

Data collection through Webscraping

Rafiq Islam

2024-08-14

Table of contents

Introduction	1
Data Source Websites	1
Web Scraping	2
Example of Webscraping from a real website	5
References	11

Introduction

Collecting data and preparing it for a project is one of the most important tasks in any data science or machine learning project. There are many sources from where we can collect data for a project, such as

- Connecting to a SQL database server
- Data Source Websites such as [Kaggle](#), [Google Dataset Search](#), [UCI Machine Learning Repo](#) etc
- Web Scraping with Beautiful Soup
- Using Python API

Data Source Websites

Data source websites mainly falls into two categories such as data repositories and data science competitions. There are many such websites.

1. The [UCI Machine Learning Repository](#)

2. The [Harvard Dataverse](#)
3. The [Mendeley Data Repository](#)
4. The [538](#)
5. The [New Yourk Times](#)

6. The [International Data Analysis Olympiad](#)
7. [Kaggle Competition](#)

Example of collecting data from [UCI Machine Learning Repository](#)

```
from ucimlrepo import fetch_ucirepo

# fetch dataset
iris = fetch_ucirepo(id=53)

# data (as pandas dataframes)
X = iris.data.features
y = iris.data.targets

# metadata
print(iris.metadata)

# variable information
print(iris.variables)
```

you may need to install the [UCI Machine Learning Repository](#) as a package using pip.

```
pip install ucimlrepo
```

Web Scrapping

We scrapping is another way of collecting the data for the research if the data is not available in any repository. We can collect the data from a website using a library called [BeautifulSoup](#) if the website has permission for other people to collect data from the website.

```
import bs4                                # library for BeautifulSoup
from bs4 import BeautifulSoup             # import the BeautifulSoup object
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

```
from seaborn import set_style
set_style("whitegrid")
```

Now let's make a html object using BeautifulSoup. Let's say we have a html website that looks like below

```
html_doc="""
<!DOCTYPE html>
<html lang="en">
<head>
    <title>My Dummy HTML Document</title>
</head>
<body>
    <h1>Welcome to My Dummy HTML Document</h1>
    <p>This is a paragraph in my dummy HTML document.</p>
    <a href="https://mrislamdb.github.io/blog" class="blog" id="blog"> Blog </a>
    <a href="https://mrislamdb.github.io/research" class="research" id="research"> Research </a>
</body>
</html>
"""
```

Now we want to grab information from the dummy html document above.

```
soup=BeautifulSoup(html_doc, features='html.parser')
```

Now that we have the object `soup` we can walk through each element in this object. For example, if we want to grab the title element,

```
soup.html.head.title
```

```
<title>My Dummy HTML Document</title>
```

Since the html document has only one title, therefore, we can simply use the following command

```
soup.title
```

```
<title>My Dummy HTML Document</title>
```

or this command to get the text only

```
soup.title.text
```

```
'My Dummy HTML Document'
```

This soup object is like a family tree. It has parents, children, greatgrand parents etc.

```
soup.title.parent
```

```
<head>
<title>My Dummy HTML Document</title>
</head>
```

Now to grab an attribute from the `soup` object we can use

```
soup.a
```

```
<a class="blog" href="https://mrslambda.github.io/blog" id="blog"> Blog </a>
```

or any particular thing from the attribute

```
soup.a['class']
```

```
['blog']
```

We can also find multiple attribute of the same kind

```
soup.findAll('a')
```

```
/tmp/ipykernel_7083/2082462312.py:1: DeprecationWarning:
```

```
Call to deprecated method findAll. (Replaced by find_all) -- Deprecated since version 4.0.0.
```

```
[<a class="blog" href="https://mrslambda.github.io/blog" id="blog"> Blog </a>,
 <a class="research" href="https://mrslambda.github.io/research" id="research"> Research </a>]
```

Then if we want any particular object from all a attribute

```
soup.findAll('a')[0]['id']
```

/tmp/ipykernel_7083/2617565345.py:1: DeprecationWarning:

Call to deprecated method findAll. (Replaced by find_all) -- Deprecated since version 4.0.0.

'blog'

For any p tag

```
soup.p.text
```

'This is a paragraph in my dummy HTML document.'

Similarly, if we want to grab all the hrefs from the a tags

```
[h['href'] for h in soup.findAll('a')]
```

/tmp/ipykernel_7083/1829196164.py:1: DeprecationWarning:

Call to deprecated method findAll. (Replaced by find_all) -- Deprecated since version 4.0.0.

['https://mrslambda.github.io/blog', 'https://mrslambda.github.io/research']

Example of Webscraping from a real website

In this example we want to obtain some information from [NVIDIA Graduate Fellowship Program](https://mrslambda.github.io/research). Before accessing this website we need to know if we have permission to access their data through webscraping.

