Data Wrangling Project :
Extraction of structured data from social network

Vincent Gouteux - Master IASD

March 2020

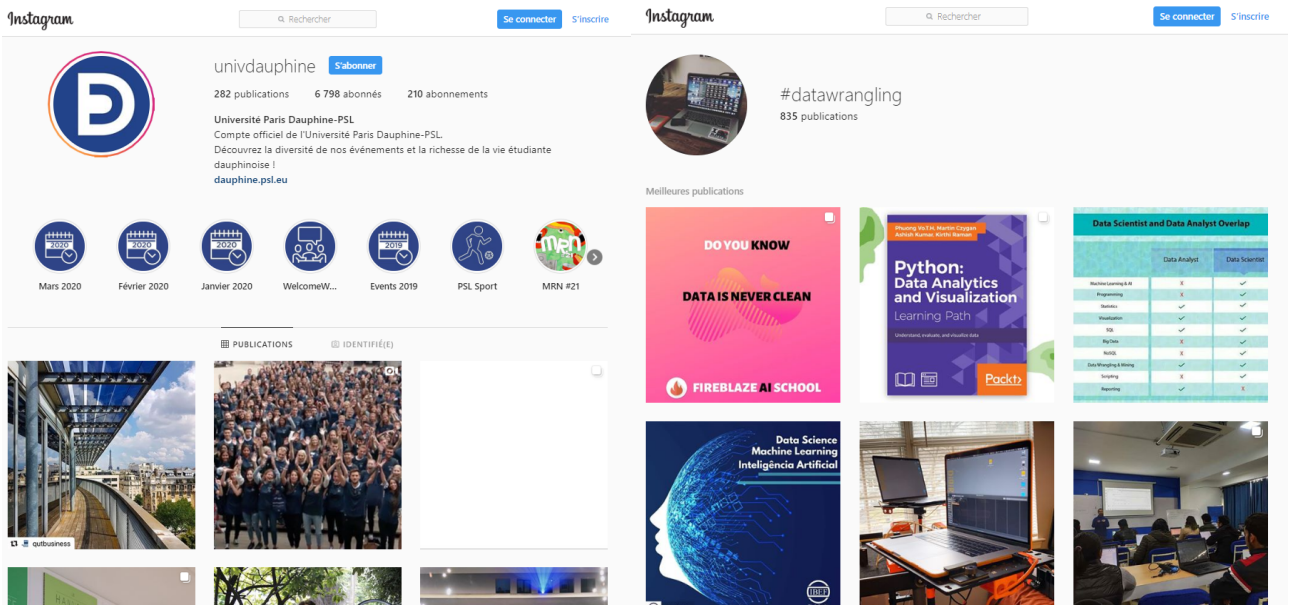
Since the creation of the web, data extraction have always been a very interesting topic. Now with the BigData a lot of people are very interested in retrieving data from the web to exploit it for different purposes. In this project we will see how to extract structured data from a social media : Instagram.

1 Introduction

A lot of work has already been done on this topic since researchers have begun to work on data extraction on the web years ago. We can cite some of them that helped a lot to produce this work and understand better the extraction of structured data : Extracting Structured Data from Web Pages by Arvind Arasu & Hector Garcia-Molina. We can also cite Mining Data Records in Web Pages by Bing Liu, Robert Grossman & Yanhong Zhai
As said above the goal of this project is to try to extract structrued data from a website using the html code of the webpage. To make this project interesting I decided to choose a very recent and famous social network which is Instagram. The interesting thing with Instagram is that this network works with posts, an image with a description, and every post has its own information such as likes, comments, place, etc. Finally I will try to scrap the site without using the API that gives a lot of tools to make the data extraction easier. Furthermore, Instagram is a very dynamic social network. Indeed there are more than 500 million daily active users and more than 100 million photos and videos uploaded per day. (Source)
As a huge quantity of data is created and uploaded everyday it could be very interesting to create a web-scrapper that retrieve the new data everyday.

2 Instagram

This section will present briefly the social network : its structure, the objects we find, etc. The thing to know is that there are two main ways to search a post on Instagram which are by using a hashtag or by looking at the posts of a specific user.
To be able to collect a lot of information using only an hashtag or a user name is quite impressive and can be very useful for people or even companies. If someone wants to track the popularity of a certain user, he can do it by simply collecting the likes on the user's posts every week. We can also measure the popularity of an idea or object by counting the number of posts which have put the hashtag of this idea...



