Universidade do Minho
Escola de Engenharia

Processamento de Linguagem Natural em Engenharia Biomédica: Plataforma Web com Dicionários Médicos

Miguel Borges Vale - PG54094
Tayan Peller - PG54451

Ano Letivo 2023/2024
June 12, 2024

Índice

1 Introdução **1**

2 Tratamento dos Datasets **2**
2.1 Alteração dos dados 2
2.2 Glossário 2
2.3 Minidicionário 3
2.4 Glossário Ministério Saúde 4

3 Enriquecimento dos Datasets **5**
3.1 Medicina 5
3.2 Web Scraping 6
3.3 União dos JSONs 7

4 Aplicação **8**
4.1 Front-End 8
4.1.1 edit_termo.html 8
4.1.2 novo_termo.html 8
4.1.3 termos_detalhes.html 8
4.1.4 index.html 8
4.2 BackEnd Flask 9
4.2.1 Página Inicial 9
4.2.2 Pesquisa 10
4.2.3 Novo Termo 11
4.2.4 Detalhe de Termos 11
4.2.5 Edição de Termos 12
4.2.6 Remoção de Termos 12

5 Funcionamento da Plataforma Web **14**

6 Conclusão **18**

Chapter 1

Introdução

O Processamento de Linguagem Natural (PLN) desempenha um papel crucial na extração, manipulação e compreensão de informações contidas em textos, especialmente na área biomédica, onde uma grande quantidade de dados é gerada diariamente. Nesse contexto, o Trabalho Prático 2 (TP2) de Processamento de Linguagem Natural em Engenharia Biomédica propõe o desenvolvimento de um sistema que permita a utilização de dados em formato JSON extraídos de dicionários médicos, como uma extensão do trabalho prático anterior.

O objetivo principal deste projeto é enriquecer o conjunto de dados inicial, adicionando novas informações provenientes de fontes externas, como websites (Web scraping) ou dicionários médicos, para aprimorar a qualidade e a abrangência dos dados. Além disso, uma análise dos conceitos extraídos anteriormente e suas possíveis relações será realizada, visando agrupá-los por domínios ou outras categorias relevantes.

Em seguida, o projeto visa a criação de uma aplicação que permita manipular e representar os dados contidos no dataset de forma adequada, explorando as relações presentes entre os conceitos. Essa não apenas fornecerá uma visualização clara dos dados, mas também facilitará a análise e a interpretação das informações extraídas. Para demonstrar a aplicação prática da ferramenta desenvolvida, será elaborado um caso de estudo, que será apresentado durante a exposição do projeto.

Chapter 2

Tratamento dos Datasets

No projeto anterior foram criados 3 diferentes ficheiros JSONS com o conteúdo de 3 diferentes glossários em pdf. Tal como foi pedido, foi decidido enriquecer o conjunto de termos. Para tal, foram adicionados 2 novos glossários de termos médicos, um derivado do ficheiro medicina.pdf, que contém a tradução de aproximadamente 5000 termos médicos diferentes em diferentes categorias que já pertencia ao grupo de glossários disponibilizados pelos docentes. O outro glossário adicionado pertence à página web Boa saúde online, que reúne dezenas de termos médicos comuns de A a Z e apresenta a definição para cada um destes.

2.1 Alteração dos dados

Para a união dos diferentes glossários e sua posterior utilização de modo simples e compacta, foi decidido utilizar a mesma estrutura de dados para todos estes. Esta é definida como uma lista de dicionários, cada um com 4 chaves referentes ao termo, sua(s) categoria(s) numa lista, sua definição e a sua tradução em inglês, respetivamente. Todos os valores dos dicionários estão no formato de string e no caso de não existir um valor num dos campos, este será igual a "N/A". A estrutura do JSON final está exemplificada no exemplo seguinte:

[{"termo": "Artéria vertebral",
"categoria": ["Anatomia","Fisiologia"],
"descrição": "Primeira porção das artérias subclávias.",
"tradução inglês": "vertebral artery"}]

2.2 Glossário

Em termos de alteração de dados dos ficheiros antigos, no caso do ficheiro glossário.pdf foi feita uma simples alteração no modo como estes foram guardados, para corroborarem com a estrutura decidida.

