Università degli Studi di Cagliari
Corso di Laurea DSBAI

Web Analytics e Analisi Testuale

http://agilegroup.eu

A.A. 2017/2018

Ing. Marco Ortu
Via Porcell 4, primo piano
mail: marco.ortu@diee.unica.it

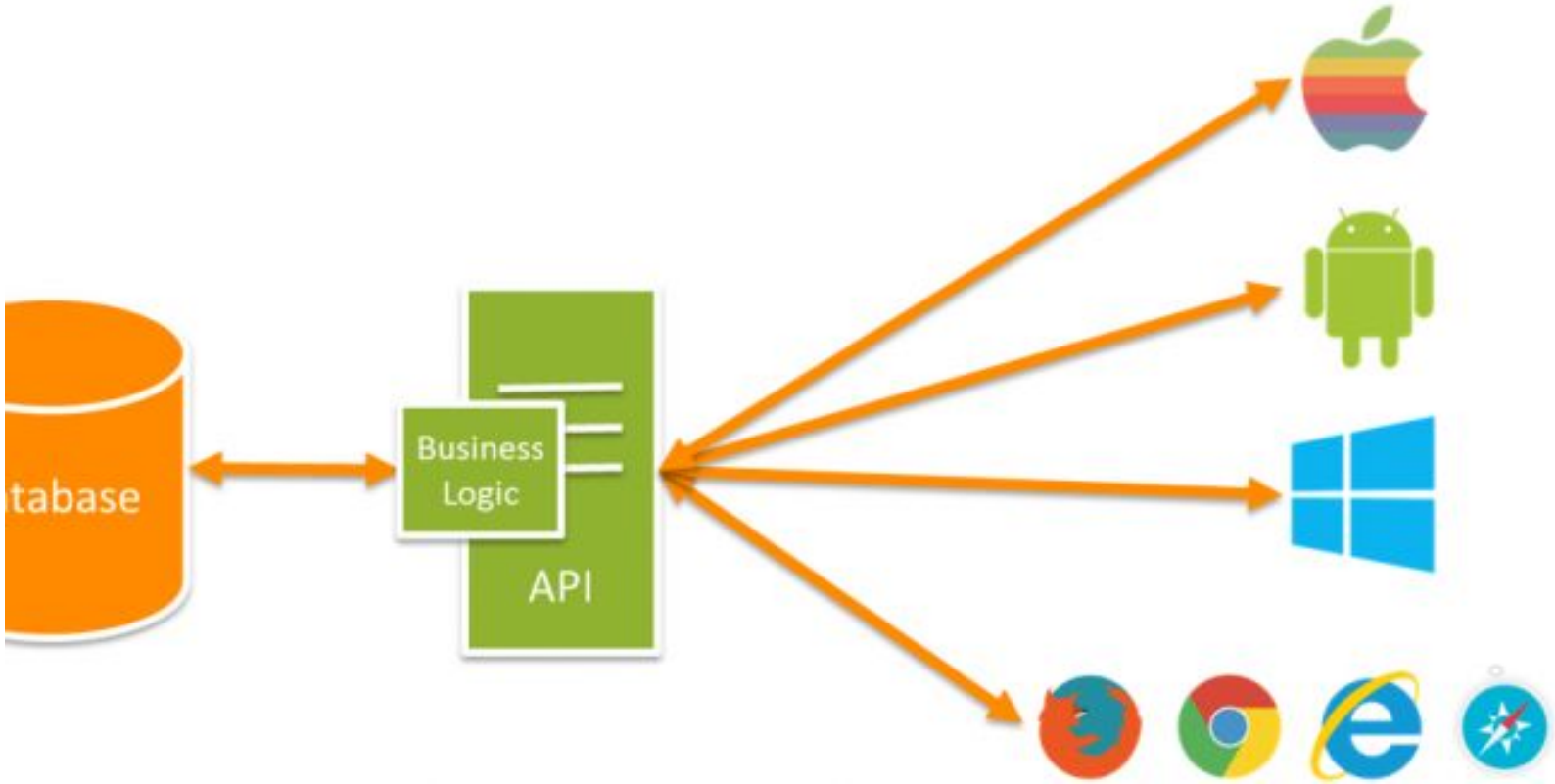
Social Web Mining

Mining Social Web

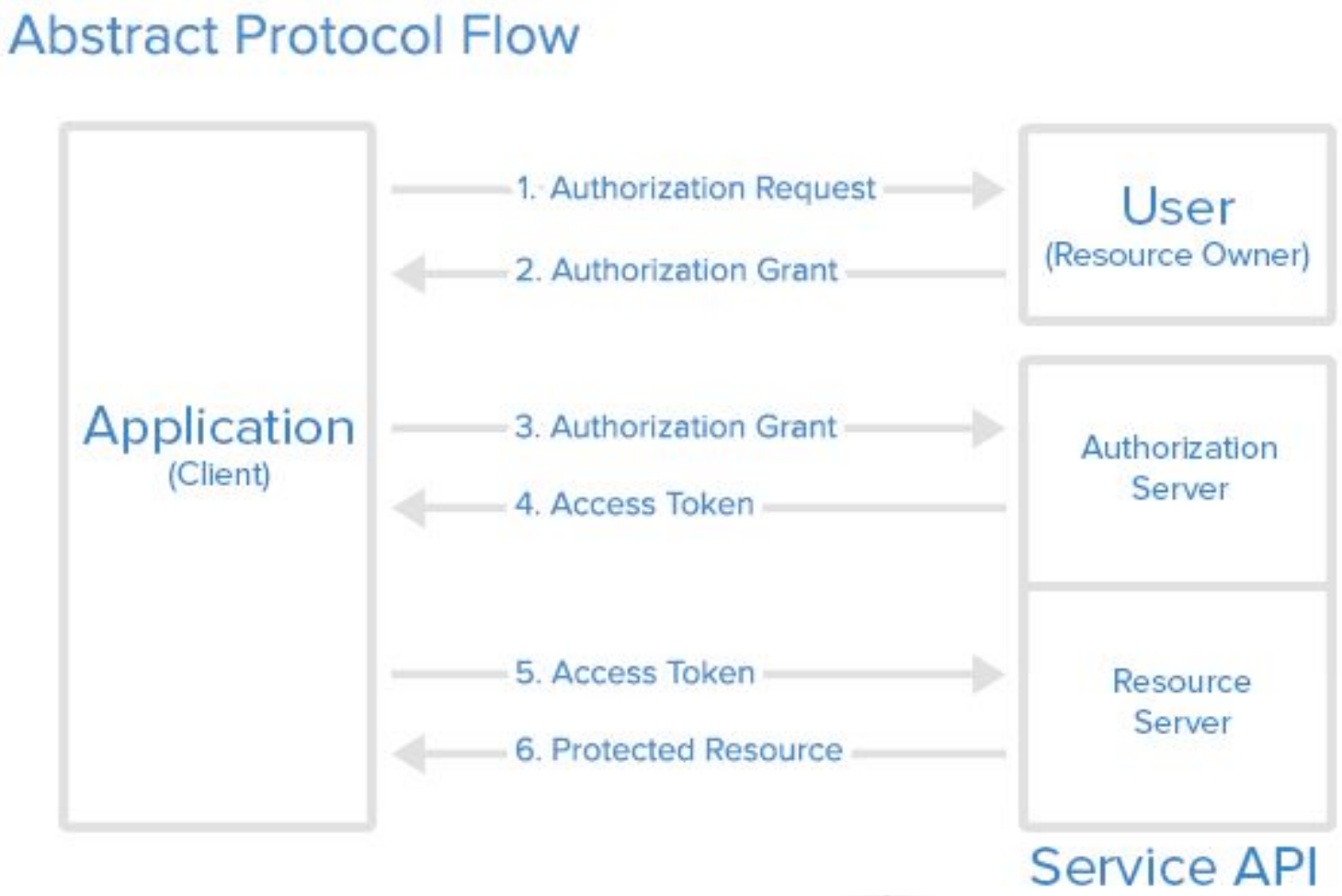
Il cosiddetto ***Social Web*** rappresenta la più grande miniera di dati sugli utenti del Web liberamente disponibile nonché un industria da miliardi di dollari.
Vedremo come analizzare le caratteristiche di tre importanti social media.

- Twitter
  - analisi dei topic trend
  - analisi di “di cosa le persone stanno parlando?”
- Facebook
  - analisi delle Fan Page
  - esplorazione delle Friendship
- LinkedIn
  - Clustering dei colleghi
  - Mining job’s titles

