Processamento de Linguagem Natural

Ana Costa, pg53062, Inês Mendes, pg53875, Luís Cunha, pg54020

7 de abril de 2024

Docentes: Luís Filipe Cunha e José João Almeida.

Resumo

Este relatório detalha o desenvolvimento de *scripts Python* com a função de extrair informação de documentos *PDF* no contexto do Processamento de Linguagem Natural em Engenharia Biomédica. Os documentos *PDF* processados neste trabalho incluem: Glossário de Termos Médicos Técnicos e Populares, Glossário Ministério da Saúde e Minidicionário de Cardiologista. O objetivo principal foi extrair informações relevantes desses documentos e armazená-las em formato *JSON* para análises e projetos futuros. O processo envolveu a conversão dos *PDFs* para *XML*, seguida pela limpeza e extração de informações usando expressões regulares em *Python*.

1 Introdução

O Trabalho Prático 1 para a unidade curricular de Processamento de Linguagem Natural em Engenharia Biomédica envolveu a aplicação dos conhecimentos adquiridos para processar documentos *PDF*. Para isto, foi disponibilizada uma lista de documentos pelo docente dos quais cada grupo teria que processar três, sendo que um era de caráter obrigatório. O principal objetivo é extrair a informação significativa desses documentos e armazená-la em formato *JSON* para análises e projetos futuros. Para isso, foram desenvolvidos quatro *scripts Python* capazes de lidar com a diversidade de formatação encontrada nos ficheiros *PDF*, garantindo precisão e eficácia na extração das informações relevantes.

O desenvolvimento destes *scripts* foi essencial para o alcance dos objetivos do trabalho, uma vez que são responsáveis por identificar e extrair informação específica dos documentos. A informação extraída foi, como anteriormente aludido, armazenada em formato *JSON*, proporcionando flexibilidade e facilidade de acesso para análises e projetos futuros.

Neste relatório está documentado o processo de desenvolvimento dos *scripts* e as estratégias utilizadas para a extração de informações dos diferentes ficheiros *PDF*. Cada PDF foi abordado de forma individual, existindo, por isso, diferenças a nível dos desafios enfrentados e das estratégias de processamento utilizadas no decorrer do processo de extração.

2 Processo e estrátegias utilizadas

Neste trabalho, foram processados os seguintes três *PDFs*: Glossário de Termos Médicos Técnicos e Populares, Glossário Ministério da Saúde e Minidicionário de Cardiologista. De modo a dar início ao processamento, foi analisada a estrutura dos documentos e identificada a informação relevante a extrair. Para possibilitar a extração da informação, os documentos PDF foram convertidos para o formato *XML* (usando o comando 'pdftohtml -xml' na *bash*). Por fim, foram criados quatro *scripts Python*, sendo que dois deles representam duas abordagens distintas ao mesmo documento.

No decorrer dos quatro *scripts*, com fim ao processamento dos documentos, as ferramentas que mais auxiliaram neste processo foram as expressões regulares (*RegEx*), através do módulo *re* do *Python*.

2.1 Glossário de Termos Médicos Técnicos e Populares

O Glossário de Termos Médicos Técnicos e Populares consiste numa compilação de vários termos médicos acompanhados dos seus termos populares correspondentes.

Para este ficheiro, foram desenvolvidas duas abordagens que serão exploradas nos seguintes pontos.

2.1.1 Primeira Abordagem

Após uma análise detalhada do *PDF*, verificou-se que este começa com um título, a fonte e informações complementares seguido do glossário, ordenado alfabeticamente. Observou-se que os termos no glossário estão destacados a negrito e a sua posição pode variar entre o início e o fim da frase. As descrições correspondentes aos termos são apresentadas na mesma linha, em itálico e com adição de '(pop)'. Se um termo inicia a frase, a sua descrição surge imediatamente a seguir. No entanto, se o termo aparecer no final da frase, a descrição é apresentada antes dele.

