

In questo notebook è presente un breve tutorial sull'uso delle API di twitter.

- **Nome:** Samuele Maria
- **Cognome:** Gallina
- **Matricola:** 100001478
- **Corso:** Social Media Management
- **Anno accademico:** 2021/2022

Per approfondimenti visitare le docs ufficiali <https://developer.twitter.com/en/docs>
(<https://developer.twitter.com/en/docs>)

Il primo step è sicuramente ottenere l'accesso alle API

[illegible]

Accade adesso

Iscriviti adesso a Twitter.

 Registrati con Google

 Iscriviti con Apple

Iscriviti tramite numero o email

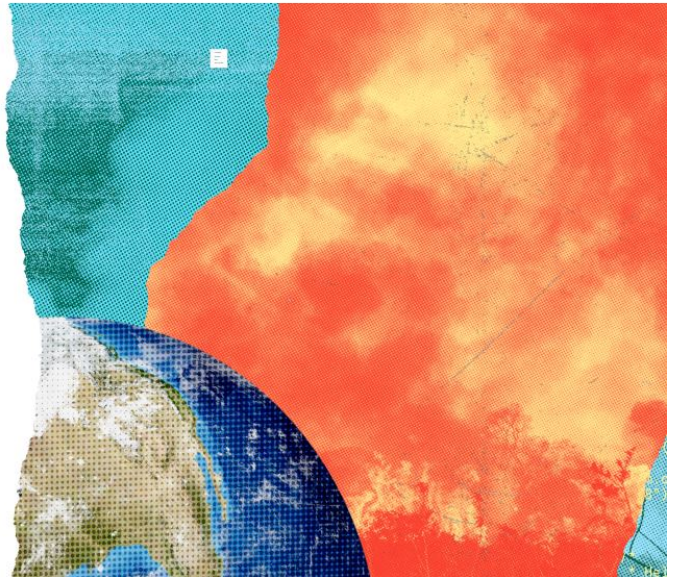
Iscrivendoti accetti i [Termini di servizio](#) e l'[Informativa sulla privacy](#), incluso l'[Utilizzo dei cookie](#).

Hai già un account? [Accedi](#)

Ottenuto l'account procediamo cercando sul nostro motore di ricerca "twitter developer" uno dei primi risultati(probabilmente il primo) sarà il sito a cui dovremo accedere. Dovrebbe comparire qualcosa del tipo:

Tap into what's happening to build what's next

Analyze how #ExtremeWeather events unfold across the globe and within the public conversation on Twitter.



Clicchiamo "Apply" in alto a destra: ci porterà in una sezione per "convertire" il nostro profilo in un account developer(quindi con accesso alle API) Probabilmente saranno richieste le credenziali. Dopo questa procedura arriviamo a dei passaggi guidati per l'ottenimento del profilo developer

Lo step2 di questa sezione sarà molto laborioso in quanto si dovrà spiegare per bene i motivi per cui vogliamo ottenere le API. Molti social media fanno attenzione a queste cose quindi meglio essere più espressivi possibile e dare quanti più dettagli possiamo.

Developer Portal

#ImportantStuff

This part of the application helps us make sure that our data will be handled in accordance with Twitter's Developer Policies.

Our ultimate goal is to keep Twitter a safe and healthy space for public conversation.

What's not allowed

Some activities (like surveillance) are never allowed on Twitter. Take a look at our restricted uses page. Double-check that your intended use is policy compliant.

Automation guidelines

If you plan on enabling any sort of automated activity on the platform (like a bot), be sure to review our automation rules.

How will you use the Twitter API or Twitter Data?

In your words

In English, please describe how you plan to use Twitter data and/or APIs. The more detailed the response, the easier it is to review and approve.

Please be thoughtful and thorough

Back Next

PRIVACY COOKIES TWITTER TERMS & CONDITIONS DEVELOPER POLICY & TERMS © 2021 TWITTER INC. FOLLOW @TWITTERDEV SUBSCRIBE TO DEVELOPER NEWS

La prima delle varie domande è la seguente(poi ne seguono altre in cui ognuno risponde come è opportuno per lui)

Lo step3 sarà una semplice review per rivedere meglio le informazioni che abbiamo inserito(per modificarle eventualmente).

Developer Portal

#Review

Before moving forward, take a moment to look over your responses. If everything looks good, you can move on to the next step.


One thing to remember, the email address you provided will be used to contact you about your account.

Basic info ▾ Edit

Intended use ▾ Edit

Back Next

Nello step4 come ogni social media che si rispetti ci sarà un documento con termini legali da acconsentire(sarebbe opportuno leggerlo)

 **Developer Portal**

#Read&Accept

We've carefully crafted our developer terms to make it readable and accessible. Our aim is to have a healthy and open platform for all.

Once you've read it and agreed, check the box below the agreement and then hit the submit application button on the bottom right.

1 Basic info

2 Intended use

3 Review

4 Terms

Developer agreement & policy

Developer Agreement

Effective: March 10, 2020


This Twitter Developer Agreement ("**Agreement**") is made between you (either an individual or an entity, referred to herein as "**you**") and Twitter (as defined below) and governs your access to and use of the

☐ By clicking on the box, you indicate that you have read and agree to this [Developer Agreement](#) and the [Twitter Developer Policy](#), additionally as


Back


Submit

Comparirà adesso una schermata del genere:

 **Developer**

Docs Community Updates Support





Time to verify your email

We just sent an email verification to [redacted]
Once verified, your application review will begin.