2.3 Minidicionário

No caso do ficheiro minidicionário.pdf, o processo de remoção de informação do ficheiro XML para a construção do dicionário foi completamente alterada. Primeiro foi feita a remoção de palavras do topo de página, números de página e letras referentes aos termos à direita a partir das suas posições na página ou modo como são definidos no XML. Em seguida foi removida informação desnecessária como as tags de página, fontspec, imagem e texto. Depois, as tags de bold à volta dos termos foram substituídas por símbolos de início e fim destes para após isto ser possível unir continuação de termos em diferentes linhas, os espaços a mais, e os símbolos de separação de linhas foram removidas. Finalmente foram resolvidos dois casos específicos que tinham uma má formatação. Foram criadas listas para os termos e para as definições encontrando todas as strings entre o símbolo de início de um termo e final deste e os termos em inglês pelas strings que se encontram entre os símbolos de final de um termo e o início deste. Iterando-se por estas duas listas para formar o dicionário final. Uma última alteração feita neste dicionário foi a adição de múltiplas traduções caso o termo em si tenha vários sinónimos separando-os entre si para ao mesma tradução.

2.4 Glossário Ministério Saúde

O ficheiro glossário_ministério_saúde.pdf foi o que sofreu o maior número de alterações para a resolução de problemas que este ainda continha, estes eram a união das categorias com o início de algumas descrições, a mesma categoria separada por duas linhas e a má formatação de algumas categorias que não estavam devidamente separadas. Para estas foi simplesmente feita uma substituição dessas linhas pela correta separação de um parágrafo entre as duas. Para os casos em que a mesma categoria estava dividida em duas linhas foi criada uma lista de todas as categorias que existem e outra lista de todas as linhas com o símbolo "#" atrás delas e que em si não correspondiam a nenhuma das categorias, que incluía todas as categorias separadas por novas linhas e algumas linhas de descrições. Esta lista foi percorrida, comparando em todos os momentos, a posição atual e a anterior, assim para o caso em que as duas linhas unidas corresponde a uma categoria, estas eram unidas para formarem a correta formatação dessa categoria. Para o resto das strings dentro da lista, estas correspondiam a descrições e, portanto, os seus símbolos precedentes foram alterados para o símbolo correto. Depois desta reparação, o mesmo processo de separar o os termos, categorias e descrições foi feita, visto que, não haveria problemas com esta, mas adicionou-se uma nova capacidade para separar as categorias em casos singulares em vez de as colocar juntas na mesma string sem uma separação clara, para tal, foi iterado pela lista de categorias e substituído o símbolo de início de categorias por um ponto de exclamação que depois foi utilizado para fazer o split destas. A componente das siglas deste ficheiro sofreram a mesma alteração de estrutura de dicionário para a escolhida.

Chapter 3

Enriquecimento dos Datasets

3.1 Medicina

Este ficheiro foi um dos dois novos que foram adicionados aos que já tinham sido utilizados. Ele apresentava um conjunto de aproximadamente 5000 traduções de múltiplos termos médicos das suas categorias em várias línguas. Por razões de simplicidade e consistência foi decidido apenas juntar as traduções em inglês. Os termos e as categorias deste ficheiro estavam escritos em espanhol, logo foi desde o início necessária a criação de uma lista das categorias que existiam e uma lista com as traduções das mesmas. Para guardar os termos em português foi primeiramente procurada pela presença do símbolo "pt" que determinada a presença da tradução do termo em português e para cada um destes foi posteriormente removido os símbolos de que tipo de português a tradução se referia. A mesma busca foi feita para a busca dos termos em inglês. Para o resgate das categorias foi procurada pelo texto contido nas linhas com um valor de font igual a 21, que eram exclusivamente utilizadas para as categorias. Para além destas foram introduzidas manualmente na lista, nas posições corretas os termos cujo tamanho do font, por uma razão ou por outra não era igual a 21. Assim é possível saber que temos todas as categorias, visto que, o número de categorias nas listas destas é igual ao número de traduções em português e em inglês, o que também é possível descobrir, pois no ficheiro é apresentado o número da iteração do termo, sendo que o último tinha o mesmo valor numérico que o tamanho das 3 listas. Para cada categoria, se estas tinham 2 ou mais espaços em branco seriam separadas numa lista de n categorias, dependendo da frequência deste padrão para conseguir ter todas as categorias diferenciadas tal como nos dicionários anteriores. Finalmente foi feita a substituição das categorias de espanhol para português, sendo que foi necessário substituir um caso especial que apenas tinhas uma palavra em vez da categoria toda. Por fim, foi construído o dicionário como nos outros casos.