Inicialmente, foi necessário fazer uma limpeza manual eliminando as linhas correspondentes ao início do documento que não pertencem ao glossário, os termos que não possuiam expressão popular associada e corrigindo as expressões regulares que ficavam separadas pela quebra de página. Para além disto, foi verificado que, na conversão para formato *XML*, quando ocorriam dois termos a bold consecutivos, entre eles existia a seguinte linha: "<text top="773"left="269"width="5"height="18"font="2"></text>". Porém, em alguns casos, como por exemplo, entre os termos "polidipsia"e "sedentário", isto não acontecia, o que também foi corrigido.

De seguida, utilizado a função *re.sub()* foram eliminadas todas as linhas que possuíssem "font="4 (linha que indicava o início de uma nova letra no glossário), linhas de formatação das páginas e do texto, linhas que possuíssem "fontspec"e outros caracteres indesejados. De seguida, de maneira a juntar expressões que fossem multilinha foram retirados as quebras de linha antes de ""e antes de "". Assim, no caso das expressões multilinha, estas ficariam numa linha só consecutivamente, o que se resolveu substituindo as expressões ""e ""por espaços. Por fim, retirou-se também as linhas em branco.

Após a limpeza do texto, utilizamos novamente expressões regulares para extrair os termos e suas respectivas expressões populares. Os termos são identificados pelas *tags* ...e as descrições são identificadas pelas *tags* .... Assim sendo, foi usado o comando *re.findall()* e grupos de captura para extrair o conteúdo delimitado pelas *tags* e armazená-lo em listas separadas.

Em seguida, foi criado um ciclo *while* que percorre as duas listas de modo a criar o dicionário em que as chaves são os termos e os valores as expressões populares respetivas. A partir deste foi criado o ficheiro "glossário.json".

2.1.2 Segunda Abordagem

Nesta segunda abordagem ao processamento do *PDF* 'Glossário de Termos Médicos Técnicos e Populares', foi numa fase inicial removido o texto do *XML* considerado irrelevante. Após o referido, foi feita a recorrência ao método *split* (através do caracter '\n') separando o documento *XML* numa lista em que cada elemento corresponde a uma linha do mesmo. Depois da lista ter sido criada esta é 'limpa', ou seja, são eliminados os elementos da lista que são insignificantes e que podiam pôr obstáculos relativamente ao processamento. Tal foi feito substituindo todos esses elementos por um carácter homogéneo a todos (neste caso '$'). No decorrer do mesmo ciclo os elementos contendo a expressão '(pop)' foram substituídos por 'SEPARADOR', de forma a posteriormente orientar o processamento.

Seguiu-se a criação de um ciclo do tipo *while* (tendo como condição de paragem a ultrapassagem do índice máximo da lista de linhas do texto). O ciclo foi concebido para, a cada iteração, verificar se as linhas atual e seguintes (até um máximo de 4 linhas) correspondem a um dado padrão (onde 'SEPARADOR' é apenas encontrado aquando do fim de uma expressão em itálico e o termo a negrito antes desta ou depois de 'SEPARADOR'; também são contemplados termos a negrito e expressões em itálico multilinha), criando uma entrada no dicionário a partir da informação recolhida e incrementando ao índice uma quantidade igual ao número de elementos que constam do padrão. Anteriormente à criação do ficheiro *JSON* e depois da criação do primeiro dicionário, é criado um novo dicionário organizado alfabeticamente com o conteúdo do primeiro. O referido foi atingido através do método *sorted* e de um ciclo *for*.

2.2 Glossário do Ministério Saúde

O Glossário do Ministério da Saúde abarca termos e conceitos utilizados para descrever e explicar os procedimentos, regulamentos e normas adotadas pelo Ministério da Saúde e outras entidades relacionadas à área da saúde pública. Além disso, incorpora uma variedade de siglas, abreviações de

expressões que representam órgãos, entidades, programas, projetos ou conceitos específicos ligados à área da saúde pública e à administração governamental.

O objetivo deste glossário é extrair as siglas e os termos. Para isso, dividimos o texto em duas partes: a primeira parte, "siglas", para extrair as siglas, e a segunda parte, "dicionário", que contém o restante do texto, para extrair os termos. Esta divisão foi feita colocando uma linha contendo a sequência de caracteres '$$$' a dividir as duas partes e utilizando o método 'split' através da sequência anteriormente mencionada para fazer a separação. As partes consideradas desnecessárias, especificamente, todo o conteúdo anterior às siglas e posterior ao dicionário, foi manualmente removido.

