

Extração de dados com Python 3

Baseado em: “[Web scraping with python](#)”

Por: Pedro de Araújo Ribeiro

Considerações:

- Este não é um estudo compreensivo de HTML5
- Este não é um estudo compreensivo de JavaScript
- Este não é um estudo compreensivo de Shell script
- IDE utilizada: Pycharm (JetBrains)
- Compilado e executado no terminal/PowerShell
- Pacotes usados:
 - Beautiful Soup 4
 - Requests
- [Github](#)
- [Kaggle](#)



Proposta:

Utilizar de técnicas simples de web scraping e extração de dados para construir um dataset e realizar um estudo estatístico (próxima apresentação) baseado nas informações obtidas.

O caso de estudo escolhido foi o site americano [CNN](#), onde extraímos a notícia e sua categoria com o intuito de construir um modelo de previsão para categorias de notícia.

Nesta apresentação abordaremos as técnicas e procedimentos para a geração desse conjunto de dados.

Python 3: Instalação

Página de [download](#)

Após a instalação, verifique se foi feita corretamente com o comando abaixo no terminal:

```
python --version
```

Instalação e execução do pycharm não será abordada, porém existem recursos online e outras IDEs mais simples

Em seguida execute a instalação dos requisitos (arquivo .txt se encontra no github) com o comando:

```
pip install -r requirements.txt
```

Extração: entendendo a ferramenta

Para extrações simples em HTML (aquelas que não usam JavaScript) utilizamos apenas as duas bibliotecas mencionadas. Especificamente, requests realiza o acesso e recuperação do conteúdo do site enquanto beautiful soup realiza a extração dos componentes relevantes.

Crie um novo arquivo python (terminado em .py) e insira as seguintes linhas de código:

```
import requests  
from bs4 import BeautifulSoup
```

Extração: entendendo a ferramenta

Começaremos realizando a requisição do conteúdo HTML:

```
url="https://edition.cnn.com/2023/11/28/india/india-tunnel-rescue-vertical-drilling-intl-hnk/index.html"

pagina = requests.get(url)
```

O que fizemos aqui foi armazenar o link do site no objeto url e em seguida realizar uma chamada de função utilizando-o como argumento, e o retorno da função foi armazenado no objeto pagina.

O objeto criado é uma cópia completa do HTML do site requisitado e deve ser transformado em um objeto compatível com o beautiful soup:

```
soup = BeautifulSoup(pagina.text, 'html.parser')
```

Extração: entendendo a ferramenta

A forma como BS realiza a extração é baseada em componentes HTML que tomam a seguinte forma:

```
<tipo conteúdo>  
<tipo> conteúdo </tipo>
```

Exemplos:

```
<meta property="og:title" content="Rescuers successfully  
drill to trapped men in Himalayan tunnel in breakthrough for  
perilous operation | CNN">
```

```
<title> India tunnel collapse: Rescuers in Uttarakhand  
successfully drill to trapped men in breakthrough for  
perilous operation | CNN </title>
```

Extração: entendendo a ferramenta

Tendo isso em mente, a chamada de função para extração é a seguinte:

```
componentes = soup.find_all('meta')
```

Esta função retorna uma lista com todas as instâncias de componentes tipo meta e os armazena em um objeto.

Como vimos no exemplo, um componente possui outros campos dentro de si, que também podem ser extraídos a parte:

```
for comp in componentes:  
    if comp.get('property')=="og:title":  
        titulo = comp.get('content')
```

