Proyecto Final TIIG

Nicolas Ramos Samboni

2021

Tratamiento de Datos Sismológicos con Python

Introducción

El objetivo principal de esta practica es extraer, transformar y representar los datos obtenidos de la pagina https://earthquake.usgs.gov/earthquakes/feed/v1.0/csv.php Para lo cual habrá que realizar una serie de procesos que se explicaran durante la práctica.

1. Automatizar la descarga diaria de datos
2. Inspeccionar ficheros de datos
3. Modificar ficheros
    1. Eliminar duplicados
    2. Eliminar columnas innecesarias
    3. Añadir columna de control
4. Uso de libreria GDAL
    1. Formar geometrias
    2. Georreferenciar
    3. Extraer en formato ShapeFile

Refencias para automaziar la extracción de datos

Para solucionar el problema de la automatización en la extracción de datos hemos buscado como referencia el siguiente link https://towardsdatascience.com/how-to-web-scrape-with-python-in-4-minutes-bc49186a8460 en la que se extrae datos de un conjunto de ficheros. Analizando el código propuesto podemos adaptarlo a nuestras necesidades en base a como esta organizada la página de USGS.

```
import requests
import urllib.request
from bs4 import BeautifulSoup

url = 'http://web.mta.info/developers/turnstile.html'
```

```
response = requests.get(url)
soup = BeautifulSoup(response.text, 'html.parser')
a=soup.findAll('a')
```

Si analizamos el resultado de ejecutar este codigo podemos ver que todos los archivos se encuentran dentro de las etiquetas <a>

Prior to 10/18/14, CurrentSaturday, December 26, 2020, Saturday, December 19, 2020, Saturday, December 12, 2020, <a href="data/nyct/turnstile/turnstile_201205.txt">

Para automatizar los procesos necesarios usamos la libreria Schedule de Python. https://pypi.org/project/schedule/ , la cual nos permite realizar procesos definidos en funciones en intervalos de tiempo definidos.

Este script descarga todos los archivos de una pagina web cada 1 minuto (Código encontrado en la primera página de referencia)

```
import schedule, time, requests, urllib.request, time
from bs4 import BeautifulSoup

def getFile():
    url = 'http://web.mta.info/developers/turnstile.html'
    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')
    line_count=1
    for one_a_tag in soup.findAll('a'):
        if line_count >= 38:
            link = one_a_tag['href']
            download_url = 'http://web.mta.info/developers/'+ link
            urllib.request.urlretrieve(download_url,'./'+link[link.find('/turnstile_')+1:])
            time.sleep(1)
        line_count+=1
schedule.every(1).minutes.do(getFile)

while True:
    schedule.run_pending()
    time.sleep(2)
```

Extracción de datos automatizada

Probamos con nuestros datos de USGS.

Analizando el resultado del código vemos que dentro de las etiquetas del HTML se encuentran los links de descarga de los datos que se actualizan en la página USGS. Por lo tanto podemos acceder directamente a ellos sin necesidad de tratar con el código fuente de la página.

```
import urllib.request, schedule, time
def scrapping():
  url = 'https://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/all_hour.csv'
  urllib.request.urlretrieve(url,'HourlyData.csv')
  df = pd.read_csv('hourlydata.csv')
schedule.every(15).minutes.do(scrapping)
while True:
    schedule.run_pending()
    time.sleep(2)
```

Inspeccionar fichero de datos

Tenemos que ir cambiando el nombre de los archivos para que no se sobreescriban. Cada vez que se ejecuta el script el archivo deberá cambiar de nombre. Como el método que estamos usando nos impide hacer esto, debemos buscar otra solución. Esto debido a que este método ejecutará el script un X numero de veces como si fuera la primera vez que se ejecutara. Por lo tanto no podemos realizar procesos iterativos que afecten a la estructura de los archivos descargados.

La solución a esto será:

1. Descaragar los datos.
2. Generamos un fichero records donde se guarden todas las modificaciones del archivo (aun que no existan datos nuevos).
3. Generamos fichero final donde se guardan los datos según nuestras preferencias.

Modificar ficheros

Usaremos pandas para el manejo de los DataFrame descargados en .csv, y generaremos el record.

```
#generamos el fichero en el cual se iran añadiendo datos si el fichero existe, en caso contrario se creara
#y añadiran los primeros datos
with open('testfile2.csv','a') as f:
  df.to_csv(f,header=f.tell()==0)
raw_data = pd.read_csv('record.csv')
  .drop_duplicates(keep='first',subset=['latitude','longitude'])
  .sort_values(by='time').reset_index(drop=True)
```

Debemos depurar los datos que queramos introducir en el archivo final que usaremos.

Introducir nueva columna en la cual se introducira el lugar (State) donde se encuentran los eventos, para asi poder diferenciar los que son de California de los que no.

