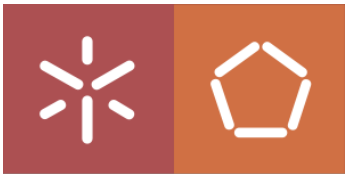

Universidade do Minho
Escola de Engenharia

Processamento de Linguagem Natural em Engenharia Biomédica

Processamento de Ficheiros em Formato PDF

Clara Cunha, PG56119
Manuel Carvalho, PG56140
Nuno Matos, PG56149

Mestrado em Engenharia Biomédica - Informática Médica

22 de abril de 2025

Resumo

O presente trabalho visa o processamento de três documentos em formato PDF para extrair informação médica. Para tal, foram desenvolvidos *parsers* personalizados que convertem os ficheiros para formatos manipuláveis (TXT/XML), processam o conteúdo mediante expressões regulares e regras específicas, e organizam os dados em estruturas JSON padronizadas. A metodologia incluiu etapas de análise documental, limpeza de texto, marcações, extração de termos, definições, traduções e metadados (como classes gramaticais, áreas temáticas e exemplos), além de correções manuais pontuais para inconsistências. Os resultados demonstram a eficácia da abordagem na extração da informação, contribuindo para uma melhor organização e acesso a terminologias médicas em contextos académicos e clínicos.

Conteúdo

1 Introdução

O presente trabalho tem como principal objetivo a análise e extração de informação a partir de documentos em formato PDF. Para tal, propõe-se o desenvolvimento de vários *parsers* capazes de identificar e recolher dados relevantes contidos nos documentos disponibilizados. A informação extraída é organizada e estruturada num ficheiro em formato `JSON`.

O desenvolvimento do trabalho é guiado por uma metodologia em várias etapas, que inclui a análise dos documentos, a definição de uma estrutura de dados adequada, a conversão dos ficheiros para formatos manipuláveis, a limpeza e anotação dos dados, a extração da informação relevante e, por fim, o seu armazenamento num formato estruturado. Esta abordagem visa assegurar um tratamento eficiente, coerente e sistemático dos dados processados.

2 Metodologia

2.1 Análise dos documentos

Numa fase inicial do desenvolvimento do trabalho, procedeu-se à seleção dos documentos a serem analisados. Para além dos dois documentos obrigatórios, `glossario_neologismos_saude.pdf` e `diccionari-multilinguee-de-la-covid-19.pdf`, foi incluído um terceiro documento, o `Glossário de Termos Médicos Técnicos e Populares.pdf`, de forma a cumprir os critérios estabelecidos e enriquecer a diversidade da informação a ser processada.

Antes de iniciar o processo de extração de informação, foi essencial realizar uma análise do conteúdo e da estrutura dos documentos selecionados. Cada documento apresenta características próprias que influenciam a abordagem de processamento a ser adotada. A seguir, são descritas as principais observações sobre os documentos analisados.

O documento `glossario_neologismos_saude.pdf` é essencialmente um glossário de termos médicos em português do Brasil. Para o trabalho de extração, o foco está entre os capítulos **3.2** e **3.3**, onde estão listados os termos que compõem o glossário. Cada entrada do glossário segue um formato consistente:

- **Termo**: O neologismo em destaque, seguido da sua classe gramatical (ex.: "abeta, *s.f.*").

- **Equivalências linguísticas**: Traduções para inglês ([ing]) e espanhol ([esp]), como "[ing] abeta; [esp] abeta".
- **Definição**: Explicação concisa do termo.
- **Informações complementares** (opcionais):
  - Siglas (ex.: "AVCI" para "acidente vascular cerebral isquêmico").
  - Notas enciclopédicas (sinalizadas com "Inf. encicl.")
  - Exemplos que ilustram o uso do termo, sinalizados entre aspas.