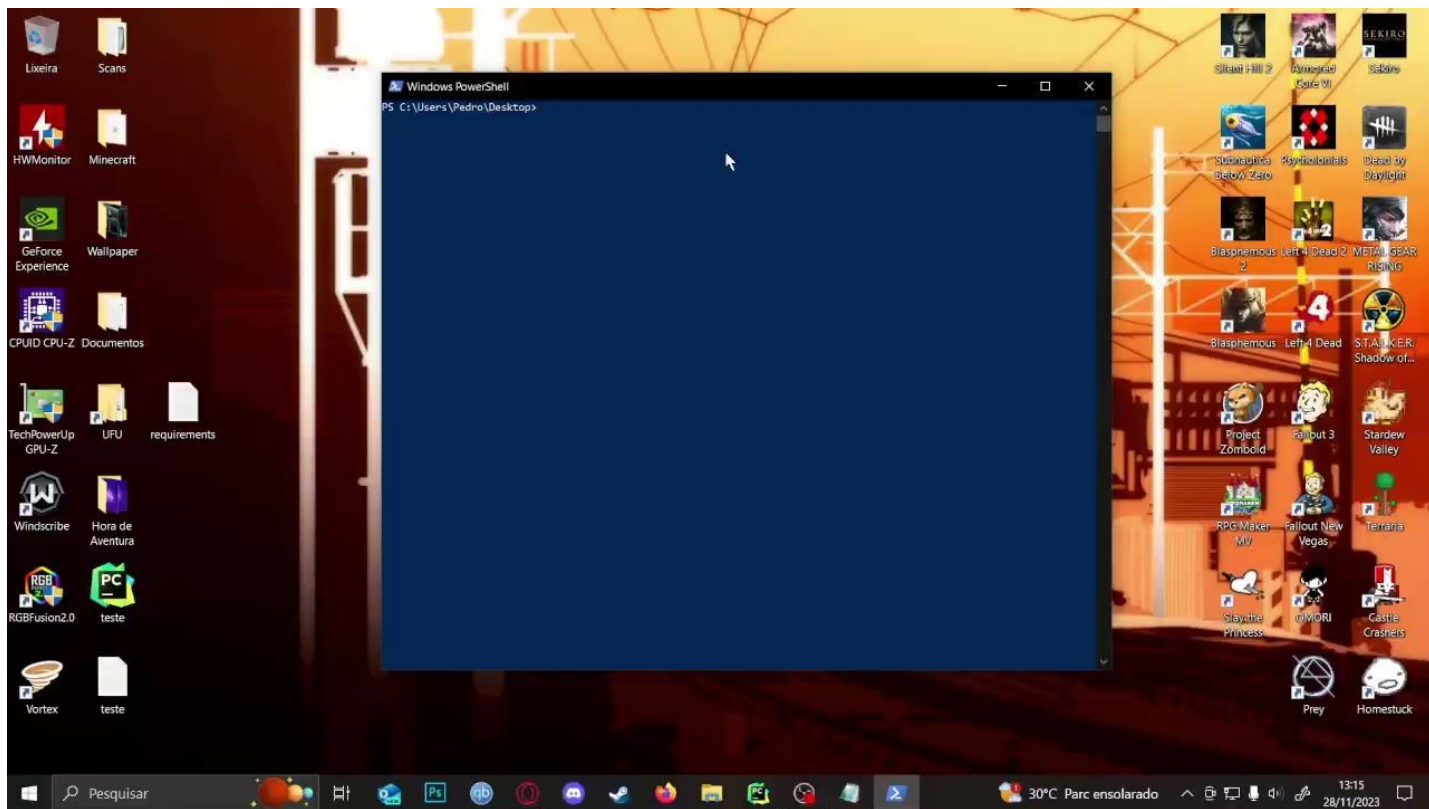

Extração: entendendo a ferramenta

O código anterior analisa todos os componentes 'meta' extraídos e verifica se seu campo 'property' é igual a 'og:title' e se sim, o resto do conteúdo do componente é armazenado no objeto titulo.

Essa lógica é repetida para todas informações que gostaríamos de extrair, porém para saber como montaremos a função devemos fazer uma análise do site escolhido.

Para exemplos anteriores e seguintes continuaremos usando [esta noticia](#).

Extração: exemplo



Análise do site: técnicas

Tendo escolhido um site usamos o atalho `[ctrl+shift+i]` para abrir o menu “inspecionar elemento” onde se encontra o HTML da página web.

Agora podemos analisar o código de maneiras diferentes, porém a mais eficiente é a ferramenta de seleção que se encontra no canto superior esquerdo da aba de inspeção de elemento (imagem no próximo slide), ela também é acessível com o atalho `[ctrl+shift+c]`. A outra alternativa é utilizar o atalho `[ctrl+f]` dentro do código e digitar algum trecho do componente que quer encontrar.

Análise do site: exemplo

The image is a composite of two screenshots. The left screenshot shows a CNN news article titled "Rescuers successfully drill through to trapped men in Himalayan tunnel in breakthrough for perilous operation". The article is by Rhea Mogul and Aishwarya Iyer, dated November 28, 2023. It reports on a rescue operation in the Uttarakhand state of India, where National Disaster Response Force (NDRF) personnel and other rescue operatives are working to reach 41 men trapped in a collapsed tunnel. The right screenshot shows the browser's developer tools for the same page. The 'Elements' panel highlights the 'img' tag for the tunnel image. The 'Styles' panel shows the default 'img' styles, and the 'Console' panel displays multiple error messages about analytics beacons.