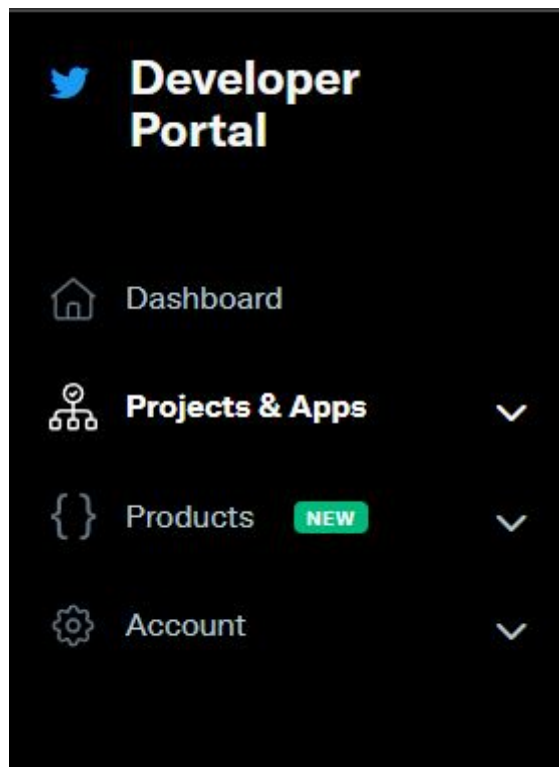
Don't see it? Try checking your spam inbox or [have it resent](#).

Verifichiamo la nostra email e avremo finito.

Arrivati a questo punto il nostro account developer dovrebbe essere pronto. Adesso cercando sempre "twitter developer" sul nostro motore di ricerca possiamo riandare nella pagina da dove siamo andati nella creazione del profilo con "apply" e ad accesso compiuto ora in alto a destra spunterà un bottone chiamato "developer portal". Cliccandolo giungeremo alla Dashboard.

Sulla sinistra comparirà una sezione simile a questa.

Cliccando su Project giungeremo a una sezione che ci farà creare un nuovo progetto (possiamo vedere il progetto come un "apps" manager)



Saranno chieste di nuovo varie informazioni

Name Your Project


1 Project name 2 Use case 3 Project description 4 App set up


Your Project helps you organize your work and monitor your usage with the Twitter API.


32


Back


Next

 **Developer Portal**


 Dashboard

 **Projects & Apps** ▼

 Products **NEW** ▼

 Account ▼

Docs ▼ Community ▼ Updates ▼ Support



Which best describes you?

1 Project name

2 Use case


3 Project description


4 App set up


This is how you intend to use the Twitter developer platform.


Student ▼


Back Next

 **Developer Portal**


 Dashboard

 **Projects & Apps** ▼

 Products **NEW** ▼

 Account ▼

Docs ▼ Community ▼ Updates ▼ Support



Describe your new Project

1 Project name

2 Use case

3 Project description

4 App set up

This info is just for us, here at Twitter. It'll help us create better developer experiences down the road.

Project description

⚠ Required

Back Next

A questo punto si dovrà creare l'app: sarà richiesto inizialmente solo il nome e dopo di che saranno generati 2 keys e 1 token(il bearer token). Facciamo attenzione a salvarli perchè poi queste chiavi non verranno più mostrate in chiaro per motivi di sicurezza. Andiamo a gestire le impostazioni della nostra app per poterci lavorare. Possiamo gestire i dettagli come nome, descrizione e foto dell'app.



App Details

[Edit](#)

NAME



APP ICON



APP ID



DESCRIPTION

This information will be visible to people who've authorized your App

This app was created to use the Twitter API.

Sarà importante gestire i permessi della nostra applicazione e le impostazioni di autenticazione. Come permessi possiamo mettere

1. Read
2. Read and write
3. Read and write and Direct message

In base a cosa dovrà fare la nostra app scegliamo 1 delle 3. Per ora lasciamo stare "Read"



App permissions

[Edit](#)**Read Only**

Read Tweets and profile information



Authentication settings

[Edit](#)**3-legged OAuth is disabled**

Use 3-legged OAuth for Sign in with Twitter, posting Tweets on behalf of other accounts and more. Get more information in the [docs](#).

E' invece importantissimo gestire le impostazioni di autenticazione. Attiviamo il 3-legged OAuth per poter accedere ai servizi delle API. Saranno richieste ora 2 informazioni:

1. Callback URLs: ovvero l'url da usare per il protocollo(potremmo inserire <https://localhost/8080> (<https://localhost/8080>) per lavorare in locale attualmente)
2. Nostro sito web(possiamo inserire anche un banale link a un nostro profilo social)



Edit authentication settings

Enable 3-legged OAuth



Use 3-legged OAuth for Sign in with Twitter, posting Tweets on behalf of other accounts and more. Get more information in the [docs](#).

Request email address from users



These additional permissions require that you provide URLs to your application or service's privacy policy and terms of service.

Adesso dovremo andare sempre dalla nostra app nella sezione "keys and tokens" Procediamo con le generazione dell'access token and secret(consumer keys e bearer token vengono generati automaticamente alla creazione dell'app) Facciamo ancora attenzione perchè anche questi token non verranno più mostrati in chiaro per motivi di sicurezza.

