MANUAL WEB SCRAPING

REALIZADO POR:
Elis, Carlos, Nacho y Javier

1. Para empezar nuestro proyecto de Scraping lo que tendremos que hacer es importar todas las **librerías** para poder empezar a trabajar. Las imprescindibles son “**bs4: BeautifulSoup**” y “**pandas**” para poder realizar el scrapeo. Luego hemos ido importando otras librerías para que el resultado nos quedase mas concreto y más limpio. Por ejemplo, la librería de “**langdetect**” para poder sacar los comentarios en el idioma deseado, en nuestro caso el español.

```
from bs4 import BeautifulSoup
import requests
import pandas
import re
from langdetect import detect
from time import *
from langdetect.lang_detect_exception import LangDetectException
```

2. Una vez importadas todas las librerías es hora de empezar a realizar el scrapeo. Para ello tendremos que pasarle la **URL** de la página que queramos sacar los datos a una variable que hemos creado. Para sacar todos los productos de una página hemos hecho una variable, que le pide el articulo introducido por el usuario y los **headers**, creados anteriormente. Una vez sacada la pagina que queremos obtener los resultados, la convertimos a **HTML**, a través de la librería “**BeautifulSoup**”. Cuando tengamos la página en HTML, creamos una variable que es una lista de las urls de todos los productos a analizar.

```
# Pedimos el articulo que quiere buscar el usuario.
Articulo = (input("Que artículo buscas: "))
# Pedimos la cantidad de productos a buscar.
cantidad = int(input("Cuantos artículos buscas: "))

# La url de la que sacaremos los productos.
URL = 'https://www.amazon.es/s?k=' + Articulo

headers = {"User-Agent": "Mozilla/3.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.0.0

# Pagina sobre la que realizaremos el scrap.
page = requests.get('https://www.amazon.es/s?k=' + Articulo, headers=headers)
# Convertimos la pagina a html.
soup1 = BeautifulSoup(page.content, "html.parser")
soup2 = BeautifulSoup(soup1.prettify(), "html.parser")

# Lista de urls de los productos a analizar.
urls = soup2.findAll('a', class_='a-link-normal s-no-outline', limit=cantidad)
```

3. Para poder almacenar todos los comentarios y valoraciones de nuestro scrapeo hemos creado una **lista** que posteriormente usaremos.

```
# Lista de comentarios del scrap.
comentarios = list()
# Lista de valoraciones del scrap.
estrellas = list()
```

4. Cuando tengamos creadas las listas, vamos a pasar a trabajar con los comentarios y las estrellas de un producto, para ello hemos creado un **método**. Primero vamos a sacar todos los comentarios de una pagina con un bucle y lo limpiamos de emoticonos y teclas especiales. Ahora con la librería "**langdetect**" vamos a comprobar el idioma del comentario, si es español lo limpiaremos y lo añadiremos a la lista de comentarios creada anteriormente.

```
# Método que escribe los comentarios de una pagina, con su valoracion convertida a 1 o 0.
def escribir(comments, stars):
    # Bucle de comentarios de una pagina.
    for comentario,  estrella in zip(comments, stars):
        # Limpiamos emoticonos, teclas especiales...
        comentario = " ".join(comentario.get_text().strip().split())
        try:
            # Comprobamos el idioma del comentario, si es español intentaremos introducirlo en la lista.
            if (detect(comentario) == "es") :
                # Limpiamos el comentario.
                comentario = comentario.lower()
                comentario = re.sub(r'[^\w\s]', '', comentario)
                comentario = re.sub(u"[àáâãäå]", 'a', comentario)
                comentario = re.sub(u"[èéêë]", 'e', comentario)
                comentario = re.sub(u"[ìíîï]", 'i', comentario)
                comentario = re.sub(u"[òóôõö]", 'o', comentario)
                comentario = re.sub(u"[ùúûü]", 'u', comentario)
                comentario = re.sub(u"[ñ]", 'n', comentario)
                comentario = re.sub(u"[ç]", 'c', comentario)
                comentarios.append(comentario)
```