```
import requests
response = requests.get(url="https://research.nvidia.com/graduate-fellowships/archive")
response.status_code
```

200

The `status_code` 200 ensures that we have enough permission to access their website data. However, if we obtain `status_code` of 403, 400, or 500 then we do not permission or a bad request. For more about the status codes [click here](#).

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

We want to make an analysis based on the institution of the past graduate fellows. Insepecting the elements in [this website](#) we see that the div those have `class="archive-group"` contains the information of the past graduate fellows.

```
pf = soup.find_all("div", class_="archive-group")
```

and the first element of this `pf` contains the information of the graduate fellows in the year of 2021.

```
pf[0]
```

```
<div class="archive-group">
<h4 class="archive-group__title">2022 Grad Fellows</h4>
<div class="views-row"><div class="views-field views-field-title"><span class="field-content
<div class="views-row"><div class="views-field views-field-title"><span class="field-content
<div class="views-row"><div class="views-field views-field-title"><span class="field-content
<div class="views-row"><div class="views-field views-field-title"><span class="field-content
<div class="views-row"><div class="views-field views-field-title"><span class="field-content
<div class="views-row"><div class="views-field views-field-title"><span class="field-content
<div class="views-row"><div class="views-field views-field-title"><span class="field-content
<div class="views-row"><div class="views-field views-field-title"><span class="field-content
<div class="views-row"><div class="views-field views-field-title"><span class="field-content
<div class="views-row"><div class="views-field views-field-title"><span class="field-content
<div class="views-row"><div class="views-field views-field-title"><span class="field-content
<div class="views-row"><div class="views-field views-field-title"><span class="field-content
<div class="views-row"><div class="views-field views-field-title"><span class="field-content
<div class="views-row"><div class="views-field views-field-title"><span class="field-content
<div class="views-row"><div class="views-field views-field-title"><span class="field-content
</div>
```

Now let's make a **pandas** dataframe using the information in this page. We can make an use of the output from the above chunk. To grab the year, we see that `archive-group__title` class with a `h4` tag contains the year for all years. With `strip=True`, the text is cleaned by removing extra whitespace from the beginning and end. We need the first element so a `split()[0]` will do the job. Then we make another group called `fellows` that contains the fellows in a certian year by using the `div` and `class="views-row"`. Once the new group created, we then iterate through this group to extract their names and corresponding institutions.

```

data=[]

for group in pf:
    year = group.find(
        "h4",class_="archive-group__title"
    ).get_text(strip=True).split()[0]

    fellows = group.find_all("div", class_="views-row")
    for fellow in fellows:
        name = fellow.find(
            "div", class_="views-field-title"
        ).get_text(strip=True)
        institute = fellow.find(
            "div", class_="views-field-field-grad-fellow-institution"
        ).get_text(strip=True)

        data.append({"Name": name, "Year": year, "Institute": institute})

data=pd.DataFrame(data)
data.head()

```

	Name	Year	Institute
0	Davis Rempe	2022	Stanford University
1	Enze Xie (Finalist)	2022	University of Hong Kong
2	Gokul Swamy (Finalist)	2022	Carnegie Mellon University
3	Hao Chen	2022	University of Texas at Austin
4	Hong-Xing (Koven) Yu (Finalist)	2022	Stanford University

Now let's perform some Exploratory Data Analysis (EDA). First, we analyze the unique values and distributions.

```

# Count the number of fellows each year
year_counts = data['Year'].value_counts().sort_values(ascending=False)
# Create a DataFrame where years are columns and counts are values in the next row
year_data = {
    'Year': year_counts.index,
    'Count': year_counts.values
}
# Create the DataFrame
year_data_counts = pd.DataFrame(year_data)

```

```
# Transpose the DataFrame and reset index to get years as columns
year_data_counts = year_data_counts.set_index('Year').T

# Display the DataFrame
print(year_data_counts)
```

```
Year    2018    2022    2019    2017    2015    2006    2012    2011    2016    2007    ...    2021  \
Count      16      15      15      15      12      12      11      11      11      11    ...      10
```

```
Year    2020    2003    2014    2008    2009    2010    2005    2004    2002
Count      10      10      10      10      10      9      8      7      6
```

