Lesson 5- Crawl and scrape

May 6, 2019

1 Lesson 5 - Crawl and Scrape

1.1 Making the request

1.1.1 Using 'requests' module

Use the requests module to make a HTTP request to http://www.tripadvisor.com - Check the status of the request - Display the response header information

```
In [ ]: import requests
        url = 'http://www.tripadvisor.com/'
        response = requests.get(url)

        print(response.status_code)
        #print(response.headers)
```

1.1.2 Get the '/robots.txt' file contents

```
In [ ]: import requests
        url = 'http://www.tripadvisor.com/robots.txt'
        response = requests.get(url)

        if response.status_code == 200:
            print(response.status_code)
            print(response.text)
        else:
            print('Failed to get a response from the url. Error code: ',resp.status_code )
```

1.1.3 Get the HTML content from the website

```
In [ ]: import requests
        url = 'http://tripadvisor.com'
        response = requests.get(url)

        if response.status_code == 200:
            print(response.status_code)
            print(response.text)
        else:
            print('Failed to get a response from the url. Error code: ',resp.status_code )
```

1.2 Scraping websites

Sometimes, you may want a little bit of information - a movie rating, stock price, or product availability - but the information is available only in HTML pages, surrounded by ads and extraneous content.

To do this we build an automated web fetcher called a crawler or spider. After the HTML contents have been retrived from the remote web servers, a scraper parses it to find the needle in the haystack.

1.2.1 BeautifulSoup Module

The bs4 module can be used for searching a webpage (HTML file) and pulling required data from it. It does three things to make a HTML page searchable- * First, converts the HTML page to Unicode, and HTML entities are converted to Unicode characters * Second, parses (analyses) the HTML page using the best available parser. It will use an HTML parser unless you specifically tell it to use an XML parser * Finally transforms a complex HTML document into a complex tree of Python objects.

This module takes the HTML page and creates four kinds of objects: Tag, NavigableString, BeautifulSoup, and Comment. * The ***BeautifulSoup*** object itself represents the webpage as a whole * A ***Tag*** object corresponds to an XML or HTML tag in the webpage * The ***NavigableString*** class to contains the bit of text within a tag

Read more about BeautifulSoup : https://www.crummy.com/software/BeautifulSoup/bs4/doc/

```
In [ ]: <h1 id="HEADING" property="name" class="heading_name   ">
            <div class="heading_height"></div>
             "
             Le Jardin Napolitain
             "
        </h1>
```

Step 1: Making the soup First we need to use the BeautifulSoup module to parse the HTML data into Python readable Unicode Text format.

Let us write the code to parse a html page. We will use the trip advisor URL for an infamous restaurant - https://www.tripadvisor.com/Restaurant_Review-g187147-d1751525-Reviews-Cafe_Le_Dome-Paris_Ile_de_France.html

```
In [ ]: import requests
        from bs4 import BeautifulSoup
        scrape_url = 'https://www.tripadvisor.com/Restaurant_Review-g187147-d1751525-Reviews-Ca
        response = requests.get(scrape_url)
        print(response.status_code)

        if response.status_code == 200:
            soup = BeautifulSoup(response.text, 'html.parser') # Soup
            print(soup.prettify)
```

Step 2: Inspect the element you want to scrape In this step we will inspect the HTML data of the website to understand the tags and attributes that matches the element. Let us inspect the HTML data of the URL and understand where (under which tag) the review data is located.

```
In [ ]: <div class="entry">
            <p class="partial_entry">
            Popped in on way to Eiffel Tower for lunch, big mistake.
            Pizza was disgusting and service was poor.
            Its a shame Trip Advisor dont let you score venues zero....
            <span class="taLnk ulBlueLinks" onclick="widgetEvCall('handlers.clickExpand',event
            </span>
            </p>
        </div>
```

Step 3: Searching the soup for the data Beautiful Soup defines a lot of methods for searching the parse tree (soup), the two most popular methods are: find() and find_all().

The simplest filter is a tag. Pass a tag to a search method and Beautiful Soup will perform a match against that exact string.

Let us try and find all the < p > (paragraph) tags in the soup:

```
In [ ]: import requests
        from bs4 import BeautifulSoup

        def scrapecontent(url):
            """This function parses the HTML page representing the url using the BeautifulSoup
            and returns the created python readable data structure (soup)"""
            scrape_response = requests.get(url)
            print(scrape_response.status_code)

            if scrape_response.status_code == 200:
                soup = BeautifulSoup(scrape_response.text, 'html.parser')
                return soup
            else:
                print('Error accessing url : ',scrape_response.status_code)
                return None

        def main():
            scrape_url = 'https://www.tripadvisor.com/Restaurant_Review-g187147-d1751525-Revie
            ret_soup = scrapecontent(scrape_url)
            if ret_soup:
                for review in ret_soup.find_all('p', class_='partial_entry'):
                    print(review.text) #We are interested only in the text data, since the rev
        main()
```

Step 4: Enable pagination Automatically access subsequent pages

```
In [ ]: import requests
        from bs4 import BeautifulSoup

        def scrapecontent(url):
            """This function parses the HTML page representing the url using the BeautifulSoup
            and returns the created python readable data structure (soup)"""
            scrape_response = requests.get(url)
            print(scrape_response.status_code)

            if scrape_response.status_code == 200:
                soup = BeautifulSoup(scrape_response.text, 'html.parser')
                return soup
            else:
                print('Error accessing url : ',scrape_response.status_code)
                return None

        def main():
            page_no = 0

            while(page_no < 60):
                scrape_url = 'https://www.tripadvisor.com/Restaurant_Review-g187147-d1751525-R
                ret_soup = scrapecontent(scrape_url)
                if ret_soup:
                    for review in ret_soup.find_all('p', class_='partial_entry'):
                        print(review.text) #We are interested only in the text data, since the
                page_no = page_no + 10

        main()
```

Using yesterdays sentiment analysis code and the corpus of sentiment found in the word_sentiment.csv file, calculate the sentiment of the reviews.

```
In [ ]: #Enter your code here
```

Expanding this further To add additional details we can inspect the tags further and add the reviewer rating and reviwer details.

```
In [ ]: import requests
        from bs4 import BeautifulSoup
        def scrapecontent(url):
            """This function parses the HTML page representing the url using the BeautifulSoup
            and returns the created python readable data structure (soup)"""
            scrape_response = requests.get(url)
            print(scrape_response.status_code)

            if scrape_response.status_code == 200:
                soup = BeautifulSoup(scrape_response.text, 'html.parser')
                return soup
            else:
                print('Error accessing url : ',scrape_response.status_code)
                return None

        def main():
            scrape_url = 'https://www.tripadvisor.com/Restaurant_Review-g187147-d1751525-Revie
```

```
    ret_soup = scrapecontent(scrape_url)
    if ret_soup:
        for rev_data in ret_soup.find_all('div', class_= 'review-container'):
            date = rev_data.find('span', class_ ='ratingDate')# Get the date if the re
            print(date.text)
            review = rev_data.find('p') # Get the review text
            print(review.text)
            rating = rev_data.find('span',class_='ui_bubble_rating') #Get the rating o
            print(int(rating['class'][1][7:])/10)
main()
```

Using the review data and the ratings available is there any way we can improve the corpus of sentiments "word_sentiment.csv" file?