Beautifulsoup Handbook : Data Collection for Ai through Webscraping

Ashraful Islam Mahi

Full Stack Python Developer | ML & Robotics Enthusiast | Founder & CEO, PytronLab

Understanding the Role of Data Collection in Data Science & Machine Learning.

What is data collection?

- Data collection includes gathering, processing, and storing data from various sources to train machine learning models. It includes structured (databases, spreadsheets) and unstructured (images, text, audio) data collected from sensors, social media, APIs, surveys, and more.

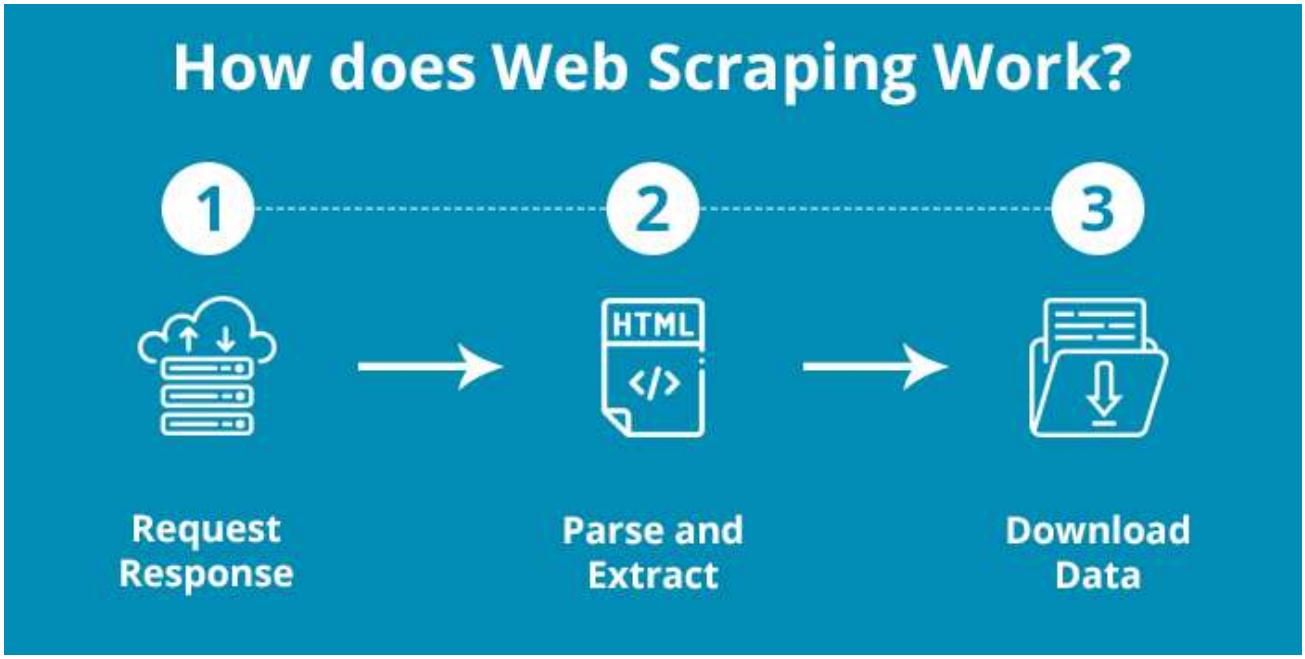
Imporatnce of data collection:

1. Model Accuracy – High-quality data ensures better predictions and decision-making.
2. Training Efficiency – More diverse and relevant data improve learning speed and performance.
3. Bias Reduction – Proper data collection minimizes biases and enhances fairness in AI models.
4. Real-world Adaptation – AI models need continuous data to stay relevant and perform well in dynamic environments.
5. Automation & Insights – Well-collected data helps automate processes and extract meaningful insights for businesses and research.

What is webscraping?

- Web scraping is the automated process of extracting data from websites. It involves using scripts or tools to access web pages, retrieve information, and store it in a structured format for analysis.

- How It Works:



1. Sending Requests – A scraper sends a request to a webpage using libraries like requests in Python.
2. Parsing HTML – Extracting relevant data using tools like BeautifulSoup or Scrapy.
3. Storing Data – Saving the extracted information in a CSV, database, or other formats for further use.