O documento `diccionari-multilinguee-de-la-covid-19.pdf` contitui um dicionário multilíngua que compila terminologias relacionadas com a COVID-19, abrangendo termos essenciais para a compreensão da pandemia. O conteúdo de interesse para a análise centra-se no corpo principal do dicionário, onde se encontram organizadas as entradas terminológicas. Cada entrada do dicionário segue uma estrutura padronizada:

- **Número de ordem**: Identificador único atrabuído a cada termo.
- **Termo em catalão**: A palavra em destaque seguida da categoria gramatical (ex.: *n m* para substantivo masculino).
- **Variantes**: Quando aplicável, termo relacionado ou equivalente que pode ser consultado no glossário, sinalizado como "veg".
- **Sinónimos complementares**: Quando aplicável, sinalizados como "sin. compl.".
- **Equivalências linguísticas**: Incluindo occitano (oc), basco (eu), galego (gl), castelhano (es), inglês (en), francês (fr), português (pt) (com distinção entre português de Portugal ([PT]) e do Brasil ([BR])), neerlandês (nl) e árabe (ar). Cada equivalente é acompanhado da respetiva categoria gramatical.
- **Área temática**: Indica o domínio de aplicação do termo (ex.: Epidemiologia, Clínica, Prevenção).
- **Definição**: Explicação concisa do termo.
- **Notas adicionais**: Informações complementares.

O documento `glossario_de_termos_medicos_tecnicos_e_populares.pdf` é um glossário abrangente de termos médicos, técnicos e populares em português de Portugal. Cada entrada do glossário contém o **termo** em destaque com a sua **definição**. Cada definição termina com a marcação `(pop)`.

2.2 Tratamento dos dados

2.2.1 Glossário de Neologismos em Saúde

A implementação das etapas orientadas à extração estruturada da informação relevante do `Glossário de Neologismos em Saúde` encontra-se no ficheiro `neologismos.py`, localizado na pasta `neologismos`.

1. Conversão do formato PDF para TXT

Como etapa inicial do processo, procedeu-se à conversão do documento original para um formato conveniente à sua manipulação. Para tal, o ficheiro em formato PDF foi convertido para TXT recorrendo, na linha de comandos, ao comando `pdftotext`.

2. Leitura do ficheiro

O código lê o conteúdo do ficheiro `neologismos.txt`, carregando-o como uma única string. Utiliza-se a codificação UTF-8 de modo a assegurar a correta interpretação de caracteres especiais e acentuados.

```
with open('neologismos.txt', 'r', encoding='utf-8') as file
    :
     content = file.read()
```

3. Extração da secção do glossário

É extraída a secção relevante do glossário, localizada entre os capítulos 3.2 e 3.3. Este processo é realizado por meio de uma expressão regular que captura o conteúdo entre essas secções, considerando que a quebra de página é representada pelo caractere especial `FF`.

```
pattern = r'FF3\.2(.*?)3\.3'
match = re.search(pattern, content, re.DOTALL)
```

Os 14 primeiros caracteres do texto extraído são eliminados, uma vez que correspondem a uma parte do texto que não faz parte das entradas do glossário propriamente ditas, mas sim ao título da secção. Assim, garante-se que o conteúdo extraído comece diretamente com as entradas relevantes do glossário.

```
glossario_raw = match.group(1)[14:].strip()
```

4. Limpeza dos dados

O conteúdo extraído passa por um processo de limpeza, onde as quebras de página são substituídas por espaços, a fim de melhorar a legibilidade.

```
glossario_raw = re.sub(r'\nFF', ' ', glossario_raw)
```

Além disso, o código também remove informações desnecessárias, como o nome de publicações que aparecem depois de exemplos dentro das definições dos termos.

```
glossario_raw = re.sub(r'" [^(]+', '" ', glossario_raw)
```

O conteúdo restante é dividido em várias entradas com base num padrão de citação, de modo a que cada entrada represente um conceito.

```
glossario = re.split(r'".?\s*\(', glossario_raw)
```

5. Extração da informação relevante

A função `parse_entry` é responsável por analisar cada entrada de conceito e extrair informações específicas.