(a) Instagram User : "univdauphine" (b) Instagram hashtag : "datawrangling"

Figure 1: The two ways of retrieving posts on Instagram

As said before, the "objects" on Instagram are posts. We are going to try to collect posts from a specific user or a specific hashtag with all the information linked to this post : date of publication, number of likes, number of comments, etc. We will see that it is very structured data and that the extraction by using the HTML code (which is a bit brutal and not very elegant) gives very satisfying results.

3 The HTML code

As said before there is an API of Instagram where it is possible to retrieve data contained on the website. But it is more interesting to scrap using the HTML code so we can really see the objects we are manipulating and the data we are collecting.

As we can see on the picture below, we have access to the HTML code directly from the post and we can know exactly where is every single information. As we can see, by reading the HTML code we can see what are the different HTML tags and therefore where is information we need.

The huge advantage is that we are actually collecting structured data, which means once we have detected all HTML tags that we need to find all the information we would like to collect, we can apply this process on every post on Instagram (except for several special cases). To sum up, we just need to collect and analyse 3 HTML codes :

- The HTML code of a user's page in order to retrieve the HTML code of his posts
- The HTML code of a hashtag for the same purpose
- The HTML code of a post, which has the same structure for every post.

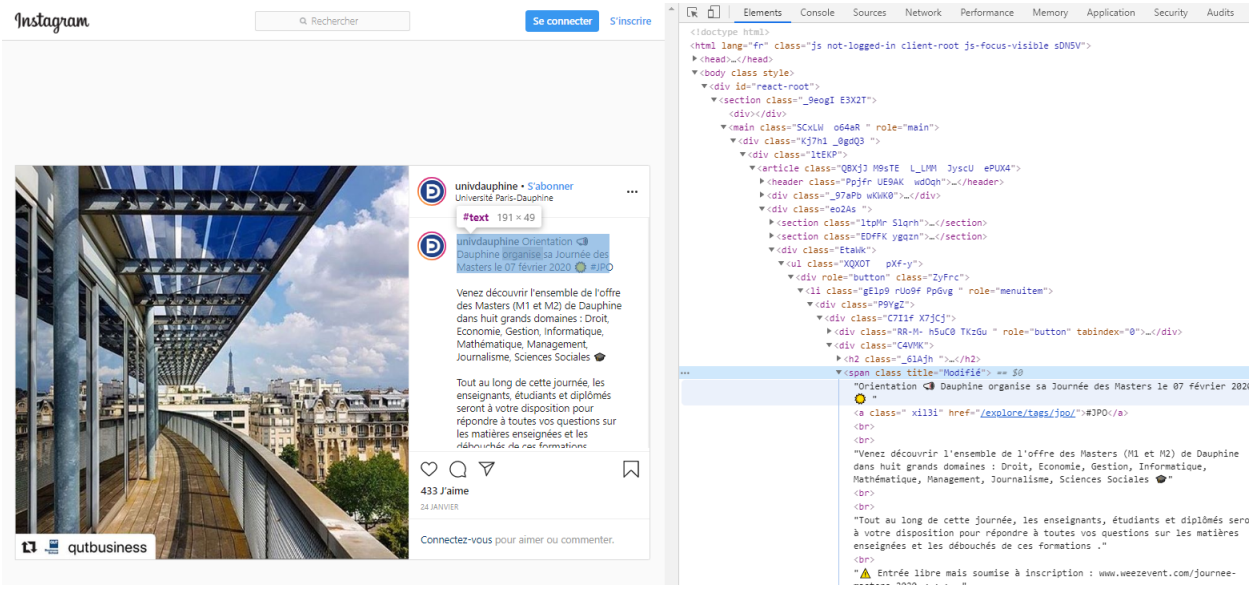


Figure 2: HTML code of a certain post of univdauphine

4 Python implementation

Web scrapping is really used to gather data in a machine learning or data science process. As we know, there are a lot of useful tools and packages available on python to develop machine learning algorithms. Some people have also created tools to make web scrapping easier. We can cite for example BeautifulSoup which is really good for extracting data from a webpage using its URL. Once again I could have used one of these tools to perform web scrapping but the main goal of this project was to see how difficult it is to collect data "from scratch" and if the scrapping is successful, it arouses a lot of new questions around these data : Is it necessary to clean the data ? What can be done with it (Machine learning algorithm) ?

