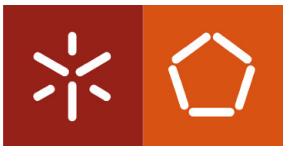

Relatório - Trabalho Prático 1

Processamento de Linguagem Natural em Engenharia Biomédica

Mestrado em Engenharia Biomédica - Informática Médica

Grupo :
Afonso Rodrigues (PG55831)
João Esteves Gil (PG56132)
Maria Inês Eusébio (PG57586)

Docentes:
José Almeida
Luís Filipe Cunha

Índice

1 Introdução

Este relatório tem como principal objetivo documentar o desenvolvimento do primeiro trabalho prático da Unidade Curricular de Processamento de Linguagem Natural, no âmbito do curso de Engenharia Biomédica. O trabalho consiste na extração e organização de informação relevante a partir de vários documentos médicos técnicos e científicos em formato *XML* (provenientes de *PDFs* convertidos).

Para identificar os dados pretendidos — como termos principais, categorias, siglas associadas, traduções em diferentes idiomas e descrições — foram utilizadas expressões regulares. Esta abordagem possibilita, de forma flexível e automatizada, localizar padrões complexos no texto, facilitando operações como extração, substituição e limpeza de dados. Paralelamente, foi realizada uma pré-processamento dos ficheiros para remover elementos desnecessários como cabeçalhos, rodapés, referências CAS e numeração.

Os dados extraídos foram estruturados em ficheiros *JSON*, promovendo a reutilização da informação em aplicações futuras de análise, visualização ou interoperabilidade com sistemas biomédicos.

2 Documentos Processados

Para a realização deste documento foram utilizados três documentos onde se realizou a extração de informação, sendo os documentos escolhidos:

- diccionari-multilinguee-de-la-covid-19.pdf
- glossario_neologismos_saude.pdf
- glossario_ministerio_saude.pdf

O primeiro documento analisado consiste num dicionário médico multilingue com termos relativos à pandemia de COVID-19. Cada entrada apresenta, além do termo principal, uma descrição textual, traduções em diversas línguas — incluindo português, espanhol, francês, inglês, entre outras — e, opcionalmente, uma ou mais siglas associadas. A presença de siglas não é obrigatória, o que exigiu uma diferenciação cuidadosa entre entradas com e sem siglas durante o processo de extração.

Relativamente ao segundo documento, trata-se de um dicionário de neologismos terminológicos no domínio da saúde humana. A cada conceito está associado o respetivo substantivo — identificado como masculino ou feminino —, bem como as suas traduções em espanhol e inglês, uma sigla correspondente (quando existente) e ainda uma citação ilustrativa. Nos casos em que não existe sigla associada, esta é simplesmente omitida, sem comprometer a integridade dos restantes dados. A estrutura uniforme deste ficheiro permitiu aplicar um tratamento sistemático com base em padrões bem definidos.

Por último, o terceiro documento consiste num glossário desenvolvido pelo Ministério da Saúde, igualmente inserido no domínio da medicina. Neste caso, a cada conceito é atribuída uma categoria temática e uma descrição explicativa que contextualiza o seu significado e uso no âmbito da saúde.

3 Arquitetura

A arquitetura do projeto assenta na extração, estruturação e junção de dados a partir de três documentos principais:

- diccionari-multilinguee-de-la-covid-19.pdf;
- glossario_neologismos_saude.pdf;
- glossario_ministerio_saude.pdf.

Cada um desses documentos foi convertido previamente para o formato XML através do comando pdftohtml -xml, de forma a facilitar o seu tratamento com expressões regulares.

A partir do diccionari-multilinguee-de-la-covid-19.pdf, obteve-se um ficheiro JSON estruturado com termos médicos multilingues relativos à COVID-19, incluindo as respetivas descrições, traduções em várias línguas e, quando aplicável, siglas associadas. Este processo resultou no ficheiro glossario_covid.json

Do glossario_neologismos_saude.pdf, extraiu-se um conjunto de termos em português relacionados com neologismos na área da saúde. Para cada termo, foram recolhidas informações como a tradução em inglês e espanhol, a categoria gramatical (substantivo masculino ou feminino), a descrição do termo, a sigla (caso exista) e uma citação ilustrativa. Este processo resultou no ficheiro glossario_neologismos.json.