3.2 Web Scraping

Como nos foi pedida a adição de pelo menos um dicionário de termos médicos online foi utilizado o site Boa Saúde para a construção do dicionário, este apresenta um conjunto de páginas dedicadas aos termos existentes divididos por letra inicial. Cada secção poderá ter ou não mais do que uma página web com termos havendo uma barra de páginas existentes por baixo da tabela de definições e as descrições destes termos em si são acedidas pelo link existente no próprio termo. Para obter os termos e descrições foram criadas duas listas para estes. Para procurar em todas as páginas iniciais dos termos iniciados por uma letra foi criado um ciclo que percorre o alfabeto e com uma fstring foi feito o request de cada página web, dentro desta foi instanciado o parser e adicionada à variável paginas a componente de paginacao definida no html da página, uma lista não estruturada que contêm os links de todas as páginas com termos iniciados por aquela letra, caso esta existisse seria criada uma lista de todas as referências dentro da tag ¡a¿ incluída na lista não estruturada de paginação, se o texto apresentado por esta tag fosse um número. Para cada link de página existente para as definições iniciadas por essa letra, será feito de modo igual ao caso de não existir paginação, que é a busca da lista não estruturada com o nome de classe listagem, que contém todos os termos dessa página e irá criar uma lista de todas as referências existentes nas tags ¡a¿ desses termos. Para cada link dos termos foi adicionado à lista o termo que está incluído no texto da tag ¡h2¿ incluída na div com o nome de id col-center e para a descrição esta foi adicionada à sua lista após ter sido buscada no texto incluído na próxima tag ¡p¿ depois da tag ¡h2¿. Após percorrer todas as páginas do website com todas as letras do abecedário, indo por todas as páginas que existiam para cada uma desta, e guardando as informações de termos e descrições incluídas no link dos nomes dos termos, estes foram adicionados ao dicionário final deste ficheiro tal como os outros casos.

3.3 União dos JSONs

Para a criação do JSON final que incluí todos os outros JSONs unidos foram criadas duas listas, uma para o JSON final e outra para os termos presentes nessa lista. Foram percorridos todos os 5 ficheiros JSON e a cada um o seu conteúdo foi guardado numa variável que foi usada para percorrer todos os termos presentes nela, se o valor da chave termo não estivesse presente na lista termos então seria adicionada e esse dicionário seria adicionado à lista final, assim foi possível criar uma lista de dicionários sem termos repetidos. Posteriormente à criação da lista de dicionários final, esta foi percorrida mais uma vez utilizando a biblioteca deep_translator para oferecer uma tradução em inglês a todos os termos que não contém uma, ou seja, o valor existente na chave "tradução inglês" seja igual a "N/A". Esta lista final foi guardada num ficheiro JSON global denominado "glossário.json".

Chapter 4

Aplicação

4.1 Front-End

4.1.1 edit_termo.html

A página de edição de termos apresenta um título com o nome do termo escolhido e por baixo deste um formulário com inputs para o utilizador conseguir enviar as informações do termo que deseja alterar, por baixo deste formulário existe um redirecionamento para a página inicial.

4.1.2 novo_termo.html

A página de adição de um novo termo à parecença da página de editar termos também apresenta um formulário com os mesmos inputs para enviar as informações do novo termo que deseja adicionar, todos os inputs são obrigatórios e tal como a página de editar termos, esta também contém um botão para redirecionar o utilizador para a página inicial.

4.1.3 termos_detalhes.html

A página sobre os detalhes do termo apresenta estes de forma dinâmica por baixo de cada caraterística, um *form* é utilizado para remover o termo em si do conjunto de termos médicos guardados no ficheiro *JSON* na forma de um botão. Dois outros botões são utilizados para redirecionar o utilizador ou para a página inicial ou para a página de edição de termos enviando para essa as informações guardadas na página.