Consumer Keys ⓘ

API Key and Secret

Regenerate

Authentication Tokens ⓘ

Bearer Token

Generated November 13, 2021

Regenerate

Revoke

Access Token and Secret

For [REDACTED]

Generate

Arrivati a questo punto dovremmo essere in grado di utilizzare le API di twitter nei nostri codici.

Lavorando con OAuth 2 ovvero il servizio che usa Twitter per l'autenticazione/generazione di bearer token usiamo un codice simile a questo per effettuare appunto una richiesta per il nostro token. Definiamo una funzione per ottenere il bearer token.

In [1]:

```
import requests
def OttieniBearerToken(apiKey, apiKeySecret):
    dati = {'grant_type': 'client_credentials'}
    endpoint = 'https://api.twitter.com/oauth2/token'
    risposta = requests.post(endpoint, data=dati, auth=(apiKey, apiKeySecret))
    return risposta.json()['access_token']
```

Definiamo ora la ottieniJSON che prende appunto un url per la richiesta e i parametri. L'autenticazione la forniamo grazie alla funzione richiestaAuth. Andiamo a controllare il codice di stato(se è andato tutto bene dovrebbe essere 200) e stampiamo il nostro json ricevuto come risposta.

In [2]:

```
import json

bearer_token = OttieniBearerToken('R2axAPh7Cctq2szNIcf3Mt9nH', 'VN0hDlPGhwtY1a20FHnJ9SwQm083')

def richiestaAuth(r):
    #headers http della nostra richiesta
    #inseriamo l'autorizzazione e il User-Agent che "dirà" al server cosa vogliamo fare
    #in questo caso chiediamo di fare una ricerca sui recenti con Python
    r.headers["Authorization"] = f"Bearer {bearer_token}"
    r.headers["User-Agent"] = "v2RecentSearchPython"
    return r

def ottieniJSON(url, params):
    #facciamo una richiesta inserendo l'url tipico per le azioni che vogliamo fare (esempio
    #dobbiamo inserire gli headers http(lo facciamo con la funzione richiestaAuth)
    #inseriamo parametri per la nostra ricerca(ad esempio degli hashtags)
    risposta = requests.get(url, auth=richiestaAuth, params=params)
    if risposta.status_code != 200:
        raise Exception(risposta.status_code, risposta.text)
    return risposta.json()
```

In [3]:

```
search_url = "https://api.twitter.com/2/tweets/search/recent"
query_params = {'query': '#football'} #con questi parametri(solo la query) troviamo post r
json_response = ottieniJSON(search_url, query_params)
print(json.dumps(json_response))
```

```
[{"id": "149484800878809089", "text": "\u9577\u8c37\u90e8\u8aa0\u0304c\u06765\u05b63\u09650\u0308a\u03067\u0306e\u073fe\u05f79\u05f15\u09000\u03092\u0201c\u05426\u05b9a\u0201d\u03000\u2015\u030d5\u030e9\u030f3\u030af\u030d5\u030eb\u030c8\u03068\u0300c\u0307e\u0305f\u06765\u05e74\u0306b\u08a71\u03057\u05408\u03046\u0300d\nhttp://t.co/hxuU83DsLc\n#urawareds #daihyo #jleague #football #soccer"}, {"id": "1494847979039215617", "text": "RT @STEMsportsUSA: The longest field goal made in pro football history was 64 yards from @Broncos Matt Prater! The longest attempted field\u0026"}, {"id": "1494847970675933184", "text": "Soccer Super League https://t.co/y7cBQStK0e (https://t.co/y7cBQStK0e) #soccer #league #Football #07"}, {"id": "1494847937607872513", "text": "RT @topfanscorner: \u00d83d\u00dc49 $88.95 \u00d83d\u00dc48\nPICNIC TIME Black New Orleans Saints Cart Cooler @topfanscorner\n#PICNIC #TIME #Black #NewOrleansSaints #Cart\u0026"}, {"id": "1494847812126707715", "text": "J1\u00795e\u06238\u030fb\u09152\u04e95\u09ad8\u05fb3\u2015\u069d9\u091ce\u0667a\u07ae0\u0300c\u030b5\u030c3\u030ab\u030fc\u06587\u05316\u03092\u0795e\u06238\u03067\u08d77\u03053\u03057\u0305f\u03044\u0300d \u0958b\u05e55\u076f4\u0524d\u030a4\u030f3\u030bf\u030d3\u030e5\u030fc\nhttps://t.co/wmLtJLUT58\n#urawareds #daihyo #jleague #football #soccer"}, {"id": "1494847802198749186", "text": "RT @NeftyDraft: \u00d83c\u00df89This $superR BOWL 2022 SEASON\nFans get to win 3 NFTs\u00d83c\u00df89 \u00d83c\u00dfc8\u00d83c\u00dfc8\u00d83c\u00dfc8\n1. Follow @NeftyDraft\n2. Like/RT\n3. Tag 2 friends\nWinners will\u0026"}, {"id": "1494847714043047938", "text": "#Detroit #Lions: All month, we are spotlighting young leaders already making their impact on Detr...\n\nhttp://t.co/Uu20fWsF8I\n#DetroitLions #Football #LionsBHM #Michigan #NationalFootballConference #NationalFootballConferenceNorthDivision #NationalFootballLeague #NFL https://t.co/9quwwRC8jc"}, {"id": "1494847651313074181", "text": "Liverpool-Norwich City\nPredicted Lineups, Head to Head, its effect on the table and more. Click the link below.\nhttp://t.co/CXYFVNj3PN\n#LIVNOR #LiverpoolFC #NorwichCity #PremierLeague #football"}, {"id": "1494847539048337415", "text": "It is easy for me to keep going when things get hard. #basketball #nba #sports #ballislife #k #football #basket #lebronjames #lakers #bball #sport #nike #basketballneverstops"}, {"id": "1494847509222486017", "text": "FC\u06771\u04eac\u030fb\u0677e\u06728\u07396\u0751f\u03001\u0958b\u05e55\u030c7\u030d3\u030e5\u030fc\u05f3e\u060dc\u03057\u03044\u2015\u0300c\u070b9\u053d6\u0308a\u0305f\u0304b\u03063\u0305f\u0300d\u09ad8\u05352\u065b0\u04eba\u030b9\u030bf\u030e1\u030f3\u03067\u05de6\u08db3\u05f37\u070c8\u030df\u030c9\u030eb\nhttps://t.co/IopFANAwll\n#urawareds #jleague #football #soccer\n\n\u04e2d\u07530\u03001\u0672c\u07530\u07cfb\u07d71\u03068\u0540c\u03058\u030cb\u030aa\u030a4\u0304c\u03059\u0308b\u03002\n\u03055\u03063\u03055\u03068\u030df\u030af\u030b7\u030a3\u06771\u04eac\u03092\u051fa\u03066\u06d77\u05916\u06311\u06226\u03057\u03066\u0307b\u03057\u03044\u03002"}], "meta": {"newest_id": "149484800878809089", "oldest_id": "1494847509222486017", "result_count": 10, "next_token": "b26v89c19zqg8o3fpe76kxjx6dvvgft172vquvw1000ot"}}
```