- Common Uses:
  - Price comparison
  - Market research
  - Sentiment analysis
  - Lead generation
  - AI training data collection

Popular webscraping frameworks of pythons & their use cases:

Frameworks	When to Use
1. Beautifulsoup & Requests	Static website with small to medium project
2. Scrapy	Static website but Large Project
3. Selenium	Dynamic website with direct interaction requirement
4. Scrapy-Playwright	Dynamic website with large project

Intro to Beautifulsoup:

What is Beautifulsoup?

- Beautiful Soup is a Python package for parsing HTML and XML documents, including those with malformed markup. It creates a parse tree for documents that can be used to extract data from HTML, which is useful for web scraping.

1. Installing Beautifulsoup:

In [ ]:
```
%pip install beautifulsoup4
%pip install requests
```

2. Importing Beautifulsoup:

In [1]:
```
from bs4 import BeautifulSoup as bs
import requests
```

3. Best Practices of using Beautifulsoup:
   - Don't send separate requests for each extraction.
   - Extract the html of the web page at once using single request.

Let's scrape a website Quotes to Scrape

Technique: First HTML retrive then Scrape

Step-1: Get the url of the web page you want to scrape:

In [15]:
```
url = 'https://quotes.toscrape.com/'
```

Step-2: Fetch the HTML code of that page:

In [16]:
```
respose = requests.get(url)
```

Step-3: Check the status code of the 'response'.

- If respose.status_code == 200 : HTML has successfully fetched.
- If respose.status_code != 200 : HTML hasn't fetched.

In [17]:
```
respose.status_code
```

Out[17]: 200

Step-4: Save the HTML code in a HTML file:

In [19]:
```
html_file_name = 'quote.html'

with open(html_file_name,'w', encoding='utf8') as file:
  file.write(respose.text)
```

Step-5: Create Soup object using beautiful soup to scrape the html:

In [ ]:
```
html_file_name = 'quote.html'

with open(html_file_name,'r',encoding='utf8') as file:
  html_content = file.read()
  soup = bs(html_content)
  print(soup.prettify())
```

Scraping using the soup object:

1. Find 'Page Title':

In [141…
```
title = soup.find('title')

print(title.text)
```

```
Quotes to Scrape
```

2. Select first quote according to class name:

In [142…
```
first_quote = soup.find('span',class_='text')

print(first_quote.text)
```

```
“The world as we have created it is a process of our thinking. It cannot be changed without ch
anging our thinking.”
```

3. Select all quotes according to class name:

In [143…
```
quotes = soup.find_all('span',class_='text')
print(f"No of quotes: {len(quotes)}\n")

print("Quotes:\n------")
for i in range(len(quotes)):
  print(f"{i+1}.{quotes[i].text}")
```

```
No of quotes: 10

Quotes:
------
1.“The world as we have created it is a process of our thinking. It cannot be changed without
changing our thinking.”
2.“It is our choices, Harry, that show what we truly are, far more than our abilities.”
3.“There are only two ways to live your life. One is as though nothing is a miracle. The other
is as though everything is a miracle.”
4.“The person, be it gentleman or lady, who has not pleasure in a good novel, must be intolera
bly stupid.”
5.“Imperfection is beauty, madness is genius and it's better to be absolutely ridiculous than
absolutely boring.”
6.“Try not to become a man of success. Rather become a man of value.”
7.“It is better to be hated for what you are than to be loved for what you are not.”
8.“I have not failed. I've just found 10,000 ways that won't work.”
9.“A woman is like a tea bag; you never know how strong it is until it's in hot water.”
10.“A day without sunshine is like, you know, night.”
```

4. Find the text of quote using attribute:

In [144…
```
quote_text = soup.find('span',attrs={'itemprop':'text'}).text

print(quote_text)
```

```
“The world as we have created it is a process of our thinking. It cannot be changed without ch
anging our thinking.”
```

5. Select "top 10 tags" using the text inside it:

In [145…
```
tags_title = soup.find('h2',string='Top Ten tags')
print(tags_title)
```

```
<h2>Top Ten tags</h2>
```

Navigation using soup object

1. Find parent of 'tags_title':

In [ ]:
```
tag_box = tags_title.parent

print(tag_box)
```

2. Find the next sibling of "tags_title':

In [107…
```
first_tag_span = tags_title.find_next_sibling()

print(first_tag_span)
```

```
<span class="tag-item">
<a class="tag" href="/tag/love/" style="font-size: 28px">love</a>
</span>
```