5. Ya tenemos limpios y añadidos todos los comentarios, ahora vamos a pasar con las estrellas. Creamos un bucle para saber si las estrellas del comentario son mayores a tres, y si lo son, añadiremos un 1 a la lista de estrellas, pero si las estrellas de esa valoración son menos de 3 añadiremos un 0 a la lista de estrellas.

```
        # Condicion para clasificar las estrellas.
        if (len(estrella.findChild("span", { "class" : "a-icon-alt" }).get_text()) == 55) :
            if (str(estrella.findChild("span", { "class" : "a-icon-alt" }).get_text())[19:20] > "3") :
                estrellas.append("1")
            else :
                estrellas.append("0")
        else :
            if (str(estrella.findChild("span", { "class" : "a-icon-alt" }).get_text())[18:19] > "3") :
                estrellas.append("1")
            else :
                estrellas.append("0")
except LangDetectException:
    print("No se ha podido introducir el comentario")
```

6. Ahora que ya hemos añadido todos los comentarios y estrellas vamos a sacar todos los productos de una página y cuando no haya más productos en la página pasaremos a la siguiente. Eso lo haremos con un bucle, y un contador de productos de la página en la que estemos.

```
# Contador de productos
producto = 1
# Bucle por producto
for i in urls:
    # Sacamos la pagina de reviews del producto
    review = i['href'].replace('dp', 'product-reviews')
    page = requests.get(("https://www.amazon.es" + review ), headers=headers)
    # Convertimos la pagina a html.
    soup1 = BeautifulSoup(page.content, "html.parser")
    soup2 = BeautifulSoup(soup1.prettify(), "html.parser")
    # Cogemos la pagina siguiente de reviews
    siguiente =  soup2.findAll('li', class_='a-last')
    # Cogemos la pagina siguiente de reviews
    final =  soup2.findAll('li', class_='a-disabled a-last')
```

7. Ahora vamos a comprobar con una condición que si es la primera pagina guardamos los comentarios y las estrellas de esa página, pero si hay una pagina siguiente analizamos la página en la que estemos y volvemos a convertir la página en **HTML** y guardamos todos los comentarios y estrellas de esa página. Con la variable creada de "**siguiente**" guardaremos las etiquetas de las paginas para poder usarlas en la condición de antes. Luego crearemos la variable "**final**" que guarda la cantidad de etiquetas que hay para que con otra condición saber si es la última página del producto.

```
# Condicion (Si es la primera pagina)
if contador == '0' :
    print('Página(1): https://www.amazon.es/' + review)
    # Guardamos las estrellas y comentarios
    stars = soup2.findAll('div', class_='a-section celwidget')
    comments = soup2.findAll('span', class_='a-size-base review-text review-text-content')
else :
    # Condicion (Si hay una pagina siguiente de reviews, la primera vez que se ejecute cogera la 2a pagina).
    if len(siguiente) > 0 :
        # Sacamos la pagina a analizar con el "siguiente" de la anterior pagina.
        page = requests.get('https://www.amazon.es/' + str((siguiente[0].findChild("a"))['href']), headers=headers)
        print('Página(' + str(int(contador) + 1) + '): https://www.amazon.es/' + str((siguiente[0].findChild("a"))['href']))
        # Convertimos la pagina a html.
        soup1 = BeautifulSoup(page.content, "html.parser")
        soup2 = BeautifulSoup(soup1.prettify(), "html.parser")
        # Guardamos las estrellas y comentarios.
        stars = soup2.findAll('div', class_='a-section celwidget')
        comments = soup2.findAll('span', class_='a-size-base review-text review-text-content')
        # Guardamos la siguiente pagina que usaremos en la condicion de arriba.
        siguiente =  soup2.findAll('li', class_='a-last')
        # Guardamos la cantidad de etiquetas "a-disabled a-last" que hay.
        final =  soup2.findAll('li', class_='a-disabled a-last')
        # Condicion (Si existe la etiqueta anterior, significa que es la ultima pagina de reviews de ese producto).
        if len(final) == 1 :
            # Ecribimos los comentarios en la lista.
            escribir(comments,stars)
            # Rompemos el bucle de este producto.
            break
```