Análise do site: exemplo

India tunnel collapse: Rescuers successfully drill through to trapped men in Himalayan tunnel in breakthrough for perilous operation

World / India

Rescuers successfully drill through to trapped men in Himalayan tunnel in breakthrough for perilous operation

h1#maincontent.headline_text.inline-placeholder 656 x 90 November 28, 2023

Color #0C0C0C

Font 24px cnn_sans_display, helvetica, sans-serif

Margin 0px 0px 24px 0px

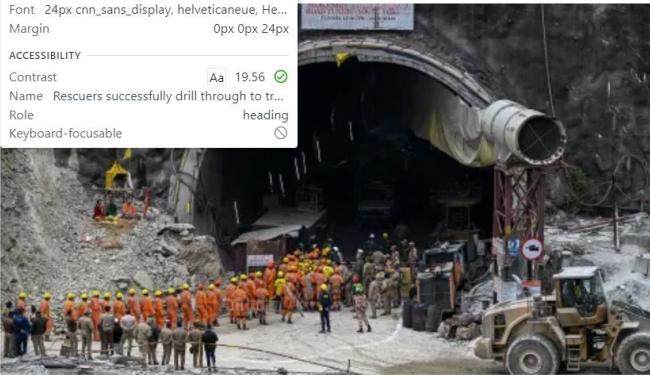
ACCESSIBILITY

Contrast Aa 19.56

Name Rescuers successfully drill through to trapped men in Himalayan tunnel in breakthrough for perilous operation

Role heading

Keyboard-focusable



Sajjad Hussain/AP/Getty Images

National Disaster Response Force (NDRF) personnel along with other rescue operatives gather near the face of the collapsed under construction Silkyara tunnel in the Uttarkashi district of India's Uttarakhand state, on November 28, 2023. Indian rescue teams digging by hand are on the verge of breaking through to reach 41 men trapped in a

```
tabindex="1"></a>
<header class="header_wrapper-outer" style="height: 40px; top: 0px; margin-bottom: 0px;">...</header>
<div class="layout_content-wrapper layout-with-rail_content-wrapper">
  <section class="layout_info layout-with-rail_info" data-editable="toplayout" data-track-zone="toplayout">...</section>
  <section class="layout_top layout-with-rail_top" data-editable="top" data-track-zone="top">
    <div class="breadcrumb">
      breadcrumb_margin-article
      " data-url="cms.cnn.com/_components/breadcrumb/instances/clpgj3l2r003126p251nvdyeg@published" data-component-name="breadcrumb" data-component-variation="breadcrumb">...</div>
    <div data-uri="cms.cnn.com/_components/headline/instances/clpgj3208084126p247nv25w@published" class="headline headline-has-lowertext" data-component-name="headline">
      <div class="headline_wrapper">
        <div data-editable="settings"></div>
        <h1 data-editable="headline_text" class="headline_text inline-placeholder" id="maincontent">...</h1>
      </div>
        <div class="headline_footer">...</div>
      </div>
    </section>
  <section class="layout_wrapper layout-with-rail_wrapper" data-sticky-anchor-pos="top" data-sticky-anchor-condition-type="pageType" data-sticky-anchor-condition-value="article" data-sticky-anchor-priority="1">
    <small div.image_container. picture.image_picture img.image_dam-img>
```

Styles

element.style {

.image_dam-img {

width: 100%;

object-fit: cover;

height: 100%;

}, :after, :before {

box-sizing: inherit;

moz-osx-font-smoothing: grayscale;

webkit-font-smoothing: antialiased;

text-rendering: optimizelegibility;

img[Attributes Style] {

height: 1333px;

aspect-ratio: auto 2000 / 1333;

width: 2000px;

img {

overflow: clip-margin: content-box;

overflow: clip;

Inherited from body.layout.layout-with-ra-

body, h1, h2, h3, h4, h5 {

font-family: cnn_sans_display, helvetica, Helvetica serif;

Console

top Filter

Default levels No Issues

Could not send out analytics beacon. window.trackMetrics is not a function index.html:6513

Could not send out analytics beacon. window.trackMetrics is not a function index.html:6513

Could not send out analytics beacon. window.trackMetrics is not a function index.html:6513

Could not send out analytics beacon. window.trackMetrics is not a function index.html:6513

Could not send out analytics beacon. window.trackMetrics is not a function index.html:6513

Análise do site: exemplo

India tunnel collapse: Rescuers

edition.cnn.com/2023/11/28/india/india-tunnel-rescue-vertical-drilling-intl-hnk/index.html

World


Audio Live TV Log In

World / India

Rescuers successfully drill through to trapped men in Himalayan tunnel in breakthrough for perilous operation

By Rhea Mogul and Aishwarya Iyer, CNN

3 minute read · Updated 7:35 AM EST, Tue November 28, 2023



Sajjad Hussain/AP/Getty Images

National Disaster Response Force (NDRF) personnel along with other rescue operatives gather near the face of the collapsed under construction Silkyara tunnel in the Uttarakhand district of India's Uttarakhand state, on November 28, 2023. Indian rescue teams digging by hand are on the verge of breaking through to reach 41 men trapped in a