3. Find the previous sibling of "tags_title':

In [108…
```
h2_again = first_tag_span.find_previous_sibling()

print(h2_again)
```

```
<h2>Top Ten tags</h2>
```

4. First children of tag box:

In [138…
```
tag_children = soup.find('div',class_ = 'tags-box').children
tag_box_children = list(tag_children)
final_children = [x for x in tag_box_children if x != '\n']

top_h2_tag = final_children[0]

print(top_h2_tag)
```

```
<h2>Top Ten tags</h2>
```

Extraction

1. Get the href value of top_tag:

In [140…
```
first_tag_span_href = first_tag_span.a['href']

print(first_tag_span_href)
```

```
/tag/love/
```

2. Extract any attribute with this method:

In [146…
```
first_quote['class']
```

Out[146…
```
['text']
```

In [148…
```
first_tag_span.a['style']
```

Out[148…
```
'font-size: 28px'
```

Let's scrape another website Book to Scrape

Step-1: Extract 1st book info:

In [4]:
```
url = 'https://books.toscrape.com/catalogue/a-light-in-the-attic_1000/index.html'
```

Step-2: Extract the html of that book page

In [5]:
```
response = requests.get(url)
```

Step-3: Check the status code

In [6]:
```
response.status_code
```

Out[6]: 200

Step-4: Write the html code into a html file

In [7]:
```
file_name = 'book.html'

with open(file_name,'w',encoding='utf8') as file:
  file.write(response.text)
```

Step-5: Read the html file and turn that into a soup object

In [ ]:
```
file_name = 'book.html'

with open(file_name,'r',encoding='utf8') as file:
  content = file.read()
  soup = bs(content)

print(soup.prettify())
```

In [10]:
```
# book name extraction
book_name = soup.find('div',class_ = 'product_main').h1
print(f"Book Name: {book_name.text}")
```

```
Book Name: A Light in the Attic
```

In [11]:
```
# catagory name extraction
items = soup.find('ul',class_ = 'breadcrumb').children
list_items = list(items)
filtered_list = [x for x in list_items if x != '\n']
catagory = filtered_list[2]
print(f"Book Catagory:{catagory.text}")
```

```
Book Catagory:
Poetry
```

In [12]:
```
# rating extraction
all_p = soup.find('div',class_ = 'product_main')
p_items = all_p.find_all('p')
rating = p_items[2]['class']

print(f"Ratings: {rating[1]}")
```

```
Ratings: Three
```

In [13]:
```
# product info extraction
left_info = soup.find_all('th')
right_info = soup.find_all('td')

print("Product Info:")
print("-----------")
for left,right in zip(left_info,right_info):
  print(f"{left.text}: {right.text}")
```

```
Product Info:
-----------
UPC: a897fe39b1053632
Product Type: Books
Price (excl. tax): Â£51.77
Price (incl. tax): Â£51.77
Tax: Â£0.00
Availability: In stock (22 available)
Number of reviews: 0
```

In [30]:
```
# Image Extraction

image_div = soup.find('div',class_ = 'item active').img['src']

print(f'Image link: https://books.toscrape.com/{image_div}')
```

```
Image link: https://books.toscrape.com/../../media/cache/fe/72/fe72f0532301ec28892ae79a629a293c.jpg
```

In [34]:
```
url = 'https://books.toscrape.com/index.html'

response = requests.get(url)

response.status_code
```

Out[34]: 200

In [35]:
```
file_name = 'books.html'

with open(file_name,'w',encoding='utf8') as file:
  file.write(response.text)
```

In [ ]:
```
file_name = 'books.html'

with open(file_name,'r',encoding='utf8') as file:
  content = file.read()

  soup = bs(content)
  print(soup.prettify())
```

In [42]:
```
#Detect page number

list = soup.find('li',class_ = 'current').text
no_of_page = int(list.strip().split(' ')[-1])
print(no_of_page)
```

```
50
```

In [ ]:
```
#Loop throuh all pages , if the page number is known
for page in range(1,no_of_page+1):
  print(f"Page no----> {page}")
  page_url = f'https://books.toscrape.com/catalogue/page-{page}.html'
  response = requests.get(page_url)
  content = response.text
  soup = bs(content)
  books = soup.find_all('article',class_ = 'product_pod')
  print(f"No of book: {len(books)}")
```