4.1.4 index.html

A página *index* apresenta o título Termos Médicos, uma *tag* de *hiperlink* para a página de adicionar um novo termo no canto superior direito, um formulário com um input para o utilizador adicionar termos, descrições ou traduções a pesquisar, por baixo deste estão definidos vários botões rádio para a pessoa escolher que tipo de dados ela quer associar a sua pesquisa anterior, uma *tag select* é utilizada para a pessoa escolher se quiser dentro de que tipo de categoria pesquisar e por fmi um botão de *submit* dentro do

formulário. Por baixo deste foi criada uma condição if, em que se existirem dados a exibir será criada dinamicamente uma tabela com o conjunto de todos os resultados possíveis pela pesquisa em que é incluído o termo, a descrição as categorias do termo e a sua tradução em inglês. Por baixo deste está definido o *script* que garantirá a criação de uma tabela *JQuery* a partir da tabela HTML criada atras, de modo a apenas a construir depois da tabela HTML estiver completamente processada, transformando a tabela *resultsTable* numa *DataTable*.

4.2 BackEnd Flask

O servidor Flask é iniciado com a abertura do JSON previamente criado que contém os termos e seus outros atributos.

```
with open('glossario.json', 'r', encoding='utf-8') as f:
    terms = json.load(f)

unique_categories = sorted(set(cat for term in terms for cat in term['categoria']))
```

Figure 4.1: Armazenamento do JSON

4.2.1 Página Inicial

A função `index()` foi criada para ser ativada quando um usuário acessa a rota padrão `'/'`. Dentro dessa função, a variável `terms` contém todos os termos do glossário, carregados a partir do arquivo JSON. A variável `unique_categories` armazena todas as categorias únicas presentes nos termos. O objetivo principal desta função é renderizar o template `index.html`, que possui botões e formulários desenvolvidos para permitir pesquisa com diferentes parâmetros.

```
@app.route('/')
def index():
    return render_template('index.html', terms=terms, unique_categories=unique_categories)
```

Figure 4.2: Função index

4.2.2 Pesquisa

Quando a função `search()` é chamada, é porque o formulário de pesquisa, presente no template inicial, foi submetido via método POST. Ela recebe os parâmetros de pesquisa, incluindo o tipo de pesquisa (termo, descrição, tradução) e a categoria selecionada.

Com base nos parâmetros recebidos, a função inicia a filtragem dos termos correspondentes, por meio da criação de listas em compreensão comparando os dados de pesquisa recebidos do formulário com os dados presentes na variável do glossário. Se o tipo de pesquisa for "termo", a função cria uma lista de termos filtrados onde o termo de busca é encontrado no campo "termo". Se o tipo de pesquisa for "descrição", ela cria uma lista de termos onde o termo de busca é encontrado no campo "descrição". Se o tipo de pesquisa for "tradução", ela cria uma lista de termos onde o termo de busca é encontrado no campo "tradução inglês". Se o tipo de pesquisa for "todos", a função cria uma lista de termos onde o termo de busca é encontrado em qualquer um dos campos mencionados acima.

Após a filtragem com base no tipo de pesquisa, a função verifica se uma categoria específica foi selecionada. Se uma categoria foi selecionada, a função filtra ainda mais os termos correspondentes para incluir apenas aqueles que pertencem à categoria especificada.

Com os termos filtrados disponíveis, a função retorna o template index.html com os resultados da pesquisa. Os resultados são passados para o template como um parâmetro, permitindo que sejam exibidos ao usuário.

```python
@app.route('/search', methods=['POST'])
def search():
    query = request.form['query'].lower()
    search_type = request.form['search_type']
    selected_category = request.form['category']

    if search_type == 'term':
        results = [term for term in terms if query in term['termo'].lower()]
    elif search_type == 'description':
        results = [term for term in terms if query in term['descrição'].lower()]
    elif search_type == 'translation':
        results = [term for term in terms if query in term.get('tradução inglês', '').lower()]
    else:
        results = [term for term in terms if query in term['termo'].lower() or query in term['descrição'].lower()
                   or query in term.get('tradução inglês', '').lower()]

    if selected_category != 'all':
        results = [term for term in results if selected_category in term['categoria']]

    return render_template('index.html', terms=terms, unique_categories=unique_categories, results=results)
```

