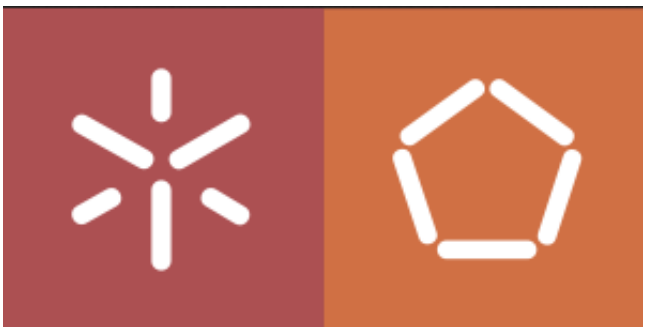


Universidade do Minho

Mestrado em Engenharia Informática

Scripting no Processamento de Linguagem Natural

Beautiful Soup - Python

Grupo:

João Vieira	A76516
Manuel Monteiro	A74036
Miguel Dias	PG41089

4 de Maio de 2020

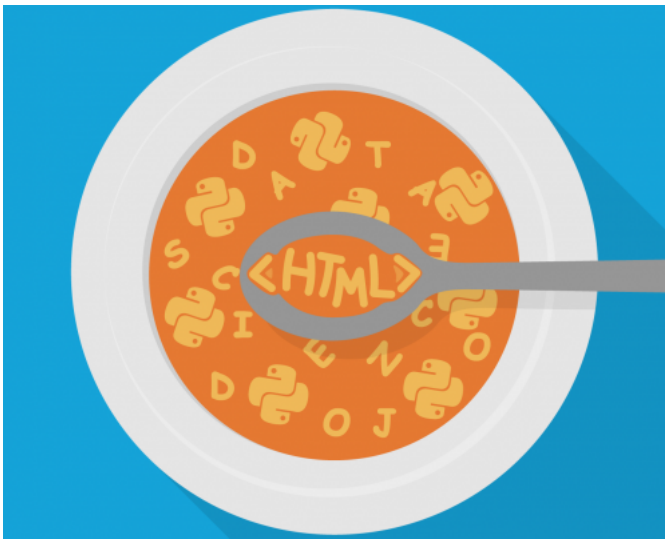
Conteúdo

1 Introdução

Este relatório é relativo a um trabalho realizado para a unidade curricular de *Scripting* no Processamento de Linguagem Natural. Foi escolhida uma ferramenta de uma lista dada pelo professor, sendo escolhida por nós o *Beautiful Soup*. Esta ferramenta de *Python* serve para extrair dados de ficheiros *HTML/XML*. Com ele podemos navegar e pesquisar e modificar a árvore de *parsing* do documento.

2 Objetivo do Projeto

O objetivo deste projeto é utilizar a ferramenta (*Beautiful Soup*) para criar um *script*. Como é sabido, temos outro projeto relativo à unidade curricular do perfil de mestrado. Como tal, decidimos utilizar este projeto para ser utilizado nesse projeto, portanto foi elaborado um *script* para ser utilizado na povoação/atualização dos dados (preço, promoções...) de jogos da plataforma *Steam*.



3 Resolução

Para tal, fazemos um pedido a uma *webpage* que tem uma tabela com todas as promoções existentes na plataforma (Steam DB), a partir do qual utilizamos a ferramenta para obtermos o preço e as promoções dos mesmos.

```html
<tr class="app appimg" data-appid="1600" data-cache="1587490676">
    <td data-sort="30">
        <a target="_blank" rel="noopener"
        ↪ href="https://store.steampowered.com/app/1600/"
        ↪ class="info-icon"
            title="Store"></a>
    </td>
    <td class="applogo">
        <a target="_blank" rel="noopener" href="/app/1600/"
        ↪ tabindex="-1" aria-hidden="true">
            <img src="/static/img/applogo.svg" alt="">
        </a>
    </td>
    <td>
        <a target="_blank" rel="noopener" class="b"
        ↪ href="/app/1600/">Dangerous Waters</a>
        <span class="subinfo">
            <span class="highest-discount">lowest price is
            ↪ <b>1,49</b> at -90%</span>
        </span>
    </td>
    <td class="" data-sort="80">-80%</td>
    <td data-sort="559">5,59</td>
    <td data-sort="70.43">70.43%</td>
    <td data-sort="1588611600" class="timeago"></td>
    <td data-sort="1587403211" class="timeago"></td>
    <td data-sort="1139270400" class="timeago"></td>
</tr>
```

*1: **Excerto HTML da tabela***

```python
http = urllib3.PoolManager()
r = http.request("GET","https://steamdb.info/sales/")

soup = BeautifulSoup(r.data, "lxml")

genRe = r'Genre:[\s]+([\w ,]+)'
devRe = r'Developer:[\s]+([\w ,]+)'
pubRe = r'Publisher:[\s]+([\w ,]+)'
rdRe = r'Release Date:[\s]+([\w ,\d]+)'

