Processamento de Linguagem Natural em Eng. Biomédica

Teste, 27-05-2024

1 Python

Considere o seguinte texto:

```
texto = """Valeu a pena? Tudo vale a pena
Se a alma não é pequena.
Quem quer passar além do Bojador
Tem que passar além da dor ..."""
```

Especifique as seguintes estruturas de dados em compreensão.

1. Lista de tokens do texto, **sem stop words**.

```
stopwords = ['a', 'há', 'da', 'já', 'num', 'de', 'do', 'e', 'um', ...]

tokens = [ ... ]
```

2. Lista formada pelas palavras do texto iniciadas por maiúscula:

```
palavras_capital = [ ... ]
```

3. Lista formada pelos primeiros carateres de cada palavra.

```
first_characters= [ ... ]
```

2 Word Embeddings

1. Crie a função `CreateWord2Vec(text, output_path)` que recebe um texto e uma diretoria como input e utiliza esse texto para gerar um modelo Word2Vec, guardando-o na diretoria recebida no input. Deve usar a biblioteca Gensim para gerar o modelo.
2. Crie uma função que recebe como input um modelo Word2Vec, uma frase (query) e um conjunto de frases e calcula qual a frase desse conjunto é mais similar da frase query.

Assuma que as seguintes funções já se encontram definidas:

- `cosine(vec1, vec2)`: recebe dois vetores como input e devolve o cosseno desses vetores.
- `mean_vec([vec])`: recebe uma lista de vetores e devolve o vetor resultante da média dos vetores da lista.

3 Jinja2

A seguinte estrutura de dados representa uma lista de artigos científicos. Use a biblioteca Jinja2 para gerar dinamicamente uma página páginas HTML que representem todos os campos desta estrutura de forma adequada. No caso de haver valores nulos, a página HTML deve mostrar o texto "N/A".

```
papers = [
   {
      "título": "O Impacto das Mudanças Climáticas na Biodiversidade",
      "autores": ["João Silva", "Maria Santos"],
      "revista": "Natureza", # Este campo pode ter valor null
      "data_publicação": "2023-07-15",
      "doi": "10.1038/s41586-023-0456-2",
      "resumo": "As mudanças climáticas representam uma ameaça significativa para a biodiversidade global...",
      "palavras-chave": ["mudanças climáticas", "biodiversidade", "ecologia"],
   },
]
```

4 Web Scrapping

Implemente um programa em Python que dado o URL de um plano de estudos de um curso, extraia o nome, descrição e créditos de todas as Unidades curriculares desse curso.

Tome como exemplo as páginas HTML apresentadas abaixo.

Exemplo de página do plano de estudos: (https://www.uminho.pt/cursos/engbiomedica)

```
<div class="disciplina-table">
  <table>
    <tr>
      <th>Semestre</th>
      <th>Disciplina</th>
      <th>ECTS</th>
    </tr>
    <tr>
      <td>1º Semestre</td>
      <td><a href="https://www.uminho.pt/cursos/engbiomedica/biomecanica"> Biomecânica </a></td>
      <td>10</td>
    </tr>
    <tr>
      <td>2º Semestre</td>
      <td><a href="https://www.uminho.pt/cursos/engbiomedica/eletronicabiomedica">Eletrônica Biomédica</a></td>
      <td>8</td>
    </tr>

    (...)
  </table>
</div>
```

Exemplo de página de uma disciplina (https://www.uminho.pt/cursos/engbiomedica/biomecanica)

```
<div class ="container">
    Ano letivo 2022-2023
</div>
<div class="container">
    <div class="disciplina-title">
        <h1> Engenharia Biomédica </h1>
        <h2> <b> Biomecânica </b> </h2>
    </div>
    <div class="disciplina-desc">
        <p>
            A disciplina de Biomecânica estuda as forças e os movimentos do corpo humano (...)
        <p>
    </div>
</div>
```

O resultado deve ser gravado num ficheiro JSON.

Exemplo de saída esperada:

```
{
   "Biomecânica": { "ECTS": "10", "desc":"A disciplina de Biomecânica estuda ..."} ,
   "Eletrônica Biomédica": {"ECTS": "8", "desc":"A disciplina de ..."},
    (...)
}
```

5 Expressões regulares

1. Escreva uma função Python que transforme os Elementos XML vazios escritos com a notação `<abc/>` em XML com notação `<abc></abc>` mantendo tudo o resto inalterado.
2. Enriqueça a função anterior de modo a poder incluir também atributos:

`<abc id="33"/>` → `<abc id="33"></abc>`

6 Flask

Suponha que dispomos de um pasta `proverbios` com vários ficheiros de provérbios, um para cada país, (exemplo: `proverbios/PT.txt`, `proverbio/EN.txt`, ...). Cada ficheiro contêm um provérbio por linha.

Pretendemos criar uma aplicação Flask `proverbio` que permita consultá-los.

1. Crie uma rota Flask `procura/<country>/<pattern>` que por exemplo responda ao URL

`http://..../proverbios/procura/PT/cavalo`

com todas as linhas de `provérbios/PT.txt` que contêm `cavalo`.

2. Crie uma rota Flask `adiciona`, que permita adicionar um novo provérbio ao ficheiro `proverbios/<country>.txt` através do seguinte formulário HTML:

```
<form action="/adiciona" method="post">
    <label for="country">País:</label>
    <input type="text" id="country" name="country" required>
    <label for="proverb">Provérbio:</label>
    <input type="text" id="proverb" name="proverb" required>
    <input type="submit" value="Add Proverb">
</form>
```

Esta rota deve ainda garantir que não existem provérbios repetidos. Tenha em conta potenciais erros associados a esta operação.