Otteniamo in questo modo un JSON che contiene le informazioni riguardo ai tweet recenti su Twitter con l'hashtag "football" passato come parametro

I Wrapper

Nella pratica è solito usare dei wrapper ovvero delle librerie/oggetti che faranno il "lavoro sporco" per noi. Noi penseremo solamente a dare i termini di autenticazione, nel nostro caso i 4 token di twitter. Al bearer token che è l'entità più difficile da gestire ci penserà il wrapper.

Il wrapper ovvero la libreria chiamata twitter ci faciliterà le richieste, ad esempio ottenere l'autenticazione diventa una semplicissima chiamata a funzione dove inseriamo le nostre keys, al bearer token ci penserà il pacchetto.

Iniziamo ottenendo un oggetto di tipo twitter.api.Api con questa chiamata a funzione(contenuta nel pacchetto twitter)

In [4]:

```
import yaml

with open("psw.yml", "r") as stream:
    psw = yaml.safe_load(stream)
```

In [5]:

```
import twitter
api = twitter.Api(consumer_key=str(psw['consumer_key']),
                  consumer_secret=str(psw['consumer_secret']),
                  access_token_key=str(psw['access_token_key']),
                  access_token_secret=str(psw['access_token_secret']))
print(type(api))
```

```
<class 'twitter.api.Api'>
```

Con la chiamata GetUser possiamo ottenere un oggetto che contiene le informazioni su un utente, estrapioliamo nell'esempio l'utente di Cristiano Ronaldo e conserviamo in una variabile ID_CR il suo id

In [6]:

```
cristiano_ronaldo=api.GetUser(screen_name='@Cristiano')
ID_CR=cristiano_ronaldo.id
```

La GetUserTimeline(user) è utile per ottenere i post recenti di un utente

In [7]:

```
statuses = api.GetUserTimeline(ID_CR,count=1)
print([statuses[0]])
tweet=statuses[0];
```

```
[Status(ID=1493711992510681096, ScreenName=Cristiano, Created=Tue Feb 15 22:
21:05 +0000 2022, Text='Back on track! Nobody gives up and there's only one
way to get back on track: hard work, team work, serious work. E... https://t.co/q1vdaSNExG')] (https://t.co/q1vdaSNExG)])
```

Vediamo quante persone segue il nostro utente, la funzione GetFriendIDs intuitivamente ci torna una lista con gli ID delle persone seguite dall'utente

In [8]:

```
friendsCR=api.GetFriendIDs(user_id=ID_CR)
print(len(friendsCR))
```

60

La `GetStatus` ottiene un oggetto che contiene informazioni su un tweet con un certo id.

In [9]:

```
status=api.GetStatus(tweet.id) #ottieni tweet con un certo id
```

Nell'oggetto sono contenuti vari campi che caratterizzano il tweet, estraiamo ad esempio il numero di mi piace(in Twitter sono detti "favorites") e il numero di retweet. Ci sono alcune informazioni estraibili tramite API solo avendo un account developer Premium Enterprise come ad esempio il `reply_count` che conteggia intuitivamente il numero di risposte che ha avuto il tweet.

In [10]:

```
print(status.favorite_count,status.retweet_count)
```

348107 36266

In [11]:

```
print(tweet.id)
```

1493711992510681096

Proviamo ad estrarre il `reply_count` usando una funzionalità di un altro pacchetto chiamato `tweepy` che permette di scorrere le risposte al tweet per conteggiarle una ad una in modo non troppo complicato. Però la chiamata potrebbe interrompersi se il tweet ha troppe risposte da conteggiare.