Inicialmente, o termo é identificado, sendo que este é a sequência de palavras que aparece antes do ponto e da letra que indica a classe gramatical.

```
title_pattern = r'(.+) \w\.'
title = re.search(title_pattern, entry).group(1).strip()
```

Na etapa seguinte, procede-se à extração da classe gramatical associada ao termo, capturando a letra que aparece após o ponto.

```
gender_pattern = r'.+ \w\.(\w)\.'
gender = re.search(gender_pattern, entry).group(1)
```

As traduções do termo para os idiomas inglês e espanhol são localizadas a partir de padrões específicos que reconhecem as marcações `[ing]` e `[esp]`, respetivamente.

```
eng_trans = re.search(r'(.*)\[ing\]', entry)
esp_trans = re.search(r';(.*)\[esp\]', entry, flags=re.
   DOTALL)
```

A definição do termo é extraída, capturando o texto que vem após ao caractere **]** até ao primeiro ponto, e é armazenada na variável `definition`. Caso nenhuma definição seja identificada, a variável correspondente recebe uma string vazia.

```
definition_pattern = r'\]\n([\s\S]*?\.)'
definition = re.search(definition_pattern, entry)

definition = definition.group(1) if definition else ""
```

A função verifica a presença de siglas, identificadas pela expressão “Sigla:” seguida do valor da sigla. Se encontrada, a sigla é extraída, armazenada separadamente e removida do corpo da definição.

```
sigla = re.search(r"Sigla: (.*)\n", definition)
if sigla:
    sigla = sigla.group(1).strip()
    definition = re.sub(r"Sigla: (.*)\n", "", definition)
```

A função também procura por uma referência enciclopédica, marcada pela expressão “Inf. encicl.:”. Se encontrada, a referência é extraída e normalizada.

```
encicl = re.search(r"Inf. encicl.: (.*?\.)", entry, flags=
   re.DOTALL)
if encicl:
    encicl = encicl.group(1).replace("\n", " ").strip()
```

A função ainda contempla a extração de um exemplo de uso do termo.

```
usage_pattern = r'"([^"]*)'
usage = re.search(usage_pattern, entry, flags=re.DOTALL)
```

Após a recolha de todas as informações relevantes, estas são organizadas num dicionário denominado `content`, o qual agrupa os dados extraídos: traduções, classe gramatical, descrição, exemplo de uso, sigla (se presente) e referência enciclopédica (se aplicável):

```
content = {
    'traducoes': {'en': eng_trans.group(1).replace("\n", "
       ").strip() if eng_trans else "",
        'es': esp_trans.group(1).replace("\n", " ").strip()
            if esp_trans else ""
        },
    'categoria' : "n " + gender,
    'descricao': definition.replace("\n", " ").strip(),
    'exemplo': usage.group(1).replace("\n", " ").strip() if
        usage else ""
```

```
}

if sigla:
    content['sigla'] = sigla
if encicl:
    content['enciclopedia'] = encicl
```

Por fim, a função `parse_entry` é aplicada a cada entrada do glossário, sendo os resultados armazenados num dicionário principal, no qual cada chave corresponde ao termo e o valor ao conjunto de informações extraídas.

```
for conceito in glossario[:-1]: #remover ultima entrada pq
      s   (19, 12, ...)
    title, content = parse_entry(conceito)
    dict[title] = content
```

6. Escrita dos dados em JSON

O dicionário `dict` é salvo num ficheiro `JSON`.

```
with open('neologismos.json', 'w', encoding='utf-8') as
   json_file:
    json.dump(dict, json_file, ensure_ascii=False, indent
       =4)
```

7. Intervenções manuais

Foram realizadas algumas correções manuais no texto original para resolver inconsistências observadas.

Na linha 3672, a abreviação “es” foi substituída por “[esp]”, para garantir a conformidade com a convenção de tradução para o espanhol.

Na linha 6109, para o termo “tripanossomicida” foi acrescentado o caractere de aspas no exemplo. Também nas linhas 4678 e 3420 foi inserido este caractere com o mesmo propósito, para os termos “genes do imprinting” e “biologia sistêmica”, respetivamente.

Na linha 3337, o apóstrofo foi substituído por aspas para o termo “beta-amiloide”, alinhando-se à norma de uso de aspas para indicar exemplos.

