Web-Scraping

Web scraping consists in automatically downloading images from the internet. In order to do web scraping, we used the html code of the page. Thanks to a library called selenium, we managed to interact with the page elements using python.
In order to use the script, you need to install :

- Selenium and urllib with python
- Firefox and its associated driver as detailed in the selenium documentation.
- Once this step is done, you need to indicate the path of the driver and the path of the destination of the data in the script.

a. Using the web scraping algorithm

The first script, dataset.py, extracts a given number of images from google image, for each research that you put in a list of research, creates a new folder corresponding to this research and saves the thumbnail associated with each image. This script can be used for classification tasks since the images downloaded are relatively small (around 200px). This script is designed to be used only once, because you can't choose which images are downloaded. This method is useful to get a huge amount of data very quickly, but after you've downloaded these images, you need to make sure that these images correspond with what you look for by reviewing them, which is a bad aspect of this method, but still requires less time than downloading images yourself.

The second script, dataSetEbirds, uses a website called Ebirds where you can find lots of photos corresponding to a given bird.

- Firstly, you need to have the english bird name in order to find something.
- Once you have its name, enter it in the algorithm (nom_recherche) and create a new folder with this name.
- Launch the first part of the algorithm (###lancement du navigateur). You should see images corresponding to the bird you choose.
- Once this step is done, choose a date in order to be able to make the database larger later without taking the same photo (could be a problem if they end up in training and testing).
- Once this is done, you can pass to the second part : this step uses a listener that can recognise if you press &,é,",',(,-,è. (correspond to 1,2,3?4?5 in my clavier), and also the down arrow key. You need to modify thoses to correspond to yours.
- Once this step is done, you can launch the algorithm. When you press a key, it saves it in a python array, which will be used later to download images. For each line, you need to press on the number associated with the photo you want to save (don't press anything if you don't want).
- Once this step is done, press the down arrow key. The page should move to correspond to the new line. You can repeat the process until you are done. To download the data, you need to change the num variable to the first indice on which your image will be

saved (for example num=0 for sander species will save sander-0 first). It is useful if you have already downloaded images (if you dont update it, it will replace the previous images).

- Once this is done, you can also choose the resolution of the image. (see commented version for full resolution). When you launch it, it should save the image in the specified directory.

b. How it works (in case you would want to change the script)

The goal of the selenium library is to find the url of each image. It is located in the html code of each web page if there is an image in this page (this html code can be seen by right click and “inspecter l’élément” in french).

Then you need to find the tag containing the image url. This tag is the thing that you will search with selenium.

We find elements thanks to their html banner by using driver.find_elements_by_xpath().

The argument is for exemple:

//div[contains(@class,"ResultsGallery-row")]'

It means we search in all the documents for a div tag which contains:

class=”ResultsGallery-row"

This function returns an element that represents this tag. If you need to find a tag inside this element, instead of *‘//’* at the beginning, you must use *‘.//’*.

Once you have found the good tag, (balise1 for example) you can get the specific text of an element *‘src’* for example with the command :

balise1.get_attribute(‘src’)

This return the text contains in this field.

Once this is done, to get the url, you need to extract the useful part of this string with python chain manipulation. Now, you have the url of the image. Urllib allows us to download the image in binary representation, and we save it using

open() (argument ‘wb’ = write binary)