8. Si vemos que no hay pagina siguiente pararemos el bucle, pero si no es así escribiremos todos los comentarios y estrellas almacenados y mostraremos a tiempo real todos los comentarios almacenados.

```
    # Condicion (Si no hay pagina siguiente)
    if (len(siguiente) == 0) :
        # Paramos bucle
        break
    else :
        # Ecribimos los comentarios en la lista.
        escribir(comments,stars)
    # Contador de primera pagina.
    contador = str(int(contador) + 1)
    # Mostramos comentarios obtenidos por el programa en tiempo real.
    print('Comentarios obtenidos: ' + str(len(comentarios)))
# Contador de producto.
producto = producto + 1
```

9. Por último, todos los datos los enviaremos a “**pandas**”, que es la librería explicada antes, y los mostraremos. También exportaremos todos los datos a un **fichero csv**, separado por comas, para posteriormente analizarlo en la **IA**.

```
# Enviamos los datos a pandas.
datos = pandas.DataFrame({"Valoracion": estrellas, "Comentario": comentarios})
print(datos)
# Exportamos el pandas a un csv, sin index, y separado con ",".
datos.to_csv('prueba.csv', encoding='utf-8', sep=',', index=False)
```

IA

1. Una vez realizado el “**Scrapeo**” vamos a ponernos a hacer la “Inteligencia Artificial”, para ello lo primero va a ser importar las librerías necesarias. Las más importantes son “pandas” que la hemos utilizado para hacer el “**Scrapeo**” y la de “**SkLearn**” que la utilizaremos posteriormente para entrenar a la “**IA**”.

```
from sklearn.model_selection import train_test_split
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.svm import SVC
from imblearn.under_sampling import RandomUnderSampler
```

2. Cuando tengamos todas las librerías importadas, nos disponemos a realizar la IA. Para ello primero tendremos que leer el **fichero**.**csv** con el que entrenaremos a nuestra máquina. Cuando le hayamos pasado el fichero crearemos un “**DataFrame**” para equilibrar los datos positivos y negativos del fichero. Para poder entrenar la máquina crearemos los datos de entrenamiento y la bolsa de palabras, que se encarga de transformar de texto a numero las palabras correspondientes y sus comunicaciones para así ser capaz posteriormente de identificar si es bueno o malo respecto a la conexión. Posteriormente entrenaremos la **IA** con los datos y comprobamos el acierto de nuestra **IA** respecto a los datos del test.

```
# Importamos el csv y lo introducimos en un DataFrame.
df = pd.read_csv('./Data.csv')

# Creamos un DataFrame en el que equilibraremos los datos positivos y negativos.
df_comentario_balanceado, df_comentario_balanceado['Valoracion'] = RandomUnderSampler().fit_resample(df[['Comentario']], df['V

# Creamos un DataFrame en el que equilibraremos los datos positivos y negativos.
train, test = train_test_split(df_comentario_balanceado, train_size=0.9, random_state=50)

# Creamos los datos de entrenamiento y de test.
x_train, y_train, x_test, y_test = train['Comentario'], train['Valoracion'], test['Comentario'], test['Valoracion']

# Creamos la bolsa de palabras, separandolas en train y test.
vectorizer = CountVectorizer(analyzer="word", lowercase=False)
x_train_transform = vectorizer.fit_transform(x_train)
x_test_transform = vectorizer.transform(x_test)

# Entrenamos la IA con los datos de entrenamiento.
clf = SVC().fit(x_train_transform, y_train)

# Probamos el acierto de nuestra IA mediante los datos de test y lo mostramos por pantalla.
print("La IA tiene un", round(clf.score(x_test_transform, y_test)*100, 2), "%","de acierto")
```

3. Para acabar la “**IA**” realizaremos un bucle, este tendrá un menú para que el usuario pueda iniciar el programa y pararlo cuando quiera. Dentro de ese bucle el usuario pondrá el comentario que quiera y valorará en base al test si el comentario es negativo o positivo.

```
while (True) :
    # Menu
    print("\n-------------------------------------")
    print("0-. Finalizar programa")
    # Pedimos el comentario a analizar
    comentario = [input("Introduce un comentario: ")]
    if (comentario[0] == '0') :
        break
    else :
        # Lo convertimos a minusculas
        comentario[0] = comentario[0].lower()
        # Condicion (Si el comentario es 0, es negativo)
        if (clf.predict(vectorizer.transform(comentario)) == 0) :
            print("El comentario es negativo")
        else :
            print("El comentario es positivo")
    print("-------------------------------------")
```