Web Scraping Project Report

Kashfia Salony, 456294
Marwan Otrok, 455363
Sugarbayar Enkhbayar, 456296

June 8, 2023

1. Description of the Webpage

Our team chose the https://concertful.com/area/poland/ website. There are One hundred upcoming concerts listed in Poland. If you click, Concert name, you will see a new window. And that window shows us detailed information about chosen concerts such as Performer, Address, Club name, Date, and Genre. In this project, We will scrape the following information for each concert. (Performer, Venue, Address, Date, Genre)

2. Description of 3 approaches to scrape

a. Beautiful Soup

We used BeautifulSoup to scrape 100 pages from the website. The script begins by importing the necessary libraries: urllib.request, BeautifulSoup from bs4, re for regular expressions, time for measuring the duration, and pandas for data manipulation. The links list is created by appending the relevant URLs to the base URL. For each link, it opens the URL, parses the HTML, and extracts relevant information using BeautifulSoup. We scraped performer, venue, address, date, and genre information from each 100 links.

The extracted information is stored in a dictionary called “add”. The add dictionary is converted into a DataFrame using pd.DataFrame(), and then concatenated with the existing df DataFrame using pd.concat(). After scraping all the links, the final df DataFrame contains the concert data. The df.to_csv() function is used to save the DataFrame as a CSV file named 'df_beautifulsoup.csv'. In order to measure time performance, we calculated the difference between start time and end time.

b. Scrapy

After creating a new project, we defined a Scrapy Spider named "ConcertSpider" that starts at the URL "https://concertful.com/area/poland/". The Spider's "parse" method is responsible for extracting the concert links from the HTML response using an XPath expression. It creates a new Scrapy Item object called "Link" and assigns the extracted link to the "link" field. The "Link" object is then yielded to be processed by Scrapy.

Then, we defined another Scrapy Spider named "ConcertinfoSpider" that reads a list of URLs from previously exported CSV file named "concertlist.csv" and uses those links as the starting points for the spider. The Spider's "parse" method is responsible for extracting concert information from the HTML responses, which defines XPath expressions to extract the performer, venue, address, date, and genre information from the response. The extracted data is then stored in a Scrapy Item object named "ConcertsInfo" where each field corresponds to a specific piece of information.

Finally, the code measures the execution time of the scraping process for both spiders and prints the duration (18 -20 sec)

c. **Selenium**

Firstly, we import the necessary packages. Then we set up Chromedriver and get information from our website. After that, we created an empty list. Then, we gathered all "href" from every 100 links. Then, we gathered Performer, Venue, Address, Date, and Genre information from every 100 links using Xpath. Then finally we saved our result as df_selenium.csv. In order to measure time performance, we calculated the difference between start time and end time. To run selenium code, we use the following commands on PowerShell.

- WSL, python3 Test_WS1.py

3. Analysis of the output

a. Analysis based on scraped data

We have 100-row data. And Table 1 shows the unique value of each variable. So we have data for 63 types of Performers, 39 types of Clubs, 10 types of addresses, and 11 types of genres.

Table 1. Count of variables

Variables	Count
Performer	63
Venue	39
Address	10
Genre	11

Table 2 shows the top 5 performers. In other words, the Disney Princess concert will be held 9 times.

Table 2. Top 5 performers

Rank	Performer	Count
1	Disney Princess: Concert	9
2	Airbourne	3
3	Anti-Flag	3
4	Gogol Bordello	3
5	Kevin Morby	3

Table 3 shows the top 5 provinces. In other words, 42 concerts will be held in Warsaw. And 21 concerts will be held in Krakow.

Table 3. Top 5 provinces

Rank	Provinces	Count
1	Warsaw	42
2	Krakow	21
3	Wroclaw	8
4	Gdansk	7
5	Poznan	7

Table 4 shows the top 5 Genres. 28 Hard Rock concerts will be held, and 18 Alternative Rock concerts will be held.

Table 4. Top 5 Genres

Rank	Genre	Count
1	Hard Rock / Heavy Metal	28
2	Alternative Rock	18
3	Pop Music / Soft Rock	16
4	Punk / Garage Rock	12
5	Electronic Music / Dance	7

b. Analysis based on performance analysis
   Scrapy is the fastest approach. The Selenium Is the slowest approach. Scrapy is almost 10 times faster than Selenium.

Table 5. Time performance of approaches

Approaches	Time performance
BeautifulSoup	130 sec
Scrapy	18-20 sec
Selenium	180-200 sec

4. Participation of the members

Table 6. Participation of members

Members	Participation
Kashfia Salony	Worked on the BeautifulSoup approach
Marwan Otrok	Worked on the Scrapy approach
Sugarbayar Enkhbayar	Worked on the Selenium approach

5. Appendix

Table 6. Python code BeautifulSoup

```
from urllib import request
from bs4 import BeautifulSoup as BS
import re
import time

start_time=time.time()

url='https://concertful.com/area/poland/'
html = request.urlopen(url)
bs = BS(html.read(), 'html.parser')
tags = bs.find_all('a')
tags=tags[3:103]
links = ['https://concertful.com/' + tag['href'] for tag in tags]

df = pd.DataFrame({'Performer':[], 'Venue':[],'Address':[],'Date':[],'Genre':[]})
for link in links:
    html = request.urlopen(link)
    bs = BS(html.read(), 'html.parser')
    Performer = bs.find('span',{'class':'performers'}).find('a').text
    Venue = bs.find('span',{'class':'venue_name'}).text
    Address = bs.find('span',{'class':'address'}).find('a').text
    Date=bs.find('th',string='Date:').next_sibling.next_sibling.text.strip()
    Date=Date.replace('\t', '')
    Date=Date.replace('\n', '')
    Genre = bs.find('th',string='Genre:').next_sibling.next_sibling.text
    add = {'Performer':Performer, 'Venue':Venue,'Address':Address,'Date':Date,'Genre':Genre}
    df = df.append(add, ignore_index = True)
df.to_csv('df_beautifulsoup.csv')

end_time=time.time()
duration=end_time-start_time
print("Time duration: ", duration)
```