4.1 Python Request

The tool we are forced to use is the request tool. Indeed, it is the only way to get the HTML code from an URL. The requests documentation is available here. The big advantage we have when scrapping websites with python requests is that we can collect data from a website and transform it in a json file. We then have a collection of nested dictionaries where the keys are the HTML tags and the values are the data we are interested in.

The lines of python code below are the ones we need to get this big HTML code dictionary.

```
user = univdauphine
r = requests.get("https://www.instagram.com/{0}/?__a=1".format(user))
htmlcode = r.text
data = json.loads(htmlcode.strip(),strict = False)
```

Once we get the raw data from the request, we have to treat it to extract the meaningful data. There are some python libraries and packages that allow us to understand and treat the

HTML in a much easier way by cleaning the code and add some pre-processing actions. But we find it more interesting to do it from scratch, by doing this we can really understand how the website is structured, and where precisely the information is stored. To do so, we had to spend a lot of time analysing the HTML code to understand which HTML tag corresponds to which information and to find where is stored the important data.

4.2 First naive method

Each post is indexed by a shortcode, to access to the web page of this post we only have to enter it in the URL. Each shortcode is unique.

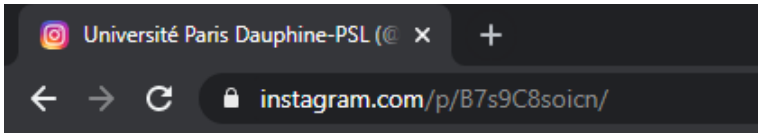


Figure 3: Shortcode of the first post of univdauphine

Our first approach was the most intuitive one. From a hashtag or a user's page, we can easily get all the shortcodes and then have access to the HTML code of every post. To do so, I created the *Data_From_Posts* function that takes a number of posts and an array of shortcodes. Here are the first lines of the HTML code we get from the request. In order to make the job easier we load it as a JSON file to detect where is the information we need and what are the HTML tags we need to select.

```
r = requests.get("https://www.instagram.com/p/B7s9C8soicn/?__a=1")
htmlcode = r.text
data = json.loads(htmlcode.strip(),strict = False)
data
```

```
{'graphql': {'shortcode_media': {'__typename': 'GraphImage',
   'accessibility_caption': 'Photo by Université Paris Dauphine-PSL in Université Paris Dauphine - PSL. Image may contain: sky, cloud and outdoor',
   'caption_is_edited': True,
   'commenting_disabled_for_viewer': False,
   'comments_disabled': False,
   'dimensions': {'height': 1080, 'width': 1080},
   'display_resources': [{'config_height': 640,
     'config_width': 640,
     'src': 'https://scontent-iad3-1.cdninstagram.com/v/t51.2885-15/sh0.08/e35/p640x640/82767862_153372909436658_2874354898234735576_n.jpg?_nc_ht=sc
    {'config_height': 750,
     'config_width': 750,
     'src': 'https://scontent-iad3-1.cdninstagram.com/v/t51.2885-15/sh0.08/e35/p750x750/82767862_153372909436658_2874354898234735576_n.jpg?_nc_ht=sc
    {'config_height': 1080,
     'config_width': 1080,
     'src': 'https://scontent-iad3-1.cdninstagram.com/v/t51.2885-15/e35/p1080x1080/82767862_153372909436658_2874354898234735576_n.jpg?_nc_ht=scontent
   'display_url': 'https://scontent-iad3-1.cdninstagram.com/v/t51.2885-15/e35/p1080x1080/82767862_153372909436658_2874354898234735576_n.jpg?_nc_ht=s
   'edge_media_preview_comment': {'count': 8,
    'edges': [{'node': {'created_at': 1581237399,
       'did_report_as_spam': False,
       'edge_liked_by': {'count': 0},
       'id': '17844218554968458',
       'is_restricted_pending': False,
       'owner': {'id': '1405010293',
```

Figure 4: Htmlcode of the first post of univdauphine

As said before, a big part of the work was analysing the HTML code and go through the dictionary to get relevant information. Once we know where we can find the information, we

collect everything and store it in a pandas DataFrame. Pandas DataFrames are very useful for manipulation of structured data, each row represents a post and each column an attribute of the post. With this method we had access to every information we need about each post. However, this technique turned out to be inefficient. As we were sending a new request for every post, it took only few requests to the Instagram system to detect we were scrapping their website. This technique worked for the 20-30 first posts but after, we were systematically kicked from the website and our requests returned errors. We then had to find an other method to overcome this problem.