Por sua vez, o glossario_ministerio_saude.pdf deu origem ao glossario_ministerio.json, contendo conceitos definidos pelo Ministério da Saúde. Para cada conceito, foi identificada a sua categoria temática e uma descrição explicativa.

Após o tratamento individual de cada documento, procedeu-se à junção dos três ficheiros. Este processo foi realizado com o objetivo de consolidar todos os conceitos num único ficheiro denominado glossario_junçao.json. Para cada conceito, é registada a fonte de onde foi extraído, evitando duplicações.

O resultado final foi um ficheiro glossario_junçao.json contendo todos os conceitos unificados, ordenados alfabeticamente, garantindo a consistência e a não duplicação da informação.

4 Tratamento de Dados

Para o tratamento dos dados, todos os ficheiros PDF selecionados foram previamente convertidos para o formato XML através do comando pdftohtml -xml, executado na linha de comandos. Posteriormente, foi realizada uma limpeza inicial dos documentos XML, removendo-se toda a informação irrelevante, tanto com o auxílio da função re.sub (expressões regulares), como através de edições manuais complementares.

4.1 diccionari-multilinguee-de-la-covid-19.pdf

O *Diccionari Multilingüe de la COVID-19* é uma obra terminológica que reúne quinhentos conceitos associados à pandemia da COVID-19, apresentando-os de forma estruturada em múltiplas línguas, incluindo o catalão, castelhano, inglês, francês, português, entre outras.

4.1.1 Processamento do documento

O processamento deste documento passou por várias fases metodológicas com o objetivo de converter o conteúdo terminológico do PDF original para um formato estruturado e reutilizável — concretamente, um ficheiro JSON. Devido à complexidade estrutural do PDF, este foi primeiramente convertido para XML, facilitando a manipulação textual através de expressões regulares e algoritmos. Foram removidos elementos indesejados como marcadores de página, metadados e outras etiquetas irrelevantes, utilizando expressões regulares.

```
xml = re.sub(r"</page>", "", xml)
xml = re.sub(r"<page[^>]*>", "", xml)
xml = re.sub(r"<fontspec[^>]*/>", "", xml)
xml = re.sub(r"\s*\n\s*", "", xml)

conceitos = re.findall(r'<text[^>]+font="11">(\d+)\s*</text>\s*<text[^>]+font="2[5,6]"><b>(.*?)</b></text>', xml)
conceitos.sort(key=lambda x: int(x[0]))

padrao_bloco = re.compile(
    r'<text[^>]+font="11">' + r'{}' + r'\s*</text>\s*<text[^>]+font="2[5-6]"><b>{}</b></text>(.*?)(?=<text[^>]+font="11">\d+\s*</text>|$)',
    re.DOTALL
)
```

Figura 1: Expressões regulares utilizadas para pré-processamento do texto.

Através de padrões definidos, foi possível identificar blocos de informação associados a cada conceito, nomeadamente: Denominações principais e sinónimos;