Figure 4.3: Função search

4.2.3 Novo Termo

A função `novo_termo()` é responsável por lidar com a rota `'/novo_termo'`. O método GET é usado para renderizar inicialmente o template que possui o formulário para adição de um novo termo ao glossário. Quando o formulário é submetido, o método da requisição é POST, permitindo que os detalhes fornecidos pelo usuário sejam utilizados para adicionar um novo termo ao glossário. Os dados do formulário são acessados através do objeto `request.form`. Após a adição do novo termo, a função redireciona o usuário de volta para a página inicial.

```
@app.route('/novo_termo', methods=['GET', 'POST'])
def novo_termo():
    if request.method == 'POST':
        new_term = {
            'termo': request.form['termo'],
            'descrição': request.form['descrição'],
            'tradução inglês': request.form.get('tradução inglês'),
            'categoria': request.form['categoria'].split(',')
        }
        terms.append(new_term)
        with open('glossario.json', 'w', encoding='utf-8') as f:
            json.dump(terms, f, ensure_ascii=False, indent=4)
        return redirect(url_for('index'))
    return render_template('novo_termo.html')
```

Figure 4.4: Função novo_termo

4.2.4 Detalhe de Termos

A função `termos_detalhes()` é acionada quando um usuário acessa a rota `'/term/string:termo'`. Ela recebe o termo como parâmetro na URL dinâmica e busca no glossário o termo correspondente. Os detalhes desse termo são então passados para o template `termos_detalhes.html` que é renderizado, mostrando os detalhes do termo.

```python
@app.route('/term/<string:termo>')
def termos_detalhes(termo):
    term_details = next((t for t in terms if t['termo'].lower() == termo.lower()), None)
    return render_template('termos_detalhes.html', term=term_details)
```

Figure 4.5: Função termos_detalhes

4.2.5 Edição de Termos

A função `edit_term()` é acessada tanto por requisições GET quanto POST e é responsável por lidar com a rota `'/term/string:termo/edit'`. Inicialmente a requisição do tipo GET ocorre para renderizar o template `edit_template.thml` que contém o formulário de edição preenchido com os detalhes do termo. Desta forma, o utilizador pode então modificar os detalhes do termo conforme o desejado. Quando o formulário é submetido, o método da requisição é POST e os dados são acessados através do objeto `request.form`. Isso permite a atualização dos detalhes do termo com base nos dados fornecidos. Após a edição, os detalhes atualizados do termo são salvos no arquivo JSON e o usuário é redirecionado de volta para a página de detalhes do termo.

```python
@app.route('/term/<string:termo>/edit', methods=['GET', 'POST'])
def edit_term(termo):
    term_details = next((t for t in terms if t['termo'].lower() == termo.lower()), None)
    if request.method == 'POST':
        term_details['descrição'] = request.form['descrição']
        term_details['tradução inglês'] = request.form.get('tradução inglês')
        term_details['categoria'] = request.form['categoria'].split(',')
        with open('glossario.json', 'w', encoding='utf-8') as f:
            json.dump(terms, f, ensure_ascii=False, indent=4)
        return redirect(url_for('termos_detalhes', termo=termo))
    return render_template('edit_termo.html', term=term_details)
```

Figure 4.6: Função edit_termo

4.2.6 Remoção de Termos

Por fim, a função `remove_term()` é ativada quando um usuário acessa a rota `'/remove/string:termo'` e submete um formulário via método POST para remover um termo específico do glossário. Esta função recebe o parâmetro `termo`, que é o termo que o usuário deseja remover, na URL dinâmica. No início da função, a variável global `terms` é modificada utilizando uma compreensão de lista para filtrar todos os termos que não correspondem ao termo

a ser removido. Isso é feito através da comparação dos valores dos termos com o valor do termo a ser removido. Em seguida, os termos atualizados são escritos de volta no arquivo JSON `glossario.json`. Isso garante que o glossário reflete a remoção do termo selecionado. Por fim, a função redireciona o usuário de volta para a página inicial (`index`) usando a função `redirect(url_for('index'))`. O objetivo principal dessa função é permitir que os usuários removam termos indesejados do glossário de forma interativa e intuitiva. Essa função segue a mesma estrutura das outras funções, começando com a manipulação dos dados recebidos, seguida pela atualização do estado interno (remoção do termo), e termina com o redirecionamento do usuário de volta para a página inicial para refletir as mudanças feitas no glossário.

```python
@app.route('/remove/<string:termo>', methods=['POST'])
def remove_term(termo):
    global terms
    terms = [term for term in terms if term['termo'] != termo]

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

    return redirect(url_for('index'))
```

Figure 4.7: Função remove_termo

Chapter 5

Funcionamento da Plataforma Web

A Plataforma Web permite aos utilizadores procurarem por termos médicos, facilitando e otimizando suas pesquisas no glossário. A pesquisa por termos permite que o termo que desejado seja inserido na barra de busca e a plataforma retornará resultados baseados no termo inserido.

