Em cada página HTML estão presentes campos que permitem a colocação de informação de qualquer um dos ficheiros, devidamente identificados, para nos podermos assegurar da sua correta classificação, como é visível na Figura 9.

## Designation: cisto pilonídeo (o quisto pilonidal)

Description:

Description DMTE:

Description DTME:

English: piliferous cyst, pilonidal cyst

Spanish: quiste pilonidal

Description AG:

Spanish CIH:

English CIH:

Figura 9 - Exemplo de uma página HTML

## Conclusão

O trabalho desenvolvido permitiu não só a consolidação de conhecimentos, assim como a criação de um ficheiro robusto contendo diversos tipos de informação diferentes, desde traduções, o termo que queríamos encontrar e a sua respectiva definição.

No entanto, existem alguns pontos que poderiam ser melhorados tendo aquilo que foi feito em conta.

Primeiro, poder-se-ia ter utilizado a biblioteca *deep-translator* para assegurar que todos os termos, assim como as suas descrições, estavam presentes em todas as linguagens. No entanto, devido ao facto que a utilização desta biblioteca é extremamente demorada, optou-se por ficar apenas com os termos incluídos nos ficheiros JSON.

Além disto, também se pode verificar que em alguns dos ficheiros, o código escrito já se revela algo lento, podendo ser útil a reestruturação ou até a reescrita de alguns dos algoritmos apresentados.

No caso do HTML, poder-se-ia melhorá-lo, assegurando-nos de que os campos vazios não são mostrados.

Apesar de tudo isto, é de referir que apesar de existirem pontos para melhoria, de uma forma geral, o programa escrito permitiu a criação de um ficheiro que contém uma quantidade de informação robusta, tanto de linguagens, como de termos e descrições, sendo que as melhorias referidas apenas serviriam para aumentar a quantidade de informação no ficheiro, assim como a eficiência dos programas.