4.3 Second method

We had to spent some time looking at different HTML codes to see if they were information available without sending new requests. By looking at the HTML code of an user's page or an hashtag, we saw that they were still a lot of data and information about each post. Of course, there was less useful information but sending one request allowed us to retrieve data of 80 post instead of 1. The main data we are loosing by using this technique are the comments. However, we had to make a compromise and it was more interesting to retrieve as many posts as we wanted with less information than only few posts with everything. As we said before we face here a new problem which is that the HTML codes of a user's page, a hashtag or a post are different and we can't apply the same *extraction function*. We had to create two different functions *Posts_Data_From_User* and *Posts_Data_From_Tag*. These functions go through all the dictionary and store in a pandas Dataframe all the information we want to keep. Now that we have seen the method we will give a look a the results, and further we will see what we could add to these functions to make them more useful and have higher-quality data.

5 Results

To follow the examples we had since the beginning we will collect posts from the user's page univdauphine and the hashtag #datawrangling We had to deal with a lot of problems and exceptions that we will explain in the next part. But after solving every problem and error we finally managed to have satisfying results. As the dataframes are very big and contain a lot of information we decided to put truncated dataframe containing less columns just to show how the data look after we collect it. We will see in the next part every attribute we kept and additional information we used to have cleaner data and tricks that will help for the future if we want to exploit these data.

	Tag_Id	Tag_Name	Extraction_Date	Posts_W_Hashtag	Owner_Id	Shortcode	Timestamp	Likes	Hashtags	Mentions	Com's	Text	Img_Description
0	17843831179043460	datawrangling	21032020	862	10602884403	B9zqLYug8bL	2020-03-16 20:11:07	39	[datatechcon\nNeed, dashboard, kpi, smallbusin...	[datatechcon]	0	Customers are the foundation of any business' ...	Video
1	17843831179043460	datawrangling	21032020	862	9299626406	B9yFEDcAKx5	2020-03-16 05:18:48	27	[businessintelligence\n, automateddatawranglin...	[]	4	Why wouldn't you prefer the speed and data pro...	Video
2	17843831179043460	datawrangling	21032020	862	13935806959	B9o5bH1gczi	2020-03-12 15:42:41	12	[datawrangling, businessintelligence, inglês, ...	[]	0	#datawrangling #businessintelligence #inglês #...	1 person, text
3	17843831179043460	datawrangling	21032020	862	9299626406	B9nlb2lAtH_	2020-03-12 03:31:24	14	[]	[]	1	Reduce human resource costs by taking complete...	Video
4	17843831179043460	datawrangling	21032020	862	12763554440	B9gIW5oAt9N	2020-03-09 06:00:01	35	[ml, machinelearningalgorithms, machinelearnin...	[data, data, data]	1	A very subtle and sharp way to understand the ...	possible text that says 'Statistics: Given th...
5	17843831179043460	datawrangling	21032020	862	12763554440	B9eJomnAl0P	2020-03-08 11:32:41	32	[]	[data, data, data]	1	Some very cool and handy R packages which help...	Null
6	17843831179043460	datawrangling	21032020	862	186032245	B9ZVgKvBMRe	2020-03-06 14:40:12	49	[CineColombiano, detrasdecamaras, datawrangling]	[ficcifestival,]	0	Después de un tiempo dando vueltas por el mund...	1 person, standing and text
7	17843831179043460	datawrangling	21032020	862	9299626406	B9TD-UGgv50	2020-03-04 04:11:35	10	[]	[]	1	Change reality into a more desirable state wit...	1 person, beard and outdoor, possible text th...
8	17843831179043460	datawrangling	21032020	862	18075761281	B9RXXdHHKji	2020-03-03 12:22:33	10	[data, datascience, datasavy, datascientist, d...	[]	0	Data visualization tools... #data#datascience#...	Null
9	17843831179043460	datawrangling	21032020	862	27474528070	B9HfkqQH0s-	2020-02-28 16:21:51	15	[fridayfeeling, fridaytreats, ditpic, davincir...	[]	1	Cheeky Friday beer to round off the weeks work...	Null