```
[1 rows x 21 columns]
```

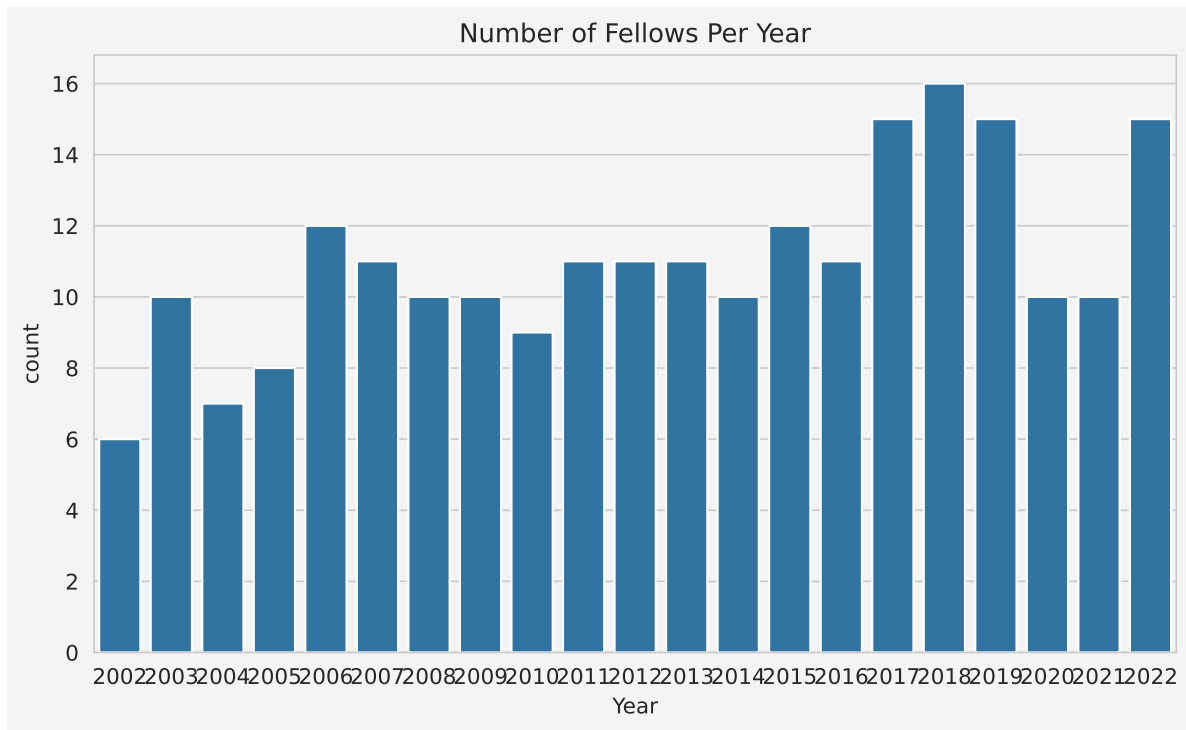
Next we see that most represented universities

```
university_counts = data['Institute'].value_counts()
print(university_counts.head(10)) # Display the top 10 universities
```

```
Institute
Stanford University                28
Massachusetts Institute of Technology  21
Carnegie Mellon University         17
University of California, Berkeley  16
University of Washington           10
University of Utah                 10
Georgia Institute of Technology     9
University of Illinois, Urbana-Champaign  9
University of California, Davis     9
Cornell University                 7
Name: count, dtype: int64
```

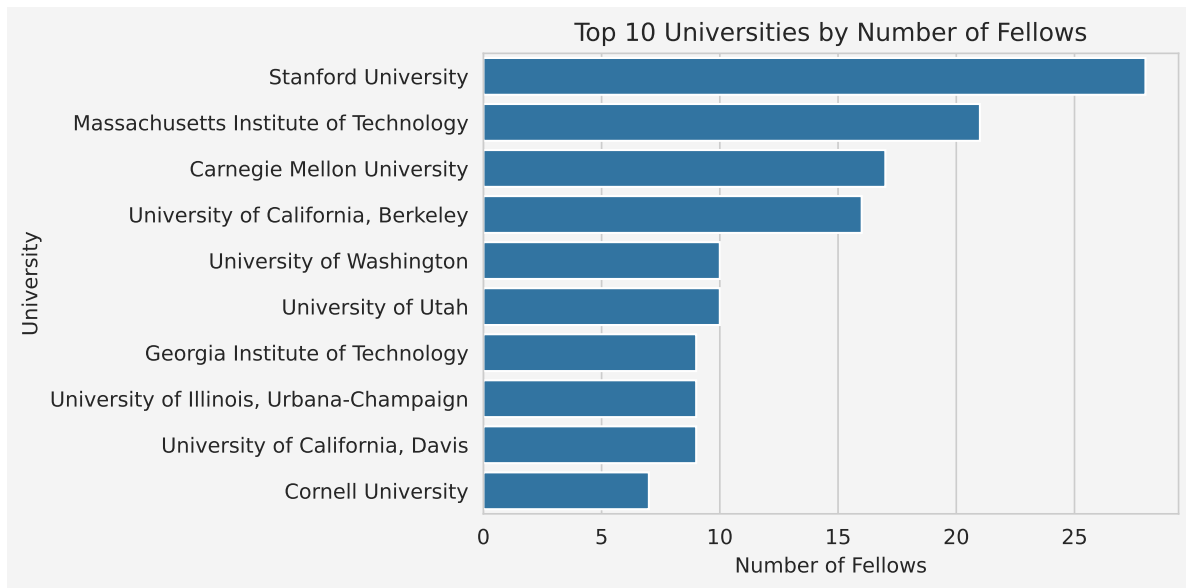
To visualize the award distributions per year,