Para extrair as siglas, iniciou-se por limpar o conteúdo desnecessário com a ajuda de expressões regulares ('*re.sub*'), e utilizamos *re.findall* para procurar e extrai-las, assim como as suas descrições. Após a extração, foi empregado um *loop*, do tipo *for*, que percorre os termos e as descrições, associando a descrição ao termo e adicionando ao dicionário designado, posteriormente usado na criação do ficheiro 'siglas.json'. Para extrair os termos, assim como todo o conteúdo descritivo associado a cada um, também foi limpo o conteúdo desnecessário com o auxílio do método *re.sub*. O texto do dicionário foi dividido numa lista de *strings*, onde cada *string* representa uma linha do ficheiro *XML*, utilizando o método *split* (através do caracter '\n'). Após este *split* foi criado um ciclo *for* que itera através dos elementos da lista previamente referida, sendo uma variável correspondente a um índice criada, antes deste, de forma a garantir acesso aos elementos anteriores e posteriores da lista dentro do ciclo. Dentro do ciclo criado é possível ver duas condições principais, que as entradas da lista deverão respeitar de modo a 'entrar' numa delas. A primeira condição 'captura' todos os termos a negrito e tenta verificar se esse mesmo termo se encontra ou não isolado, ou seja, se os elementos anterior e posteriores a ele não são também termos a negrito. Após certificado que o termo seguinte já não é um termo a negrito a concatenação dos anteriores termos consecutivos é adicionada a uma lista de termos do nosso dicionário. É necessário acrescentar que também é verificado se o elemento após o fim do termo é uma linha contendo a expressão 'categoria' a itálico. Se o *output* for positivo então assinala aquele termo como contendo categoria (com_categoria), caso contrário assinala como não contendo categoria (sem_categoria). Desta forma, cada elemento da lista de termos será uma lista de dois elementos onde o elemento correspondente ao índice 0 é o termo e o elemento correspondente ao índice 1 a existência, ou não, de categoria. A segunda condição 'retém' qualquer termo que não se encontra a negrito e não corresponde a uma linha contendo a expressão 'categoria' em itálico. Desta forma 'apanha' todas as categorias e descrições de termos. Tendo isto em conta, este diferencia uma categoria de uma descrição através da *width* da linha *XML*. Sabendo que uma categoria nunca ultrapassa uma *width* de 223 e começa sempre após a expressão 'categoria' a itálico, conseguimos encontrar todas as categorias, que são concatenadas no caso de serem multi-linha e adicionadas a uma lista de categorias, homologamente à lista de termos. Para os casos em que o início de uma descrição é muito curta (inferior a uma *width* de 223) o código também foi pensado para lidar com o referido obstáculo, verificando se o elemento 'atual' tem uma *width* inferior a 218 (ou seja, que não atinge o limite de uma linha de categoria e que, por isso, indica o fim da atual categoria). As restantes linhas, por exclusão de partes, serão referentes a descrições. Dessa forma elas são concatenadas até um termo a negrito ser identificado como sendo o seguinte elemento da lista (indicando que a descrição acabou e começou um novo termo) e, por fim, adicionadas a uma lista de descrições.

A vantagem desta estratégia de processamento é o facto das listas estarem organizadas, querendo isto dizer que ao elemento de índice 0 da lista de termos vai corresponder o elemento de índice 0 da lista de descrições. Tal funciona ligeiramente diferente para as categorias, visto que nem todos os termos são dotados de uma. Assim, antes do ciclo *for* concebido para a criação do dicionário *Python*, que contém nele o dicionário presente no *PDF*, e que foi guardado no ficheiro 'dicionario.json', são iniciadas variáveis correspondentes ao índice da lista de categorias e da lista de descrições. Em cada iteração é verificado se o termo 'atual' tem uma categoria ou não. Caso positivo é criada uma entrada no dicionário *Python* final cuja chave é o termo e cujo valor é um novo dicionário *Python* detentor de duas chaves, 'categoria', associada à categoria e 'descricao', associada à descrição. Em caso negativo uma entrada semelhante é criada, porém, como valor da chave 'categoria' é dada a *string* 'N/A'.