Elements Console Sources Network Performance Memory Application

```
<script id="guid" type="text/javascript" src="https://www.ugdturner.com/...>
</script>
window.env={
  "AD_SLOT_CLIENT_INJECTOR_REGISTRY": "https://cdn.cnn.com/logo.png",
  "APPLE_NEWS_LOGO_NAME_POLITICS": "https://media.cnn.com/[audio, custom, feed, interactive, inventory, profile, script hub],
  "CONTENT_HUB_UNIQUE_DEPLOYMENT_KEY": "rnl120cd",
  "CONVIVA_CUSTOM_SCORED_PROD_INIT_JS": "ENABLE_PW_RESET_AROSE": true,
  "ENABLE_L1": {
    "enabled": true,
    "profile": "sqNMPXefWm",
    "liveAuth": {
      "enabled": true,
      "symbolSearch": true,
      "symbolSearch": "J50N&q=",
      "brand": "CNN&platform=web&country=US",
      "TOP_AUTH_SERVICE_APP_ID": "eyj",
      "brand": "cnplusee&platform=web",
      "TRINITY_CONFIGURATION": "domestic.mic"
    }
  }
}
</script>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1, >
<!-- data-editable="head" -->
<!-- data-uri="cms.cnn.com/_components/meta-title/instances/clpg33i" -->
<title>
  "India tunnel collapse: Rescuers in Uttarakhand successfully drill to reach trapped men"
</title>
<meta property="og:title" content="Rescuers successfully drill to reach trapped men in Himalayan tunnel" />
<meta name="twitter:title" content="Rescuers successfully drill to reach trapped men in Himalayan tunnel" />
<!-- data-uri="cms.cnn.com/_components/meta-description/instances/c" -->
<meta name="description" content="Rescuers on Tuesday successfully drill through to reach trapped men in Himalayan tunnel" />
<meta name="twitter:description" content="Rescuers on Tuesday successfully drill through to reach trapped men in Himalayan tunnel" />
<meta name="og:description" content="Rescuers on Tuesday successfully drill through to reach trapped men in Himalayan tunnel" />
<!-- data-uri="cms.cnn.com/_components/meta-image/instances/clne33i" -->
</script>
<!-- t-entry-br.userconsent-state-mg.userconsent-reg-other-optout head meta -->
<script>
  (function() {
    window.rescuersSuccessfully = true;
  })();
</script>
</body>
</html>
```

Styles Computed Layout Event Listeners

Filter

element.style {

*:after, :before index.html:561

box-sizing: inherit;

moz-osx-font-smoothing: grayscale;

webkit-font-smoothing: antialiased;

text-rendering: optimizelegibility;

meta { user agent stylesheet

display: none;

Inherited from html.userconsent-cntry-br...

@media (max-width: 959px) index.html:3

:root {

--social-sharing-display: none;

@media (min-width: 480px) index.html:3

:root {

--theme-section-headline-text_margin-bottom: 20px;

--theme-container_margin-bottom-grid-3: 32px;

--theme-container_margin-bottom-feature-grid-3: 0;

:root {

--theme-primary: #cc0000;

--theme-background: #0c0c0c;

--theme-divider: #404040;

Console

at e.value (index.html:8417:724)

at index.html:8417:608

POST https://logs.browser-intake-datagogh.com/api/v2/logs?ddsource=browser&ddta_0&dd-e index.html:6697

vp-origin=browser&dd-request-id=bdd3e02-92ff-4508-h73f-51541c051c26 net::ERR_BLOCKED_BY_ADBLOCKER

POST https://logs.browser-intake-datagogh.com/api/v2/logs?ddsource=browser&ddta_0&dd-e index.html:6699

vp-origin=browser&dd-request-id=bb60658d-0c31-4518-9a15-028db764893f net::ERR_BLOCKED_BY_ADBLOCKER

Análise do site: o que buscar

Para cada site que se deseja extrair informações deve-se tomar a importante decisão de quais informações são relevantes para estudos futuros. No caso de notícias CNN, foram identificadas como relevantes o título da notícia, subtítulo, palavras chave, corpo do texto e a categoria. Além disso mais tarde abordaremos web crawlers e para isso devemos também buscar quaisquer referências a outros artigos CNN dentro de um dado artigo.

