**TODAY’s HOT NEWS**

**FINAL PROJECT FROM QMNB COURSE**

**Purpose**

Web scraping is one of must-know knowledge as a data analysis to collect the data efficiently in these days, because it helps access good data sources and save valuable time before going to data analysis phase. From this significant importance of accessing proper data sources, my project brings the today’s headlines from two different news websites (BBC and CNN). It provides the web digital news service such as google news. This project is, especially, highly relevant to my interest for my future career, which is related to web development, so that I decided to go more further to present this project on the website.

**Tasks**

1. Scrap the main headlines from each websites   
2. Save the headlines into CSV file  
3. Present the headlines on the website for users

**Process**

* **Selenium Webdriver, BeautifulSoup**

Instead of Python Basic Library *urllib*, I used more powerful library called *BeautifulSoup*, which we handled also in lecture. However, only with *BeautifulSoup*, I couldn't get some elements, what I want to scrap, which were used by Java Script. So, I used *selenium* with *webdriver* for keeping stronger and more stable.

Only disadvantage of this selenium webdriver was that it takes long time to load pages. Since the website was opened by selenium for collecting the data, it was also quite burdensome to wait until the end of whole process.

Here was my idea, that users don’t see the whole process from opening each website browsers by selenium to closing. The option has to be also added to not open the browser every time, when the *selenium* works with *webdriver* as following:

|  |
| --- |
| op = webdriver.ChromeOptions()  op.add\_argument('headless')  driver = webdriver.Chrome(ChromeDriverManager().install(), options=op) |

* **Analysis of HTML Structures**

Each website has different HTML Structures, so that I had to spend a lot of times at first to analyzes about the structure and how I can get most efficiently to get the results without duplication or with using a lot of for loops. And I had a difficulty to use the function such as *find\_all*, because sometimes I couldn’t use again *find\_all* after using it in same flat. I found out later that *find\_all* can be used again in for loop under itself.

* + **CNN Website**

|  |
| --- |
| articles = page\_soup.find\_all("div", {"class":"cd\_\_content"})  for article in articles:  for a in article.find\_all('a', href=True):  ...  headline = article.find("span", {"class":"cd\_\_headline-text"})  headline\_text = headline.get\_text()  headlines.append(headline\_text) |

* + **BBC Website**

|  |
| --- |
| s1 = page\_soup.find\_all("a", {"class":"media\_\_link"})  for s in s1:  href = s.attrs['href']  links.append(URL\_new+href)  ...  #strip for getting rid of unnecessary spaces  headline\_text = s.get\_text().strip()  headlines.append(headline\_text) |

* **Make Lists and Dictionaries**

I learned from this part that the *for* loops can makes everything way easier and also difficult your tasks depends on where you put elements (inside or outside of for loops). Because, this was the most time-consuming part for me due to small mistake, that I put the empty lists [] for the data sets not outside of for loop, but inside of for loops.

After solving this problem, I put each list from URL and news headlines into dictionary by sorting keys ‘headline’ and ‘url’ as following:

|  |
| --- |
| links\_with\_text = []  headlines = []  ...  for article in articles:  ...  dict\_cnn = [{'headline' : headlines, 'url' : links\_with\_text} for headlines,  links\_with\_text in zip(headlines,links\_with\_text)] |

There was a problem for scraping the URL from each headline, because they have different patterns. Some of them starts only short version of URL without main website address such as <https://www.cnn.com/> and some of them has this main address included. It was necessary to make the standard URL address for all with if-statement.

* **Saving a list of the dictionaries into CSV files**

Keys and Values are saved in different columns. According to Python 3.X, It was necessary to use the option ‘w’, newline= ‘ ’ for eliminating the blanks after each row.

|  |
| --- |
| keys = dict\_cnn[0].keys()  a\_file = open('output\_cnn.csv', 'w', newline='')  dict\_writer = csv.DictWriter(a\_file, keys)  dict\_writer.writeheader()  dict\_writer.writerows(dict\_cnn)  a\_file.close() |

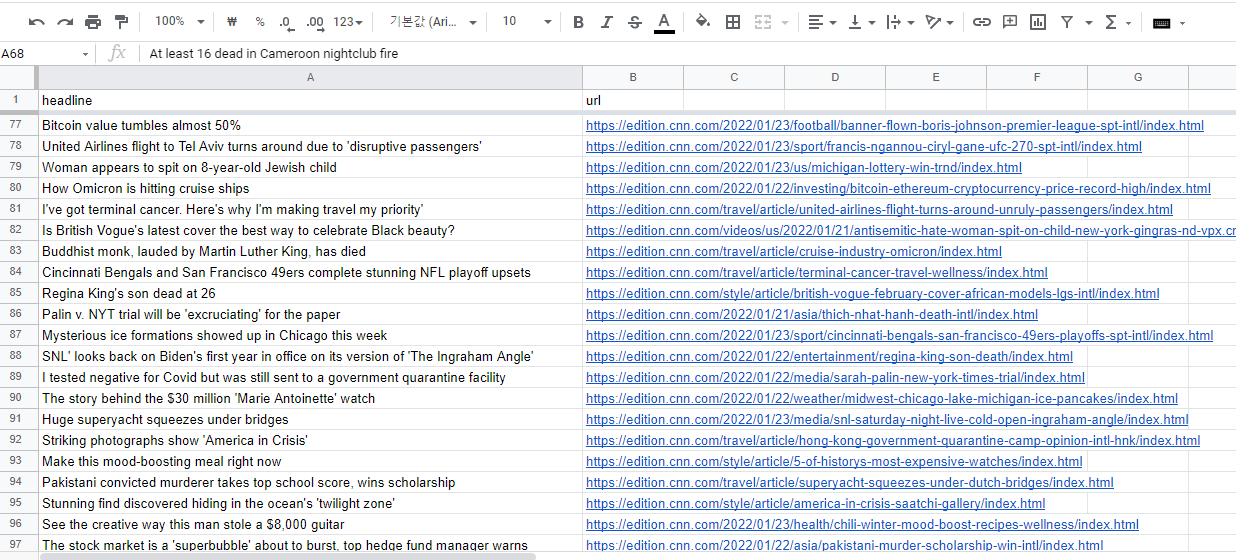
* **Merge two different CSV files into one combined CSV file**

Figure 1 Combined CSV file in Excel Sheet

* **Merge two different CSV files into one combined CSV file**