```

```
f = open('dataset.json','w');
f.write("[")
counter = 0

page = soup.tbody.find_all("tr")
for row in page:

    counter = counter+1
    print(counter)

    td_list = row.find_all("td")
    game_page = td_list[0].a["href"]

    name = td_list[2].a.text

    discount = td_list[3].text
    price = td_list[4].text
    rating = td_list[5].text

    time_now = dt.datetime.now()
    sale_end =
    ↪ dt.datetime.fromtimestamp(float(td_list[6]["data-sort"]))
    ↪ - time_now
    sale_start = time_now -
    ↪ dt.datetime.fromtimestamp(float(td_list[7]["data-sort"]))
```

2: ***Script que extrai informação da tabela***

Além disso para cada jogo encontrado fazemos um outro pedido para obter mais informações sobre o jogo tais como a sua descrição, género, data de lançamento entre outros. Este último pedido é efetuado na página da loja de cada jogo baseado na (Loja da Steam).

```
<meta name="Description" content="Counter-Strike: Global Offensive
 ↪  (CS: GO)
expands upon the team-based action gameplay that it pioneered when
 ↪  it was launched 19 years ago.
CS: GO features new maps, characters, weapons, and game modes,
and delivers updated versions of the classic CS content (de_dust2,
 ↪  etc.).">

<div class="details_block">
    <b>Título:</b> Counter-Strike: Global Offensive<br>
    <b>G&eacute;nero:</b>
    <a href="...">Ação</a>,
    <a href="...">
        Grátis para Jogar</a>
    <br>
    <div class="dev_row">
        <b>Desenvolvedor:</b>
        <a href="...">
            Valve
        </a>,
        <a href="...">
            Hidden Path Entertainment</a>
    </div>
    <div class="dev_row">
        <b>Editora:</b>
        <a href="...">Valve</a>
    </div>
    <b>Data de lançamento:</b> 21 Ago, 2012<br>
</div>
```

3: ***Informações do jogo***

```
# PÁGINA DO JOGO
    g = requests.get(game_page)

    if(g.status_code == 200):

        game = BeautifulSoup(g.content, "lxml")
        gdesc = game.find("meta",{'name':
         ↪  "Description"})["content"]
```

```
        gtable = game.find("div",{'class': "details_block"})

        if(gtable):
            genre = re.findall(genRe,gtable.text)
            if(genre):
                genrel = map(lambda x :
                 ↪  x.strip(),genre[0].split(","))
            else:
                genrel = []


            developer = re.findall(devRe,gtable.text)
            if(developer):
                developerl = map(lambda x :
                 ↪  x.strip(),developer[0].split(","))
            else:
                developerl = []


            publisher = re.findall(pubRe,gtable.text)
            if(publisher):
                publisherl = map(lambda x :
                 ↪  x.strip(),publisher[0].split(","))
            else:
                publisherl = []

            releaseDate = re.findall(rdRe,gtable.text)
            if (releaseDate):
                releaseDate = releaseDate[0]
            else:
                releaseDate = ""
```

*4: **Script que extrai a informação de um jogo***

Por fim, é criado um ficheiro em formato *JSON* que guarda toda a informação extraída.

```
#File write
            f.write("{\n")
            f.write(f'"name":"{name}",\n') #Nome
            f.write(f'"description":"{gdesc}",\n') #Descrição
            f.write(f'"discount":"{discount}",\n') #Desconto
            f.write(f'"price": "{price}",\n') #Preço
            f.write(f'"rating": "{rating}",\n') #Avaliação
            f.write(f'"sale_start": "{sale_start}",\n') #Inicio de
             ↪ promoção
            f.write(f'"sale_end": "{sale_end}",\n') #Fim de
             ↪ promoção
            f.write(f'"genres": {json.dumps(list(genrel))},\n')
             ↪ #Generos
            f.write(f'"developers":
             ↪ {json.dumps(list(developerl))},\n') #Developers
            f.write(f'"publishers":
             ↪ {json.dumps(list(publisherl))},\n') #Publisher
            f.write(f'"release_date": "{releaseDate}"\n') #Data de
             ↪ Lançamento
            if (counter < len(page)):
                f.write("},\n")
            else:
                f.write("}\n")

f.write("]")
f.close()
```

5: ***Script que guarda a informação no ficheiro***

4 Conclusão

A resolução deste pequeno projeto e a utilização do *Beautiful Soup*, permitiu com que nos fosse introduzido o conceito de *Web Scraping*. O módulo estudado fez com que o conceito fosse bem consolidado, devido à facilidade que este dá ao programador de extrair informações de qualquer sítio da *web*. A ferramenta permite criar *script's* que extraem muita informação e ao mesmo tempo têm um número bastante reduzido de linhas de código, o que o torna bastante apelativo.

Em suma, o *Beautiful Soup* revelou ser o módulo ideal para quem quer começar a fazer mineração de dados através de *Web Scraping*, em comparação com outras ferramentas semelhantes e com o mesmo intuito.