Para estes artigos:

- links estão no componente `<a>` no campo “href”
- título, subtítulo, palavras chaves e categoria estão no componente `<meta>` em campos variados
- corpo do texto está em um componente a parte que veremos em breve

Extração: busca direcionada

Primeiro preparamos os objetos que receberão os dados

```
metas = soup.find_all('meta')
```

```
theme_element = None
```

```
title_element = None
```

```
description_element = None
```

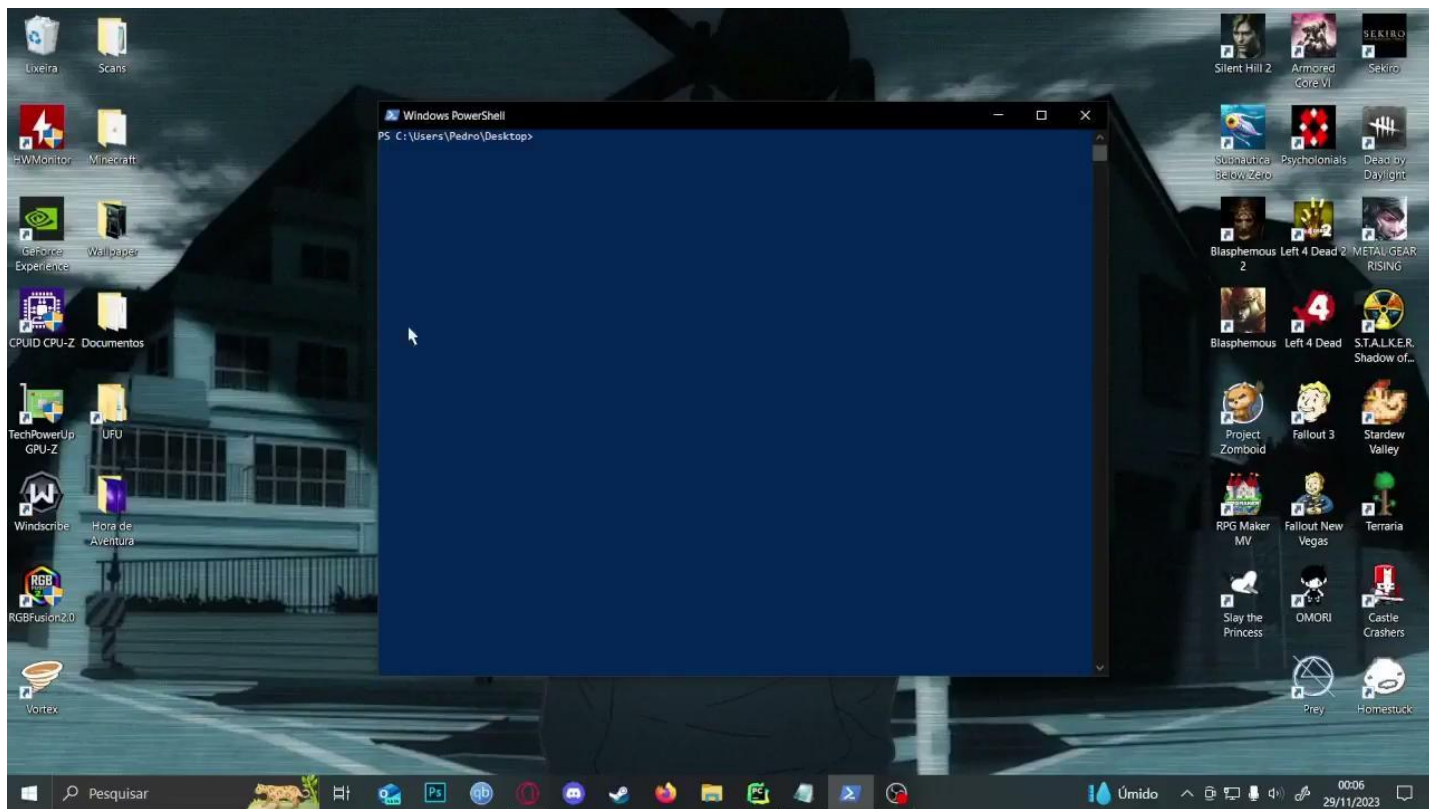
```
tags_element = None
```

Python é uma linguagem que não exige pré alocação, portanto essa etapa serve um propósito diferente: verificar após a extração se todos os campos foram preenchidos.

Extração: busca direcionada

```
for m in metas:
    if m.get('name') == 'description':
        description_element = m.get('content')
    if m.get('name') == 'theme':
        theme_element = m.get('content')
    if m.get('property') == 'og:title':
        title_element = m.get('content').split('|', 1)[0]
    if m.get('name') == 'keywords':
        tags_element = m.get('content')
```

Extração: busca direcionada - exemplo



Extração: busca direcionada

Para o corpo do texto será mais complicado pois ele toma a seguinte forma sempre:

```
<script type="application/ld+json">
```

```
{"@type": "NewsArticle", "@context": "https://schema.org", "articleBody": "Corpo do texto", "articleSection": continua com muitas outras informações}
```

```
</script>
```

O que temos ao fim é um ponto de início e de parada bem definido em um componente enorme, possibilitando que cortemos somente a parte relevante.