2.3 Minidicionário de Cardiologista do Autor Ricardo Silveira Mello

O Minidicionário de Cardiologista corresponde a um dicionário dos termos e expressões mais frequentes empregados na cardiologia mundial, com as peculiaridades dessa especialidade médica.[**?**] Ao analisar o documento, constatou-se que a primeira metade consistia na listagem dos termos em inglês acom-

panhados das respetivas traduções/descrições em português, enquanto a segunda metade apresentava a listagem dos termos em português e suas traduções/descrições em inglês. Além disto, todos os termos estavam destacados a negrito, uma característica que posteriormente facilitou a extração das informações.

Inicialmente, foi necessário fazer um processamento do documento *XML* de modo a eliminar as informações não relevantes. Assim, foram manualmente removidas as linhas correspondentes à parte inicial do documento, que incluíam, por exemplo, a capa, a apresentação e os agradecimentos e também todas as que possuíssem a expressão 'fontspec'. A zona de separação entre as duas secções do Minidicionário foi substuída por um '@'.

De seguida, foram usadas diversas expressões regulares para limpar o texto e torná-lo mais uniforme. Usando a função *re.sub()* foram removidas todas as linhas cuja *font* fosse igual a 5, 6, 9, 12 e 15, todas as marcações de imagens, as informações de formatação sobre as páginas, incluindo cabeçalhos e números de página, todas as linhas em branco e também caracteres especiais indesejados.

Depois disto, foi usada a função *re.split()* pelo caracter '@' de modo a dividir o conteúdo do documento nas suas duas secções. Cada uma destas, foi processada de modo a tornar o texto coeso e simples para a extração das informações. Foi novamente usada a função *re.sub()* para remover todas as *tags* '<text>' e os seus atributos, todas as *tags* '</text>', substituir duas ou mais quebras de linha consecutivas por apenas uma quebra de linha, espaços em branco no início de cada linha do texto e juntar sequências de texto em negrito (...) que estavam separadas por uma quebra de linha (ou seja, termos multi-linha). Por fim, foram destacados todos os termos de modo que cada conjunto termo/descrição fosse apresentado da seguinte forma: termos entre '@', quebra de linha, tradução, duas quebras de linha.

As listas correspondentes a cada secção do documento foram usadas para gerar dois novos arquivos *XML* denominados 'EN-PT.xml' e 'PT-EN.xml', respetivamente. Em seguida, foi usado o comando *re.findall()* em conjunto com a expressão '@(.+)@\n(.*)' para extrair os termos e as suas descrições. Além disso, utilizou-se o comando *dict* para criar dicionários nos quais os termos atuam como chaves e as descrições como valores correspondentes. A partir destes, foram criados os documentos 'EN_PT.json' e 'PT_EN.json'.

3 Conclusões

O objetivo deste trabalho era extrair informações relevantes dos documentos e armazená-las em formato *JSON* para análise e projetos futuros. Para isso, processamos três documentos, o Glossário de Termos Médicos Técnicos e Populares, o Glossário do Ministério da Saúde e o Minidicionário de Cardiologista. O processamento envolveu o desenvolvimento *scripts* em *Python* para extrair informações dos documentos *PDF*. A conversão dos *PDFs* para *XML*, foi fundamental para as manipulação do texto e a utilização de expressões regulares em *Python* foi importante para a extração da informação pretendida.

Cada documento *PDF* , tinha as suas próprias características e desafios. Ao abordar cada documento, foram identificados padrões que distinguem as diferentes partes dos dicionários concebidos e foram adaptadas estratégias de processamento para lidar com os desafios de cada um. Isso demonstra a flexibilidade e adaptabilidade dos métodos desenvolvidos.

Os scripts desenvolvidos foram capazes extrair as informações desejadas de forma precisa e eficiente, contribuindo para futuras análises e projetos nesta área.

Em última análise, o sucesso na extração e organização dessas informações essenciais não apenas demonstra a eficácia dos métodos utilizados, mas também abre novas oportunidades para pesquisas e desenvolvimentos futuros.