Table 7. Python code Selenium

```
from genericpath import exists
from pickle import TRUE
from tracemalloc import start
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
import time
import pandas as pd
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.common.exceptions import StaleElementReferenceException
from selenium.common.exceptions import NoSuchElementException
from selenium import webdriver

start_time=time.time()

path = "/mnt/c/Users/nomin/chromedriver_linux64/chromedriver"
ser = Service(path)
driver = webdriver.Chrome( service=ser)
### Following page is the starting page to scrap
url='https://concertful.com/area/poland/'
driver.get(url)
href_links = []

elems = driver.find_elements(by=By.XPATH, value="//a[@href]")
for elem in elems[3:103]:
    l = elem.get_attribute("href")
    if l not in href_links:
        href_links.append(l)

Performer_list=[]
Venue_list=[]
Address_list=[]
Date_list=[]
Genre_list=[]
#elems
for i in href_links:
    driver.get(i)

Performer=driver.find_element(by=By.XPATH,value='/html/body/div/div[2]/div[1]/div[1]/div/table/tbody/tr[
1]/td/span/a/abbr')
    Performer=Performer.text
    Performer_list.append(Performer)

Venue=driver.find_element(by=By.XPATH,value='/html/body/div/div[2]/div[1]/div[1]/div/table/tbody/tr[2]/t
d/span[1]')
    Venue=Venue.text
    Venue_list.append(Venue)

Address=driver.find_element(by=By.XPATH,value='/html/body/div/div[2]/div[1]/div[1]/div/table/tbody/tr[2]
/td/span[2]/abbr[1]/a')
    Address=Address.text
    Address_list.append(Address)

Date=driver.find_element(by=By.XPATH,value='/html/body/div/div[2]/div[1]/div[1]/div/table/tbody/tr[3]/td')
    Date=Date.text
    Date_list.append(Date)

Genre=driver.find_element(by=By.XPATH,value='/html/body/div/div[2]/div[1]/div[1]/div/table/tbody/tr[4]/td
')
    Genre=Genre.text
    Genre_list.append(Genre)
```

```
        #driver.quit()
        dictionary = {'Performer': Performer_list,'Venue':Venue_list,
              'Address': Address_list,'Date':Date_list,
              'Genre': Genre_list}
        df = pd.DataFrame(dictionary)
        df.to_csv('df_selenium.csv')

        end_time=time.time()
        duration=end_time-start_time
print("Time duration: ", duration)
```

Table 8. Python code Scrapy (Part 1)

```
import scrapy
import pandas as pd
import time

page_limit = True

if page_limit == True:
    pages = 10
else: pages = 1

start = time.time()


class Link(scrapy.Item):
    link = scrapy.Field()

class ConcertSpider(scrapy.Spider):
    name = "concert"
    allowed_domains = ["https://concertful.com/"]
    start_urls = ["https://concertful.com/area/poland/"]
    for i in range(pages):
        start_urls.append('https://concertful.com/area/poland/' + "/?page=" + str(i))

    def parse(self, response):
        xpath = "//span[@class = 'eventName']/a/@href"

        selection = response.xpath(xpath)
        for s in selection:
            l = Link()
            l['link'] = "https://concertful.com" + s.get()

            yield l

        end = time.time()
        print('Here:', end - start)
```

Table 9. Python code Scrapy (Part 2)

```
   import scrapy
import time

start = time.time()

class ConcertsInfo(scrapy.Item):

    Performer = scrapy.Field()
    Venue = scrapy.Field()
    Address = scrapy.Field()
    Date = scrapy.Field()
    Genre = scrapy.Field()


class ConcertinfoSpider(scrapy.Spider):
    name = "concertInfo"
    allowed_domains = ["https://concertful.com/"]
    try:
        with open("concertlist.csv", "rt") as f:
            start_urls = [url.strip() for url in f.readlines()][1:]
    except:
        start_urls = []


    def parse(self, response):
        r = ConcertsInfo()

        Performer_xpath = '//span[@class = "performers"]//a/abbr/text()'
        Venue_xpath = '//span[@class = "venue_name"]/text()'
        Address_xpath = '//th[text()="Venue:"]/following-
sibling::td[@class="event_info"]//span[@class="address"]//text()[normalize-space() and not(contains(., ", "))
and not(contains(., ","))]'
        Date_xpath = '//th[contains(text(),"Date:")]/following-sibling::td[@class="event_info"]//text()'
        Genre_xpath = '//th[contains(text(),"Genre:")]/following-sibling::td//text()'



        r['Performer'] = response.xpath(Performer_xpath).getall()
        r['Venue'] = response.xpath(Venue_xpath).getall()
        r['Address'] = [address.replace('\t', '').replace('\n', '').strip() for address in
response.xpath(Address_xpath).extract()]
        r['Date'] = [date.replace('\t', '').replace('\n', '').strip() for date in response.xpath(Date_xpath).extract()]
        r['Genre'] = response.xpath(Genre_xpath).getall()

        yield r

        end = time.time()
        print('Here:', end - start)
```