2.2.2 Dicionário Multilíngua da COVID-19

A implementação das etapas destinada à extração estruturada da informação relevante do `Dicionário Multilíngua da COVID-19` encontra-se no ficheiro `multilingues.py`, localizado na pasta `neologismos`.

1. Conversão do formato PDF para XML

A primeira fase do processo consistiu na conversão do ficheiro original, em formato PDF, para o formato XML. Esta conversão foi realizada por meio do comando `pdftohtml -xml`, executado na linha de comandos.

2. Leitura do ficheiro

Posteriormente, procedeu-se à abertura e leitura do ficheiro `multilingue.xml`, recorrendo à codificação UTF-8 para assegurar a correta interpretação dos caracteres.

```
with open('multilingue.xml', 'r', encoding='utf-8') as file
    :
    content = file.read()
```

3. Extração das páginas

O ficheiro `multilingue.xml` é processado utilizando a biblioteca `BeautifulSoup`. O algoritmo percorre todas as páginas do documento, identificando o número de cada uma e seleciona apenas aquelas cujo número se encontra no intervalo compreendido entre 30 e 183, consideradas relevantes para o processo de extração.

Em cada página selecionada, o conteúdo textual é separado em duas colunas — `left_column` e `right_column` — com base no valor do atributo `left`. Este elemento difere para páginas da esquerda e da direita.

```
for page in soup.find_all('page'):
    page_number = int(page['number'])

    if page_number % 2 == 0:
        left = 500
    else:
        left = 440

    if 30 <= page_number <= 182:
        left_column = ''
```

```
        right_column = ''

        for text in page.find_all('text'):
            if int(text['left']) < left:
                left_column += str(text) + '\n'
            else:
                right_column += str(text) + '\n'
```

Finalmente, o conteúdo de ambas as colunas é concatenado e armazenado na variável `pages`.

```
        page = left_column + right_column
        pages += (page)
```

4. Limpeza dos dados

Inicialmente, procede-se à remoção de toda a informação irrelevante, como números de página, cabeçalhos (todos estes são identidivéis pela sua font distinta) e das tags de xml, que já não são relevantes, usando expressões regulares.

```
pages = re.sub(r'<text font="6".*/text>\n', r'', pages)
pages = re.sub(r'<.*font="14".*/text>\n', r'', pages)
pages = re.sub(r'<.*font="15".*/text>\n', r'', pages)
pages = re.sub(r'<.*font="36".*/text>\n', r'', pages)
pages = re.sub(r'<.*font="38".*/text>\n', r'', pages)
pages = re.sub(r'<.*?>', r'', pages)
```

5. Marcações

São inicialmente definidos todos os marcadores que serão utilizados bem como todos os elementos que visam separação.

```
markers = "@#§*∞  $"

categorias = {
    r"n",
    r"n pl",
    r"n m",
    r"n m pl",
    r"n f",
    r"n f pl",
```

```
    r"n m, f",
    r"n m/f",
    r"adj",
    r"v tr",
    r"v tr/intr",
    r"v intr"
}

remis = {
    r'sin. compl.',
    r'sin.',
    r'den. com.',
    r"sigla",
    r"veg."
}

langs = {
    r"oc",
    r"eu",
    r"gl",
    r"es",
    r"en",
    r"fr",
    r"pt",
    r"pt \[PT\]",
    r"pt \[BR\]",
    r"nl",
    r"ar"
}

codigos = {
    r"sbl",
    r"nc",
    r"CAS"
}

campos = {
```

```
    r"CONCEPTES GENERALS",
    r"EPIDEMIOLOGIA",
    r"ETIOPATOG NIA",
    r"DIAGN STIC",
    r"CL NICA",
    r"PREVENCI ",
    r"TRACTAMENT",
    r"PRINCIPIS ACTIUS",
    r"ENTORN SOCIAL"
}
```

Sempre que uma linha coincide com uma das `categorias` previamente definidas, essa linha é destacada com o prefixo `##`, de modo a facilitar a sua identificação nas etapas seguintes.

```
new_pages = ''
for line in pages.split('\n'):
    line = line.strip()
    if line in categorias:
        new_pages += '##' + line + '\n' # marcar as
           categorias com ##
    else:
        new_pages += line + '\n'
```