Extração: busca direcionada

```
start = '<script  
type="application/ld+json">{"@type":"NewsArticle", "@conte  
xt":"https://schema.org", "articleBody":' '  
end = '", "articleSection":'  
  
s = str(soup.find('script', type='application/ld+json'))  
  
article_element = (s.split(start))[1].split(end)[0]
```

Extração: busca direcionada e web crawlers

O que fizemos no código anterior foi cortar a string nos pontos relevantes encontrados, de forma que temos somente o corpo do texto armazenado no objeto.

Agora sobraram somente os links para outros artigos, porém para entender sua relevância devemos abordar web crawlers:

Consistem de scripts e funções responsáveis por automatizar a expansão da base inicial de alvos de extração, semelhante a um processo auto-replicante. Para cada artigo extraímos seus links de outros artigos e para cada um desses repetimos o processo até que não sobrem mais links não visitados.

Extração: busca direcionada e web crawlers

Links estão armazenados de duas formas diferentes, porém ainda no mesmo componente e mesmo campo:

```
<a href="/2023/11/28/india/india-tunnel-rescue-vertical-drilling-intl-hnk/index.html" /a>
```

```
<a href="https://edition.cnn.com/2023/11/28/india/india-tunnel-rescue-vertical-drilling-intl-hnk/index.html" /a>
```

Portanto criaremos uma contingência para ambos os casos e com tudo visto até agora podemos construir nossa primeira função.

Extração: busca direcionada e web crawlers

```
def getlinks(links, url):

    page = requests.get(url)

    soup = BeautifulSoup(page.text, features: 'html.parser')

    links1 = soup.find_all('a')

    for li in links1:
        temp = li.get('href')
        if str(temp).startswith('/2023/') and str('https://edition.cnn.com' + temp) not in links:
            links = links + ['https://edition.cnn.com' + temp]
        if (str(temp).startswith('https://edition.cnn.com/2023/')
            and str(temp) not in links and '/videos/' not in str(temp)
            and '/gallery/' not in str(temp)):
            links = links + [temp]

    return links
```

Extração: busca direcionada e web crawlers

A função anterior recebe a lista de links alvos, sendo eles visitados ou não, e a URL a ser analisada.

Vários componentes `<a>` são extraídos e para todos verificamos seu campo `“href”`, caso caia no primeiro caso o conteúdo é concatenado com o prefixo do site e adicionado à lista de links, no segundo caso não há concatenação e é adicionado imediatamente.

A primeira condição é simplesmente não estar na lista, enquanto na segunda além disso ele não deve pertencer aos temas `“video”` e `“galeria”` pois estes não possuem corpo de texto.

Em seguida criaremos nossa segunda função, já implementando o crawler.

Extração: busca completa

```
5 def pageread(idn, data_page, url, links):
6
7     links = getlinks(links, url)
8
9     page = requests.get(url)
10
11     soup = BeautifulSoup(page.text, features: 'html.parser')
12
13     metas = soup.find_all('meta')
14
15     theme_element = None
16     title_element = None
17     description_element = None
18     tags_element = None
19     start = ('<script type="application/ld+json">{"@type":'
20             '"NewsArticle","@context":"https://schema.org","articleBody":""')
21     end = '","articleSection":'
```

Extração: busca completa

```
23     s = str(soup.find( name= 'script', type='application/ld+json'))
24     if not s.startswith(start):
25         return [data_page, links]
26
27     article_element = (s.split(start))[1].split(end)[0]
28     link_element = url
29
30     for m in metas:
31         if m.get('name') == 'description':
32             description_element = m.get('content')
33         if m.get('name') == 'theme':
34             theme_element = m.get('content')
35         if m.get('property') == 'og:title':
36             title_element = m.get('content').split('|', 1)[0]
37         if m.get('name') == 'keywords':
38             tags_element = m.get('content')
```

Extração: busca completa

```
40     if theme_element is None or title_element is None:
41         return [data_page, links]
42     return [data_page + [(idn, title_element, description_element,
43                           article_element, tags_element, theme_element,
44                           link_element)], links]
```

1º trecho do código extrai os links, requisita o html e o converte em um objeto soup e por fim inicializa os objetos

2º trecho verifica se existe um corpo do texto, se não a função retorna os parâmetros iniciais sem mudanças, se sim ele continua para a extração das outras informações

3º trecho verifica se existe título e tema, se não a função retorna os parâmetros iniciais sem mudanças, se sim ele retorna a lista de dados anteriores concatenada com a 7-upla obtida, e a lista de links

Extração: tipos de dados e seu armazenamento

Na função anterior passamos os seguintes parâmetros: Um contador de iterações que simplesmente atribui um número ao artigo, uma lista de informações de artigos, o link do artigo e a lista de links gerais.