Para esto abra que manipular la columna 'place' la cual nos indica la direccion postal aproximada del evento, incluido el Estado/País donde se encuentran.

```
state=[]
for row in raw_data['place']:
  x=re.split(',',row)
  state.append(x[-1])
raw_data['state']=state
index_names=raw_data[raw_data['state']!=' CA'].index
```

La libreria Pandas nos permite eliminar duplicados con drop_duplicates, pero no funciona al escribir documentos por lo tanto leemos el archivo de registro de los datos extraidos 'con duplicados' y depuramos todos los datos según se van introduciendo en nuestro fichero final con el que trabajaremos después para el restro del proyecto.

```
#pandas nos permite eliminar duplicados con drop_duplicates, pero no funciona al escribir documentos,
#leemos el archivo de registro de los datos extraidos 'con duplicados'
#y depuramos todos los datos segun se van introduciendo en nuestro fichero final con el
#que trabajaremos después para el restro del proyecto
raw_data.drop(columns=['Unnamed: 0','locationSource','magSource',
                       'status','magType','nst','gap','dmin','rms',
                       'net','magNst'],index=index_names).to_csv('final.csv')
```

Aplicacion de libreria GDAL/OGR

Ahora que ya tenemos el fichero final en formato .csv con coordenadas de los eventos sísmicos podemos generar geometrías puntuales gracias a la libreria GDAL:OGR, para posteriormente generar un fichero ShapeFile que podremos representar en alguna herramienta GIS.

Usaremos el siguiente link como ayuda para generar el archivo .shp: https://pcjericks.github.io/py-gdalogr-cookbook/index.html

Usando las herramientas que nos da la libreria GDAL podemos ir construyendo el fichero shp:

```
data = pd.read_csv('final.csv')
driver = ogr.GetDriverByName("ESRI Shapefile")
data_source = driver.CreateDataSource("/Users/nickramos/Desktop/Coding/seismos/seismos.shp")
```

Para asociar la proyección WGS84 a nuestros datos realizaremos el siguiente paso. Esto con el objetivo de superponer los datos sobre un mapa base de Estados Unidos (fichero incluido en la documentación de la practica).

```
srs = osr.SpatialReference() #WGS84
srs.ImportFromEPSG(4326)
```

Ahora creamos la capa e instanciamos todos los campos de información que queremos que aparezcan en la información de los eventos.

```
layer = data_source.CreateLayer("Eventos Sismicos", srs, ogr.wkbPoint)

field_name = ogr.FieldDefn("Id", ogr.OFTString)
field_name.SetWidth(24)
layer.CreateField(field_name)
field_region = ogr.FieldDefn("Place", ogr.OFTString)
field_region.SetWidth(24)
layer.CreateField(field_region)
layer.CreateField(ogr.FieldDefn("Time", ogr.OFTDate))
layer.CreateField(ogr.FieldDefn("Latitude", ogr.OFTReal))
layer.CreateField(ogr.FieldDefn("Longitude", ogr.OFTReal))
layer.CreateField(ogr.FieldDefn("Mag", ogr.OFTReal))
layer.CreateField(ogr.FieldDefn("Depth", ogr.OFTReal))
```

Gracias a la libreria Pandas podemos iterar sobre dos columnas del dataframe con mayor facilidad. Extraemos las coordenadas de los eventos y generamos la geometria puntual que cada uno de ellos.

```
for index,row in data.iterrows():
  feature = ogr.Feature(layer.GetLayerDefn())
  feature.SetField("Id", row['id'])
  feature.SetField("Place", row['place'])
  feature.SetField("Time", row['time'])
  feature.SetField("Latitude", row['latitude'])
  feature.SetField("Longitude", row['longitude'])
  feature.SetField("Mag", row['mag'])
  feature.SetField("Depth", row['depth'])
  wkt = "POINT(%f %f)" %  (float(row['longitude']) , float(row['latitude']))
```

Geocodificar

Al generarse eventos sobre cualquier punto de la superficie terrestre sobre el estado de California es muy dificil encontrar una solución exacta de una dirección postal asociada a las coordenadas. Aun asi se puede aproximar al condado o territorio sobre el que se encuentra el punto.

Una manera fácil y gratis de conseguir la geocodificación mediante la libreria 'inverse_geocoder', la cual no necesita internet para encontrar las soluciones. Y comprobando los resultados vemos que es suficientemente exacto para el objetivo de la practica.

```
def reverseGeocode(coordinates):
  result = rg.search(coordinates)
  #pprint.pprint(result[0].get('admin2'))

if __name__=="__main__":
  coordinates =(42.4, -3.7)
  reverseGeocode(coordinates)
```

NOTAS

Todos los script finales se encuentran en la documentación de la práctica.

Deben ejecutarse de forma secuencial: usgs.py → geometry.py → geocode.py

Los archivos .csv se modifican al ejecutar la secuencia de scripts, no es necesario volver a ejecutarlos para comprobar los resultados (a excepción de geocode.py)

Este script devuelve resultados por pantalla, no los adjunta al archivo .csv