Com o intuito de assinalar os diferentes códigos, como "CAS", "sbl"e "nc"quando estes existem, o código utiliza uma expressão regular para acrescentar o caractere “§”.

```
codigos_pattern = rf"^({'|'.join(codigos)})\n(?=[^{markers
   }])"
pages = re.sub(codigos_pattern, r'(marcador referido acima)
   \1\n', pages, flags=re.MULTILINE)
```

Com o intuito de assinalar as diferentes traduções, como "en", "pt"e "es"quando estas existem, o código utiliza uma expressão regular para acrescentar o caractere “★”.

```
traducao_pattern = rf"^({'|'.join(langs)})\n(?=[^{markers
   }])"
pages = re.sub(traducao_pattern, r'*\1\n', pages, flags=re.
   MULTILINE)
```

Com o intuito de assinalar diferentes tipos de remissão e siglas, como "sigla", "sin"(sinónimo)

e "veg"(ver) quando estes existem, o código utiliza uma expressão regular para acrescentar o caractere “∞”.

```
remis_pattern = rf"^({'|'.join(remis)})\n(?=[^{markers}])"
pages = re.sub(remis_pattern, r'∞\1\n', pages, flags=re.
    MULTILINE)
```

Com o intuito de assinalar os diferentes tipos de descrição, como "CONCEPTES GENERALS"e "EPIDEMIOLOGIA"quando estes existem, o código utiliza uma expressão regular para acrescentar o caractere “”.

```
campos_pattern = rf"^({'|'.join(campos)})\.(?=[^{markers}])
    "
pages = re.sub(campos_pattern, r' \1\.', pages, flags=re.
    MULTILINE)
```

Com o intuito de assinalar as notas associadas a cada termo, o código utiliza uma expressão regular para acrescentar o caractere “”.

```
nota_pattern = r"^(Nota:.*)"
pages = re.sub(nota_pattern, r' \1', pages, flags=re.
    MULTILINE)
```

Com o intuito de assinalar o identificador único associado a cada termo, o código utiliza uma expressão regular para acrescentar o caractere “@”.

```
concept_pattern = r"(^[1-9]\d*(\n+.*){1,2}\n##.+?)"
pages = re.sub(rf"{concept_pattern}", r"@\1", pages, flags=
    re.MULTILINE)
```

6. Extração da informação relevante

Após as marcações referidas acima procede-se à segmentação do corpus utilizando o caractere “@” como delimitador. O conteúdo segmentado é armazenado na lista `glossario`, sendo selecionados todos os elementos menos o primeiro, correspondentes às entradas relevantes do dicionário.

```
glossario = pages.split('@')
glossario = glossario[1:]
```

O título do conceito é extraído usando uma expressão regular.

```
title_pattern = r"^\d+((\n.*){1,2})\n##(.*)"
```

```
title = re.search(title_pattern, concept, flags=re.
    MULTILINE)
```

A categoria do conceito é extraído usando uma expressão regular.

```
categoria_pattern = r"^\d+(\n.*){1,2}\n##(.*)"
categoria = re.search(categoria_pattern, concept, flags=re.
    MULTILINE)
```

As traduções do conceito, quando presentes, são extraídas usando uma expressão regular.

```
traducoes_pattern = rf"*((.*\n)*?)(@|§|∞| | |$)"
traduc = re.findall(traducoes_pattern, concept, flags=re.
    MULTILINE)
```

Como cada termo pode apresentar mais que uma tradução, e cada tradução pode ter associada uma categoria são usadas expressões regulares e segmentações pelas marcas anteriormente faladas, para identificar cada tradução e respetiva categoria, se presente.

As remissões do conceito, quando presentes, são extraídas usando uma expressão regular.

```
remi_pattern = rf"∞((.*\n)*?)(@|§|*| | |$)"
remiss = re.findall(remi_pattern, concept, flags=re.
    MULTILINE)
```

Como já referido para as traduções pode ocorrer que haja mais que uma remissão, são usados os mesmos métodos para extrair a informação.