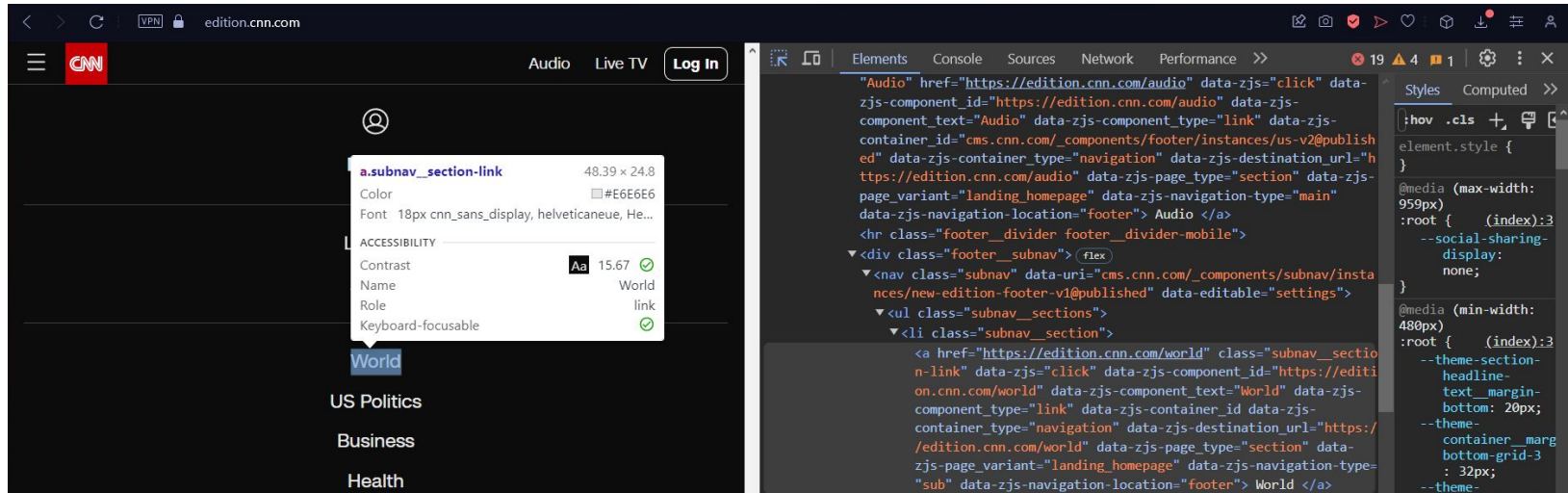
Agora devemos dar atenção a essa lista de informações e do que ela é constituída:

```
data_page = [("ID", "Title", "Description", "Body",  
             "Keywords", "Theme", "Link")]
```

O que fizemos aqui foi iniciar a lista com uma 7-upla referente aos nomes das informações que estamos extraindo. Ao final essa primeira 7-upla será o header de um arquivo CSV e as demais serão as observações.

Extração: coleta de links iniciais

No site principal da CNN podemos observar além de links de notícias, bem no fim da página, links para subseções da CNN referentes aos temas de notícias. Essas subseções serão chamadas de agregadores e criaremos uma nova função para extraí-los, depois aplicaremos a função `getlinks` em cada um deles para conseguir a maior quantidade possível de artigos logo no início.



Extração: coleta de links iniciais

```
def getagregator(ags, url):  
  
    page = requests.get(url)  
  
    soup = BeautifulSoup(page.text, features: 'html.parser')  
  
    links1 = soup.find_all('a')  
  
    for li in links1:  
        temp = li.get('href')  
        if str(temp).startswith('https://edition.cnn.com/') and str(temp) not in ags:  
            ags = ags + [temp]  
  
    return ags
```

Extração completa:

A função anterior é uma simples variação de `getlinks` com menos condicionais.

Agora, finalmente, temos todos os componentes necessários para realizar um web scrape extensivo.

As três funções foram salvas em um arquivo nomeado `pageReader.py` e o restante do processo ocorrerá no arquivo `main.py`.

Utilizaremos também a biblioteca `csv` nativa do python para converter todas as informações extraídas em um arquivo compatível com o R Studio.

Extração completa:

```
1 import pageReader
2 import csv
3
4 data_page = [("ID", "Title", "Description", "Body", "Keywords", "Theme", "Link")]
5 links = []
6 agrs = []
7
8 agrs = pageReader.getagregator(agrs, url="https://edition.cnn.com")
9
10 print("Leitura dos agregadores concluida")
11
12 for a in agrs:
13     links = pageReader.getlinks(links, a)
14
15 print("Leitura dos links concluida, total de noticias: " + str(len(links)))
```