Os filtros de pesquisa são ferramentas que permitem um refinamento ainda maior das buscas. Existem 4 tipos de filtros, para termos, descrição, tradução e para todos. Nos filtros específicos apenas há resultados para a existência desses itens no parâmetro escolhido. Ou seja, o filtro de termos limita a busca nos termos, o filtro de descrição limita a pesquisa nas descrições e o filtro de traduções também limita a pesquisa no parâmetro das traduções. Ainda é realizar uma pesquisa abrangente que considera tanto os termos, as descrições quanto as traduções utilizando o filtro todos.

A seleção da categoria é um recurso importante e prático onde é disponibilizado um menu suspenso textitdropbox de categorias, contendo todas as categorias existentes no glossário, para refinar sua busca.

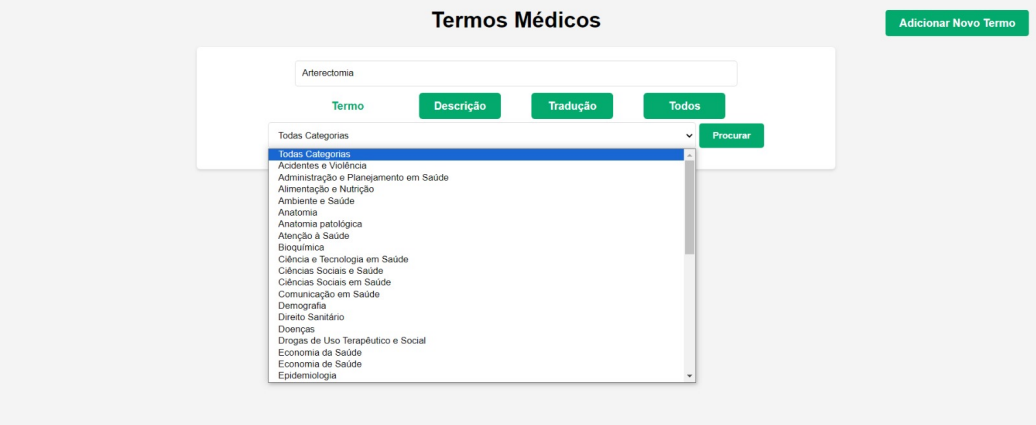

Figure 5.1: Página Inicial

Figure 5.2: Exemplo de pesquisa por termos

Na página inicial também é possível selecionar a opção de adicionar um novo termo. Neste é apresentado um novo *template* com um formulário onde deve ser introduzido informações sobre o termo. Todos os campos são obrigatórios, assim forçando que o glossário mantenha uma estrutura organizada. Além disso, um termo pode ter mais do que uma categoria, desde que seja separado por vírgula.