Bisogna fare attenzione al fatto che, come c'è scritto nelle docs., ci sono dei rate limit(per ogni 15 minuti) da rispettare. Superati questi limiti iniziamo ad avere degli errori. Anche una sola 'ispezione' a un tweet potrebbe darci problemi se i numeri sono grandi.

Possiamo ottenere errori dovuti soprattutto al fatto che i calcoli sono fatti in "real time".

In [12]:

```

import tweepy

consumer_key=str(psw['consumer_key'])
consumer_secret=str(psw['consumer_secret'])
access_token_key=str(psw['access_token_key'])
access_token_secret=str(psw['access_token_secret'])

auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token_key, access_token_secret)
apiTweepy = tweepy.API(auth)

name = cristiano_ronaldo.screen_name

tweet_idINT = tweet.id
tweet_idSTR = str(tweet_idINT)

repliesCR=0
try:
    for tweetT in tweepy.Cursor(apiTweepy.search_tweets,q='to:'+name, result_type='recent')
        if hasattr(tweetT, 'in_reply_to_status_id_str'):
            if (tweetT.in_reply_to_status_id_str==tweet_idSTR):
                repliesCR = repliesCR + 1
    print(repliesCR)
except:
    print("Rate Limit dei 15 mins infranto.")

```

194

Il tweet di Cristiano Ronaldo ha numeri grandi. Proviamo ad eseguire lo stesso script però con un post con numeri più contenuti. Prendiamo un post di Matteo Salvini che sembra avere numeri ragionevoli(non è detto comunque funzioni, potremmo aver comunque infranto il rate limit con precedenti chiamate)

In [14]:

```

nameMS = 'matteosalvinimi'
tweet_idMS = '1460958935083634688'

repliesMS=0
try:
    for tweetT in tweepy.Cursor(apiTweepy.search_tweets,q='to:'+nameMS, result_type='recent')
        if hasattr(tweetT, 'in_reply_to_status_id_str'):
            if (tweetT.in_reply_to_status_id_str==tweet_idMS):
                repliesMS = repliesMS + 1
    print(repliesMS)
except:
    print("Rate Limit dei 15mins infranto.")

```

Rate Limit dei 15mins infranto.

In conclusione per quanto riguarda il reply_count non è facile/possibile estrarre sempre informazioni usando le API. Un'altra tecnica sarebbe il web scraping che però risulta altrettanto difficoltoso in quanto quelle che ci da Twitter non sono pagine statiche HTML da cui estrarre facilmente informazioni ma sono degli script. Usare BeautifulSoup è quindi inutile, bisognerebbe usare altre librerie che in pratica aprono una vera e propria finestra di un browser per estrarre l'html dato in output al client il che rende tutto molto lento.

Cerchiamo ora di fare una statistica.

Raccogliamo da tweet recenti di Ronaldo 3 informazioni: il testo, il favorite_count e il retweet_count.

Dal testo proveremo a suddividere i tweet in 2 sottogruppi: tweet in cui Ronaldo parla della sua squadra(Manchester United o nazionale del Portogallo o Juventus) e tweet in cui non ne parla(quindi potrebbe parlare di se stesso, la sua famiglia sponsor ecc) Suddivisi i tweet andremo a vedere i campi favorite_count e retweet_count per creare 2 medie per ognuno dei 2 sottogruppi. Ottenute queste informazioni riusciremo a valutare se i suoi seguaci preferiscono uno dei 2 tipi di tweets all'altro o se il tipo di tweet è influente(dunque le medie sarebbero circa uguali)

Usiamo la GetUserTimeline() per ottenere un campione di 200 post recenti del nostro utente

In [12]:

```
campione = api.GetUserTimeline(ID_CR,count=200)
```

In ogni istanza di campione ci sono molti campi: a noi interessano il text, il favorite_count e il retweet_count e anche gli hashtags in quanto potrebbe anche esserci un #ManchesterUnited ad esempio. Definiamo una lista di stringhe, che se presenti, fanno intendere il post parli della squadra(potrebbe essere una tra Manchester United, Portogallo e Juventus):

In [13]:

```
#(La chiamiamo LTAS che sta per ListTalkAboutSquad)
LTAS = ["PT", "PORTUGAL", "seleção", "vamoscomtudo", "VamosComTudo", "EQUIPA", "equipa",
        "Man. United", "Old Trafford", "Man. United", "RED DEVILS", "⚫⚪", "⚫", "mufc",
        "MANCHESTER UNITED", "MAN. UNITED",
        "FINO ALLA FINE", "🏆", "Fino Alla Fine", "Juve", "JUVENTUS", "Juventus", "juve",
        "juventus", "Winning team", "Vittoria", "Fino Alla Fine",
        "finoallafine", "FINOALLAFINE", "FinoAllaFine"]
length = len(LTAS)
print(length)
```

31

L'oggetto dato in output dalla GetStatus ha un attributo chiamato "Hashtags" da cui vogliamo estrarre gli hashtags del tweet.

In [14]:

```
idPostEsempio = 1355599316300292097
post = api.GetStatus(idPostEsempio)
print(type(post.hashtags[0]))
print(post.hashtags[0].text)
```

```
<class 'twitter.models.Hashtag'>
finoallafine
```

Come si vede, da un oggetto di tipo hashtag è facilmente estraibile la parte testuale che è quella che ci interessa. Andiamo a suddividere il nostro campione in 2 sottoinsiemi.