- Traduções para outras línguas (detetadas por padrões língua e respetivo termo);
- Siglas e abreviaturas relevantes;
- Descrições e definições contextuais;
- Notas explicativas ou citações.

```python
resultados = []

for i, (num, nome) in enumerate(conceitos):
    padrao_conceito = padrao_bloco.pattern.format(re.escape(num), re.escape(nome))
    match = re.search(padrao_conceito, xml)

    if not match:
        print(f"[!] Bloco não encontrado para: {nome}")
        continue
    bloco = match.group(1)

    sub_match = re.search(r'<i>\s*n\s*m?\s*</i>', bloco)
    substantivo = sub_match.group(0).strip("<i> </i>") if sub_match else ""

    traducoes = re.findall(r'<i>([a-z\[\]\s]+)</i></text>\s*<text[^>]+font="11">(.*?)</text>',bloco)

    traducoes_formatadas = []
    for lang, termo in traducoes:
        lang = lang.strip()
        termo = termo.strip()

        if not re.match(r'^n(\s+[mf])?$', lang):
            traducoes_formatadas.append(f"{lang}: {termo}")


    sigla_match = re.search(r'sigla\s*</text>\s*<text[^>]+font="25"><b>(.*?)</b></text>', bloco)
    sigla = sigla_match.group(1).strip() if sigla_match else ""


    descricoes = re.findall(r'<text[^>]+font="(?:11|27)">(.+?)</text>', bloco)
    descricoes_filtradas = []

    for d in descricoes:
        texto = d.strip()

        if re.fullmatch(r'[0-9,\s]+', texto):
            continue
        if texto.lower() == nome.lower():
            continue
        if re.match(r'^[A-Z]{2,5}$', texto):
            continue
        if texto.lower() in [t.split(": ")[-1].lower() for t in traducoes_formatadas]:
            continue
        if re.search(r'sigla|Nota:', texto, re.IGNORECASE):
            continue

        descricoes_filtradas.append(texto)

    descricao = " ".join(descricoes_filtradas)


    citacao_match = re.search(r'(Nota:.*?)(?=<text[^>]+font="11">|\Z)', bloco, re.DOTALL)
    if citacao_match:
        citacao_bruta = citacao_match.group(1)

        citacao_limpa = re.sub(r'<[^>]+>', '', citacao_bruta).strip()
        citacao = re.sub(r'\s+', ' ', citacao_limpa)
    else:
        citacao = ""

    resultados.append({
        "Conceito": nome,
        "Substantivo": substantivo,
        "Traduções": traducoes_formatadas,
        "Sigla": sigla,
        "Descricao": descricao,
        "Citação": citacao
    })
```

Figura 2: Tratamento do xml e transformação em dicionário

4.1.2 Tratamento e organização da informação:

Toda a informação extraída foi processada para remover duplicados, ruído e inconsistências, ficando formatada de forma uniforme.

4.1.3 Criação do ficheiro JSON:

Os dados tratados foram armazenados num ficheiro glossario_covid.json, contendo uma lista de dicionários, cada um representando um termo com os seus atributos principais.

Este processamento automatizado permite não só preservar a riqueza do conteúdo terminológico original, como também facilita a sua utilização futura em plataformas digitais, bases de dados ou aplicações lexicográficas. A estrutura JSON final assegura compatibilidade com diversas linguagens de programação e ambientes de desenvolvimento.

```
file_out="glossario_covid.json"
with open(file_out, "w", encoding="utf-8") as f:
    json.dump(resultados, f, ensure_ascii=False, indent=2)
print(f"Arquivo JSON exportado: {file_out}")
```

Figura 3: Código de criação do json

4.2 glossario_neologismos_saude.pdf

O Glossário de Neologismos da Saúde é um ficheiro que descreve um conjunto de termos e respetivas informações, desde traduções e descrições até usos, na forma de exemplos, dos conceitos em contexto prático. O objetivo é criar um ficheiro JSON com as informações agrupadas e organizadas uniformemente.

4.2.1 Processamento do documento

Antes de se iniciar o pré-processamento e processamento do ficheiro, é importante que seja feita a conversão do ficheiro PDF para XML, para que possa ser manipulado de uma forma mais fácil.

Após esta conversão, o processo inicia-se com a aplicação de expressões regulares. Estas, tem como objetivo retirar partes indesejadas do ficheiro, que vão desde marcadores de início e fim de página até numeração de página ou outros tipos de elementos textuais que não são úteis para o trabalho em questão.

```
texto = re.sub(r'</page>\s*\n?', '', texto) #remove todas as trocas de página do ficheiro
texto = re.sub(r'<text top="\d+" left="\d+" width="\d+" height="\d+" font="23"> </text>*\n?', '',texto)
texto = re.sub(r'<page number="\d+" position="absolute" top="\d+" left="\d+" height="\d+" width="\d+">*\n?', '', texto)
texto = re.sub(r'<text[^>]*?>\(((\d+)\)\s*</text>*\n?', '', texto)
```

Figura 4: Expressões regulares utilizadas para pré-processamento do texto.

São criados padrões de organização dos conceitos, texto e siglas, para que seja mais fácil a recolha destes dados.

São também inicializadas funções para que possam ser extraídas, do texto, todas as traduções (extrair_traducoes()), siglas (extrair_sigla()), descrições (extrair_descricao()) e citações (extrair_citacao()). Em todas as funções, os blocos de informação são extraídos e processados de forma a que esta fique limpa e utilizável.

```
# Função para extrair tradução
def extrair_traducoes(parte):
    blocos = padrao_texto.findall(parte)
    partes = []

    for bloco in blocos:
        texto_limpo = bloco.strip().lower()

        if "[ing]" in texto_limpo or "[esp]" in texto_limpo:
            partes.append(bloco.strip())
        else:
            break

    if partes:
        return " ".join(partes)
    return None
```

Figura 5: Exemplo de função utilizada para extração de informação.

Após tudo isto, é feito um match de todos os campos que dizem respeito aos conceitos e, depois, toda esta informação é guardada, na forma de dicionário, numa lista de resultados.

```python
resultado = []

for match in padrao_conceito_substantivo.finditer(texto):
    conceito = match.group("conceito").strip()
    genero = match.group("substantivo")
    fim = match.end()
    parte_pos_termo = texto[fim:fim+6500]

    traducoes = extrair_traducoes(parte_pos_termo)
    sigla = extrair_sigla(parte_pos_termo[:600])
    descricao = extrair_descricao(parte_pos_termo)
    citacao = extrair_citacao(parte_pos_termo)


    resultado.append({
        "Conceito": conceito,
        "Substantivo": genero,
        "Traduções": traducoes,
        "Sigla": sigla,
        "Descrição": descricao,
        "Citação": citacao,
    })
```

Figura 6: Processamento da informação.

Finalmente, esta lista é escrita num ficheiro glossario_neologismos.json, onde os dados ficam armazenados. Abaixo (Figura), mostra-se a formatação com que cada um dos dados fica guardado no ficheiro JSON.


```
{
  "Conceito": "ácido acetilsalicílico",
  "Substantivo": "s.m.",
  "Traduções": "acetilsalicilic acid [ing]; ácido acetilsalicílico [esp]",
  "Sigla": "Sem sigla associada",
  "Descrição": "Medicamento que pode ser utilizado pelo ser humano, como prevenção ao enfarte, desde que o usuário não possua altos níveis de colesterol.",
  "Citação": "“...A aspirina (acido acetilsalicílico) pode reduzir o risco de enfarte em mais de 30%, mas isso só vale para 75% da população...”"
},
```


Figura 7: Estrutura de dados no ficheiro glossario_neologismos.json.

4.3 glossario_ministerio_saude.pdf

O Glossário do Ministério da Saúde é um ficheiro PDF que tem informações relativas a vários conceitos dentro da categoria da saúde pública. Este documento foi criado com o objetivo de sistematizar, estruturar e indexar todos estes conceitos que se veem úteis no panorama da saúde, promovendo maior clareza e precisão na comunicação institucional.

Analisando o documento, é possível notar que existe um conjunto de informações características associadas a cada conceito, que são o conceito, a categoria em que este se insere, e a descrição. Serão estas as informações que será pretendido que se retire do glossário. Tudo o resto será eliminado.

4.3.1 Processamento do documento

O ficheiro glo_ministerio.py é o script onde todo o processamento do glossário é realizado.

A primeira fase, e antes de se entrar no script, consiste na conversão do documento de PDF para XML. Assim, torna-se mais fácil a tarefa de limpeza e processamento de dados, devido à maior acessibilidade e editabilidade de um ficheiro deste tipo. Após isto, o documento está pronto para ser processado pelo script.

Assim que o ficheiro é lido, vários caracteres e estruturas do documento são removidas através de expressões regulares, como se pode ver a figura 8. Este recurso permite que todas as ocorrências que seguem um determinado conjunto de restrições/características possam ser processadas e transformadas uniformemente. Aqui, são eliminadas estruturas como quebras de página, ou informações contidas presentes nos rodapés, por exemplo.

```
texto = re.sub(r'</page>\s*\n?', '', texto)
texto = re.sub(r'<page number="\d+" position="absolute" top="\d+" left="\d+" height="\d+" width="\d+">*\n?', '', texto)
texto = re.sub(r'<text top="\d+" left="\d+" width="\d+" height="\d+" font="\d+">\d+</text>*\n?', '', texto)
texto = re.sub(r'<text top="\d+" left="\d+" width="\d+" height="\d+" font="22">.</text>*\n?', '', texto)
texto = re.sub(r"<i>*\n?", "", texto)
texto = re.sub(r"</i>*\n?", "", texto)
texto = re.sub(r'</b>\n<b>(.*?)</b>\n', r' \1</b>\n', texto)
texto = re.sub(r"ü", "u", texto)
texto = re.sub(r"<b>*\n?", "", texto)
texto = re.sub(r"</b>*\n?", "", texto)
texto = re.sub (r"- ", "", texto) #usado para tirar quando é quebra de linha
texto = re.sub (r" – ", "", texto)
texto = re.sub (r"-*\n?", "", texto)
```

Figura 8: Expressões regulares utilizadas para limpeza do ficheiro XML.

A função extrair_descrição() (Figura 9) tem como objetivo recolher a descrição de um conceito, tal como o nome indica, ou seja, é capaz de extrair e agrupar (caso necessário, por exemplo quando existem quebras de página a meio da descrição) todas as partes de texto relativas a um termo e retornar esse mesmo texto.

```
def extrair_descricao(texto, inicio, fim_proximo=None):
    """
    Extrai todos os blocos font="14" entre 'inicio' e 'fim_proximo'.
    Se fim_proximo for None, vai até ao fim do texto.
    """
    descricao_partes = []
    limite = fim_proximo if fim_proximo else len(texto)

    for match in re.finditer(r'<text[^>]*font="14"[^>]*>(.*?)</text>', texto):
        if inicio <= match.start() < limite:
            descricao_partes.append(match.group(1).strip())

    descricao = " ".join(descricao_partes)

    descricao = re.sub(r'(\w+)-\s+(\w+)', r'\1\2', descricao)

    return descricao
```

Figura 9: Função extrair_descrição().

Após O pré-processamento e a inicialização da função, os conceitos são percorridos segundo um padrão criado em expressões regulares, e as informações vão sendo, de forma iterativa, recolhidas, agrupadas e organizadas. Após isso, as informações processadas são guardadas numa lista de dicionários (Figura 10).

```python
padrao_conceito_categoria = re.compile(
    r'<text[^>]*font="21"[^>]*>(?P<conceito>.*?)</text>\s*'
    r'<text[^>]*font="16"[^>]*>\s*(<i>)?Categoria:(</i>)?\s*</text>\s*'
    r'<text[^>]*font="14"[^>]*>(?P<categoria>.*?)</text>',
    re.DOTALL
)

resultado = []
matches = list(padrao_conceito_categoria.finditer(texto))

for i, match in enumerate(matches):
    conceito = match.group("conceito").strip()
    categoria = match.group("categoria").strip()
    fim_atual = match.end()

    if i + 1 < len(matches):
        inicio_proximo = matches[i + 1].start()
    else:
        inicio_proximo = None

    descricao = extrair_descricao(texto, fim_atual, inicio_proximo)

    resultado.append({
        "Conceito": conceito,
        "Categoria": categoria,
        "Descrição": descricao
    })
```

Figura 10: Processamento e organização das informações relativas a cada um dos conceitos.

Finalmente, o documento glossario_ministerio.json é aberto com premissões de escrita ("w"), e todas as informações presentes na lista "resultado"são transcritas para o ficheiro JSON.

As informações ficam guardadas como é possível ver na figura 11.

```json
{
    "Conceito": "Acidentes de trânsito",
    "Categoria": "Acidentes e Violência",
    "Descrição": "Acidentes com veículos, ocorridos na via pública."
},
```

Figura 11: Estrutura da informação guardada no ficheiro glossario_ministerio.json .

4.4 Junção dos ficheiros Json

Para além do tratamento individual dos diferentes documentos, foi ainda realizado um processo de fusão entre os três ficheiros *json* gerados. Este processo teve como objetivo combinar os conceitos presentes em ambos os ficheiros num único ficheiro consolidado, designado por glossario_junçao.json

Para isso, começou-se por fazer o *import* necessário e a leitura dos ficheiros, como consta na Figura 12.

```python
ministerio_json = "glossario_ministerio.json"
neologismos_json = "glossario_neologismos.json"
covid_json = "glossario_covid.json"
junçao_conceitos = "glossario_juncao.json"


with open(ministerio_json, "r", encoding="utf-8") as f:
    ministerio = json.load(f)

with open(neologismos_json, "r", encoding="utf-8") as f:
    neologismos = json.load(f)

with open(covid_json, "r", encoding="utf-8") as f:
    covid = json.load(f)
```

Figura 12: *Import* e leitura dos ficheiros *json*.