Bulk Extraction

In [3]:
```
import requests
from bs4 import BeautifulSoup as bs
import builtins
import pandas as pd

file_name = 'image_link.txt'
# with open(file_name,'w') as file:
#     pass



book_dict = {
  'name': [],
  'price': [],
```

```
  'catagory': [],
  'ratings': [],
  'upc': [],
  'availability': [],
  'in_stock': [],
  'image_link': []
}

headers = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) '
                  'AppleWebKit/537.36 (KHTML, like Gecko) '
                  'Chrome/134.0.0.0 Safari/537.36'
}

page = 1
while True:
    print("\n************************")
    print(f"Page no----> {page}")
    print("************************\n")

    page_url = f'https://books.toscrape.com/catalogue/page-{page}.html'
    response = requests.get(page_url, headers=headers)
    content = response.text
    # Use a separate variable for the listing page soup
    listing_soup = bs(content, 'html.parser')
    books = listing_soup.find_all('article', class_='product_pod')

    for book in books:
        # Fix quotes in the f-string below
        book_href = book.find("a")["href"]
        book_url = f'https://books.toscrape.com/catalogue/{book_href}'
        response = requests.get(book_url, headers=headers)
        response.encoding = 'utf-8'
        book_soup = bs(response.text, 'html.parser')

        # Book name extraction
        book_name_tag = book_soup.find('div', class_='product_main').h1
        book_name = book_name_tag.text if book_name_tag else "No Title"
        book_dict['name'].append(book_name)

        # Price extraction
        price_tag = book_soup.find('div', class_='product_main').p
        price = price_tag.text if price_tag else "No Price"
        book_dict['price'].append(price)

        # Catagory extraction
        items = book_soup.find('ul', class_='breadcrumb').children
        list_items = builtins.list(items)
        filtered_list = [x for x in list_items if x != '\n']
        catagory = filtered_list[2].get_text(strip=True) if len(filtered_list) >= 3 else "Not
        book_dict['catagory'].append(catagory)

        # Rating extraction: assumes the third <p> tag in product_main holds the rating
        product_main = book_soup.find('div', class_='product_main')
        p_items = product_main.find_all('p')
        rating = p_items[2]['class'][1] if len(p_items) >= 3 and len(p_items[2].get("class",
        book_dict['ratings'].append(rating)

        # UPC extraction
        upc_th = book_soup.find('th', string='UPC')
        upc = upc_th.find_next_sibling().text if upc_th else "Not Available"
        book_dict['upc'].append(upc)

        # Availability extraction
        availability_th = book_soup.find('th', string='Availability')
        availability = availability_th.find_next_sibling().text if availability_th else "Not
```

```
        book_dict['availability'].append(availability)

        # Instock extraction
        if '(' in availability:
            instock = availability.split('(')[1].split(' ')[0]
        else:
            instock = "Not Available"
        book_dict['in_stock'].append(instock)

        # Image extraction and convert relative URL to absolute URL
        image_tag = book_soup.find('div', class_='item active').img
        if image_tag:
            image_src = image_tag.get('src', '')
            image_src = image_src.replace('../', '')
            image_link = f'https://books.toscrape.com/{image_src}'
        else:
            image_link = "Not Available"
        book_dict['image_link'].append(image_link)
        with open(file_name,'a',encoding='utf8') as file:
            file.write(image_link+'\n')

    # Use listing_soup to check for the next button
    # next_button = listing_soup.find('li', class_='next')
    if page > 3:
        print("\nScraping Done")
        break
    else:
        page += 1

df = pd.DataFrame(book_dict)
df.to_excel('book.xlsx', index=False)
```

```
*************************
Page no----> 1
*************************


*************************
Page no----> 2
*************************


*************************
Page no----> 3
*************************


*************************
Page no----> 4
*************************


Scraping Done
```

Download Images:

In [ ]:

```
import os

os.mkdir('book_images')

headers = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) '
                  'AppleWebKit/537.36 (KHTML, like Gecko) '
                  'Chrome/134.0.0.0 Safari/537.36'
}
```

```
with open('image_link.txt', 'r') as f:
  links_txt = f.read()
  links_list = links_txt.split('\n')

for i ,image_url in enumerate(links_list):
  response = requests.get(image_url,headers=headers)
  with open(f'book_images/{i+i}.jpg', 'wb') as imagefile:
    imagefile.write(response.content)
```