In [15]:

```

Tweet_squad = []
Tweet_Nosquad = []

for tweet in campione:
    aggiuntoATweetSquad = False

    #vediamo se almeno una parola della nostra lista sta nel text
    for parola in lTAS:
        if parola in tweet.text:
            Tweet_squad.append(tweet)
            aggiuntoATweetSquad = True
            break

    #vediamo se almeno una parola della nostra lista sta tra gli hashtags del tweet
    #usiamo il primo if per evitare di inserire 2 volte nella lista squad un tweet
    #in cui nel testo si parla della squadra e che contiene anche hashtags che fanno
    #match con la nostra lista
    if aggiuntoATweetSquad == False:
        for hashtag in tweet.hashtags:
            for parola in lTAS:
                if parola in hashtag.text:
                    Tweet_squad.append(tweet)
                    aggiuntoATweetSquad = True
                    break

    #se il valore è false vuol dire che il tweet non è stato aggiunto nella lista squad qui
    if aggiuntoATweetSquad == False:
        Tweet_Nosquad.append(tweet)

print(len(Tweet_squad))
print(len(Tweet_Nosquad))

```

67
133

Adesso facciamo la somma di tutti i favorite_count e retweet_count dei 2 sottogruppi e facciamo le medie per ogni tweet.

In [16]:

```

somma_FS = 0
somma_RS = 0
for t in Tweet_squad:
    somma_FS = somma_FS + t.favorite_count
    somma_RS = somma_RS + t.retweet_count
media_FS = somma_FS/(len(Tweet_squad))
media_RS = somma_RS/(len(Tweet_squad))
print("Un post che parla della squadra ha in media: " + str(media_FS) + " likes e " + str(m

```

Un post che parla della squadra ha in media: 216879.74626865672 likes e 1300
1.34328358209 retweets

In [17]:

```
somma_FNS = 0
somma_RNS = 0
for t in Tweet_Nosquad:
    somma_FNS = somma_FNS + t.favorite_count
    somma_RNS = somma_RNS + t.retweet_count
media_FNS = somma_FNS/(len(Tweet_Nosquad))
media_RNS = somma_RNS/(len(Tweet_Nosquad))
print("Un post che NON parla della squadra ha in media: " + str(media_FNS) + " likes e " +
```

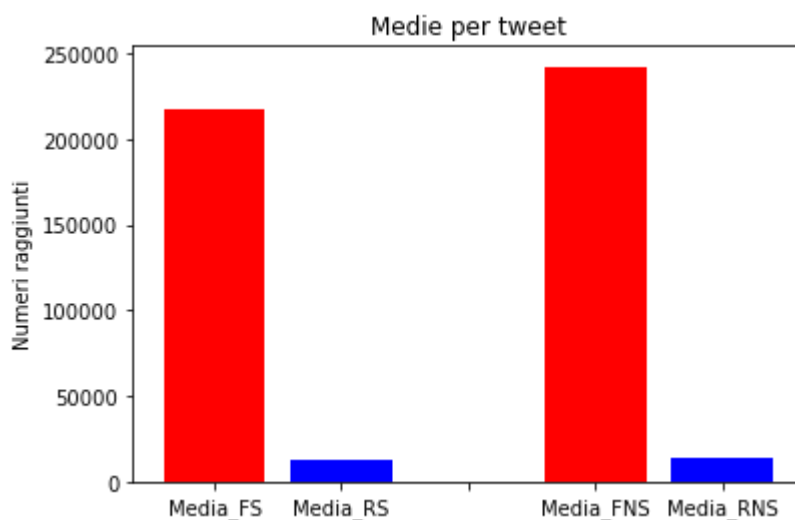
Un post che NON parla della squadra ha in media: 242045.22556390977 likes e
14029.142857142857 retweets

Disegniamo alcuni grafici per mostrare più visivamente se ci sono differenze grandi tra le varie medie

1. Media_FS indica la media dei like in un tweet di tipo Squad
2. Media_RS indica la media dei retweet in un tweet di tipo Squad
3. Media_FNS indica la media dei like in un tweet di tipo NoSquad
4. Media_RNS indica la media dei retweet in un tweet di tipo NoSquad

In [18]:

```
import matplotlib.pyplot as plt
import numpy as np
medie = ["Media_FS", "Media_RS", "", "Media_FNS", "Media_RNS"]
valori = [media_FS , media_RS ,0, media_FNS , media_RNS]
x_pos = np.arange(len(valori))
plt.bar(x_pos, valori, align='center',color=["red" , "blue" ,"white", "red" , "blue"])
#La stringa vuota, lo 0 e il "white" sono usati solo per creare un pò di spazio
plt.xticks(x_pos, medie)
plt.ylabel('Numeri raggiunti')
plt.title('Medie per tweet')
plt.show()
```



In [19]:

