2.1.web_crawler

October 2, 2023

[1]:
```
#%% Get the content of a page using the requests library
import requests
mywebpage_url='https://www.sis.uta.fi/~tojape/'
#mywebpage_url='https://www.tuni.fi/en/'
mywebpage_html=requests.get(mywebpage_url)
```

[2]:
```
#%% Parse the HTML content using beautifulsoup
import bs4
mywebpage_parsed=bs4.BeautifulSoup(mywebpage_html.content,'html.parser')
```

[3]:
```
#%% Get the text content of the page
def getpagetext(parsedpage):
    # Remove HTML elements that are scripts
    scriptelements=parsedpage.find_all('script')
    # Concatenate the text content from all table cells
    for scriptelement in scriptelements:
        # Extract this script element from the page.
        # This changes the page given to this function!
        scriptelement.extract()
    pagetext=parsedpage.get_text()
    return(pagetext)

mywebpage_text=getpagetext(mywebpage_parsed)
# print(mywebpage_text)
```

[4]:
```
# Find HTML elements that are table cells or 'div' cells
tablecells=mywebpage_parsed.find_all(['td','div'])
# Concatenate the text content from all table or div cells
pagetext=''
for tablecell in tablecells:
    pagetext=pagetext+'\n'+tablecell.text.strip()
# print(pagetext)
```

[5]:
```
#%% Find linked pages in Finnish sites, but not PDF or PS files
def getpageurls(webpage_parsed):
    # Find elements that are hyperlinks
    pagelinkelements=webpage_parsed.find_all('a')
    pageurls=[];
```

```
    for pagelink in pagelinkelements:
        pageurl_isok=1
        try:
            pageurl=pagelink['href']
        except:
            pageurl_isok=0
        if pageurl_isok==1:
            # Check that the url does NOT contain these strings
            if (pageurl.find('.pdf')!=-1)|(pageurl.find('.ps')!=-1):
                pageurl_isok=0
            # Check that the url DOES contain these strings
            if (pageurl.find('http')==-1)|(pageurl.find('.fi')==-1):
                pageurl_isok=0
        if pageurl_isok==1:
            pageurls.append(pageurl)
    return(pageurls)
mywebpage_urls=getpageurls(mywebpage_parsed)
# print(mywebpage_urls)
```

[6]:
```
#%% Basic web crawler
def basicwebcrawler(seedpage_url,maxpages):
    # Store URLs crawled and their text content
    num_pages_crawled=0
    crawled_urls=[]
    crawled_texts=[]
    # Remaining pages to crawl: start from a seed page URL
    pagestocrawl=[seedpage_url]
    # Process remaining pages until a desired number
    # of pages have been found
    while (num_pages_crawled<maxpages)&(len(pagestocrawl)>0):
        # Retrieve the topmost remaining page and parse it
        pagetocrawl_url=pagestocrawl[0]
        print('Getting page:')
        print(pagetocrawl_url)
        pagetocrawl_html=requests.get(pagetocrawl_url)
        pagetocrawl_parsed=bs4.BeautifulSoup(pagetocrawl_html.content,'html.
↪parser')
        # Get the text and URLs of the page
        pagetocrawl_text=getpagetext(pagetocrawl_parsed)
        pagetocrawl_urls=getpageurls(pagetocrawl_parsed)
        # Store the URL and content of the processed page
        num_pages_crawled=num_pages_crawled+1
        crawled_urls.append(pagetocrawl_url)
        crawled_texts.append(pagetocrawl_text)
        # Remove the processed page from remaining pages,
        # but add the new URLs
        pagestocrawl=pagestocrawl[1:len(pagestocrawl)]
```

```
        pagestocrawl.extend(pagetocrawl_urls)
    return(crawled_urls,crawled_texts)
mycrawled_urls_and_texts=basicwebcrawler('https://www.sis.uta.fi/~tojape/',10)
mycrawled_urls=mycrawled_urls_and_texts[0]
mycrawled_texts=mycrawled_urls_and_texts[1]
```

```
Getting page:
https://www.sis.uta.fi/~tojape/
Getting page:
https://www.tuni.fi/en
Getting page:
https://www.tuni.fi/en/about-us/faculty-information-technology-and-
communication-sciences
Getting page:
https://www.tuni.fi/en/about-us/computing-sciences
Getting page:
http://cs.aalto.fi/en/
Getting page:
http://www.cis.hut.fi/projects/mi
Getting page:
http://users.ics.aalto.fi/jtpelto/
Getting page:
http://research.ics.aalto.fi/coin/
Getting page:
https://www.tuni.fi/en/study-with-us/computing-sciences-data-
science?navref=curated--list
Getting page:
https://www.tuni.fi/en/study-with-us/computing-sciences-statistical-data-
analytics?navref=curated--list
```