Os códigos do conceito, quando presentes, são extraídos usando uma expressão regular.

```
codigos_pattern = rf"§((.*\n)*?)(?={'|'.join(markers)})"
codigos = re.findall(codigos_pattern, concept, flags=re.
    MULTILINE)
```

A descrição do conceito, quando presente, é extraída usando uma expressão regular.

```
campos_pattern = rf" .*\.((.*\n)*?)(?={'|'.join(markers)})"
campos = re.search(campos_pattern, concept, flags=re.
    MULTILINE)
```

As notas do conceito, quando presentes, são extraídas usando uma expressão regular.

```
nota_pattern = rf" Nota:((.*\n)*?)(?={'|'.join(markers)})"
notas = re.search(nota_pattern, concept, flags=re.MULTILINE
    )
```

Toda a informação extraída é de alguma forma processada de forma a melhorar a legibilidade final.

7. Escrita dos dados em JSON

Após a extração e organização das informações relevantes de cada conceito, os dados estruturados são exportados para um ficheiro no formato `JSON`.

```
with open('glossario.json', 'w', encoding='utf-8') as
   json_file:
    json.dump(dict, json_file, ensure_ascii=False, indent
       =4)
```

2.2.3 Glossário de Termos Médicos Técnicos e Populares

A implementação das etapas orientadas à extração estruturada da informação relevante do `Glossário de Termos Médicos Técnicos e Populares` encontra-se no ficheiro `popular.py`, localizado na pasta `neologismos`.

1. Conversão do formato PDF para XML

O documento original, inicialmente em formato PDF, foi convertido para o formato XML, na linha de comandos, através da utilização do comando `pdftohtml -xml`

2. Leitura do ficheiro

O código procede à abertura e leitura do conteúdo integral do ficheiro `populares.xml`, codificado em UTF-8:

```
with open('populares.xml', 'r', encoding='utf-8') as file:
    content = file.read()
```

3. Extração da secção do glossário

O ponto de início relevante do glossário, correspondente à entrada da letra "A", é identificado. Através de uma expressão regular, localiza-se esse ponto e descarta-se o conteúdo anterior, garantindo que apenas a secção pertinente é processada.

```
start_pattern = r"<text[^>]*><b>A</b></text>"
start_match = re.search(start_pattern, content)
content = content[start_match.start():] if start_match else
    ""
```

4. Limpeza dos dados

Segue-se uma fase de limpeza do conteúdo XML, em que são removidas diversas tags estruturais (<fontspec>, <page>, <text>, <i>), que não contribuem semanticamente para os dados a extrair. Esta limpeza permite reduzir ruído textual e facilita o tratamento posterior.

Adicionalmente, são eliminadas linhas que contenham exclusivamente letras maiúsculas soltas, que indicam a divisão por letra inicial no glossário (ex. A", B", etc.).

```
tags = [
r"</?fontspec.*?>",
r"</?page.*?>",
r"</?text.*?>",
r"<i>",
r"</i>"
]
content = re.sub(r"<.*>[A-Z]<.*>", "", content) # letras
for tag in tags:
    content = re.sub(tag, "", content)
```

De modo a isolar as definições, os termos são removidos do conteúdo, e o texto restante é sujeito a uma limpeza: remoção de parágrafos, vírgulas e excesso de espaços em branco.

```
terms_re = r"<b>(.*?)</b>"

processed_content = re.sub(terms_re, "", content)
processed_content= re.sub(r"\n+", " ", processed_content)
processed_content = re.sub(r"\s+,", "", processed_content)
processed_content = re.sub(r"\s{2,}", " ",
   processed_content)
```

5. Marcações

Com o intuito de facilitar a extração das definições, a marcação "(pop)", comum a todas as entradas, é substituída pelo marcador "@@".