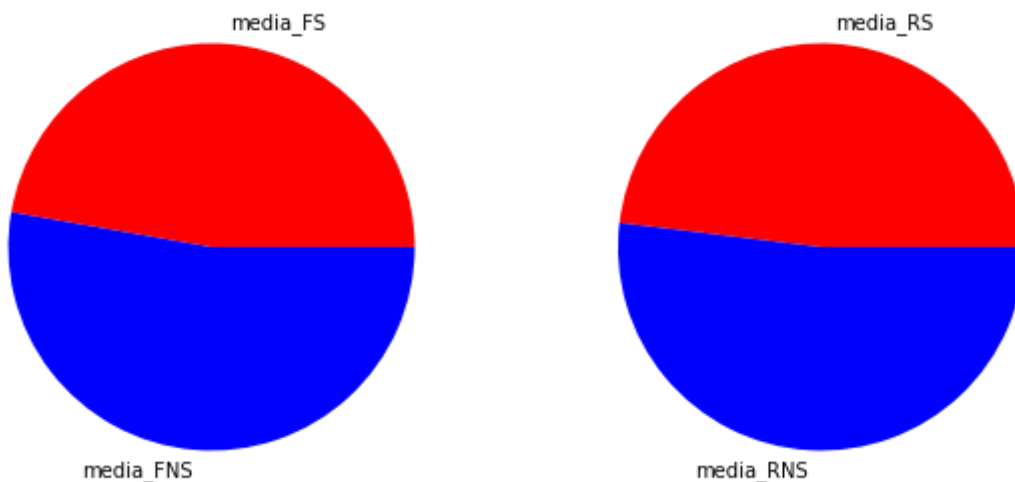
```
fig, (ax1,ax2) = plt.subplots(1,2,figsize=(10,10))

lista_fav = [media_FS , media_FNS]
label1 = ["media_FS","media_FNS"]
ax1.pie(lista_fav, labels=label1,colors=["red","blue"])

lista_r = [media_RS , media_RNS]
label2 = ["media_RS","media_RNS"]
ax2.pie(lista_r, labels=label2,colors=["red","blue"])

print("Grafici a torta per mostrare le differenze tra le varie medie")
```

Grafici a torta per mostrare le differenze tra le varie medie



Dai dati che otteniamo dalle medie potremmo dire che effettivamente i seguaci di Ronaldo non sono molto influenzati dal tipo di tweet dell'utente, sembrano sempre apprezzare le sue attività.

Correlazione tra favorite_count e retweet_count

Vediamo ora se il favorite_count e il retweet_count sono correlate secondo il coefficiente di Pearson (lo vediamo nuovamente per tutto il campione e per i sottoinsiemi separati). Verifichiamo dunque se c'è una corrispondenza lineare tra le 2 variabili. Plottiamo per vedere almeno a occhio se c'è un qualche andamento lineare tra le 2 variabili nell'intero campione.

Creiamo intanto 2 strutture, una conterrà i valori dei favorites del campione e l'altra conterrà i valori dei retweets. Creiamo anche delle strutture che contengono i favorites dei tweet che sono classificati come squad e come Nosquad

In [23]:

```
favCampione = []
retCampione = []

for t in campione:
    favCampione.append(t.favorite_count)
    retCampione.append(t.retweet_count)

favSquad = []
retSquad = []

for t in Tweet_squad:
    favSquad.append(t.favorite_count)
    retSquad.append(t.retweet_count)

favNoSquad = []
retNoSquad = []

for t in Tweet_Nosquad:
    favNoSquad.append(t.favorite_count)
    retNoSquad.append(t.retweet_count)
```


In [24]:

```

from sklearn.linear_model import LinearRegression

x = np.array(favCampione)
y = np.array(retCampione)

x = x.reshape(-1, 1)
y = y.reshape(-1, 1) #il metodo fit richiede che l'input abbia questa struttura

model = LinearRegression().fit(x, y)
coeff_ang = model.coef_[0][0]
interc = model.intercept_

a = np.linspace(0,max(favCampione),max(favCampione))
b = coeff_ang*a+interc
#abbiamo creato la retta di regressione
plt.plot(favCampione,retCampione,'g^',a,b)
#plottiamo i punti reali che mettono in relazione favorites e retweets e la retta di regres

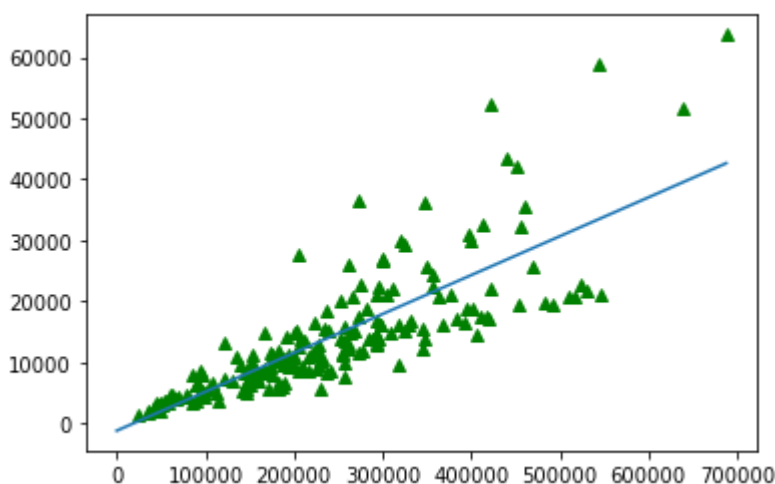
```

Out[24]:

```

[<matplotlib.lines.Line2D at 0x1906b730490>,
 <matplotlib.lines.Line2D at 0x1906b730310>]

```



Notiamo visivamente una correlazione lineare abbastanza forte soprattutto nei tweets con favorites e retweets bassi. Possiamo a questo punto predire il numero di retweets dato il numero di favorites, sfruttiamo il metodo `predict()`.

