

| **Name: Khan shoab akhat vakil ahamad** | **Course Name: SMA-LAB** |
| --- | --- |
| **Class: BE-CO** | **Batch: 01** |
| **Roll no: 19CO33** | **Experiment No: 02** |

**AIM :** Data Scraping and Storage- Preprocess, filter and store social media data for business (Using Python, MongoDB, R, etc).

**1.INSTAGRAM :**

**CODE :**

try:

import argparse

import json

import requests

from fake\_headers import Headers

except ModuleNotFoundError:

print("Please download dependencies from requirement.txt")

except Exception as ex:

print(ex)

'''can scrap only public instagram accounts'''

class Instagram:

@staticmethod

def build\_param(username):

params = {

'username': username,

}

return params

@staticmethod

def build\_headers(username):

return {

'authority': 'www.instagram.com',

'accept': '\*/\*',

'accept-language': 'en-US,en;q=0.9',

'referer': f'https://www.instagram.com/{username}/',

'sec-ch-prefers-color-scheme': 'dark',

'sec-ch-ua': '"Not?A\_Brand";v="8", "Chromium";v="108", "Microsoft Edge";v="108"',

'sec-fetch-dest': 'empty',

'sec-fetch-mode': 'cors',

'sec-fetch-site': 'same-origin',

'user-agent': Headers().generate()['User-Agent'],

'x-asbd-id': '198387',

'x-csrftoken': 'VUm8uVUz0h2Y2CO1SwGgVAG3jQixNBmg',

'x-ig-app-id': '936619743392459',

'x-ig-www-claim': '0',

'x-requested-with': 'XMLHttpRequest',

}

@staticmethod

def make\_request(url, params, headers, proxy=None):

response = None

if proxy:

proxy\_dict = {

'http': f'http://{proxy}',

'https': f'http://{proxy}'

}

response = requests.get(

url, headers=headers, params=params, proxies=proxy\_dict)

else:

response = requests.get(

url, headers=headers, params=params)

return response

@staticmethod

def scrap(username, proxy = None):

try:

headers = Instagram.build\_headers(username)

params = Instagram.build\_param(username)

response = Instagram.make\_request('https://www.instagram.com/api/v1/users/web\_profile\_info/',

headers=headers, params=params, proxy=proxy)

if response.status\_code == 200:

profile\_data = response.json()['data']['user']

return json.dumps(profile\_data)

else:

print('Error : ', response.status\_code, response.text)

except Exception as ex:

print(ex)

if \_\_name\_\_ == '\_\_main\_\_':

parser = argparse.ArgumentParser()

parser.add\_argument("username", help="username to search")

parser.add\_argument("--proxy", help="Proxy to use", default=None)

args = parser.parse\_args()

data = Instagram.scrap(args.username, args.proxy)

f = open("instagramtData.txt","w")

f.write(data)

**OUTPUT :** {"biography": "", "bio\_links": [], "biography\_with\_entities": {"raw\_text": "", "entities": []}, "blocked\_by\_viewer": false, "restricted\_by\_viewer": null, "country\_block": false, "external\_url": null, "external\_url\_linkshimmed": null, "edge\_followed\_by": {"count": 0}, "fbid": "17841400963000146", "followed\_by\_viewer": false, "edge\_follow": {"count": 0}, "follows\_viewer": false, "full\_name": "", "group\_metadata": null, "has\_ar\_effects": false, "has\_clips": false, "has\_guides": false, "has\_channel": false, "has\_blocked\_viewer": false, "highlight\_reel\_count": 0, "has\_requested\_viewer": false, "hide\_like\_and\_view\_counts": false, "id": "9376100", "is\_business\_account": false, "is\_professional\_account": false, "is\_supervision\_enabled": false, "is\_guardian\_of\_viewer": false, "is\_supervised\_by\_viewer": false, "is\_supervised\_user": false, "is\_embeds\_disabled": false, "is\_joined\_recently": false, "guardian\_id": null, "business\_address\_json": null, "business\_contact\_method": "UNKNOWN", "business\_email": null, "business\_phone\_number": null, "business\_category\_name": null, "overall\_category\_name": null, "category\_enum": null, "category\_name": null, "is\_private": true, "is\_verified": false, "edge\_mutual\_followed\_by": {"count": 0, "edges": []}, "profile\_pic\_url": "https://instagram.fruh4-6.fna.fbcdn.net/v/t51.2885-19/44884218\_345707102882519\_2446069589734326272\_n.jpg?\_nc\_ht=instagram.fruh4-6.fna.fbcdn.net&\_nc\_cat=1&\_nc\_ohc=z9RV69IotL0AX8xkz8m&edm=ABmJApABAAAA&ccb=7-5&ig\_cache\_key=YW5vbnltb3VzX3Byb2ZpbGVfcGlj.2-ccb7-5&oh=00\_AfBfQ7cfIRZfut9SHQBtOph0X7EqRpeSdcspGezCVHueiA&oe=63DFB9CF&\_nc\_sid=6136e7", "profile\_pic\_url\_hd": "https://instagram.fruh4-6.fna.fbcdn.net/v/t51.2885-19/44884218\_345707102882519\_2446069589734326272\_n.jpg?\_nc\_ht=instagram.fruh4-6.fna.fbcdn.net&\_nc\_cat=1&\_nc\_ohc=z9RV69IotL0AX8xkz8m&edm=ABmJApABAAAA&ccb=7-5&ig\_cache\_key=YW5vbnltb3VzX3Byb2ZpbGVfcGlj.2-ccb7-5&oh=00\_AfBfQ7cfIRZfut9SHQBtOph0X7EqRpeSdcspGezCVHueiA&oe=63DFB9CF&\_nc\_sid=6136e7", "requested\_by\_viewer": false, "should\_show\_category": false, "should\_show\_public\_contacts": false, "transparency\_label": null, "transparency\_product": "STATE\_CONTROLLED\_MEDIA", "username": "uniliver", "connected\_fb\_page": null, "pronouns": [], "edge\_felix\_video\_timeline": {"count": 0, "page\_info": {"has\_next\_page": false, "end\_cursor": null}, "edges": []}, "edge\_owner\_to\_timeline\_media": {"count": 9, "page\_info": {"has\_next\_page": false, "end\_cursor": ""}, "edges": []}, "edge\_saved\_media": {"count": 0, "page\_info": {"has\_next\_page": false, "end\_cursor": null}, "edges": []}, "edge\_media\_collections": {"count": 0, "page\_info": {"has\_next\_page": false, "end\_cursor": null}, "edges": []}, "edge\_related\_profiles": {"edges": []}}

**2.PINTEREST :**

try:

import argparse

from fake\_headers import Headers

import requests

import json

except ModuleNotFoundError:

print("Please download dependencies from requirement.txt")

except Exception as ex:

print(ex)

class Pinterest:

'''This class scraps pinterest and returns a dict containing all user data'''

@staticmethod

def \_generate\_url(username):

return "https://pinterest.com/resource/UserResource/get/?source\_url=%25{}%2F&data=%7B%22options%22%3A%7B%22field\_set\_key%22%3A%22profile%22%2C%22username%22%3A%22{}%22%2C%22is\_mobile\_fork%22%3Atrue%7D%2C%22context%22%3A%7B%7D%7D&\_=1640428319046".format(username, username)

@staticmethod

def \_make\_request(url):

headers = Headers().generate()

response = requests.get(url, headers=headers)

return response

@staticmethod

def scrap(username):

try:

try:

url = Pinterest.\_generate\_url(username)

response = Pinterest.\_make\_request(url)

if response.status\_code == 200:

response = response.json()

else:

print("Failed to get Data!")

exit()

except Exception as ex:

print("Error", ex)

exit()

json\_data = response

data = json\_data['resource\_response']['data']

return json.dumps(data)

except Exception as ex:

print(ex)

if \_\_name\_\_ == '\_\_main\_\_':

parser = argparse.ArgumentParser()

parser.add\_argument("username", help="username to search")

args = parser.parse\_args()

data = Pinterest.scrap(args.username)

f= open("pinterestData.txt","w")

f.write(data)

**OUTPUT :** {"last\_name": "\u0422\u0430\u043d\u0447\u0435\u0432\u0438", "listed\_website\_url": null, "impressum\_url": null, "is\_ads\_only\_profile": false, "username": "uniliver", "profile\_reach": -1, "last\_pin\_save\_time": "Tue, 31 Jan 2023 08:05:42 +0000", "storefront\_search\_enabled": false, "seo\_title": "\u0418\u0432\u0430\u043d \u0414\u0435\u043b\u044f\u043d\u0430 \u0422\u0430\u043d\u0447\u0435\u0432\u0438 (uniliver) - Profile | Pinterest", "is\_primary\_website\_verified": false, "is\_tastemaker": false, "domain\_verified": false, "joined\_communities\_count": 0, "website\_url": null, "show\_discovered\_feed": null, "image\_small\_url": "https://i.pinimg.com/30x30\_RS/af/ed/ae/afedae86768749d0938c0d1ae25d3cd3.jpg", "type": "user", "seo\_description": "See what \u0418\u0432\u0430\u043d \u0414\u0435\u043b\u044f\u043d\u0430 \u0422\u0430\u043d\u0447\u0435\u0432\u0438 (uniliver) has discovered on Pinterest, the world's biggest collection of ideas.", "profile\_discovered\_public": null, "storefront\_management\_enabled": true, "first\_name": "\u0418\u0432\u0430\u043d \u0414\u0435\u043b\u044f\u043d\u0430", "show\_creator\_profile": false, "full\_name": "\u0418\u0432\u0430\u043d \u0414\u0435\u043b\u044f\u043d\u0430 \u0422\u0430\u043d\u0447\u0435\u0432\u0438", "native\_pin\_count": 0, "domain\_url": null, "has\_catalog": false, "pins\_done\_count": 0, "seo\_canonical\_domain": "www.pinterest.com", "verified\_identity": {}, "eligible\_profile\_tabs": [{"id": "8782887847649", "type": "profiletab", "tab\_type": 1, "name": "Created", "is\_default": false}, {"id": "8782887847653", "type": "profiletab", "tab\_type": 0, "name": "Saved", "is\_default": true}], "blocked\_by\_me": false, "has\_published\_pins": false, "image\_large\_url": "https://i.pinimg.com/140x140\_RS/af/ed/ae/afedae86768749d0938c0d1ae25d3cd3.jpg", "board\_count": 4, "id": "640637253151885408", "profile\_views": -1, "show\_impressum": false, "show\_engagement\_tab": false, "interest\_following\_count": 3, "profile\_cover": {"id": "640637253151885408"}, "following\_count": 16, "is\_verified\_merchant": false, "partner": null, "image\_medium\_url": "https://i.pinimg.com/75x75\_RS/af/ed/ae/afedae86768749d0938c0d1ae25d3cd3.jpg", "has\_showcase": false, "indexed": true, "pin\_count": 69, "has\_shopping\_showcase": false, "image\_xlarge\_url": "https://i.pinimg.com/280x280\_RS/af/ed/ae/afedae86768749d0938c0d1ae25d3cd3.jpg", "has\_custom\_board\_sorting\_order": false, "explicitly\_followed\_by\_me": false, "is\_inspirational\_merchant": false, "video\_pin\_count": 0, "explicit\_user\_following\_count": 13, "ads\_only\_profile\_site": null, "follower\_count": 0, "pronouns": [], "seo\_noindex\_reason": null, "is\_partner": false, "group\_board\_count": 0, "about": "", "story\_pin\_count": 0, "repins\_from": [{"id": "31525403550082472", "full\_name": "Mm", "image\_medium\_url": "https://s.pinimg.com/images/user/default\_75.png", "repins\_from": [], "image\_small\_url": "https://s.pinimg.com/images/user/default\_30.png", "username": "milano\_222"}, {"id": "389913417646895016", "full\_name": "Greg Billman", "image\_medium\_url": "https://s.pinimg.com/images/user/default\_75.png", "repins\_from": [], "image\_small\_url": "https://s.pinimg.com/images/user/default\_30.png", "username": "gbillman"}, {"id": "415175796806655134", "full\_name": "Tienda En L\u00ednea", "image\_medium\_url": "https://i.pinimg.com/75x75\_RS/e9/97/8c/e9978ca5a46c3222443d2636a2f3cc89.jpg", "repins\_from": [], "image\_small\_url": "https://i.pinimg.com/30x30\_RS/e9/97/8c/e9978ca5a46c3222443d2636a2f3cc89.jpg", "username": "latiendaenlinea"}], "has\_board": true}

**3.TWITTER :**

try:

import requests

import argparse

from fake\_headers import Headers

import json

except ModuleNotFoundError:

print("Please download dependencies from requirement.txt")

except Exception as ex:

print(ex)

AUTHORIZATION\_KEY = 'Bearer AAAAAAAAAAAAAAAAAAAAANRILgAAAAAAnNwIzUejRCOuH5E6I8xnZz4puTs%3D1Zv7ttfk8LF81IUq16cHjhLTvJu4FA33AGWWjCpTnA'

class Twitter:

@staticmethod

def find\_x\_guest\_token():

try:

headers = {

'authorization': AUTHORIZATION\_KEY,

}

response = requests.post(

'https://api.twitter.com/1.1/guest/activate.json', headers=headers)

return response.json()['guest\_token']

except Exception as ex:

print("Error at find\_x\_guest\_token: {}".format(ex))

@staticmethod

def make\_http\_request(URL, headers):

try:

response = requests.get(URL, headers=headers)

if response and response.status\_code == 200:

return response.json()

except Exception as ex:

print("Error at make\_http\_request: {}".format(ex))

@staticmethod

def build\_headers(x\_guest\_token, authorization\_key):

headers = {

'authority': 'twitter.com',

'accept': '\*/\*',

'accept-language': 'en-US,en;q=0.9',

'authorization': authorization\_key,

'sec-fetch-dest': 'empty',

'sec-fetch-mode': 'cors',

'sec-fetch-site': 'same-origin',

'user-agent': Headers().generate()['User-Agent'],

'x-guest-token': x\_guest\_token,

'x-twitter-active-user': 'yes',

'x-twitter-client-language': 'en',

}

return headers

@staticmethod

def scrap(username):

try:

# generating URL according to the username

guest\_token = Twitter.find\_x\_guest\_token()

headers = Twitter.build\_headers(guest\_token, AUTHORIZATION\_KEY)

response = Twitter.make\_http\_request(

"https://api.twitter.com/1.1/users/show.json?screen\_name={}".format(username),

headers=headers)

if response:

return json.dumps(response)

else:

print("Failed to make Request!")

except Exception as ex:

print(ex)

if \_\_name\_\_ == '\_\_main\_\_':

parser = argparse.ArgumentParser()

parser.add\_argument("username", help="username to search")

args = parser.parse\_args()

data = Twitter.scrap(args.username)

f = open("twitterData.txt","w")

f.write(data)

**OUTPUT :** {"id": 1703704490, "id\_str": "1703704490", "name": "Hindustan Unilever", "screen\_name": "HUL\_News", "location": "Mumbai, India", "profile\_location": null, "description": "Hindustan Unilever Limited has a clear purpose - to make sustainable living commonplace #sustliving", "url": "https://t.co/FcFpucU5JR", "entities": {"url": {"urls": [{"url": "https://t.co/FcFpucU5JR", "expanded\_url": "https://www.hul.co.in", "display\_url": "hul.co.in", "indices": [0, 23]}]}, "description": {"urls": []}}, "protected": false, "followers\_count": 103401, "fast\_followers\_count": 0, "normal\_followers\_count": 103401, "friends\_count": 94, "listed\_count": 193, "created\_at": "Tue Aug 27 04:17:55 +0000 2013", "favourites\_count": 3167, "utc\_offset": null, "time\_zone": null, "geo\_enabled": false, "verified": true, "statuses\_count": 10334, "media\_count": 1889, "lang": null, "status": {"created\_at": "Wed Feb 01 11:50:11 +0000 2023", "id": 1620751360705822720, "id\_str": "1620751360705822720", "text": "9 years of Prabhat. \n\nOne mission. \n\nTo create sustainable &amp; inclusive communities around our factory operations ac\u2026 https://t.co/oWPjpPaeZs", "truncated": true, "entities": {"hashtags": [], "symbols": [], "user\_mentions": [], "urls": [{"url": "https://t.co/oWPjpPaeZs", "expanded\_url": "https://twitter.com/i/web/status/1620751360705822720", "display\_url": "twitter.com/i/web/status/1\u2026", "indices": [121, 144]}]}, "source": "<a href=\"https://mobile.twitter.com\" rel=\"nofollow\">Twitter Web App</a>", "in\_reply\_to\_status\_id": null, "in\_reply\_to\_status\_id\_str": null, "in\_reply\_to\_user\_id": null, "in\_reply\_to\_user\_id\_str": null, "in\_reply\_to\_screen\_name": null, "geo": null, "coordinates": null, "place": null, "contributors": null, "is\_quote\_status": false, "retweet\_count": 1, "favorite\_count": 3, "favorited": false, "retweeted": false, "possibly\_sensitive": false, "possibly\_sensitive\_editable": true, "lang": "en", "supplemental\_language": null}, "contributors\_enabled": false, "is\_translator": false, "is\_translation\_enabled": false, "profile\_background\_color": "C0DEED", "profile\_background\_image\_url": "http://abs.twimg.com/images/themes/theme1/bg.png", "profile\_background\_image\_url\_https": "https://abs.twimg.com/images/themes/theme1/bg.png", "profile\_background\_tile": true, "profile\_image\_url": "http://pbs.twimg.com/profile\_images/778884220903104512/HzxN3b\_J\_normal.jpg", "profile\_image\_url\_https": "https://pbs.twimg.com/profile\_images/778884220903104512/HzxN3b\_J\_normal.jpg", "profile\_banner\_url": "https://pbs.twimg.com/profile\_banners/1703704490/1575614759", "profile\_link\_color": "36ABD6", "profile\_sidebar\_border\_color": "FFFFFF", "profile\_sidebar\_fill\_color": "DDEEF6", "profile\_text\_color": "333333", "profile\_use\_background\_image": true, "has\_extended\_profile": false, "default\_profile": false, "default\_profile\_image": false, "pinned\_tweet\_ids": [], "pinned\_tweet\_ids\_str": [], "has\_custom\_timelines": false, "following": null, "follow\_request\_sent": null, "notifications": null, "advertiser\_account\_type": "promotable\_user", "advertiser\_account\_service\_levels": ["dso", "dso", "dso", "dso", "dso", "reseller"], "business\_profile\_state": "none", "translator\_type": "regular", "withheld\_in\_countries": [], "require\_some\_consent": false}

**Conclusion:**

We have successfully collected Unilever social media data from websites like

Instagram, Facebook, and Twitter. In order to connect and capture social media

data for businesses (scraping, crawling, parsing).