Figure 5: Dataframe all the first 10 posts of #datawrangling

User_Id	User_Name	Is_Verified	Followers	Extraction_Date	Nb_Of_Posts	Owner_Id	Shortcode	Timestamp	Likes	Hashtags	Mentions	Com's	Text	I
257662767	univdauphine	False	6868	21032020	283	257662767	B99yqg6oJxi	2020-03-20 18:27:40	58	['Covid19']	[]	0	[#Covid19] Dauphine est fermée jusqu'à nouvel ...	
257662767	univdauphine	False	6868	21032020	283	257662767	B7s9C8soicn	2020-01-24 12:28:31	436	['JPO\n\nVenez', 'Repost', 'assoetudiante', 'v...	['qutbusiness', 'univdauphine', 'itsabrina']	8	Orientation 📣 Dauphine organise sa Journée des...	
257662767	univdauphine	False	6868	21032020	283	257662767	B7YHb_zHKZG	2020-01-16 10:19:19	423	[]	['univdauphine', 'figaroetudiant']	10	🏆 Vous avez classé @univdauphine en 2e place d...	
257662767	univdauphine	False	6868	21032020	283	257662767	B7EPCLhCon_	2020-01-08 16:56:50	450	['parcoursup']	[]	34	Nouveauté 2020 📢 Dès le 22 janvier, postulez e...	
257662767	univdauphine	False	6868	21032020	283	257662767	B5DM9JxoSb9	2019-11-19 14:16:38	255	['SEEPH2019]', 'SemaineduHandicap', 'DauphineD...	[]	1	[#SEEPH2019] Handi'cap sur Dauphine ! L'univer...	
257662767	univdauphine	False	6868	21032020	283	257662767	B3_s2JCAwhi	2019-10-24 09:06:57	225	['DauphineDurable', 'ODD']	[]	11	Un projet, une idée ou une initiative concerna...	
257662767	univdauphine	False	6868	21032020	283	257662767	B3PeofXohs0	2019-10-05 15:39:12	716	[]	[]	4	Félicitations aux près de 1400 diplômées et di...	
257662767	univdauphine	False	6868	21032020	283	257662767	B269Z0ECofG	2019-09-27 16:24:02	445	['HousingByDauphine,']	['dauphine-housing']	5	Hier soir, Isabelle Huault, Présidente de Daup...	
257662767	univdauphine	False	6868	21032020	283	257662767	B2WpRIPCevD	2019-09-13 13:55:29	606	['RentréeDauphine', 'LSO', 'assoetudiante', 'v...	['univdauphine', 'psl_univ']	9	#RentréeDauphine des L1 #LSO \nL'occasion de d...	
257662767	univdauphine	False	6868	21032020	283	257662767	B0YVFkhiESV	2019-07-26 12:34:46	303	['summer', 'paris', 'dauphine', 'psl', 'univer...	['univdauphine', 'psl_univ']	1	[Fermeture estivale] Le campus parisien de @un...	
								2019-07-		['10ans', 'Carthage'				

Figure 6: Dataframe all the first 10 posts of univdauphine

6 Additional options

Of course, we had a lot of issues with the different HTML codes and some exceptions, we will go through the problems we had and how we solved them. Once we saw that our scrapping method was working, we decided to add some options to have cleaner data, and other information that could be interesting to keep even if we first thought that it was useless data.

6.1 Problems

As we already said, the biggest problem we faced was that we could not send as many requests as possible. The first method we implemented consisted in sending a new request for every post. We quickly saw that after 20-30 posts, the Instagram system kicked us and our requests returned errors. To deal with this problem we had to retrieve the posts through the user's page directly. We also had to add some timesleep functions in our code in order to space out the request and be sure that we would get the HTML code we needed.

Another problem we faced was that we had to deal with every exception possible. When a post does not have any description, place, likes... The HTML code is not the same and to face this problem we had to add a lot of tests to generalize our function. The tests were easy ones: looking if the HTML tag was present in the code, dealing with None values etc.

Once these problems were solved and the scrapping function worked for every kind of post, we wanted to scale it for big quantities of posts. The last problem we faced was that one page contains a limited number of posts, and as we scroll, we are going through the pages without noticing. The request we make gives only the posts of the first *page*. We had to deal with end cursors and to look deeply in the architecture of the web page to understand how to automatically send a request on the second page and so on. This problem was partially solved but not in a very rigorous way. We had to look at the code to see what was changing in the HTML code as we were scrolling and how we could scrap the next page.