In [25]:

```

fav = [[500000]] #il metodo predict richiede che l'input abbia questa struttura
print(model.predict(fav))

```

```
[[30659.98906814]]
```

Verifichiamo ora quanto quantitativamente la correlazione lineare tra le 2 variabili, per farlo usiamo il coefficiente di Pearson, un valore tra -1 e 1 che indica quanto esse sono correlate linearmente. Nella pratica un buon valore per il coefficiente di Pearson è in valore assoluto maggiore o uguale a 0.8

In [26]:

```
def coefficienteAttendibile(c):
    if c >= 0.8:
        print("Il coefficiente supera 0.8, si può considerare attendibile")
    elif c <= -0.8:
        print("Il coefficiente è meno di -0.8, si può considerare attendibile")
    else:
        print("Il coefficiente ha valore (assolutamente) inferiore a 0.8, è inattendibile")
```

In [27]:

```
coeffCampione = np.corrcoef(favCampione,retCampione)
#restituisce una matrice 2*2 dove nella principale ci sono i coefficienti tra favCampione e
# se stessi(quindi il valore è 1). Nella secondaria invece ci sarà il coefficiente tra favC

print("Il coefficiente di Pearson tra favCampione e retCampione è: "+str(coeffCampione[0,1])
coeffienteAttendibile(coeffCampione[0,1])

coeffSquad = np.corrcoef(favSquad,retSquad)
print("Il coefficiente di Pearson tra favSquad e retSquad è: "+str(coeffSquad[0,1]))
coeffienteAttendibile(coeffSquad[0,1])

coeffNoSquad = np.corrcoef(favNoSquad,retNoSquad)
print("Il coefficiente di Pearson tra favNoSquad e retNoSquad è: "+str(coeffNoSquad[0,1]))
coeffienteAttendibile(coeffNoSquad[0,1])

lista_coeff = [coeffNoSquad[0,1],coeffCampione[0,1],coeffSquad[0,1]]
if max(lista_coeff) == coeffNoSquad[0,1]:
    print("Sembra che il coefficiente di Pearson più alto si veda nei tweet di tipo NoSquad")
elif max(lista_coeff) == coeffSquad[0,1]:
    print("Sembra che il coefficiente di Pearson più alto si veda nei tweet di tipo Squad")
elif max(lista_coeff) == coeffCampione[0,1]:
    print("Sembra che il coefficiente di Pearson più alto si veda nel campione intero")
```

```
Il coefficiente di Pearson tra favCampione e retCampione è: 0.81577313418074
56
Il coefficiente supera 0.8, si può considerare attendibile
Il coefficiente di Pearson tra favSquad e retSquad è: 0.8609276090239888
Il coefficiente supera 0.8, si può considerare attendibile
Il coefficiente di Pearson tra favNoSquad e retNoSquad è: 0.8201598494695587
Il coefficiente supera 0.8, si può considerare attendibile
Sembra che il coefficiente di Pearson più alto si veda nei tweet di tipo Squ
ad
```

Verifichiamo ora invece se i nostri risultati sono statisticamente significativi, viene in nostro aiuto il p-value che è facilmente calcolabile grazie alla funzione `pearsonr()` all'interno di `scipy.stats`. La statistica ci dice che possiamo ritenere statisticamente significativo un risultato qualora il relativo p-value sia un valore vicino a 0. Nei test di ipotesi si rifiuta l'ipotesi nulla quando il nostro p-value è basso. Nel nostro caso potremmo considerare come ipotesi nulla quella che afferma l'assenza di correlazione lineare tra le 2 variabili, noi la vogliamo confutare. Se il p-value è basso (di solito con basso si intende inferiore a una soglia che solitamente è 0.05) confutiamo l'ipotesi nulla e consideriamo dunque i risultati statisticamente significativi.

La `pearsonr()` dunque ci permette di calcolare il p-value riguardante i coefficienti della retta per verificare se il nostro è un risultato statisticamente significativo.

In [28]:

```
from scipy import stats
_, p_value = stats.pearsonr(favCampione, retCampione)
#La funzione ritorna il coefficiente di Pearson e il p-value, il primo lo abbiamo calcolato
# e commentato prima. In quel caso abbiamo utilizzato la funzione corrcoef()
print("Il p-value ha valore uguale a " + str(p_value))
if p_value < 0.05:
    print("Il risultato è statisticamente significativo")
```

Il p-value ha valore uguale a 5.72064929688008e-49

Il risultato è statisticamente significativo

Conclusioni

L'uso delle API di Twitter può essere molto difficoltoso ma l'esistenza dei Wrapper ci facilita la vita. Abbiamo visto che non sempre tutto è possibile con le API (ad esempio esiste il rate_limit nei 15 minuti) Twitter si rivela un ottimo social media per estrarre informazioni riguardanti soprattutto persone famose che siano politici, calciatori, attori ecc. Nel tutorial abbiamo usato un modo molto grezzo per valutare se un tweet fosse un tweet che parlasse della squadra o no in quanto effettivamente il problema da risolvere non è difficoltoso (quando Ronaldo parla della squadra inserisce praticamente sempre una delle stringhe in ITAS).

Se ci imbattessimo in un problema più difficoltoso (più di 2 etichette, tweet meno strutturati, informazioni date in maniera meno chiara ecc...) potremmo usare algoritmi di classificazione veri e propri per appunto attribuire delle etichette ai nostri tweet in maniera più precisa.

In []: