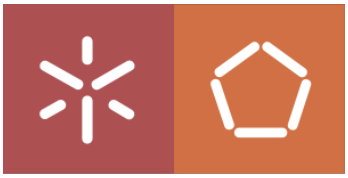

Universidade do Minho
Escola de Engenharia

Processamento de Linguagem Natural em Engenharia Biomédica

Enriquecimento e Plataforma de Visualização de um Dicionário Médico

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

Mestrado em Engenharia Biomédica - Informática Médica

15 de junho de 2025

Resumo

O presente projeto teve como objetivo a correção, enriquecimento e desenvolvimento de uma plataforma *web* para visualização de termos médicos. O enriquecimento do vocabulário foi realizado através de técnicas de *web scraping* aplicadas a fontes institucionais de referência.

Para análise semântica, foram implementados os modelos `Word2Vec` e `BERT` que permitiram classificar os termos médicos em categorias temáticas e identificar relações conceptuais entre eles.

O sistema desenvolvido inclui uma plataforma *web* interativa construída com o framework `Flask`, que oferece funcionalidades completas de pesquisa, visualização detalhada e gestão de conteúdos.

Os resultados demonstram a viabilidade da abordagem proposta e abrem perspetivas para aplicações mais avançadas no domínio do processamento de linguagem natural em contextos médicos.

Conteúdo

1 Introdução

O presente relatório descreve o processo de correção e enriquecimento de um conjunto de dados composto por termos médicos, extraídos a partir de documentos em formato PDF durante o trabalho prático anterior, bem como o desenvolvimento de uma plataforma *web* destinada à manipulação e visualização destes dados.

No decurso da análise da versão anterior do conjunto de dados, foram identificadas limitações relativas à integração do dados, em particular no que concerne à estruturação dos dados e à utilização de chaves em idiomas diferentes. Estas questões foram devidamente corrigidas, assegurando uma unificação mais consistente e coerente do conjunto de dados.

Para além da correção dos erros detetados, procedeu-se ao enriquecimento do conjunto de dados com informações suplementares obtidas através de *websites* institucionais, mediante a aplicação da técnica de *web scraping*.

Paralelamente, foi desenvolvida uma plataforma *web* interativa que permite a visualização e manipulação do conjunto de dados enriquecido. Esta ferramenta possibilita uma navegação eficiente, explorando as relações existentes entre conceitos por meio de *word embeddings*. Assim, é possível organizar e ordenar o conteúdo com base em diferentes atributos ou categorias, facilitando a análise e o acesso às informações relevantes. A plataforma permite ainda a atualização contínua do conjunto de dados, assegurando a sua manutenção e expansão futuras.

2 Correções da versão anterior

Antes de se avançar para o desenvolvimento do presente trabalho, foi necessário proceder a algumas correções relativas à versão anterior, conforme descrito a seguir.

2.1 Estrutura de dados

No ficheiro `populares.py`, o dicionário `glossario_dict` apresentava uma estrutura heterogénea, contendo simultaneamente valores únicos (do tipo *string*) e listas. Com o objetivo de uniformizar esta estrutura, procedeu-se à reformulação do processo de construção do dicionário: de forma a que todas as descrições associadas a um termo fossem armazenadas exclusivamente sob a forma de listas, mesmo nos casos em que apenas existe uma descrição. Esta padronização assegura a consistência estrutural dos dados, contribuindo para uma maior facilidade de manutenção e para um acesso mais simples, previsível e eficiente à informação armazenada.

Exemplo da estrutura anterior:

```
{
    (...)
    "anastomose": {
        "descricao": [
            "comunicação natural ou artificial entre dois vasos ou
               nervos",
            "abocamento"
        ]
    },
    "aborto": {
        "descricao": "abortamento, desmancho"
    },
    (...)
}
```

Exemplo da estrutura atual (após correção):

```
{
    (...)
    "anastomose": {
        "descricao": [
            "comunicação natural ou artificial entre dois vasos ou
               nervos",
            "abocamento"
        ]
    },
    "aborto": {
        "descricao": [
            "abortamento, desmancho"
        ]
    },
    (...)
}
```

2.2 Padronização das chaves

Na versão anterior do código, a junção dos dados dos ficheiros `populares.json`, `multilingue.json` e `neologismos.json` era feita de forma direta, com uma fusão básica dos campos. Contudo, esta abordagem apresentava limitações, nomeadamente na gestão de conceitos duplicados. Tal devia-se ao facto de as chaves em `multilingue.json` estarem originalmente em catalão, enquanto nos restantes ficheiros se encontravam em português. Consequentemente, termos equivalentes nas duas línguas não eram fundidos,

mas adicionados separadamente, dado que a comparação das chaves era feita diretamente, sem considerar as traduções.

Na versão corrigida, foi implementada uma etapa prévia de padronização das chaves no `multilingue.json`. Para cada termo, verifica-se se existe uma tradução para português (pt [PT] ou pt [BR]). Caso exista, a chave do termo é renomeada para o valor da tradução em português. A tradução original em português é removida do campo `traducoes` e é adicionado um novo campo `ca` (catalão), contendo a chave antiga e a respetiva categoria.

Com este processamento, as chaves do `multilingue.json` passam a corresponder aos termos em português, o que facilita a fusão com os restantes ficheiros, garantindo maior consistência e evitando redundâncias.

Versão anterior (chave em catalão)

```
"anòsmia": {
  "categoria": "n f",
  "traducoes": {
    "oc": {"tradução": "anòsmia",
       "categoria": "n f"},
    "eu": {"tradução": "anosmia",
       "categoria": "n"},
    "gl": {"tradução": "anosmia",
       "categoria": "n f"},
    "es": {"tradução": "anosmia",
       "categoria": "n f"},
    "en": {"tradução": "anosmia",
       "categoria": "n"},
    "fr": {"tradução": "anosmie",
       "categoria": "n f"},
    "pt": {"tradução": "anosmia",
       "categoria": "n f"},
    "nl": {
      "1": {"tradução": "anosmie",
          "categoria": "n"},
      "2": {"tradução": "
         geurverlies", "categoria
         ": "n"},
      "3": {"tradução": "
         reukverlies", "categoria
         ": "n"}
    }
  }
}
```

Versão atual (chave em português)

```
"anosmia": {
  "categoria": "n f",
  "traducoes": {
    "oc": {"tradução": "anòsmia",
       "categoria": "n f"},
    "eu": {"tradução": "anosmia",
       "categoria": "n"},
    "gl": {"tradução": "anosmia",
       "categoria": "n f"},
    "es": {"tradução": "anosmia",
       "categoria": "n f"},
    "en": {"tradução": "anosmia",
       "categoria": "n"},
    "fr": {"tradução": "anosmie",
       "categoria": "n f"},
    "nl": {
      "1": {"tradução": "anosmie",
          "categoria": "n"},
      "2": {"tradução": "
         geurverlies", "categoria
         ": "n"},
      "3": {"tradução": "
         reukverlies", "categoria
         ": "n"}
    },
    "ca": {"tradução": "anòsmia",
       "categoria": "n f"}
  }
}
```

Além disso, os termos presentes no ficheiro `multilingue.json` que não dispunham de tradução para português foram excluídos, uma vez que não podiam ser integrados de forma coerente na estrutura unificada.

2.3 Atualização do `multilingue.json`

Foram efetuadas alterações no processo de extração e estruturação de dados do ficheiro `multilingue.json`.

Primeiramente, foi identificado e removido um cabeçalho (`font == 16`). Além disso, o padrão de captura da descrição foi ajustado para também extrair o campo associado ao termo. Anteriormente, apenas a descrição textual era guardada; com a nova abordagem, esse campo — que indica, por exemplo, a área temática do termo — passa a ser guardado separadamente, conforme ilustrado abaixo:

```
{
  "key": {
    (...)
    "campo": "TRACTAMENT",
    (...)
  }
}
```

3 Enriquecimento dos dados

O *website* glossário de termos de saúde, mantido pelo CHLO – Centro Hospitalar de Lisboa Ocidental, foi utilizado para adicionar mais informações ao *dataset*.

Para extrair os conteúdos do glossário foi desenvolvido o *script* `scraper_min_saude.py`, que recorre às bibliotecas `requests` e `BeautifulSoup` e realiza um processo de *web scraping* estruturado em três etapas:

1. **Recolha de páginas**

   A função `get_pages()` percorre iterativamente todas as páginas do glossário, identificando o botão "seguinte" e armazenando os URLs correspondentes. Este passo garante a cobertura completa dos termos disponíveis, respeitando a estrutura de navegação do site.

2. **Extração de conteúdo**

   Para cada página, a função `get_glossary_data()` identifica os elementos HTML que contêm os termos e respetivas definições. Cada entrada é separada numa estrutura em dicionário contendo:

   - o termo principal,
   - a descrição,
   - e os sinónimos (quando indicados sob a etiqueta "`Sinónimos –`").

3. **Serialização para JSON**

   O resultado final é um ficheiro `min_saude.json`, no qual os dados recolhidos são organizados em formato JSON, com codificação UTF-8, respeitando os caracteres acentuados da língua portuguesa.

Após a adição de informação ao conjunto de dados, foi necessário ajustar o processo de fusão dos dados no ficheiro `merge.py`, de modo a assegurar a compatibilidade estrutural e semântica entre os dados previamente existentes e os novos registos. As principais modificações realizadas incluem:

- **Uniformização da estrutura das entradas**: Todas as entradas passaram a obedecer a uma estrutura completa e padronizada, incluindo campos como `descricao`, `categoria`, `traducoes`, `remis`, `codigos`, `Notas`, `enciclopedia` e `exemplo`. Para assegurar a consistência entre os dados já existentes e os novos, o processo de fusão foi modificado para normalizar todos os registos com essa mesma estrutura.
- **Inicialização de campos ausentes**: No momento da fusão, todos os campos que poderiam ser esperados a partir da inserção foram explicitamente inicializados em entradas que não os contivessem. Por exemplo, entradas provenientes de fontes como `popular.json` ou `min_saude.json` passaram a conter campos vazios para `remis`, `traducoes`, `codigos`, entre outros, evitando erros ou inconsistências no momento de leitura e apresentação posterior no sistema.

4 Classificação Semântica

4.1 Word2Vec

O ficheiro `train_class.py` implementa um modelo baseado na arquitetura `Word2Vec` com o objetivo de realizar a classificação semântica de termos médicos. O processo de implementação é estruturado em três etapas principais, conforme descrito a seguir.

1. **Tradução dos Campos**

   Inicialmente, procede-se à tradução dos campos semânticos originalmente em catalão para a língua portuguesa. Esta tarefa é realizada por meio de um dicionário de correspondência léxica, conforme exemplificado abaixo:

   ```
   trad_dict = {
       "anatomia": "anatomia",
       "farmacologia": "farmacologia",
       # ... outras traduções
   }
   ```

2. **Definição de Descrições**

   Posteriormente, cada campo semântico é enriquecido com um conjunto de palavras-chave associadas ao seu domínio específico:

   ```
   campos_descricoes = {
       "anatomia": ["órgão", "tecido", "estrutura", ...],
       "farmacologia": ["medicamento", "dosagem", ...],
       # ... outros campos
   }
   ```

3. **Treino do Modelo**

   Por fim, o modelo `Word2Vec` é treinado utilizando a biblioteca `Gensim`:

   ```
   from gensim.models import Word2Vec

   model = Word2Vec(
       sentences=corpus,
       vector_size=100,
       window=5,
       min_count=1,
       workers=4
   )
   ```

O resultado é armazenado no ficheiro `final_with_predicted_fields.json`.

4.2 BERT e Clustering de Termos

O notebook `clusters.ipynb` implementa uma metodologia baseada em *embeddings* gerados por modelos BERT, com o objetivo de realizar agrupamento semântico de termos médicos. O processo é composto por quatro etapas principais, descritas a seguir.

1. **Geração de Embeddings**

   Utiliza-se um modelo BERT pré-treinado específico para a língua portuguesa:

   ```
   from transformers import AutoTokenizer, AutoModel

   model_name = "neuralmind/bert-base-portuguese-cased"
   tokenizer = AutoTokenizer.from_pretrained(model_name)
   model = AutoModel.from_pretrained(model_name)
   ```

2. ***Clustering* Automático**

   Os *clusters* são criados com base nos *embeddings* de forma a selecionar o número ótimo:

   ```
   from sklearn.cluster import KMeans

   wcss = []
   for i in range(1, 11):
       kmeans = KMeans(n_clusters=i)
       kmeans.fit(embeddings)
       wcss.append(kmeans.inertia_)
   ```

3. **Visualização**

   É realizada uma redução dimensional de forma a permitir a visualização dos *clusters* criados:

   ```
   from sklearn.manifold import TSNE

   tsne = TSNE(n_components=2)
   embeddings_2d = tsne.fit_transform(embeddings)
   ```

4. **Nomeação dos Clusters**

   Os *clusters* são nomeados manualmente através da análise dos termos mais representativos dos *clusters*. A Tabela 1 apresenta exemplos representativos de termos agrupados em categorias semânticas.

Tabela 1: Exemplos de termos agrupados por categorias semânticas.

Termos	Categoria Semântica
bulbar, visual, atrial, venéreo, subconjuntival	Anatomia e Localizações Corporais
perfusão, irrigação, hiper-reflexia, reidratação, pessário	Fisiologia e Processos Corporais
hiperostose, hipervolemia, oncolítico, hipocondria, hipovolemia	Distúrbios Metabólicos e Sanguíneos

Os *clusters* nomeados são adicionados ao *dataset* final `final_final.json`, enriquecendo a análise semântica.

5 Plataforma *web*

A plataforma *web* foi desenvolvida com o *framework* `Flask`, recorrendo a *templates* `Jinja2` para a renderização dinâmica das páginas. A lógica principal da aplicação está implementada no ficheiro `app.py`, que centraliza a definição das rotas e o tratamento dos dados. A interface gráfica foi concebida com o apoio da biblioteca `Bootstrap`, permitindo uma apresentação visual consistente. As principais funcionalidades da plataforma incluem:

- **Página inicial**: Apresenta um campo de pesquisa, onde o utilizador pode procurar termos por designação, descrição ou correspondência exata. Estão disponíveis filtros adicionais, como idioma de tradução e navegação alfabética. Os resultados são exibidos em cartões interativos, que mostram a designação do termo, categoria gramatical, breve descrição e um botão para aceder à página de detalhes. Estes cartões encontram-se ordenados segundo a ordem alfabética da designação.
- **Detalhe do termo**: Exibe todas as informações associadas ao termo selecionado, incluindo a descrição, traduções em vários idiomas, exemplos de uso contextualizados, informações enciclopédicas relevantes, códigos normalizados (como CAS, ICD, entre outros), sinónimos, siglas e notas adicionais.
- **Gestão de termos**: O utilizador pode inserir novos termos, bem como editar ou excluir termos existentes. O formulário de adição/edição está dividido em várias secções para facilitar o preenchimento: informações básicas (designação, categoria, descrição), traduções (com campos dinâmicos para múltiplos idiomas), remissões (sinónimos, antónimos e siglas), códigos normalizados, exemplos de uso, informação enciclopédica e notas.

- **Interatividade**: A plataforma utiliza `JavaScript` para proporcionar uma experiência de utilização mais dinâmica e intuitiva, permitindo adicionar ou remover campos de tradução, sinónimos e siglas de forma interativa.

A Tabela 2 apresenta os diferentes *endpoints* disponibilizados pela plataforma, bem como os respetivos métodos HTTP, funções Python e descrições das funcionalidades associadas.

Tabela 2: Endpoints da plataforma.

Método	Endpoint	Função Python	Descrição
GET	`/`	`index()`	Exibe a página inicial com funcionalidades de procura, filtros e navegação alfabética
GET	`/search`	`search()`	Realiza a pesquisa de termos com base nos filtros definidos
GET	`/entry/<term>`	`entry_detail(term)`	Exibe a página de detalhes completos de um termo específico
GET	`/edit/<term>`	`edit_entry_form(term)`	Exibe o formulário para edição de um termo existente
GET	`/add`	`add_entry_form()`	Exibe o formulário para inserção de um novo termo
POST	`/add`	`add_entry_submit()`	Processa a submissão de um novo termo no dicionário ou a edição de um termo existente
POST	`/delete/<term>`	`delete_entry(term)`	Remove um termo existente

Além das funcionalidades implementadas diretamente nas rotas da aplicação, a plataforma recorre a um conjunto de funções auxiliares responsáveis por operações fundamentais de tratamento e estruturação dos dados:

- `load_data()`: Responsável por carregar os dados do dicionário a partir do ficheiro `final.json`, convertendo-os para um objeto estruturado, apto a ser manipulado no restante da aplicação.
- `normalize(text)`: Realiza a normalização de uma *string*, eliminando acentos e convertendo o texto para letras maiúsculas. Esta operação visa facilitar em comparações, pesquisas e ordenações alfabéticas.
- `extract_translation(lang_data)`: Extrai e organiza as traduções já existente de um termo para um idioma específico.
- `extract_remis_list(remis_dict, key)`: A partir da estrutura de remissões de um termo, esta função extrai listas de sinónimos, antónimos ou siglas, conforme a chave fornecida.

- `process_translation(translation_text, category)`: Processa novas traduções e categorias gramaticais inseridas pelo utilizador, permitindo múltiplas opções por idioma. Organiza os dados de forma sistemática para serem guardados ou apresentados posteriormente.

- `highlight_term(text, search_term, exact_match=False)`: Realça visualmente a ocorrência de um termo de pesquisa dentro de um texto, aplicando marcação em HTML (`<b>...</b>`) ao termo encontrado.

- `find_similar_terms(target_term, target_entry, data, max_terms=5)`: Identifica e devolve uma lista dos termos mais semelhantes ao termo-alvo, com base em critérios de similaridade. A função considera a coincidência de campos temáticos (`campo`), agrupamentos (`cluster`), sobreposição lexical entre os termos e semelhanças nas descrições (`descricao`).

6 Conclusões e trabalho futuro

O presente trabalho permitiu alcançar com sucesso os objetivos propostos, resultando num dicionário médico enriquecido e numa plataforma *web* funcional para a sua exploração. A correção das inconsistências da versão anterior, particularmente na estrutura dos dados e padronização das chaves, eliminou redundâncias e melhorou significativamente a qualidade do conjunto de dados. O enriquecimento semântico através de técnicas de *web scraping* e a aplicação de modelos de linguagem (`Word2Vec` e `BERT`) trouxeram uma nova dimensão analítica ao projeto, permitindo não apenas organizar os termos mas também compreender as suas relações conceptuais.

A plataforma *web* desenvolvida com `Flask` demonstrou ser uma solução eficaz para disponibilizar os dados de forma acessível e intuitiva. As funcionalidades implementadas - incluindo sistemas de pesquisa, visualização detalhada dos termos e ferramentas de gestão de conteúdo - criam um ecossistema completo para exploração do dicionário.

Como trabalho futuro, sugere-se expandir o âmbito do projeto em várias direções. Em primeiro lugar, a incorporação de novas fontes de dados especializadas permitirá aumentar a abrangência do vocabulário médico. Paralelamente, os modelos `Word2Vec` e `BERT` e a sua implementação poderão ser refinados, visando a melhoria da sua capacidade de representação semântica no domínio médico.