Pràctica 1 – 2.951 Tipologia i cicle de vida de les dades

Novembre 2022

Autors:

Oriol Caravaca Müller

Pau Casanova Pedrol

1. Context

En aquesta primera pràctica de l'assignatura Tipologia i cicle de vida de les dades s'ha implementat una tènica de web scraping mitjançant una web crawler o aranya, per a generar una base de dades d'enfonsaments de vaixells a tot el món al llarg de la història a partir de la pàgina "List of shipwrecks" de Wikipedia. La font original es pot consultar en aquest enllaç:

https://en.wikipedia.org/wiki/Lists_of_shipwrecks

Com sabem, Wikipedia es una famosa enciclopèdia lliure editada de manera lliure, gestionada per la Fundació Wikimedia, una organització sense ànim de lucre que es finança a partir de donacions.

2. Títol

El títol del projecte és Shipwrecks, el dataset generat a partir del procés de web scraping té el nom de shipWrecks.csv.

3. Descripció del dataset

El dataset està format per 4460 instàncies cadascuna de les quals correspon al naufragi d'un vaixell. La informació es troba descrita en anglès. Cada instància conté 11 atributs, que especifiquen diferents dades sobre el naufragi: nom del vaixell, bandera (país del vaixell), data del naufragi, tipus de nau, continent, regió, país, coordenades del naufragi, notes (un breu comentari sobre les circumstàncies del naufragi) i la imatge de l'embarcació.

El dataset resultant conté les característiques anteriorment descrites de tots els naufragis registrats en aquesta pàgina de Wikipedia.

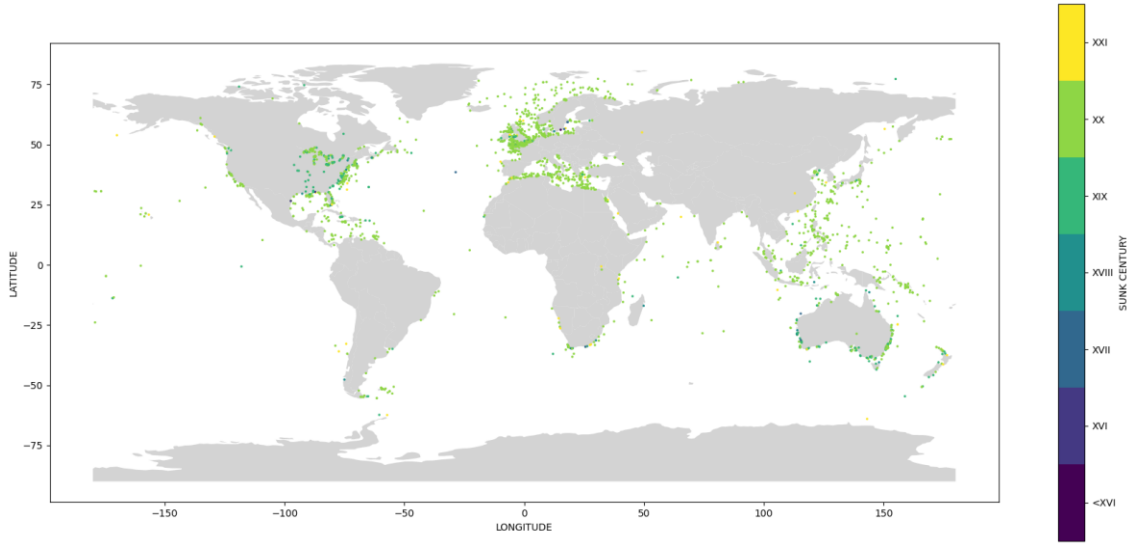
4. Representació gràfica

Una manera de representar gràficament les dades generades a partir del procés de Web Scraping és fer un diagrama de dispersió amb la localització dels naufragis al llarg i ample del món, diferenciades pel segle en que va ocórrer el naufragi.

Després d'un procés de neteja de les dades generades amb el web scraping, el codi del qual també es pot trobar al repositori GitHub en aquest enllaç:

https://github.com/biglool/ShipWrecks/blob/main/source/shipwreckPlot.py

El resultat del diagrama de dispersió és aquest:



Gràcies a aquesta visualització podem veure les zones amb més densitat de naufragis, com el canal de la Manxa al nord d'Europa o la costa est dels Estats Units. També podem observar que la majora d'instàncies registrades a la pàgina de Wikipedia són naufragis ocorreguts en el segle XX.

5. Contingut

Com hem dit, el dataset generat conté diversos camps sobre cada naufragi registrat a la pàgina List of shipwrecks de Wikipedia. Els camps són els següents:

- (string): index de la instància segons ordre d'extracció
- SHIP (string): conté el nom de l'embarcació.
- FLAG (string): conté el nom del país que dóna bandera a l'embarcació.
- SUNK DATE (date): conté la data de naufragi de l'embarcació.
- VESSEL TYPE (string): Conté el nom del tipus d'embarcació.
- ZONA1 (string): Conté el nom del continent on va succeir el naufragi.
- ZONA2 (string): Conté la regió on va succeir el naufragi.
- ZONA3 (string): Conté el nom del país on va succeir el naufragi.
- ZONA4 (string): Conté el nom de la subregió on va succeir el naufragi.
- COORDINATES (string): Conté les coordenades del lloc geogràfic on va succeir el naufragi.

- NOTES (string): conté una breu descripció escrita sobre les circumstàncies del naufragi.
- IMAGE (): imatge de l'embarcaci

El període de temps de les dades va des del 2200 AC, que es la data del naufragi més antic registrat per arqueòlegs, fins al 16 de novembre de 2022, que és la data de l'últim naufragi reportat a Wikipedia.

6. Propietari

El propietari de les dades originals és Wikimedia Foundation, que va ser creada per Jimmy Wales i Larry Sanger l'any 2001. Els editors de les pàgines de Wikipedia tenen els drets d'autor sobre els textos i les imatges, i la majoria es troben sota la llicència Creative Commons Attribution-ShareAlike License 3.0 i la llicència GNU Free Documentation License. Les condicions de reutilització de les dades sota aquestes llicències es poden trobar en aquests enllaços:

https://en.wikipedia.org/wiki/Wikipedia:Text_of_the_GNU_Free_Documentation_License

https://en.wikipedia.org/wiki/Wikipedia:Text_of_Creative_Commons_Attribution-ShareAlike_3.0_Unported_License

Aquesta llicència permet copiar, distribuir i transmetre el material, i adaptar-lo per qualsevol propòsit, fins i tot comercial, amb les següents condicions:

-Atribució: s'ha d'atribuir el material tal i com especifica l'autor, per mitjà amb un hyperlink o una URL a les pàgines que s'estan reutilitzant.

-Share alike: el material generat a partir de les dades obtingudes ha de tenir una llicència igual o equivalent a les dades originals

Per tant, per donar compliment als principis legals de l'ús i la distribució de les dades extretes de Wikipedia n'hi ha prou amb publicar el projecte desenvolupat sota una llicència equivalent, com s'exposa en el punt 8 d'aquest document i atribuir el material al seu autor per mitjà d'un hipervincle a la pàgina original.

D'altra banda alguns dels projectes similars als nostres que hem trobat són aquests:

-Wikipedia Web Scraper: Web scraper que recopila llançaments de coets a l'òrbita terrestre i compta el nombre de llançaments assolits amb èxit o fracassats per data.

https://github.com/bvyshali/Wikipedia-Web-Scraper

-Football clubs logo scraper: fa servir python i Beautifulsoup per extraure de Wikipedia imatges dels escuts d'equips de diferents lligues nacionals de futbol.

https://github.com/milosmladenovic5/football_clubs_logo_scraper

-soccerdata-scraper: extreu dades de wikipedia sobre equips de les principals lligues de futbol europees i fa visualitzacions interactives.a partir d'aquesta informació.

https://github.com/zz-xx/soccerdata-scraper

-Basketball Players Statistics: extrau dades estadístiques de jugadors de bàsquet a wikipedia i genera un gràfic comparatiu:

https://github.com/SDibla/PYTHON-Basket_Players_stats

7. Insipiració

Recopilar aquesta llista de pàgines de Wikipedia sobre naufragis en un sol dataset amb les dades més destacades de cadascun pot tenir diverses utilitats pràctiques. D'una banda pot permetre visualitzar de manera immediata diverses característiques sobre els naufragis per facilitar la comprensió del fenòmen. Podem observar en quines zones dels oceans hi ha una concentració més alta de naufragis, l'evolució temporal dels naufragis al llarg de cada segle i dècada, quins són els països d'origen amb més naufragis, etc. Des d'un punt de vista arqueològic, també pot servir per determinar si algun tipus de vaixell ha patit més naufragis al llarg de la història, o detectar localitzacions amb possibles restes arqueològiques d'interès històric.

8. Llicència

Tal i com s'especifica en la llicència del contingut de wikipedia, convé optar per una llicència equivalent o similar a la Creative Commons Attribution-ShareAlike License 3.0 pel dataset generat. Per tant nosaltres escollirem la mateixa llicència pel projecte, CC BY-SA 3.0, les condicions de la qual estan descrites en el punt 6 d'aquesta pràctica.