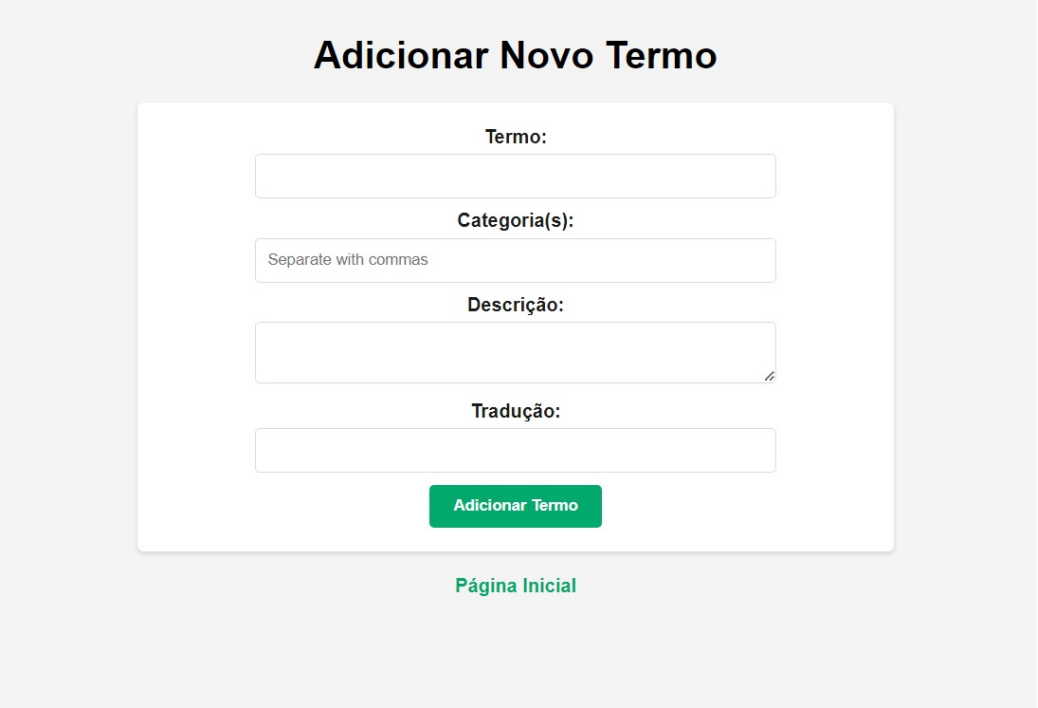

Figure 5.3: Página de adicionar novos termos

Voltando aos resultados de uma pesquisa na página inicial, é possível clicar no termo e assim o utilizador é redirecionado para um novo *template* onde são visualizados os detalhes do termo.

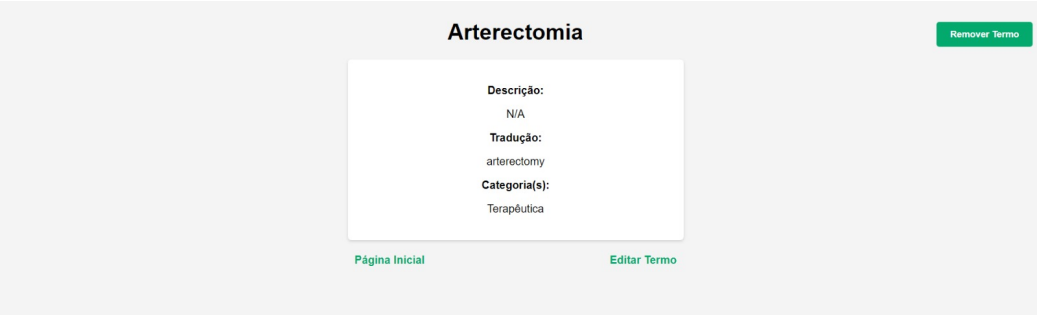

Figure 5.4: Página para ver termo

Neste *template* de visualização do termo existe um botão que permite a edição do mesmo. É renderizado um novo *template* com um formulário parecido com os anteriores mas contendo as informações atuais do termo, o

que permite perceber detalhes errados ou sem conteúdo. Isso garante que os dados serão mudados tendo o conhecimento do que já lá estava.

Figure 5.5: Página para editar termos

Chapter 6

Conclusão

Através deste projeto, foi possível explorar e aprimorar os dados inicialmente extraídos de PDFs médicos, enriquecendo-os com informações adicionais obtidas de fontes externas, como websites e dicionários médicos.

A análise dos conceitos previamente extraídos e suas relações possibilitou uma melhor compreensão e organização dos dados, permitindo agrupá-los por domínios ou categorias relevantes. Essa análise não apenas ampliou o conjunto de dados, mas também facilitou a identificação de padrões e insights valiosos para aplicações futuras.

A criação de uma ferramenta para manipular e representar os dados de forma adequada foi fundamental para explorar as relações presentes entre os conceitos, proporcionando uma visualização clara e acessível das informações extraídas. Essa ferramenta não apenas simplifica a análise dos dados, mas também abre novas oportunidades para a utilização prática dessas informações na área biomédica.

Por fim, o caso de estudo desenvolvido durante o projeto demonstrou a aplicação prática da ferramenta desenvolvida, destacando sua utilidade e relevância no contexto biomédico. Este projeto não apenas contribui para o avanço do campo de Processamento de Linguagem Natural, mas também oferece benefícios tangíveis para profissionais da área da saúde, facilitando a interpretação e o uso de dados clínicos em suas práticas diárias e pesquisas.