Extração completa:

```
17 cnt = 0
18
19 while True:
20     l2 = links[cnt]
21     cnt += 1
22     if len(data_page) < 5000 and cnt < len(links):
23         print("Progresso: " + str(cnt) + "/" + str(len(links)))
24         temp = pageReader.pageread(cnt, data_page, l2, links)
25         data_page = temp[0]
26         links = temp[1]
27     else:
28         break
29
30 print("Gerando arquivo csv")
31
32 csv_file = open('news.csv', 'w', encoding='utf-8', newline='')
33 writer = csv.writer(csv_file)
34
35 for p in data_page:
36     writer.writerow(p)
```

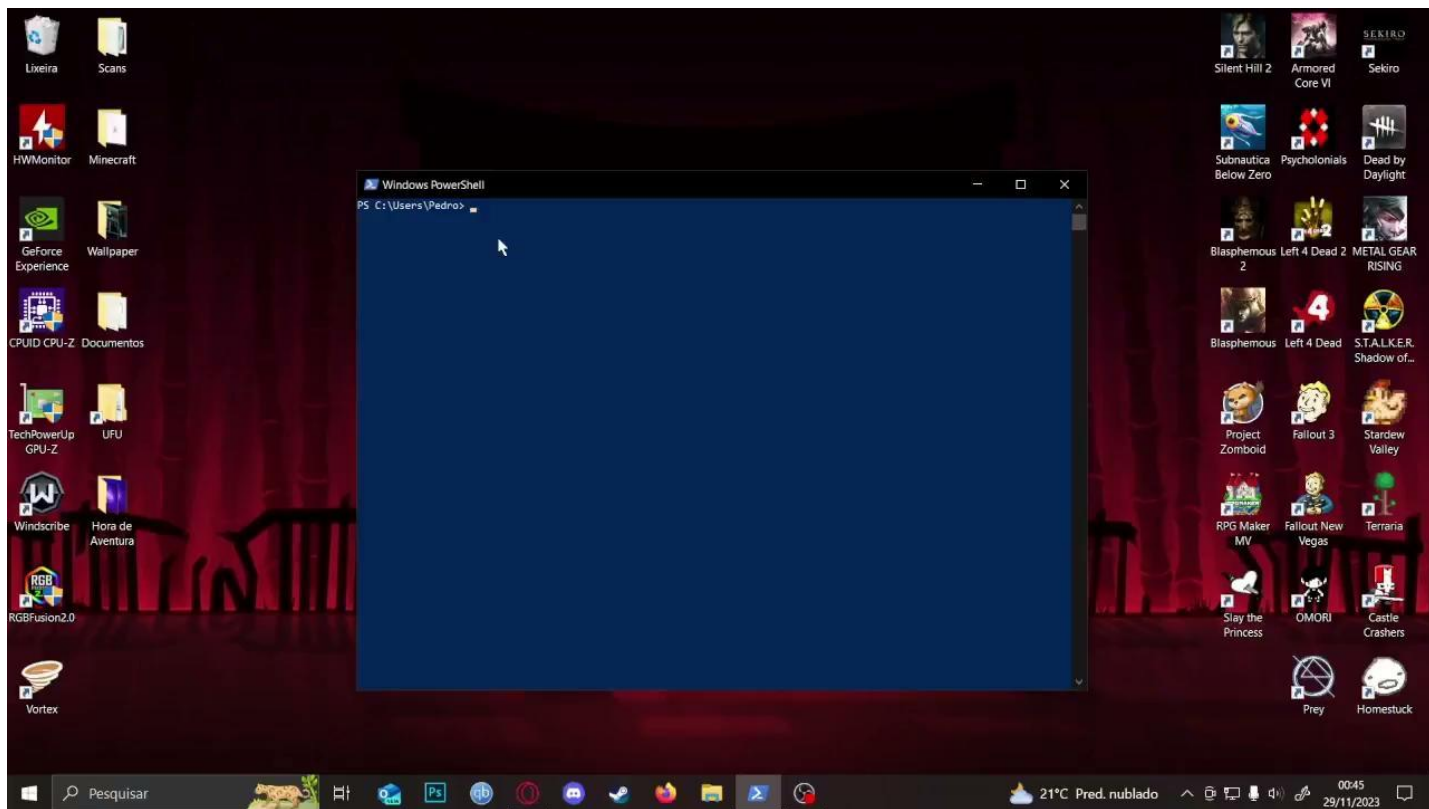
Extração completa:

1º trecho inclui as bibliotecas, inicializa as listas de informações de notícias, links e agregadores e em seguida são lidos os agregadores e os links contidos neles.

2º trecho é feita a repetição da leitura de informações para todo link da lista enquanto a quantidade de links lidos for menor que 5000 (limite arbitrário estipulado para forçar o encerramento do programa) ou a quantidade lida for igual a de links. Ao final, o arquivo CSV é gerado e o programa se encerra

Com isso se encerra o processo de extração de dados e permite o começo das aplicações de algoritmos de classificação (próxima apresentação).

Extração completa: exemplo



FIM.