9. Codi

Per fer aquest projecte, que s'ha desenvolupat en python, s'ha utilitzat el framework de codi obert Scrapy. Scrapy es framework que ens permet crear web-crawlers, tot i gestionant crides asincròniques, definició dels User-Agents, retrasos entre crides, gestió de profunditat, etc.

El codi utilitzat per al projecte de web scraping s'ha carregat al repositori GitHub creat per a aquesta pràctica, amb el nom ShipWrecks. El programa sencer està distribuït en diferents fitxers, a continuació es defineixen els fitxer que s'han editat i que contenen les parts funcionals del programa:

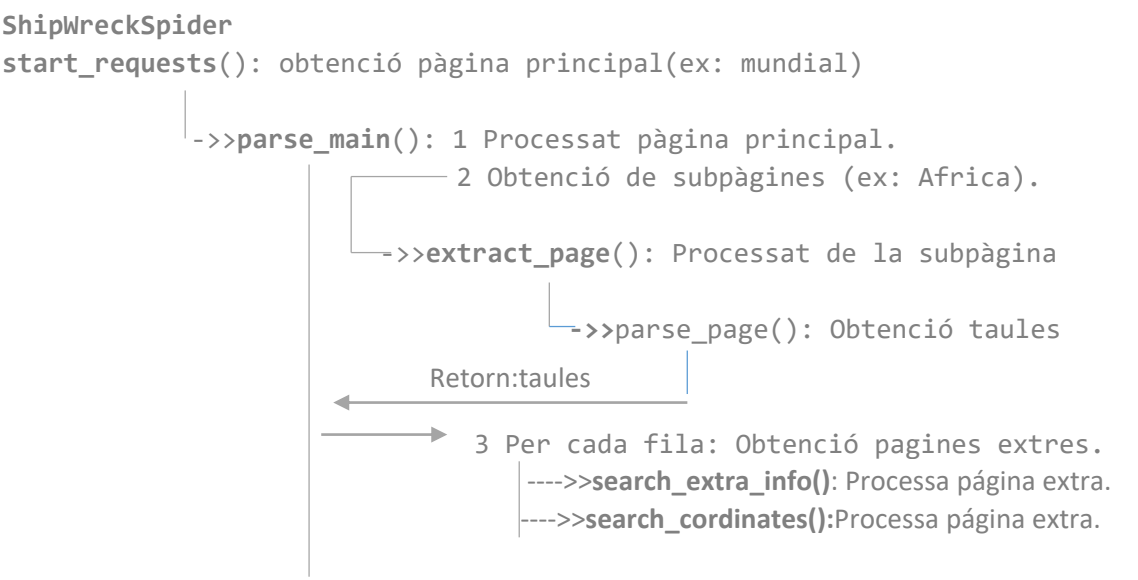
https://github.com/biglool/ShipWrecks:

- **source/requirements.txt**: Conte la informació dels requeriments de l'entorn.
- **source/main.py**: Punt de entrada del programa. Inicia el procés de webscraping.
- **source/shipWrecks/settings.py**: Conte la configuració especifica per a les bones practiques del webscaping. Com per exemple el User-agent.
- **source/shipWrecks/spiders/shipwreck.py**: Conte la implementació de la classe que s'encarga de fer el webscraping
- **source/postprocess/cleaning.py**: Conte la implementació de funcions per netejar les dades.
- **source/visualitzacio/shipwreckPlot.py**: Conte la implementació de funció que s'encarrega de generar la visualització.

Per tal d'entendre el funcionament del programa es defineixen els punts mes importants del fitxer **shipwreck.py**:

9.1 shipwreck.py

Aquesta classe conté el codi encarregat de fer el webscraping. Donada una pàgina inicial, de forma esquemàtica les crides són les seüents:



A continuació es mostra part del codi juntament amb algunes anotacions.

Creem la classe principal ShipWreckSpider que conté tots els passos per extraure les dades que volem de la página Lists of shipwrecks de Wikipedia.

```
class ShipWreckSpider(Spider):
```

Definim les URL a partir de les quals fem la petició amb la funció start_requests()

```
    name = 'shipwreck'
    allowed_domains = ['en.wikipedia.org']
    start_urls = ['https://en.wikipedia.org/wiki/Lists_of_shipwrecks']
    base_url='https://en.wikipedia.org'
    taules=[]

    def start_requests(self):
        for u in self.start_urls:
            yield Request(u, callback=self.parse_main)
```

La petició ens retorna tota la informació de la pàgina principal a través de la variable response.

Amb la funció parse_main() filtrem la informació continguda en la variable response, seguint els tags de les dades que volem extraure (en aquest cas els links de llistes 'By location').

```
    #procesem la pagina principal
    def parse_main(self, response):

        data=response.xpath("//a[not(ancestor::table) and starts-
with(@title, 'List of shipwrecks')]")
        for link in data:
            next_page_url =self.base_url+link.xpath('@href').extract()[0]
            yield Request(next_page_url,self.extract_page)
```

Amb la funció extract_page (), de cada llista de naufragis per regió (on apareixen les taules amb tots els naufragis registrats), extraiem el títol de cada taula (que defineix la regió dels naufragis que en formen part) i el contingut de les taules amb cadascun dels naufragis, els seus atributs (nom del vaixell, bandera, data del naufragi, coordenades, etc.), i el títol de cada columna.

```
    #procesem cada subpagina
    def extract_page(self, response):

        #EXTRAIEM
        main_title=response.xpath("//span[contains(@class, 'mw-page-
title-main')]/text()[1]").extract()[0]
        data=response.xpath("//table[contains(@class,
'wikitable')]/preceding-sibling::*[self::h2 or self::h3 or self::h4][1] |
//table[contains(@class, 'wikitable')]")
        data_titles=response.xpath("//span[contains(@class, 'mw-
headline') and not(contains(@id, 'Further_reading')) and
not(contains(@id, 'References'))  and not(contains(@id,
'External_links'))]/ancestor::*[self::h2 or self::h3 or self::h4][1]")
        zones =self.expand_zones(data_titles)
        df= self.parse_page(data,main_title,zones)


        #obtenim informacio extra dels links Que hem guardat.  Per cada
fila fem una nevacio del link relleants..
        self.taules.append(df)
```

```
        for index, row in df.iterrows():
            links=BeautifulSoup(str(row["extra_links"]),
'html5lib').find_all('a')

            for link in links:
                url=self.base_url+link.get('href')

                #per cadalinks de les notes busquem una posiple
localitzacio !! no sera perfecte pero solen coincidir.
                if row["COORDINATES"]=="" and row["SHIP"]!=link.text:
                    yield Request(url,self.search_cordinates,
cb_kwargs={'row_index':index, 'taula_index':len(self.taules)-1} )

                # aqui podriem buscar més informació, donat l'scope de la
practica simplement ens guardem el link dela imatge
                # poseteriorment es podrien baixar.
                if row["SHIP"]==link.text :
                    yield Request(url,self.search_image,
cb_kwargs={'row_index':index, 'taula_index':len(self.taules)-1} )
```

De la informació obtinguda de cada pàgina de llista de naufragis segons regió, creem una taula amb totes les dades, inserint els títols de les taules com a les columnes de Zona de cada fila. Al final concatenem totes les taules generades per crear un únic dataframe per totes les pàgines.

```
    #procesem la informacio de cada subpagina obtenint una taula
unificada
    def parse_page(self,data,main_title,zones_expandit):

        main_title=main_title.replace("List of shipwrecks","").replace("
of ","").replace(" in the ","")
        titol=""
        zones=["","","",""]
        dfs=[]

        for info in data:

            dades=BeautifulSoup(info.get(), 'html5lib')
            tipus = dades.find('body').find_all(recursive=False)[0].name

            if tipus !='table':
                # buscame la llista de zones correcta a la qual correspon
la taula sgüent
                titol=dades.find('span').text
                index=int(tipus.replace("h", ""))-2
                for info_zones in zones_expandit:
```

```
                    if info_zones[index]==titol and
int(info_zones[3])==index:
                        zones=info_zones.copy()

            else:
                #obtenim columnes i generem taula
                zones.insert(0, main_title)
                columnes = [cela.text.replace("\n","") for cela in
dades.find_all('tr')[0].find_all('th')]
                columnes = ['Zona1','Zona2','Zona3','Zona4'] + columnes
                files_raw=dades.find("tbody").find_all('tr')

                df=self.create_table(columnes,zones,files_raw)
                dfs.append(df)

        #anexem totes les taules de la subpagina
        df_final=pd.concat(dfs, axis=0)
        df_final.reset_index(drop=True, inplace=True)
        return df_final


    #Extraiem les dades de la taula i anexem les columnes
    def create_table(self,columnes,zones,raw):


        #donat que el format no es estandard per totes les tules errors
detectats
        files=[]
        for info_fila in raw:
            fila= [cela.text.strip() for cela in
info_fila.find_all(recursive=False)]
            fila = zones[:4] +fila

            columnes=self.unify_columns(columnes)

            if len(fila)< len(columnes):
                for i in range(len(columnes)-len(fila)):
                    fila.append("")

            if len(fila)> len(columnes):
                fila.pop()

            if len(fila)== len(columnes):
                files.append(fila)
            else:
                print(columnes)
                print(fila)

        #guardem els links de cada un dels vaixells
```

```
        columnes.append("extra_links")
        fila.append(''.join(str(link) for link in
info_fila.find_all('a')))

        #guardem la taula i fem una mica de neteja
        df=pd.DataFrame(files, columns=columnes)
        df.drop(index=df.index[0], axis=0,inplace=True)
        return df
```

Unifiquem els títols de les columnes ja que en les diferents pàgines de Wikipedia, al no seguir un mètode estàndard a l'hora de presentar la informació, apareixen conceptes iguals amb noms diferents.

```
    #corretgim noms de columnes
    def unify_columns(self,columns):

        columns= [str.upper(col) for col in columns]

        columns= ["SUNK DATE" if col=='DATE WRECKED' else col for col in
columns]
        columns= ["SUNK DATE" if col=='END OF SERVICE' else col for col
in columns]
        columns= ["SUNK DATE" if col=='DATE' else col for col in columns]

        columns= ["SHIP" if col=='NAME' else col for col in columns]

        if 'LOCATION' in columns:
            a, b = columns.index('LOCATION'), columns.index('ZONA3')
            columns[b], columns[a] = columns[a], columns[b]

        if 'RIVER' in columns:
            a, b = columns.index('RIVER'), columns.index('ZONA4')
            columns[b], columns[a] = columns[a], columns[b]

        if 'COORDINATES' not in columns:
            columns.append('COORDINATES')

        if 'IMAGE' not in columns:
            columns.append('IMAGE')

        return columns


    def search_cordinates(self, response,row_index,taula_index):

        cord=response.xpath("//span[contains(@class, 'geo-default')]")
        if len(cord) >0:
```

```
            self.taules[taula_index].loc[row_index,"COORDINATES"]=Beautif
ulSoup(cord.get(), 'html5lib').text


    def search_image(self, response,row_index,taula_index):

        info=response.xpath("//table[contains(@class, 'infobox')]")
        #resum de info extra
        if len(info) >0 :
            info=BeautifulSoup(info.get(), 'html5lib')
            imatges=info.find_all('img')

            if len(imatges) >0:
                self.taules[taula_index].loc[row_index,"IMAGE"]=imatges[0
].get('src')



    #obtenim el llistat de zones complet
    def expand_zones(self,titles):
```

Ordenem els noms dels títols de cada zona de naufragi per a poder incloure'ls de manera estructurada en el dataset final, ja que en la pàgina de Wikipedia no apareix aquesta informació de manera estandarditzada.

```
      #aixo simplement es una funcio de ordenacio que guarda les zones de
les capçaleres de la pagina
        #basicament estructura informació que com a humans obtenim
contextualment de la pagina
        last_index=-1
        prefix=['','','','']
        llista=[]
        index=0
        for title in titles:
            dades=BeautifulSoup(title.get(), 'html5lib')
            index=dades.find('body').find_all(recursive=False)[0].name.re
place("h", "")
            index=int(index)-2
            text=dades.find('span').text
            if  index > last_index:
                prefix[index] = text
            elif index == last_index:
                prefix[3]=last_index
                llista.append(prefix.copy())
                prefix[index] = text
            else:
                prefix[3]=last_index
```

```
                llista.append(prefix.copy())
                prefix[index+1] =""
                if index==0:
                    prefix=['','','','']
                else:
                    prefix[2] =""
                prefix[index] = text
            last_index=index
        prefix[3]=index
        llista.append(prefix.copy())
        return llista
```

10. Dataset

El dataset generat per aquesta pràctica és un arxiu CSV i també està carregat al Zenodo i al repositori Github del projecte, amb el nom shipWrecks.csv. Conté 4460 instàncies amb 12 atributs. Es pot veure en el següent enllaç:

https://doi.org/10.5281/zenodo.7347768

11. Vídeo

El vídeo explicatiu de la pràctica es pot trobar en el següent enllaç:

https://drive.google.com/drive/folders/1TvYKrD8coW_3XDT1vA3BT6Vw5Y5eLSVC?usp=share_link

12. Taula de contribucions

Contribucions	Signatura
Investigació prèvia	OCM, PCP
Redacció de les respostes	OCM, PCP
Desenvolupament del codi	OCM, PCP
Participació al vídeo	OCM, PCP