Cada um destes ficheiros JSON é lido e transformado numa lista de dicionários, onde cada dicionário representa um conceito com os seus respetivos campos (como descrição, categoria, traduções, etc.).

Para consolidar todos os conceitos num único dicionário unificado, é utilizada uma função chamada adicionar_conceito. Esta função percorre cada lista de conceitos e insere cada termo no dicionário final (conceitos_unificados). Se o conceito ainda não existir no dicionário, ele é adicionado e associado à fonte de onde provém (por exemplo, "ministerio", "neologismos" ou "covid"). Caso o conceito já exista, a informação da nova fonte é simplesmente adicionada ao mesmo conceito, permitindo assim saber de onde veio cada entrada e evitando duplicações. Este processo encontra-se na Figura 13.

```python
conceitos_unificados = {}


def adicionar_conceito(lista, origem):
    for item in lista:
        nome = item["Conceito"]
        if nome not in conceitos_unificados:
            conceitos_unificados[nome] = {"Conceito": nome, "Fontes": {origem: item}}
        else:
            conceitos_unificados[nome]["Fontes"][origem] = item


adicionar_conceito(ministerio, "ministerio")
adicionar_conceito(neologismos, "neologismo")
adicionar_conceito(covid, "covid")
```

Figura 13: Junção dos ficheiros *json*.

O resultado final é um dicionário onde cada conceito aparece uma única vez, com uma estrutura que regista as diferentes versões ou detalhes desse conceito conforme aparecem nas várias fontes. Esta estrutura facilita comparações entre os glossários e fornece uma base consolidada de conhecimento médico e terminológico.

```
{
  "Conceito": "acidose metilmalônica",
  "Fontes": {
    "neologismo": {
      "Conceito": "acidose metilmalônica",
      "Substantivo": "s.f.",
      "Traduções": "metilmalonic acidose [ing]; acidemia metilmalónica.[esp]",
      "Sigla": "Sem sigla associada",
      "Descrição": "Doença que afeta o ser humano, causada em função de uma deficiência metabólica genética e que cau
      "Citação": ""...a acidose metilmalônica, doença que causa retardo do desenvolvimento infantil. Em testes feitos
    }
  }
},
{
  "Conceito": "Acompanhamento do crescimento e desenvolvimento infantil",
  "Fontes": {
    "ministerio": {
      "Conceito": "Acompanhamento do crescimento e desenvolvimento infantil",
      "Categoria": "Atenção à Saúde",
      "Descrição": "Garantir a melhoria da qualidade de vida das crianças, permitindo pôr em evidência, precocemente,
    }
  }
},
```


Figura 14: *Json* com os termos e a sua respetiva informação.

5 Conclusão

O trabalho prático desenvolvido possibilitou a extração de informação relevante e a limpeza de elementos desnecessários a partir de diferentes documentos médicos, recorrendo à utilização de expressões regulares e técnicas de manipulação de texto. O principal objetivo — a construção de um ficheiro JSON contendo diversos termos médicos com as respetivas descrições, categorias e traduções — foi alcançado com sucesso.

No entanto, ao longo do desenvolvimento do projeto, surgiram algumas exceções durante o processamento dos documentos, o que implicou desafios adicionais no seu manuseamento.

Como trabalho futuro, destaca-se a necessidade de simplificação do código com vista à melhoria da eficiência e escalabilidade, permitindo assim o processamento de um maior número de documentos. Esta expansão contribuiria para o enriquecimento do ficheiro JSON com a junção de dois ficheiros menores deste formato, com a inclusão de mais termos e respetiva informação associada.

Por fim, sublinha-se a relevância do Processamento de Linguagem Natural (PLN) no contexto da Engenharia Biomédica, pelo seu contributo para o acesso mais eficiente e preciso à informação médica, potenciando a qualidade da tomada de decisão clínica e, consequentemente, a melhoria do serviço de saúde.