Exercise 2.1: Data acquisition - Building a better web crawler. 1.It can crawl the same page multiple times, if a link on a later crawled page points to the already-crawled page -> Solution: Use a list to store the crawled pages, and check if the page is already crawled before crawling it.

2.It inserts all links from each page in order as pages to be crawled. If some page contains thousands of links, the crawling will crawl those first and may never get to the links from the next page, especially if the total number of pages are limited -> Solution: Restrict the returned links of each page to a value to avoid crawling too many links from one page.

[7]:
```
# %% Improve the web crawler by the solution above
crawled_urls = []  # Store URLs crawled


def improvewebcrawler(seedpage_url, maxpages):
    global crawled_urls
    # Store URLs crawled and their text content
    num_pages_crawled = 0
    # crawled_urls = []
```

```
    crawled_texts = []
    # Remaining pages to crawl: start from a seed page URL
    pagestocrawl = [seedpage_url]
    # Process remaining pages until a desired number
    # of pages have been found
    while (num_pages_crawled < maxpages) & (len(pagestocrawl) > 0):
        # Retrieve the topmost remaining page and parse it
        pagetocrawl_url = pagestocrawl[0]
        print("Getting page:")
        print(pagetocrawl_url)
        pagetocrawl_html = requests.get(pagetocrawl_url)
        pagetocrawl_parsed = bs4.BeautifulSoup(pagetocrawl_html.content, "html.
↪parser")

        ### Condition for not overloading the crawled_urls

        if pagetocrawl_url not in crawled_urls:
            # Get the text and URLs of the page
            pagetocrawl_text = getpagetext(pagetocrawl_parsed)
            pagetocrawl_urls = getpageurls(pagetocrawl_parsed)

            ### Restrict the number of urls to be crawled to be 5 or less
            if len(pagestocrawl) + len(pagetocrawl_urls) > 5:
                pagetocrawl_urls = pagetocrawl_urls[0 : 5 - len(pagestocrawl)]

            # Store the URL and content of the processed page
            num_pages_crawled = num_pages_crawled + 1
            crawled_urls.append(pagetocrawl_url)
            crawled_texts.append(pagetocrawl_text)
            # Remove the processed page from remaining pages,
            # but add the new URLs
            pagestocrawl = pagestocrawl[1 : len(pagestocrawl)]
            pagestocrawl.extend(pagetocrawl_urls)
    return (crawled_urls, crawled_texts)
mycrawled_urls_and_texts=improvewebcrawler('https://www.sis.uta.fi/~tojape/',10)
mycrawled_urls=mycrawled_urls_and_texts[0]
mycrawled_texts=mycrawled_urls_and_texts[1]
print(crawled_urls)
```

```
Getting page:
https://www.sis.uta.fi/~tojape/
Getting page:
https://www.tuni.fi/en
Getting page:
https://www.tuni.fi/en/about-us/faculty-information-technology-and-
communication-sciences
Getting page:
https://www.tuni.fi/en/about-us/computing-sciences
```

```
Getting page:
http://cs.aalto.fi/en/
Getting page:
https://research.aalto.fi/
Getting page:
https://ourblogs.aalto.fi/
Getting page:
https://ada.aalto.fi/
Getting page:
https://booking.aalto.fi
Getting page:
https://www.aalto.fi/en/acris-instructions
['https://www.sis.uta.fi/~tojape/', 'https://www.tuni.fi/en',
'https://www.tuni.fi/en/about-us/faculty-information-technology-and-
communication-sciences', 'https://www.tuni.fi/en/about-us/computing-sciences',
'http://cs.aalto.fi/en/', 'https://research.aalto.fi/',
'https://ourblogs.aalto.fi/', 'https://ada.aalto.fi/',
'https://booking.aalto.fi', 'https://www.aalto.fi/en/acris-instructions']
```