Social Web: Web API



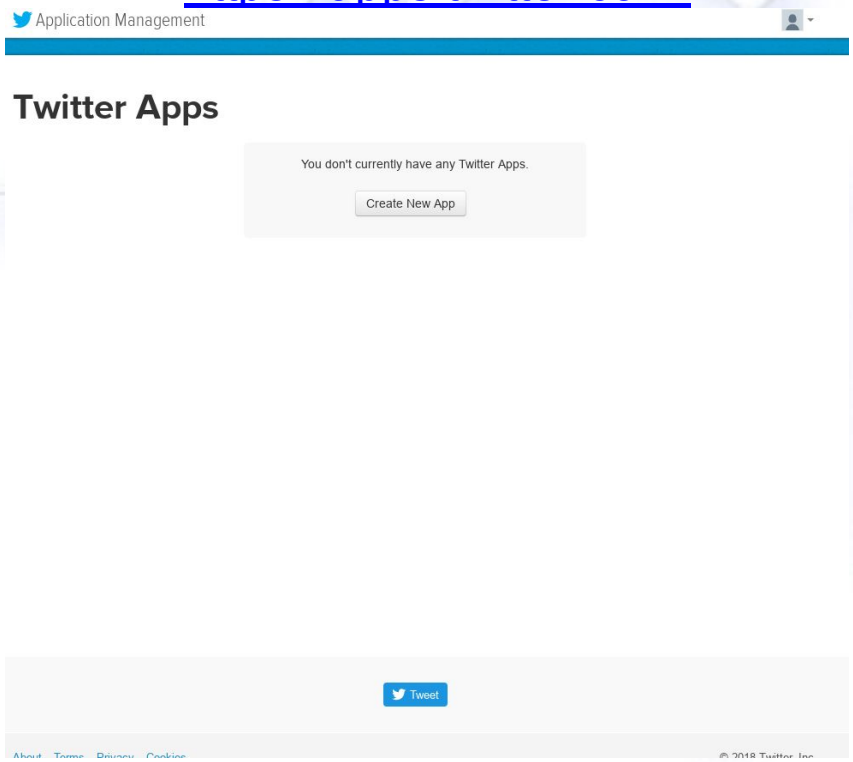
Social Web: Protocollo OAuth-2



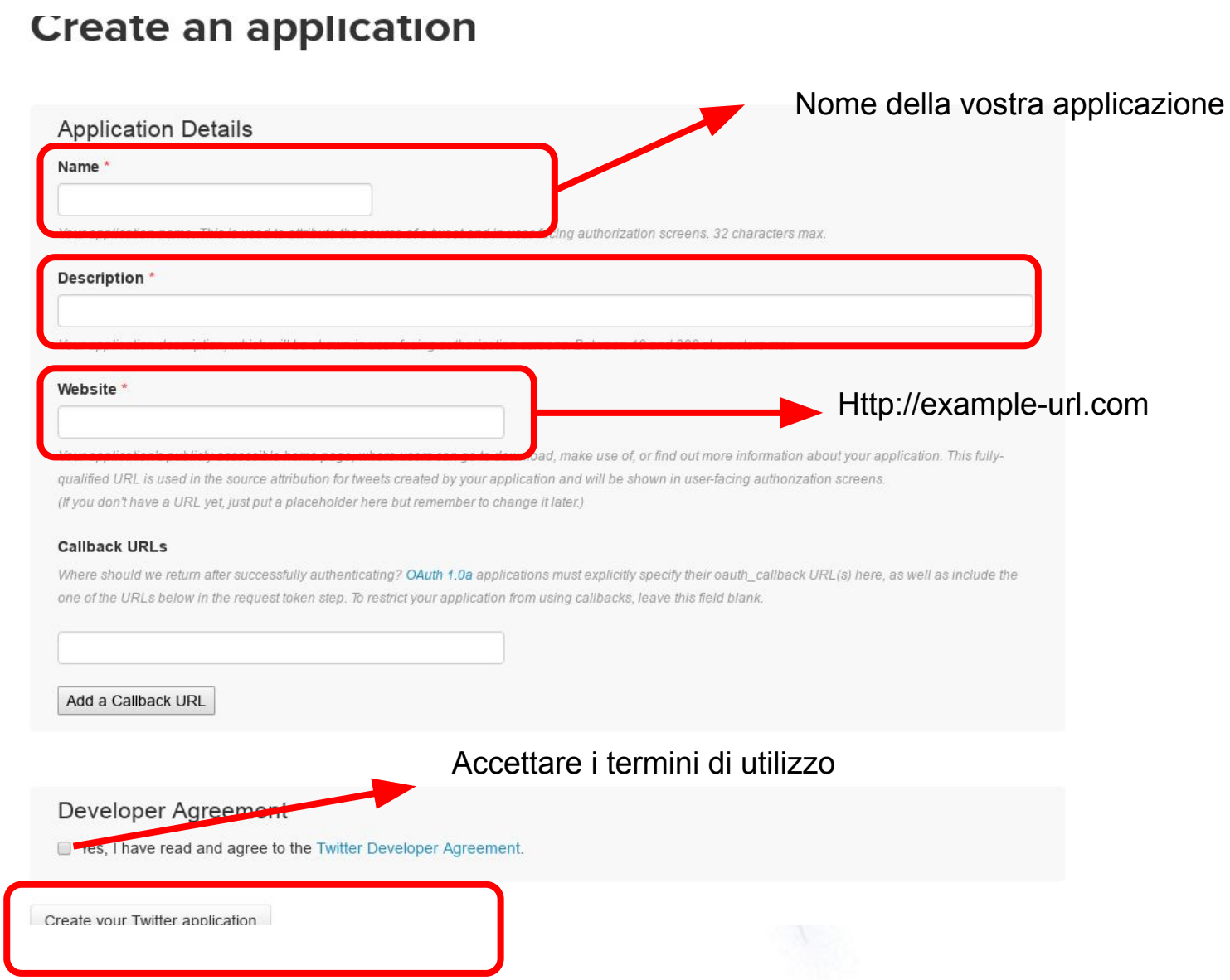
Social Web: Twitter

Il primo step per poter lavorare con Twitter è quello di ottenere delle credenziali per poter accedere alle API fornite.
Per ottenere delle credenziali è necessario creare una "***app***" al seguente indirizzo:

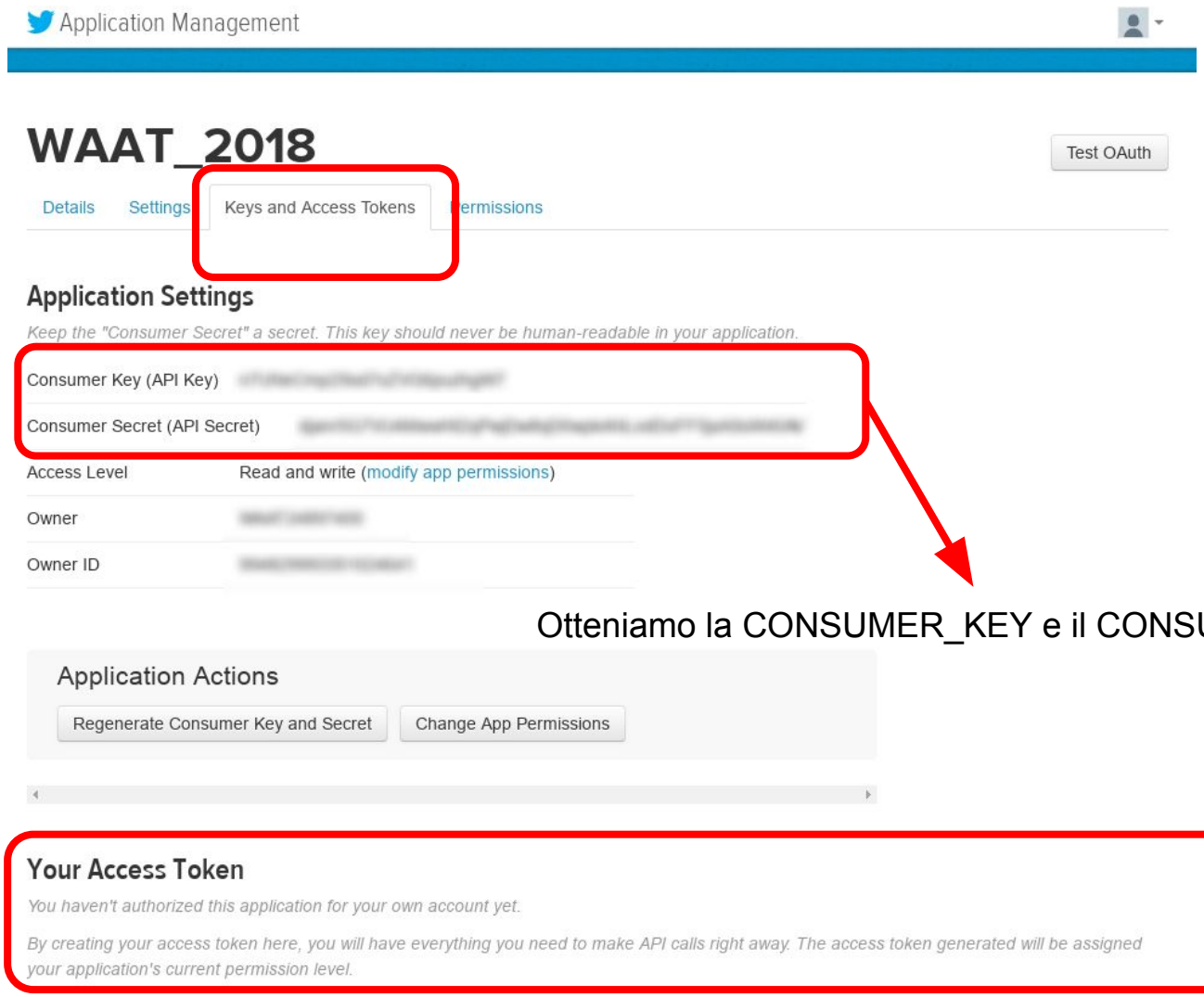
https://apps.twitter.com/



Social Web: Twitter



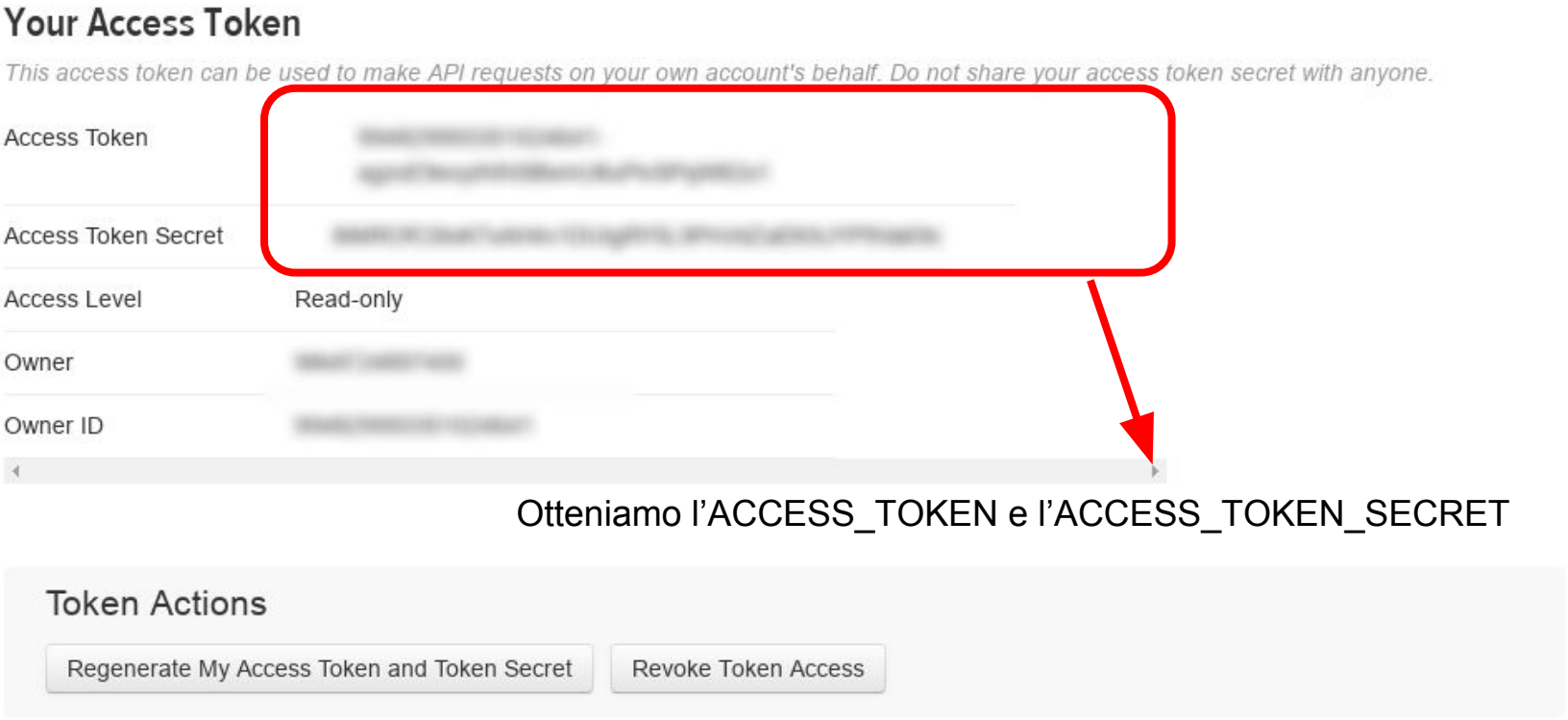
Social Web: Twitter



Otteniamo la CONSUMER_KEY e il CONSUMER_SECRET

Social Web: Twitter

Generiamo adesso gli access tokens per la nostra applicazione



Otteniamo l'ACCESS_TOKEN e l'ACCESS_TOKEN_SECRET

Social Web: Twitter

1. Dal repository ***https://github.com/marcoortu/WAAT-2019*** fare il checkout del branch ***social.***

2. Creare il file ***credentials.py*** che contiene le credenziali per i diversi social media che analizzeremo, sostituire le credenziali appena ottenute nelle seguenti variabili.

```
# Mettere qui i dati corretti per Twitter
TWITTER_CONSUMER_KEY = 'XXXXXXXXXXXXXXXXXXXXXXXX'
TWITTER_CONSUMER_SECRET = 'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
TWITTER_ACCESS_TOKEN = 'XXXXXXXXXXXXXXXXXXXXXXXXXXXX'
TWITTER_ACCESS_TOKEN_SECRET = 'XXXXXXXXXXXXXXXXXXXXXXXX'
```

Social Web: Twitter

Utilizziamo la libreria di appoggio ***tweepy*** (http://www.tweepy.org/) che permette di fare richieste alle Web API di Twitter in maniera molto semplice.
Creiamo un client di autenticazione con le credenziali ottenute nei passaggi precedenti.

```
import tweepy
from tweepy import OAuthHandler
auth = OAuthHandler(
  TWITTER_CONSUMER_KEY,
  TWITTER_CONSUMER_SECRET
)
auth.set_access_token(TWITTER_ACCESS_TOKEN,
TWITTER_ACCESS_TOKEN_SECRET)
twitterAPI = tweepy.API(auth)
```

Otteniamo l'oggetto ***twitterAPI*** che ci permette di interagire con le Web API fornite da Twitter.

Social Web: Twitter

Testiamo le nostre credenziali facendo una ricerca di tweets dato un testo di ricerca.

```
fetchedTweets = twitterAPI.search(q="data scientist", count=100)
for tweet in fetchedTweets:
   pprint(tweet)
```

Il metodo ***search*** richiede la query di ricerca e un numero massimo di tweet.
Il nostro oggetto ***twitterAPI*** è semplicemente un wrapper che si occupa di fare le chiamate alle Web API di Twitter per noi.
Nella documentazione troviamo tutte le informazioni per utilizzare la chiamata.

https://developer.twitter.com/en/docs/tweets/search/api-reference/get-search-tweets

Social Web: Twitter

Standard search API

Returns a collection of relevant Tweets matching a specified query.

Please note that Twitter's search service and, by extension, the Search API is not meant to be an exhaustive source of Tweets. Not all Tweets will be indexed or made available via the search interface.

To learn how to use Twitter Search effectively, please see the Standard search operators page for a list of available filter operators. Also, see the Working with Timelines page to learn best practices for navigating results by `since_id` and `max_id`.

Resource URL

https://api.twitter.com/1.1/search/tweets.json

URL da chiamare tramite protocollo HTTP

Resource Information

Response formats	JSON
Requires authentication?	Yes
Rate limited?	Yes
Requests / 15-min window (user auth)	180
Requests / 15-min window (app auth)	450

Limiti delle chiamate

Parametri della richiesta

Parameters

Name	Required	Description	Default Value	Example
q	required	A UTF-8, URL-encoded search query of 500 characters maximum, including operators. Queries may additionally be limited by complexity.		*@noradio*

Social Web: Twitter

```
fetchedTweets = twitterAPI.search(q="data scientist", count=100)
for tweet in fetchedTweets:
   print tweet
```

Il parametro count specifica quanti risultati mettere per pagina, con un massimo di 100 e un valore predefinito di 15.

count	optional	The number of tweets to return per page, up to a maximum of 100. Defaults to 15. This was formerly the “rpp” parameter in the old Search API.		*100*

Per una lista completa delle funzionalità fornite dalle Web API di Twitter si può consultare la pagine ufficiale della documentazione.

https://developer.twitter.com/en/docs

Social Web: Twitter Limits

Le richieste che si possono fare alle API sono limitate.

https://developer.twitter.com/en/docs/basics/rate-limiting

Una app con permessi standard (default) può effettuare un massimo di 15 richieste ogni 15 minuti. Superato questo limite per ogni richiesta successiva verrà restituito il seguente errore:

HTTP 429 “Too Many Requests”

Social Web: Twitter

Creiamo ora una piccola applicazione per la sentiment analysis. Per semplicità utilizziamo una libreria di appoggio che si occupa di valutare il sentiment di un dato testo: ***TextBlog***.

TextBlob è una libreria di alto livello costruita utilizzando la libreria ***NLTK***. Quando passiamo del testo per creare un oggetto ***TextBlob***, le seguenti elaborazioni vengono effettutate:

1. Tokenize del tweet, cioè le parole divise dal corpo del testo.
2. Rimuove le stopwords dai token (le stopwords sono le parole comunemente usate che sono irrilevanti nell'analisi del testo come I, am, you, are, ecc.)
3. Fai tagging POS (part of speech) dei token e seleziona solo features/token significative come aggettivi, avverbi, ecc.
4. Passa i token a un classificatore di sentimenti che classifica il sentimento dei tweet come positivo, negativo o neutro assegnandogli una polarità tra -1.0 e 1.0.

Social Web: Twitter

```
from textblob import TextBlob
import re

def cleanTweet(tweet):
   return ' '.join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t]) | (\w+:\ / \ / \S+)", " ",
tweet).split())

def getTweetSentiment(tweet):
   analysis = TextBlob(cleanTweet(tweet))
   if analysis.sentiment.polarity > 0:
      return 'positive'
   elif analysis.sentiment.polarity == 0:
      return 'neutral'
   else:
      return 'negative'
```

Social Web: Twitter

```
def cleanTweet(tweet):
  return ' '.join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t]) | (\w+:\ / \ / \S+)", " ", tweet).split())
```

La funzione di appoggio ***cleanTweet*** serve a rimuovere dal testo links e caratteri speciali che creano problemi nell'analisi del sentiment, utilizzando le regular expressions.

```
def getTweetSentiment(tweet):
  analysis = TextBlob(cleanTweet(tweet))
  if analysis.sentiment.polarity > 0:
    return 'positive'
  elif analysis.sentiment.polarity == 0:
    return 'neutral'
  else:
    return 'negative'
```

La funzione ***getTweetSentiment*** crea un oggetto ***TextBlog,*** passando il testo ripulito del tweet, che contiene i valori del sentiment calcolati.

Social Web: Twitter

Ecco come viene creato il classificatore dei sentimenti:

1. ***TextBlob*** utilizza il corpus delle Movie Reviews di ***NLTK*** in cui le recensioni sono già state contrassegnate come positive o negative.
2. Le feature positive e negative sono estratte rispettivamente da ciascuna recensione positiva e negativa.
3. I dati di addestramento comprendono features positive e negative con etichetta. Questi dati sono addestrati su un classificatore ***Naive Bayes***.

Usiamo il metodo ***sentiment.polarity*** della classe ***TextBlob*** per ottenere la polarità del tweet tra -1 e 1. Quindi, classifichiamo la polarità come:

```
if analysis.sentiment.polarity > 0:
  return 'positive'
elif analysis.sentiment.polarity == 0:
  return 'neutral'
else:
  return 'negative'
```

Social Web: Twitter

Un aspetto molto importante quando si estraggono dati dai social media è la mole dei dati da estrarre. Sarebbe infatti impensabile andare a scaricare centinaia di migliaia di tweet con una solo richiesta alle Web API.

Per questi motivi la maggior parte dei social media fornisce le proprie API con un sistema di paginazione basato sui ***Cursori***. Consideriamo il seguente esempio.

```
page = 1
while True:
  statuses = twitterAPI.user_timeline(page=page)
  if statuses:
    for status in statuses:
      # process status here
      pass
  else:
    # All done
    break
  page += 1  # next page
```

Social Web: Twitter

In questo caso bisogna considerare manualmente la paginazione, continuando ad iterare finchè ci sono nuove pagine. Ad ogni iterazione il nostro client farà una nuova richiesta per ottenere i nuovi tweet. Utilizzando un oggetto ***Cursor*** invece possiamo ottenere un oggetto direttamente iterabili:

```
for status in tweepy.Cursor(twitterAPI.user_timeline, id="twitter").items():
    print status
```

In questo caso il cursore continuerà ad ottenere dati finchè questi saranno disponibili, gestendo automaticamente la paginazione.

Social Web: Twitter

```
for status in tweepy.Cursor(twitterAPI.user_timeline, id="twitter").items():
    print status
```

In questo caso il cursore continuerà ad ottenere dati finchè questi saranno disponibili, gestendo automaticamente la paginazione.

```
# Itera solo i sui primi 200 elementi
for status in tweepy.Cursor(twitterAPI.user_timeline).items(200):
    print status
# itera solo sulle prime 3 pagine
for page in tweepy.Cursor(twitterAPI.user_timeline).pages(3):
    print page
```

Social Web: Twitter

Analizziamo nel dettaglio le informazioni contenute negli oggetti restituiti dalla ricerca.

```
tweets = tweepy.Cursor(twitterAPI.search, q=query).items(count)
status_texts, screen_names, hashtags= [],[],[]
for tweet in tweets:
    status_texts.append(tweet.text)
    screen_names += [userMention['screen_name'] for userMention in tweet.entities['user_mentions']]
    hashtags += [hashtag['text'] for hashtag in tweet.entities['hashtags']]
words = [word for text in status_texts for word in word_tokenize(text)]
for label, data in [('Word', words),
                    ('Screen Names', screen_names),
                    ('Hashtag', hashtags)
                    ]:
    prettyTable = PrettyTable(field_names=[label, 'Count'])
    counter = Counter(data)
    [prettyTable.add_row(kv) for kv in counter.most_common()[:10]]
    prettyTable.align[label] = 'l'
    prettyTable.align['Count'] = 'r'
    print prettyTable
```

Social Web: Twitter

PrettyTable è una libreria Python per la generazione di semplici tabelle ASCII.
È ispirata dalle tabelle ASCII utilizzate nella shell di PostgreSQL psql.

Possiamo controllare molti aspetti di una tabella, come la larghezza del riempimento della colonna, l'allineamento del testo o il bordo della tabella. Possiamo ordinare i dati.
Dato un insieme di tweet estraiamo dai risultati, il testo, il nome utente, le ***user mentions*** (@username), le parole contenute nel testo e gli hashtag presenti.

```
tweets = tweepy.Cursor(twitterAPI.search, q=query).items(count)
  status_texts, screen_names, hashtags= [],[], []
  for tweet in tweets:
     status_texts.append(tweet.text)
     screen_names += [userMention['screen_name'] for userMention in tweet.entities['user_mentions']]
     hashtags += [hashtag['text'] for hashtag in tweet.entities['hashtags']]
  words = [word for text in status_texts for word in word_tokenize(text)]
```

Social Web: Twitter

Utilizzando PrettyTable costruiamo poi quattro tabelle per visualizzare le quattro colonne metriche raccolte.

```
for label, data in [('Word', words),
                    ('Screen Names', screen_names),
                    ('Hashtag', hashtags)
                    ]:
    prettyTable = PrettyTable(field_names=[label, 'Count'])
    counter = Counter(data)
    [prettyTable.add_row(kv) for kv in counter.most_common()[:10]]
    prettyTable.align[label] = 'l'
    prettyTable.align['Count'] = 'r'
    print prettyTable
```

Social Web: Twitter

Word	Count
#	410
:	123
Trump	105
RT	78
@	69
.	64
https	63
you	54
believe	52
Donald	49

Hashtag	Count
Trump	66
Donald	46
donald	40
DonaldJTrump	26
USA	9
Kim	8
alt	6
Alte	6
Treffen	6
lobt	6

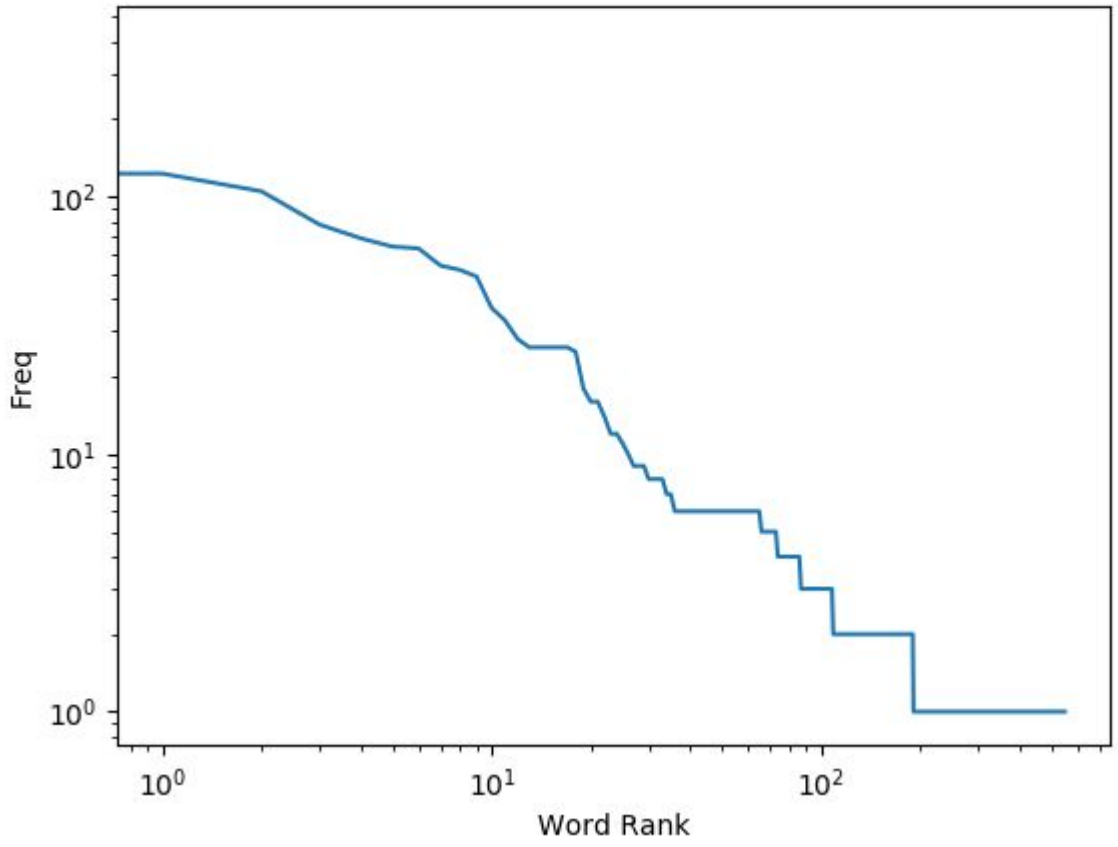
Social Web: Twitter

Possiamo visualizzare la frequenza delle parole.

```
def tweetsWordFrequency(query, count=10):
    tweets = tweepy.Cursor(twitterAPI.search, q=query).items(count)
    status_texts = []
    screen_names = []
    hashtags = []
    for tweet in tweets:
        status_texts.append(tweet.text)
        screen_names += [userMention['screen_name'] for userMention in tweet.entities['user_mentions']]
        hashtags += [hashtag['text'] for hashtag in tweet.entities['hashtags']]
    words = [word for text in status_texts for word in word_tokenize(text)]

    wordCounts = sorted(Counter(words).values(), reverse=True)
    plt.loglog(wordCounts)
    plt.ylabel("Freq")
    plt.xlabel("Word Rank")
    plt.show()
```

Social Web: Twitter



Social Web: Twitter

Possiamo visualizzare i retweet più popolari.

```
def retweetFrequency(query, count=10):
  tweets = tweepy.Cursor(twitterAPI.search, q=query).items(count)
  retweets = [
      (tweet.retweet_count,
       tweet.retweeted_status.user.screen_name,
       tweet.text
       )
      for tweet in tweets
      if hasattr(tweet, 'retweeted_status')
  ]
  prettyPrint = PrettyTable(field_names=['Count', 'Screen', 'Text'])
  [prettyPrint.add_row(row) for row in sorted(retweets, reverse=True)[:5]]
  prettyPrint.max_width['Text'] = 50
  prettyPrint.align = 'l'
  print prettyPrint
```

Social Web: Twitter

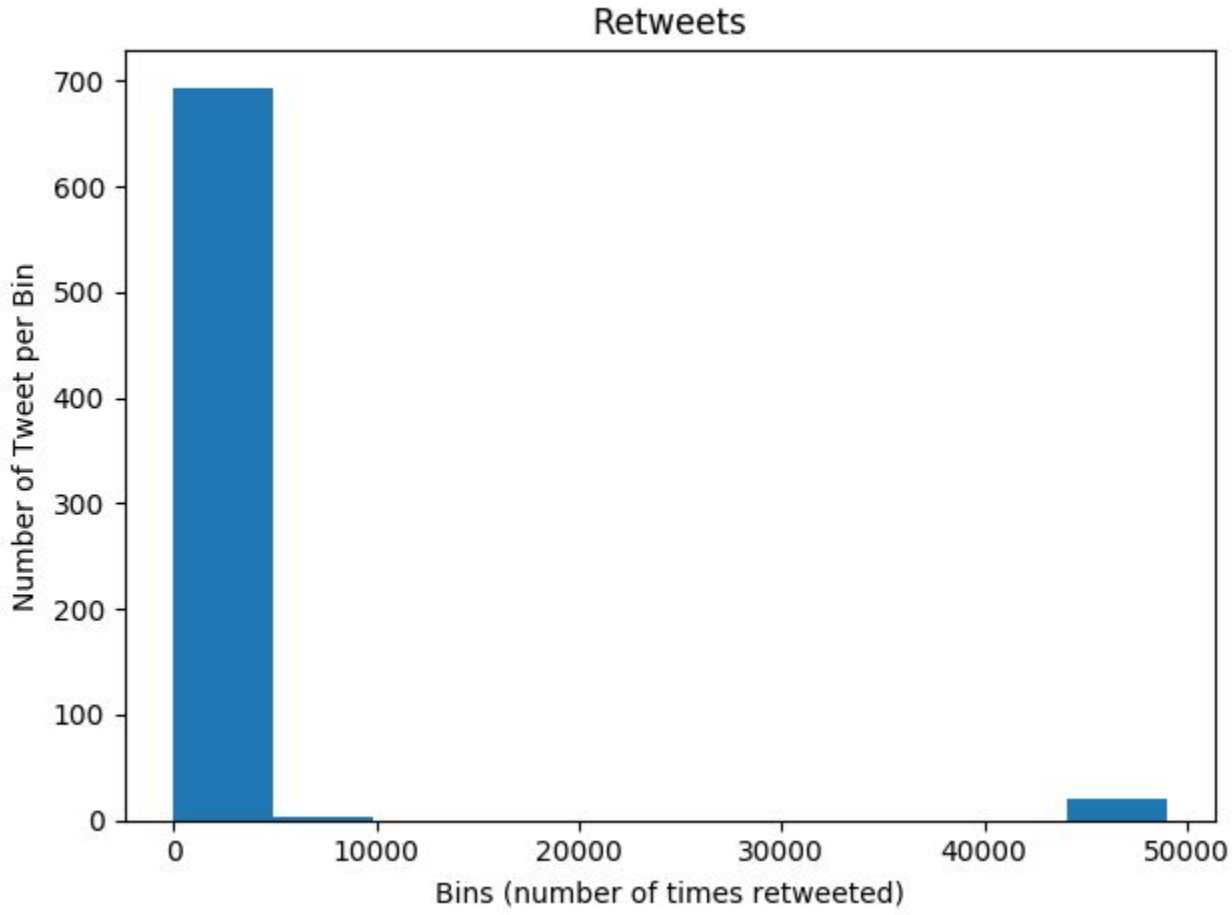
Count	Screen	Text
49062	acnewsitics	RT @acnewsitics: Donald Trump is: 1) "a billionaire" but you can't see his taxes 2) "a genius" but you can't see his grades 3) "exonerat…
4137	ProudResister	RT @ProudResister: We are all being gaslighted. Donald Trump and his campaign colluded with the Russian government. We watched it all unfol…
2805	DonaldJTrumpJr	RT @DonaldJTrumpJr: Donald Trump Jr.: It Became A Business Model For Media Organizations To Push Hoax Trump-Russia Narrative | Video | Real…
1661	EdKrassen	RT @EdKrassen: The Democratic party is more split right now than the Republican party, and that's a sad thing. Bernie Sanders is a good pe…
1482	kimguilfoyle	RT @kimguilfoyle: Donald Trump Jr.: It Became A Business Model For Media Organizations To Push Trump-Russia Narrative https://t.co/YNxv8cW…

Social Web: Twitter

Possiamo visualizzare l'istogramma dei retweet.

```
def retweetHistogram(query, count=10):
  tweets = tweepy.Cursor(twitterAPI.search, q=query).items(count)
  retweets = [
     (tweet.retweet_count,
      tweet.retweeted_status.user.screen_name,
      tweet.text
      )
     for tweet in tweets
     if hasattr(tweet, 'retweeted_status')
  ]
  counts = [count for coun, _, _ in retweets]
  plt.hist(counts)
  plt.title("Retweets")
  plt.xlabel("Bins (number of times retweeted)")
  plt.ylabel("Number of Tweet per Bin")
  plt.show()
```

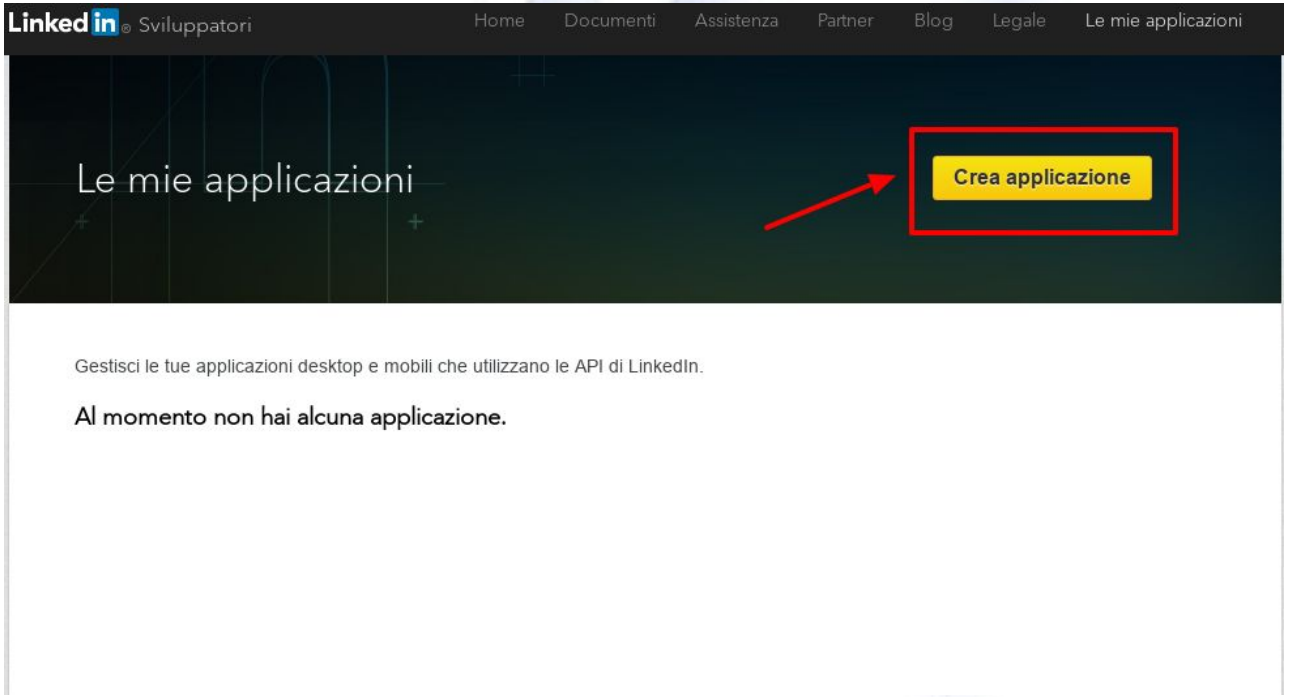
Social Web: Twitter



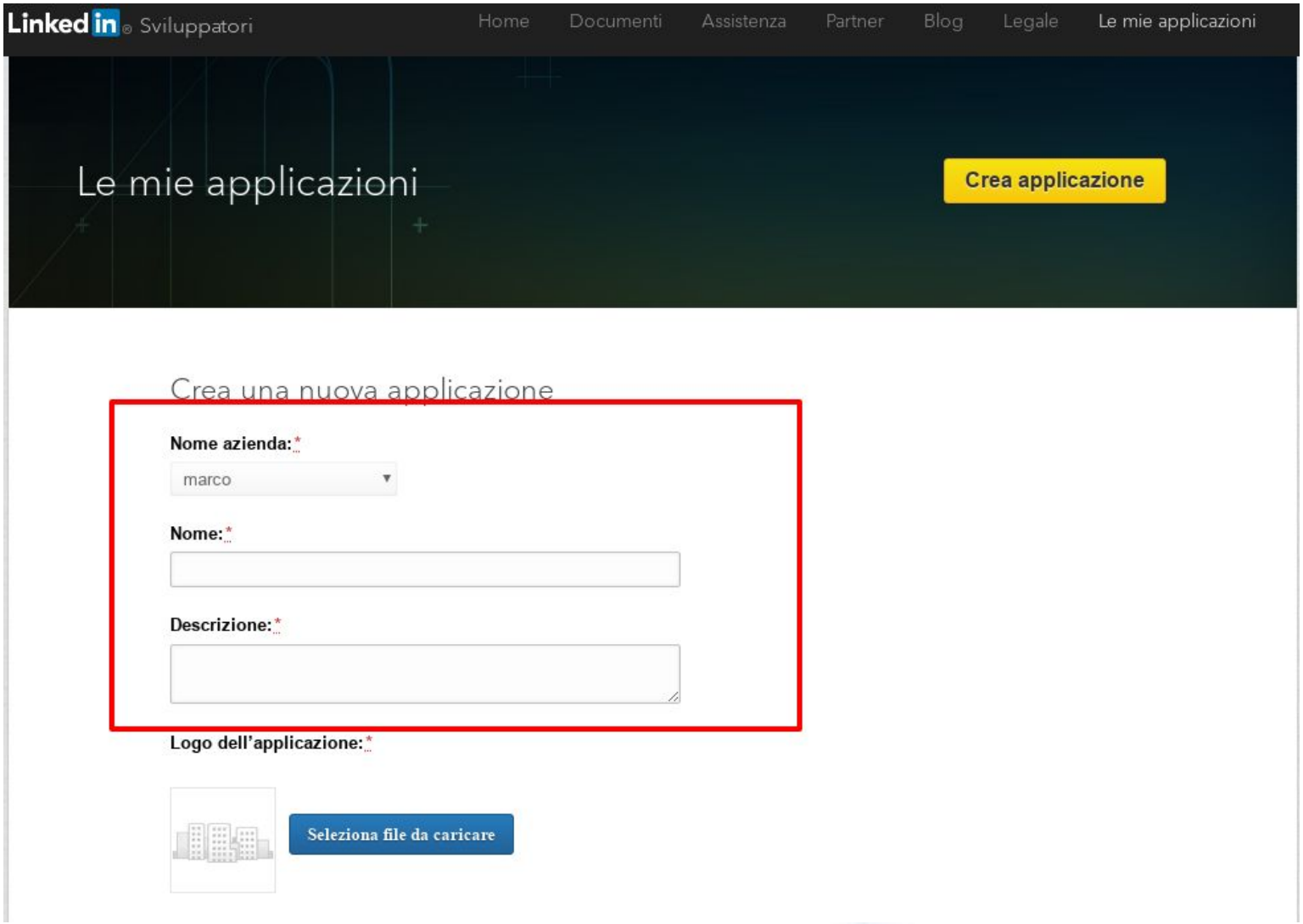
Social Web: LinkedIn

Il primo step per poter lavorare con LinkedIn è quello di ottenere delle credenziali per poter accedere alle API fornite.
Per ottenere delle credenziali è necessario registrarsi come sviluppatori e creare una "***app***" al seguente indirizzo:

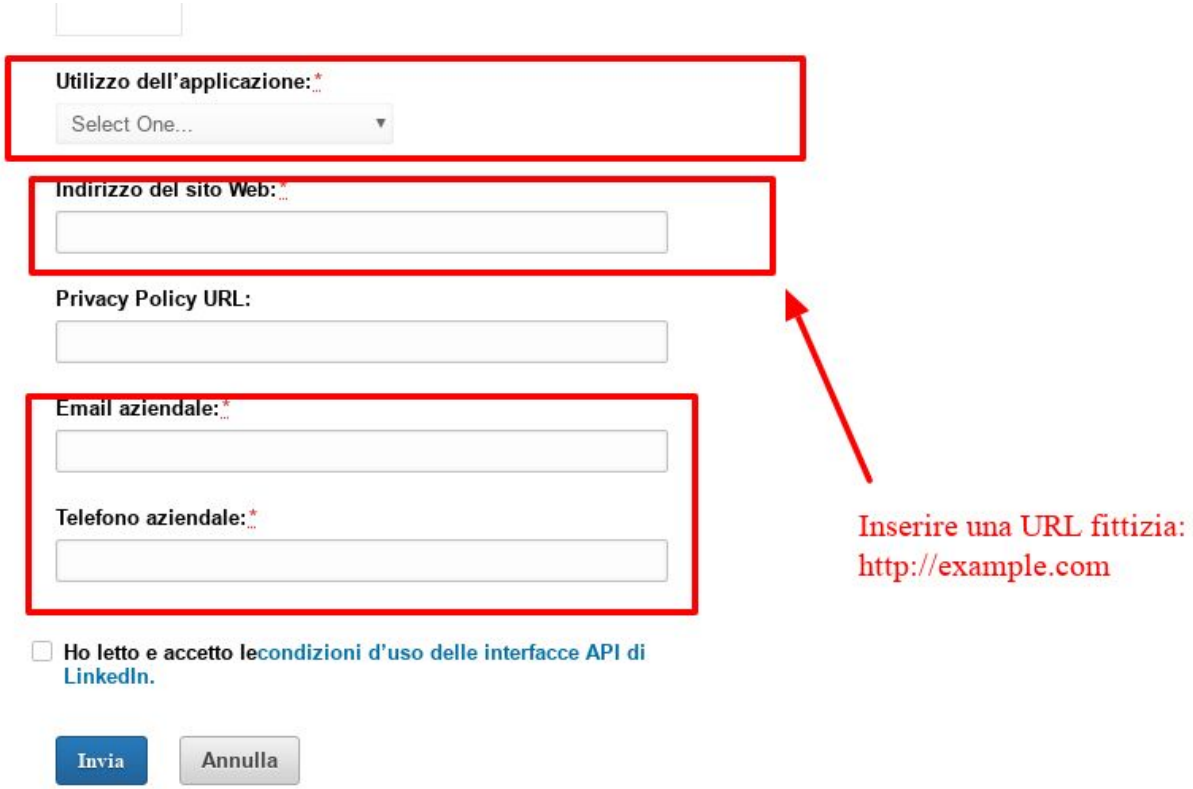
https://www.linkedin.com/developer/apps



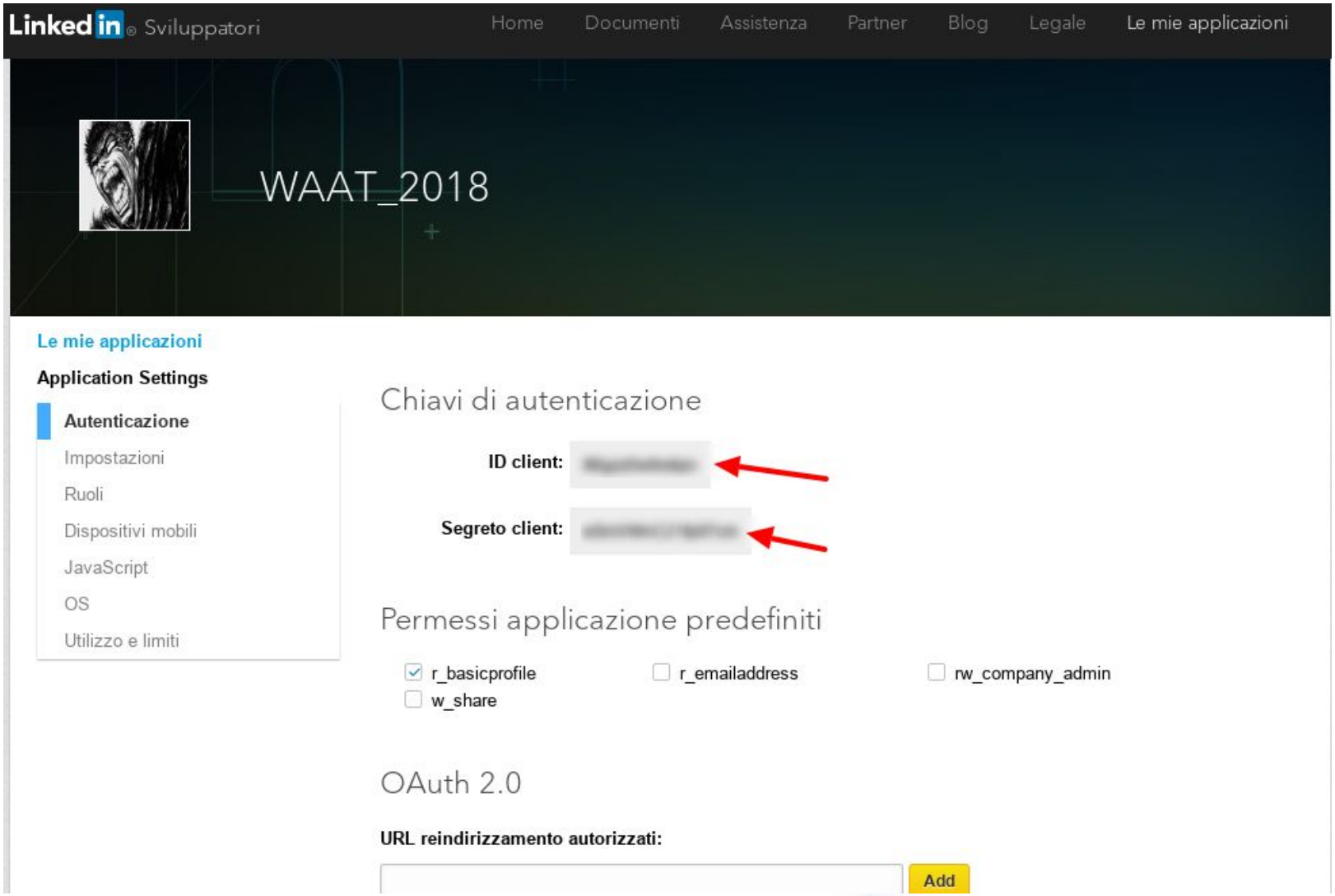
Social Web: LinkedIn



Social Web: LinkedIn



Social Web: LinkedIn



Social Web: LinkedIn

1. Dal repository ***https://github.com/marcoortu/WAAT-2019*** fare il checkout del branch ***social.***

2. Il file ***credential.py*** contiene le credenziali per i diversi social media che analizzeremo, sostituire le credenziali appena ottenute nelle seguenti variabili.

```
# Mettere qui i dati corretti per LinkedIn

LINKEDIN_CLIENT_KEY = 'XXXXXXXXXXXXXXXXXXXXXXXX'
LINKEDIN_CLIENT_SECRET = 'XXXXXXXXXXXXXXXXXXXXXXXX'
LINKEDIN_USERNAME = 'XXXXXXXXXXXXXXXXXXXXXXXX'
LINKEDIN_PASSWORD = 'XXXXXXXXXXXXXXXXXXXXXXXX'
```

Social Web: LinkedIn

Possiamo ora interagire con le API fornite da LinkedIn:

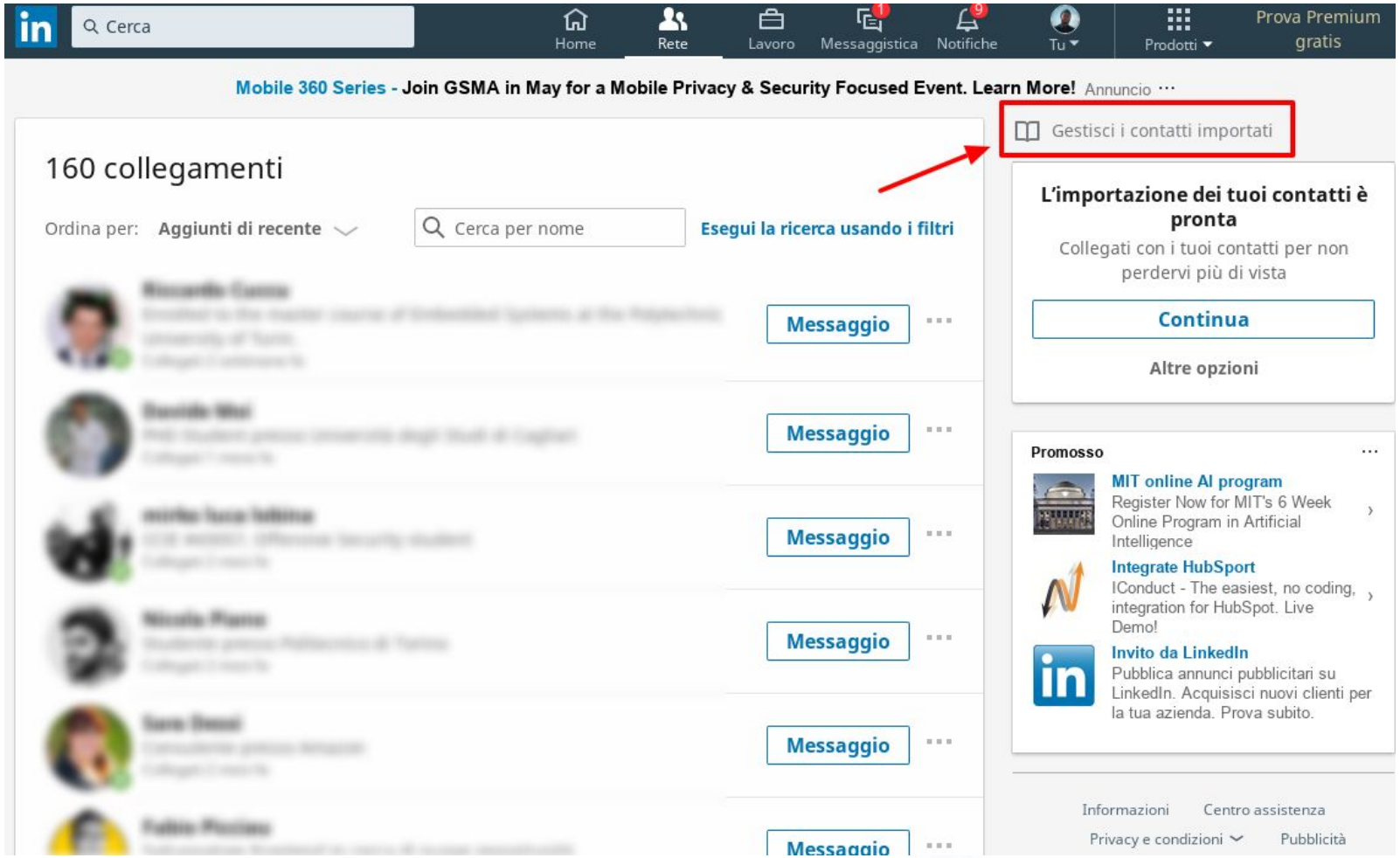
```
credential = getCredential()
auth = linkedin.LinkedInDeveloperAuthentication(LINKEDIN_CLIENT_KEY,
                                                LINKEDIN_CLIENT_SECRET,
                                                credential['oauth_token'],
                                                credential['oauth_token_secret'],
                                                '',

permissions=linkedin.PERMISSIONS.enums.values()
                                                )
app = linkedin.LinkedInApplication(auth)

if __name__ == '__main__':
  print app.get_profile()
```

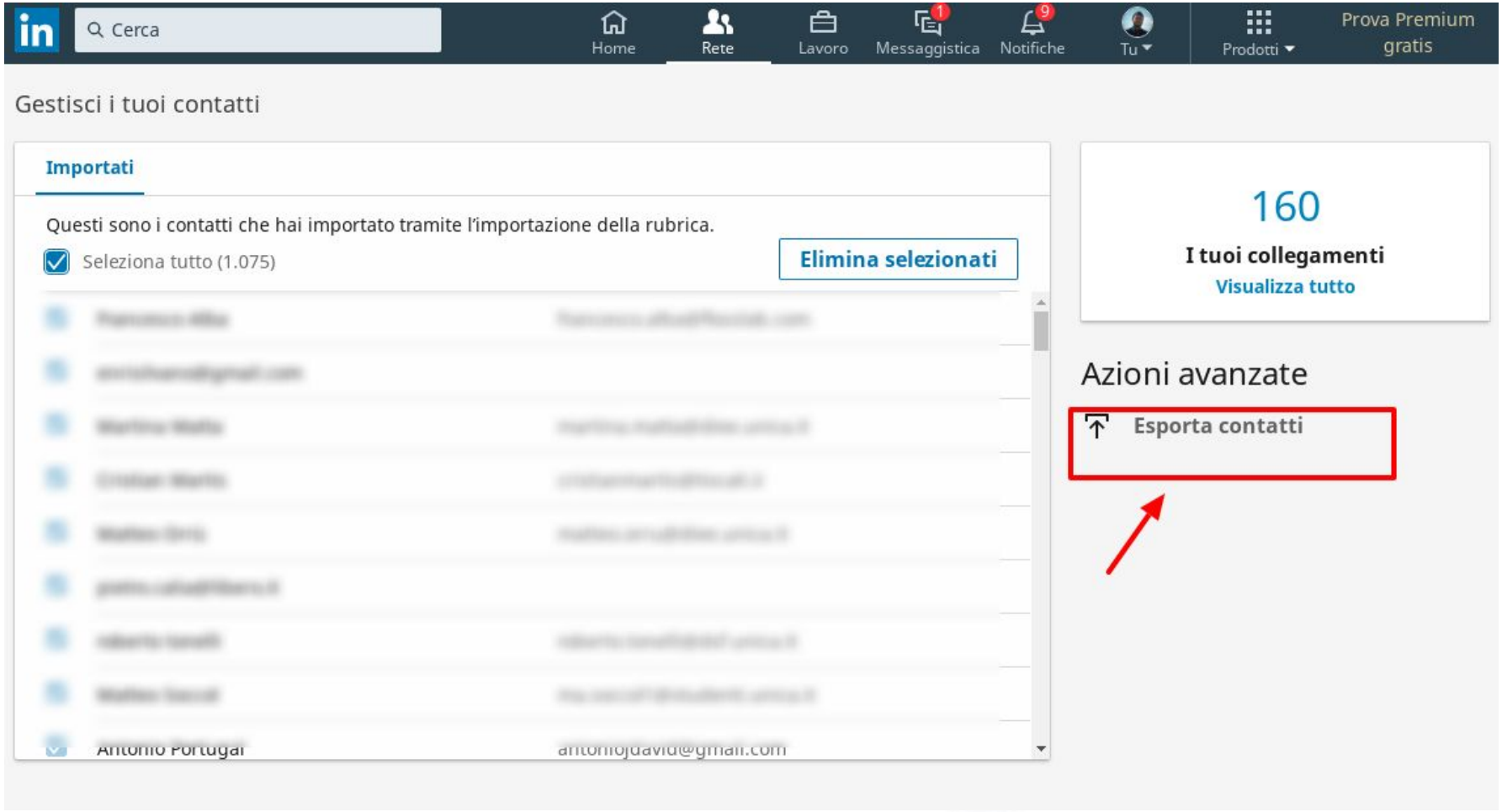
Social Web: LinkedIn

Esportare i propri contatti in csv:



Social Web: LinkedIn

Esportare i propri contatti in csv e salvare il file nella cartella data del progetto.



Social Web: K-Means

Per ottenere una stima del K migliore da utilizzare come numero di clusters del K-Means si può utilizzare il così detto metodo ***Elbow***.

Immaginiamo di ottenere una trasformazione TF-IDF data una lista di testi ***textList***.

```
tfidfVectorizer = TfidfVectorizer(max_features=100,
                                  stop_words='english',
                                  use_idf=True,
                                  tokenizer=word_tokenize,
                                  ngram_range=(1, 2))
tfidfMatrix = tfidfVectorizer.fit_transform(textList)
```

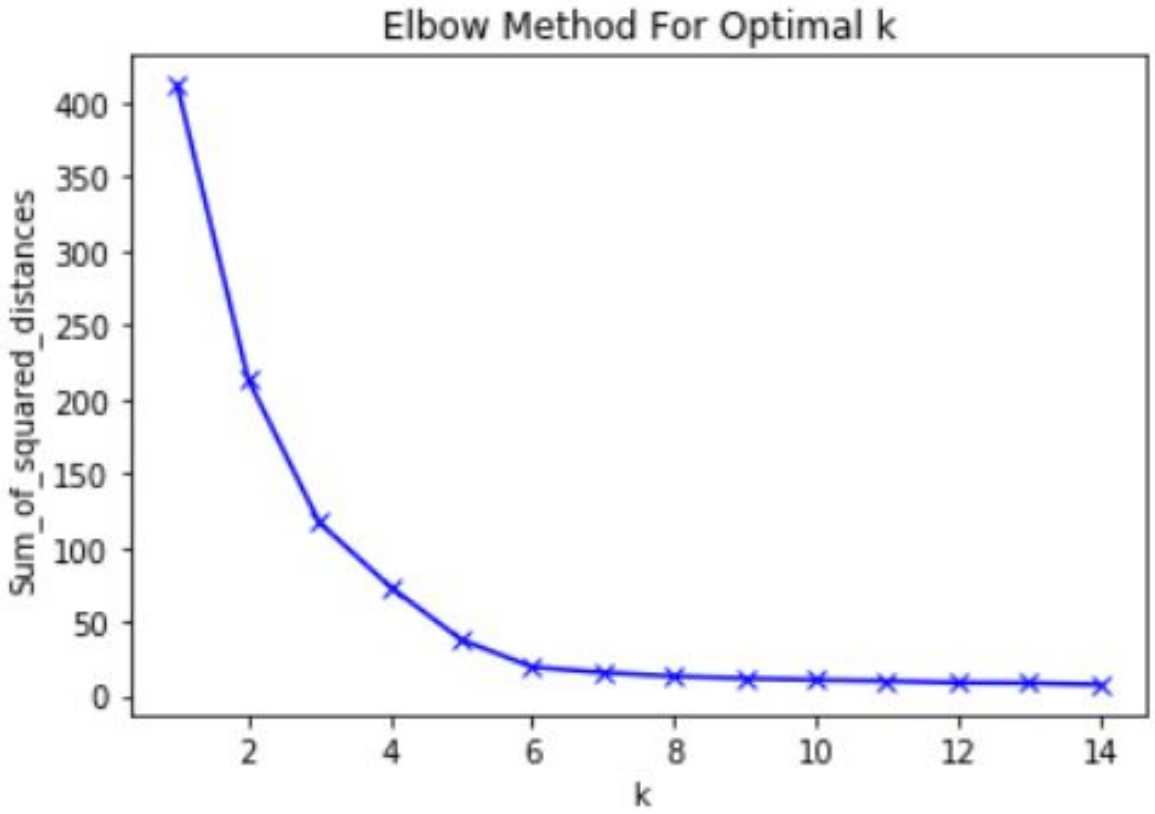
Social Web: K-Means

Definiamo una funzione che calcola la somma dei quadrati delle distanza più vicine ad ogni cluster ad ogni iterazione (parametro ***inertia_*** del kmeans).

```
def getBestKForKMeans(tfidfMatrix, maxClusters=15):
    sumOfSquaredDistances = []
    K = range(1, maxClusters)
    for k in K:
        km = KMeans(n_clusters=k)
        km = km.fit(tfidfMatrix)
        # inertia_: Sum of squared distances of samples to their closest cluster center.
        sumOfSquaredDistances.append(km.inertia_)
    print sumOfSquaredDistances
    plt.plot(K, sumOfSquaredDistances, 'bx-')
    plt.xlabel('k')
    plt.ylabel('Sum of Squared Distances')
    plt.title('Elbow Method For Optimal k')
    plt.show()
```

Social Web: K-Means

Si ottiene una curva al variare di K. Se la curva ha la forma di un gomito (Elbow appunto) allora il valore di K ottimale si trova in prossimità del gomito.



Social Web: LinkedIn

1. Dal repository ***https://github.com/marcoortu/WAAT-2019*** fare il checkout del branch ***social.***

2. Scaricare l'archivio dei vostri dati su linkedin per svolgere l'esercizio 2.

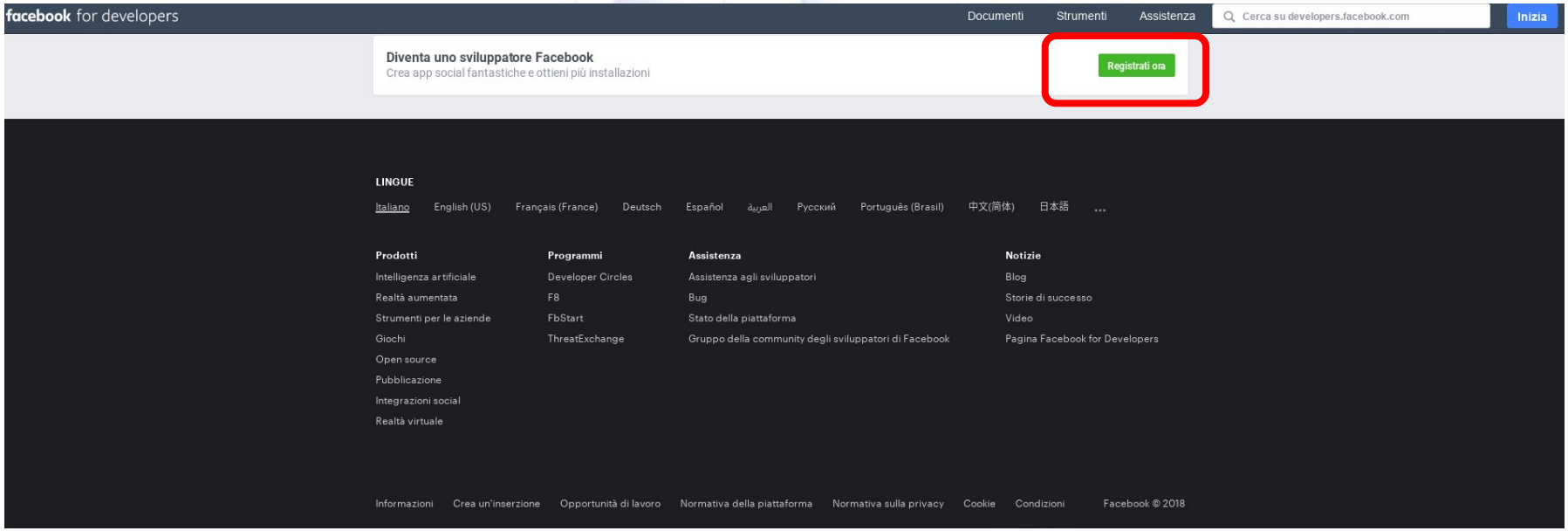
Social Web: Facebook Lockdown

https://newsroom.fb.com/news/2018/04/restricting-data-access/

Social Web: Facebook

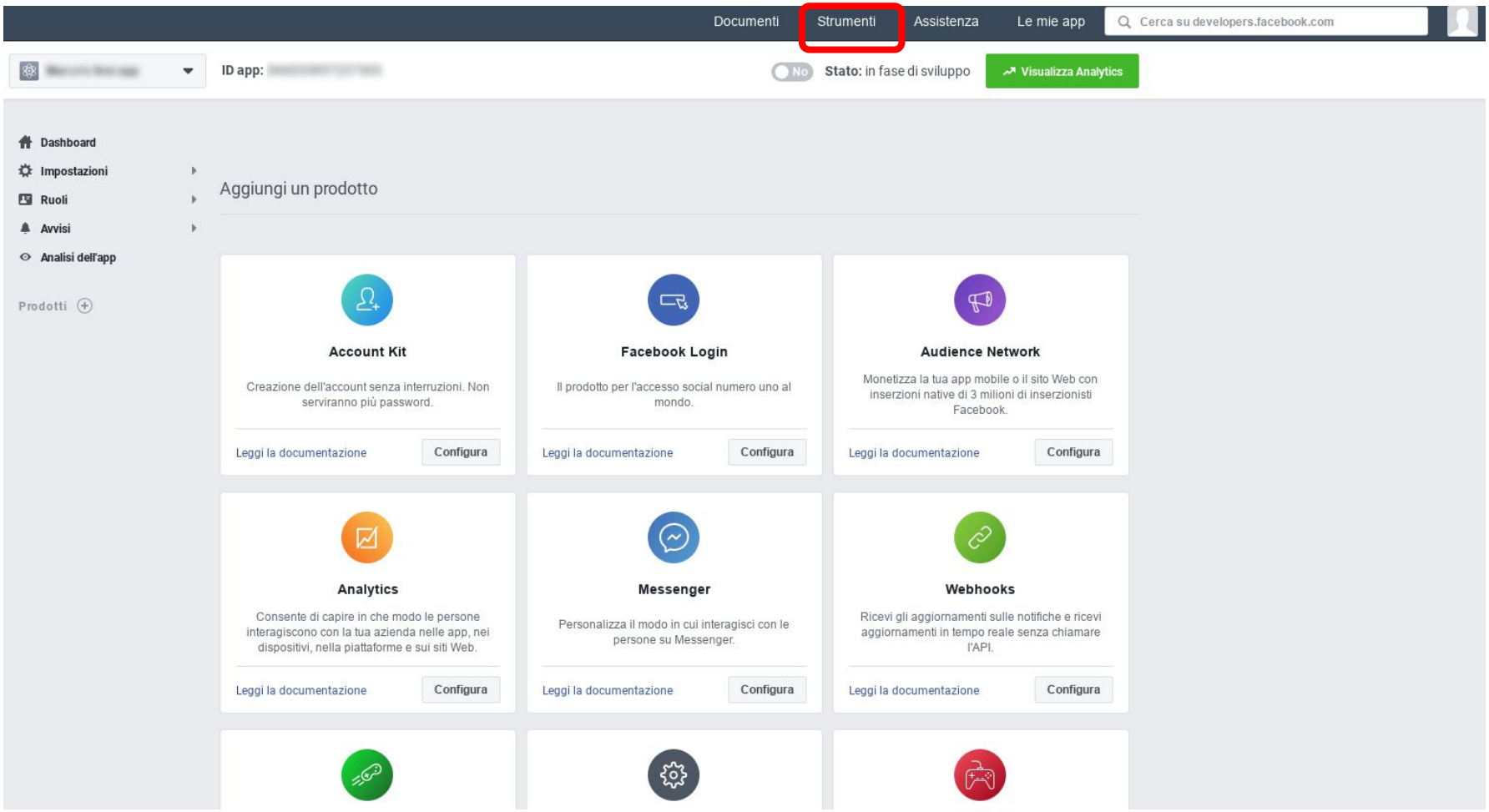
Il primo step per poter lavorare con Facebook è quello di ottenere delle credenziali per poter accedere alle API fornite.
Per ottenere delle credenziali è necessario registrarsi come sviluppatori e creare una "***app***" al seguente indirizzo:

https://developers.facebook.com/apps



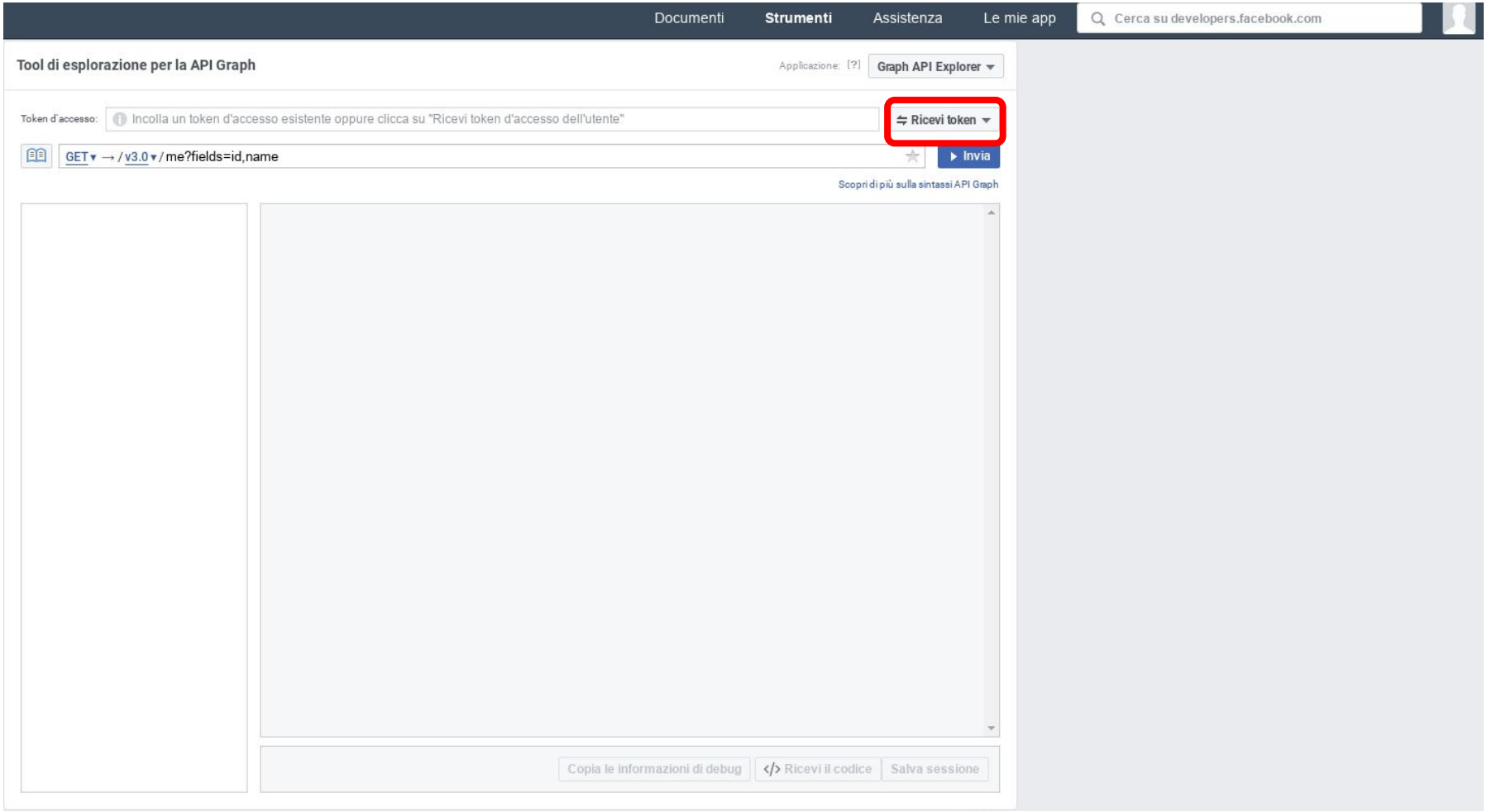
Social Web: Facebook

Dopo aver completato il wizard di installazione di arriva alla Dashboard delle applicazioni. Andare su Strumenti -> Tool di esplorazione per la API Graph



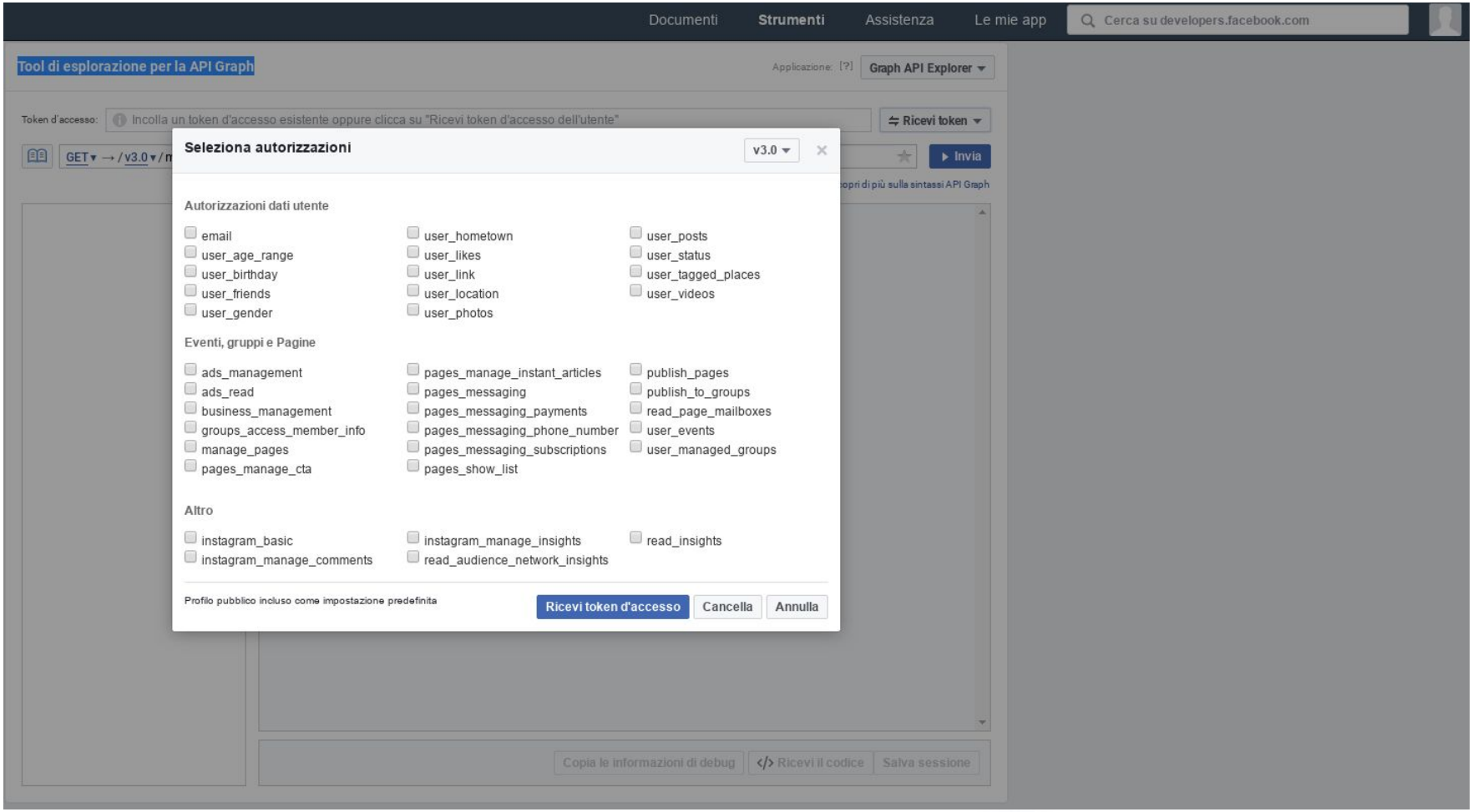
Social Web: Facebook

Generare un token di accesso cliccando sul pulsante “Ricevi Token”.



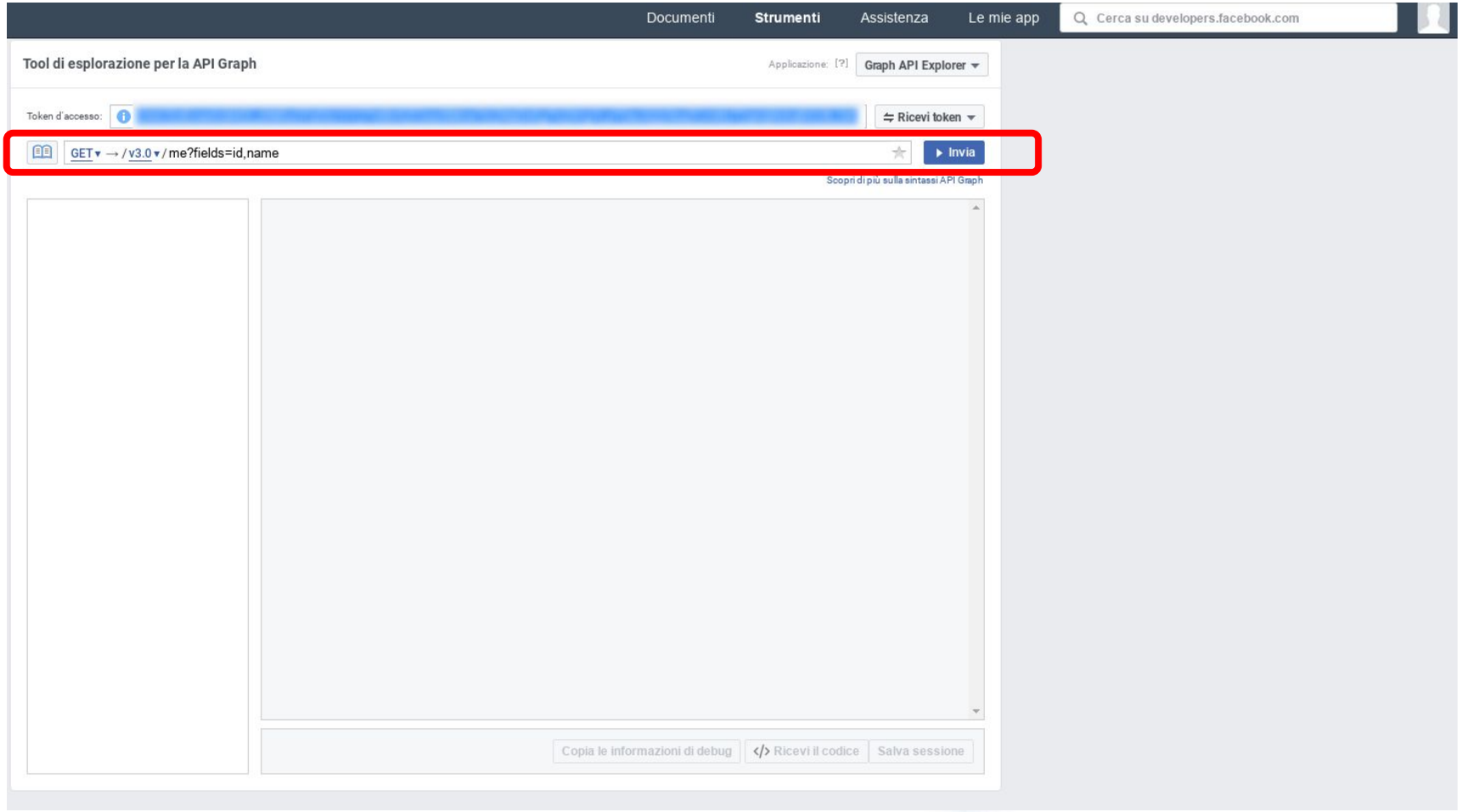
Social Web: Facebook

Selezionare i permessi di accesso della vostra applicazione.
https://developers.facebook.com/docs/graph-api/using-graph-api/



Social Web: Facebook

Copiare l’access token ricevuto.



Social Web: Facebook

1. Dal repository ***https://github.com/marcoortu/WAAT-2019*** fare il checkout del branch ***social.***

2. Il file ***credential.py*** contiene le credenziali per i diversi social media che analizzeremo, sostituire le credenziali appena ottenute nelle seguenti variabili.

```
# Mettere qui i dati corretti per Facebook

FACEBOOK_ACCESS_TOKEN = 'XXXXXXXXXXXXXXXXXXXXXXXX'
```

Social Web: Facebook

Con l'access token ottenuto precedentemente possiamo creare il nostro client per utilizzare le Graph API di Facebook.

```
import facebook

from credentials import FACEBOOK_ACCESS_TOKEN

graph = facebook.GraphAPI(access_token=FACEBOOK_ACCESS_TOKEN)

if __name__ == '__main__':
   print graph
```

Social Web: Facebook

Il metodo search permette di ricercare vari tipi di oggetti su facebook.

```
# Search for places near 1 Hacker Way in Menlo Park, California.
places = graph.search(type='place',
                      center='37.4845306,-122.1498183',
                      fields='name,location')

# Each given id maps to an object the contains the requested fields.
for place in places['data']:
  print('%s %s' % (place['name'].encode('utf8'), place['location'].get('zip')))
```

Social Web: Facebook

Creaiamo una funzione di supporto che permette di serializzare gli oggetti in python e visualizzarli in formato JSON (Javascript Object Notation).

```
def serialize(object):
   return json.dumps(object, indent=1)
```

Possiamo utilizzare questa funzione per stampare il contenuto dei vari oggetti.

```
print(serialize(graph.get_object('me')))
```

```
{
 "name": "Marco Ortu",
 "id": "XXXXXXXXXXXXX"
}
```

Social Web: Facebook

```
print(serialize(graph.get_connections('me', "friends")))
{
 "data": [],
 "summary": {
  "total_count": 0
 }
}
```

Estraiamo i post dalla pagina personale di Bill Gates:

```
user = 'BillGates'
profile = graph.get_object(user)
posts = graph.get_connections(profile['id'], 'posts')
print serialize(posts)
```

Social Web: Facebook


```
{
 "paging": {
  "next":
"https://graph.facebook.com/v2.6/216311481960/posts?access_token=EAACEdEose0cBAKuLoNGTmFaF9aVAY7ZAEGS
MZCfrfLfCQ4NVMuyskbGUnPHZBVr8goTCZCkPRxe6AISybaoD4ZBFiCwaHaMfqYXhBC05PEV8rG0xdQ8tVa3OgGZAZB
GB9AahaAxEt747fa3KRogH2fpSAmsIOdZC5SxSswPolERHPhb7hYuj2GKFbPZAMmxfs4QccPtfkKAs5RZAMPyclVzTs2&li
mit=25&until=1523383499&__paging_token=enc_AdBJi9PnCZA2BSN8ZCwtjocpY7kpCIRkEN4xZCnOF9WjlLV3pp5giObH
1mP7WtRFZAbOtKAfLdpE0mhr3nhNUquPY3Ygpny1ilCAyu609Om4ySrrvQZDZD",
  "previous":
"https://graph.facebook.com/v2.6/216311481960/posts?since=1526385603&access_token=EAACEdEose0cBAKuLoNGTm
FaF9aVAY7ZAEGSMZCfrfLfCQ4NVMuyskbGUnPHZBVr8goTCZCkPRxe6AISybaoD4ZBFiCwaHaMfqYXhBC05PEV8rG0x
dQ8tVa3OgGZAZBGB9AahaAxEt747fa3KRogH2fpSAmsIOdZC5SxSswPolERHPhb7hYuj2GKFbPZAMmxfs4QccPtfkKAs5
RZAMPyclVzTs2&limit=25&__paging_token=enc_AdBlJoYhZCoOoDRhlQSxnqrAkFLHzZCdjclf20tgcumhWJZBuvlMowobR
7e1IRuL0ZB8Cyo5aaQ9hY3SIncN5gCIgriEJkHw7vMEmUMXKQD0E6UQEgZDZD&__previous=1"
 },
 "data": [
  {
   "created_time": "2018-05-15T12:00:03+0000",
   "message": "It\u2019s always thrilling to be back on campus \u2014 and it makes me wish I could be a student all over
again. Here are some of the things I learned from my visit to University of Central Florida: https://b-gat.es/2L09Crm",
   "id": "216311481960_10155451558261961"
  }
]
}
```

Social Web: Facebook

Utilizziamo ora la paginazione per ottenere tutti i post di una determinata pagina.

```
user = 'DonaldTrump'
profile = graph.get_object(user)
posts = graph.get_connections(profile['id'], 'posts')
while True:
  try:
    for post in posts['data']:
      print serialize(post)
    posts = requests.get(posts['paging']['next']).json()
  except KeyError:
    break
```

Social Web: Facebook


```
{
 "created_time": "2018-05-15T17:29:07+0000",
 "message": "Trump Just Had Every Single One Of Them Arrested! The ENTIRE Democratic Party Is FURIOUS!",
 "id": "780469468772262_1142542515898287"
}
{
 "created_time": "2018-05-15T17:01:04+0000",
 "message": "BAM! Trump isn't playing around!",
 "story": "President Donald J Trump shared a post.",
 "id": "780469468772262_1142524029233469"
}
{
 "created_time": "2018-05-15T16:10:34+0000",
 "message": "SICK! Scumbag Author Stephen King Thinks Melania Lied, Knows Why She Was Really Hospitalized",
 "id": "780469468772262_1142495322569673"
}
```

Social Web: Facebook

Esercizio 2

Eseguire una analisi comparativa del sentiment associato a due pagine Facebook comparabili (Pepsi e Coca Cola per esempio).

1. Estrarre 100 post dalle pagine Facebook.
2. Eseguire la sentiment analysis utilizzando la libreria ***TextBlob***.
3. Riportare i risultati con ***prettyTable*** stampando:
    a. sentiment medio
    b. percentuale negative/neutral/positive
4. Visualizzare il boxplot dei sentiment delle sue pagine.
5. Visualizzare la variazione temporale del sentiment delle due pagine.

Social Web: Facebook

Esercizio 3

Eseguire una analisi comparativa del sentiment associato a due pagine Facebook comparabili.

1. Estrarre tutti i post dalle pagine Facebook di entrambe le pagine a partire da uno stesso intervallo di tempo (esempio un mese).
2. Eseguire la sentiment analysis utilizzando la libreria ***TextBlog***.
3. Visualizzare il grafico della variazione temporale del sentiment delle due pagine.
4. Visualizzare il grafico del sentiment medio giornaliero.

Referenze

- Mining the Social Web: Data Mining Facebook, Twitter, LinkedIn, Google+, GitHub, and More (ISBN-13: 978-1449367619)
- Best K for Kmeans: https://blog.cambridgespark.com/how-to-determine-the-optimal-number-of-clusters-for-k-means-clustering-14f27070048f
- http://facebook-sdk.readthedocs.io/en/latest/
  - capitoli dal 1 al 3