6.2 Processing

Once we solved all the technical problems we decided to add some other options in the scrapping functions. This part may be more considered as data cleaning than data extraction but the goal here is to add some options and functionalities that regard data extraction and will help for the cleaning. Indeed, depending on what we want to use the data for, it can be useful to have some more options to have a more precise scrapping and retrieve some particular data that we are the most interested in. As we said before, social networks such as Instagram contain a huge amount of data that can be exploited to make simple statistics or for marketing purposes. Indeed, we can quantify trends, new products, make sentiment analysis on posts, etc. We then thought about simple functionalities in our scrapping function that are easy to implement and can be very useful for further work. The goal of this little additional options is that it can reduce a lot data cleaning and aim directly at useful information. We wanted to have more relevant scrapping functions to reduce data processing and cleaning, than scrap everything we could and after that, take the time to clean the data.

As we said at the beginning, there are about 100 millions new posts everyday. Of course, a huge part of them are "useless" posts as they are private and concern unknown people. Most of the time when we want to scrap a social media we are looking at relevant posts : for example posts from celebrities, brands etc. To avoid cleaning and we decided to add some options to our scrapping method :

- A number minimum of likes : This parameter can be significant when we collect posts from a hashtag, we usually want only relevant posts. Which in most of the cases means posts with a high number of likes. This consists in a simple test on the likes count: if the value is too low, we just skip to the next post.
- A particularity on Instagram is that celebrities or famous accounts can be verified. That means that we are sure that it is their personal account. When we are scrapping users, or posts from an hashtag, it can be interesting to see if the account that posted it is verified. Once again this can help us to retrieve only relevant and interesting posts .
- Keeping the extraction data. If we want to quantify trends or evolution, it is really important to attach to every scrapped post the timestamp at which we collected it. As we may want to collect this post several times, and see the evolution in time.
- We also decided to create a csv file based on the pandas DataFrame. This is only to store every dataframe in the same format and for data processing it is always useful to have them stored in a csv file.

A very important part of a post is the text description. And in the descriptions we can notice relevant information as hashtags or mentions. Most of the time, hashtags are very useful data, they can be interpreted as people labelling their post themselves. Mentions of other users can also be very relevant. We decided to create two functions, that are not really data collecting from the web but can be seen as word or noun extraction from text. The method is really simple, we just go through the text and collect every string that follows a # or a .

Figure 7: Text from a post of @univdauphine

7 Further possible work

Scrapping allows to collect huge amounts of data and can be useful for statistics or machine learning models. As we said before, a social network is a continuous source of data. Everyday very large amounts of data are generated and can be used for studies. Celebrities or brands can quantify their popularity, a lot of companies can use those data for marketing issues etc. When we have structured data, as we have here, we can think of methods or algorithms that will treat the data and, get relevant information from it.

- The automating : for statistical purposes, it can be very interesting to quantify trends based on quantitative data. The number of likes on a post, the number of comments, the post frequency of a user, the frequency of posts containing a certain hashtag, etc. To do so we have to run the same scrapping at different periods, this is why keeping the scrapping date is really important. By scrapping the same user everyday we can hope that we will be able to produce relevant information.

- Text is really important in a post. As we saw in the previous section collecting hashtags and mentions allows us to get very useful information. In addition we can perform NLP algorithms. For example it could be interesting to run some topic modelling algorithm on description, or even sentiment analysis on comments.

- Finally we discovered a very intriguing data in the HTML code in the HTML tag *"accessibility_caption"*. It turns out that Instagram is running its own image segmentation algorithm on the Instagram posts. This is very interesting and we see how performing it is by looking at it. We decided to include it in our scrapping as it can be very useful data. As posts are mostly characterized by the image, it can be really interesting to perform deep learning algorithm on the images such as classification or image segmentation, we can even use the data to train some models, as the image are already labelled by the Instagram algorithm.

8 Conclusion

As said before we are in the Big Data era. Everyday tremendous amounts of data are generated. With the recent progress in machine learning, the data can be very valuable. We decided to focus here on social networks. As we already know social network are a huge source of data. The particularity of social media is that the data flow is continuous, and with all people on that social media, data can be really personal and private. We chose the particular case of Instagram because the data is very structured. It is easier for extraction and and for further works. The main goal of this project was to collect Instagram datas *from scratch* i.e without using APIs or scrapping tools. It took a lot of time as we have to go through a lot of HTML codes and understand how the Instagram website is built. But once we know were to find the data we want to collect, the functions that allow us to retrieve them are quite simple. The other advantage with having structured data is that we can compare them, store them in nice structures (pandas DataFrames) and use them for different purposes. Of course as there are some quantitative data, it allows us to make some basic statistics. But we also have image and text data that can be very useful for machine learning