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

6. Extração da informação relevante

Nesta etapa, procede-se à extração integral dos termos e das respetivas definições, com base no marcador delimitador previamente estabelecido. Tanto os termos como as definições são limpos de espaços em excesso.

```
terms = re.findall(terms_re, content)
terms = [termo.strip() for termo in terms]

design = re.findall(r"(?:@(?:\s|,|X)|^)(.*?)@",
   processed_content)
design = [desc.strip() for desc in design]
```

Com os termos e respetivas definições extraídos, é construído um dicionário (`glossario_dict`) que associa cada termo à sua definição. O algoritmo contempla também a possibilidade de existirem múltiplas definições para um mesmo termo, agrupando-as adequadamente.

```
glossario_dict = dict(zip(terms, design))

for termo, descricao in zip(terms, design):
    if termo in glossario_dict:
        if isinstance(glossario_dict[termo], list):
            if descricao not in glossario_dict[termo]:
                glossario_dict[termo].append(descricao)
        else:
            if descricao != glossario_dict[termo]:
                glossario_dict[termo] = [glossario_dict[
                   termo], descricao]
    else:
        glossario_dict[termo] = descricao

for termo, descricao in glossario_dict.items():
    glossario_dict[termo] = {'descricao': descricao}
```

7. Escrita dos dados em JSON

Finalmente, o dicionário é exportado em formato `JSON`, utilizando a codificação UTF-8.

```
with open("populares.json", "w", encoding="utf-8") as
   output:
    json.dump(glossario_dict, output, ensure_ascii=False,
```

```
        indent=4)
```

7. Intervenções manuais

Apesar da automatização de grande parte do processo de extração e tratamento dos dados, foi necessário realizar algumas correções manuais no glossário, decorrentes de inconsistências na estrutura do ficheiro XML.

Em alguns casos, termos compostos foram separados por quebras de linha indevidas, o que inviabilizou a sua identificação automática. Foram, por isso, reconstruídos manualmente:

- *pré-medicação*
- *infeção cruzada*
- *efeito colateral*
- *tremor intencional*

Além disso foram identificadas definições duplicadas para certos termos, em que uma das ocorrências continha erros ortográficos. Essas definições foram corrigidas, mantendo-se ambas as versões com as devidas correções:

- *protrombina* – uma das definições continha a palavra "actor" em vez de "factor" (omissão da letra inicial), e portanto foi adicionada a letra "f" à versão incorreta.
- *pré-medicação* – uma ocorrência apresentava "gera" em vez de "geral", adicionando-se a letra "l" na versão com erro.

Por fim, foram retirados símbolos extraídos automaticamente, resultante da conversão das aspas, na palavra "*dooping*".

2.3 Junção dos documentos

Por fim, procede-se à consolidação dos três ficheiros JSON previamente gerados, com o objetivo de centralizar a informação disponível. Para tal, os três ficheiros finais discutidos anteriormente são carregados para a memória.

Adota-se como base o ficheiro correspondente ao Dicionário Multilíngua da COVID-19, uma vez que este apresenta maior completude e complexidade estrutural. A partir dele, incorporam-se os dados provenientes dos demais ficheiros JSON.

Nos casos em que uma entrada já se encontra presente no ficheiro-base, a informação adicional é integrada de forma complementar — por exemplo, agregando traduções em inglês sob a forma de listas. Quando a informação se encontra ausente, como no caso de exemplos de uso, os novos dados são inseridos como campos adicionais.

3 Conclusão

O desenvolvimento deste trabalho permitiu a implementação de metodologias para a extração e estruturação de informação médica a partir de documentos em formato PDF, demonstrando a viabilidade do processamento automático de terminologia especializada. A abordagem adotada, baseada na análise individualizada de cada documento, possibilitou a criação de *parsers* adaptados às particularidades estruturais de cada fonte, garantindo a correta identificação de termos, definições e metadados associados.

A conversão dos documentos para formatos intermeédios (TXT e XML) foi essencial para viabilizar o processamento automático, com alguns desafios adicionais como a necessidade de correções manuais em casos pontuais de inconsistências textuais. A utilização de expressões regulares permitiu a normalização do conteúdo, preservando a integridade semântica da informação.

No geral, os resultados obtidos evidenciam a eficiência das técnicas aplicadas, com a extração bem-sucedida de termos médicos, as suas definições e informações complementares.