```
plt.figure(figsize=(9,5))
sns.countplot(x='Year', data=data, order=sorted(data['Year'].unique()))
plt.gca().set_facecolor('#f4f4f4')
plt.gcf().patch.set_facecolor('#f4f4f4')
plt.title('Number of Fellows Per Year')
plt.show()
```

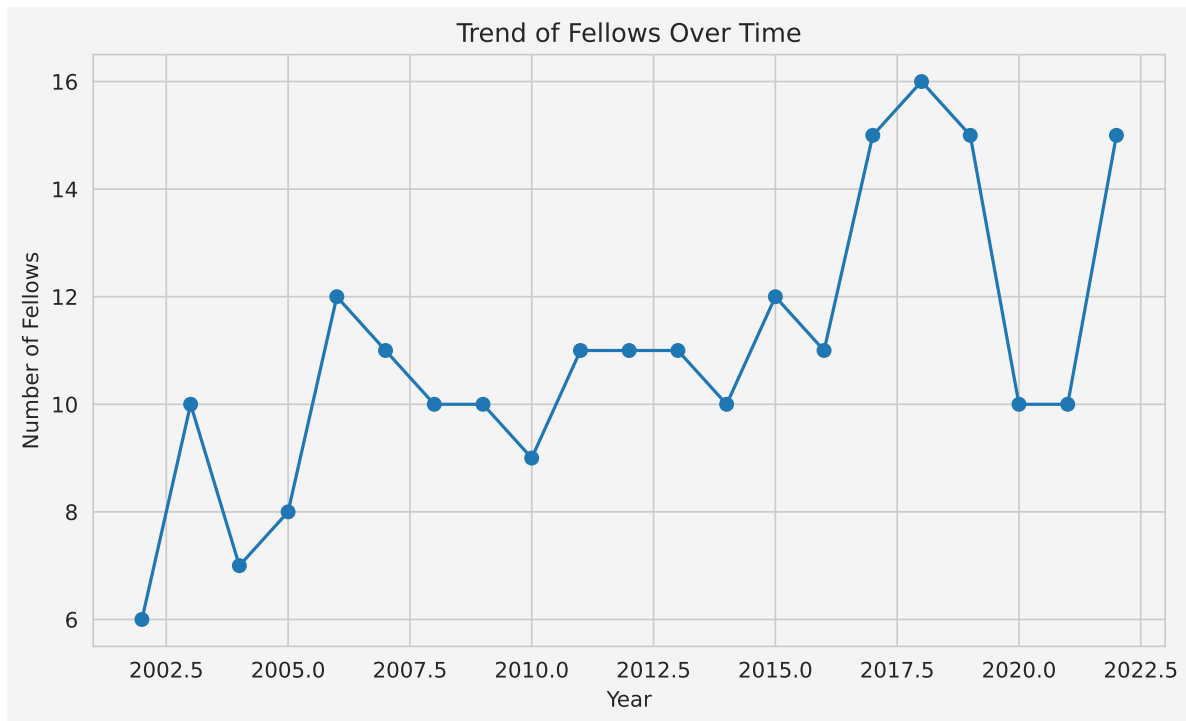
Top 10 universities visualization

```
plt.figure(figsize=(6,4))
top_universities = data['Institute'].value_counts().head(10)
sns.barplot(y=top_universities.index, x=top_universities.values)
plt.gca().set_facecolor('#f4f4f4')
plt.gcf().patch.set_facecolor('#f4f4f4')
plt.title('Top 10 Universities by Number of Fellows')
plt.xlabel('Number of Fellows')
plt.ylabel('University')
plt.show()
```



Trend over time

```
plt.figure(figsize=(9,5))
data['Year'] = data['Year'].astype(int)
yearly_trend = data.groupby('Year').size()
yearly_trend.plot(kind='line', marker='o')
plt.gca().set_facecolor('#f4f4f4')
plt.gcf().patch.set_facecolor('#f4f4f4')
plt.title('Trend of Fellows Over Time')
plt.xlabel('Year')
plt.ylabel('Number of Fellows')
plt.show()
```



This is just a simple example of collecting data through webscraping. This `BeautifulSoup` has endless potentials to use in many projects to collect the data that are not publicly available in cleaned or organized form. Thank you for reading.

References

- [Fisher, R. A.. \(1988\). Iris. UCI Machine Learning Repository.](#)

Share on



You may also like