Twitter Analysis

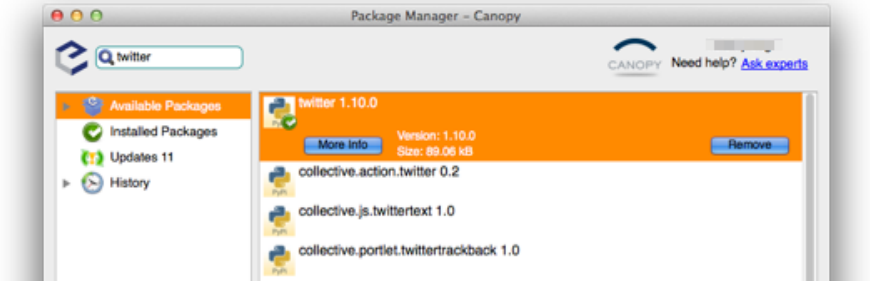
TWITTER ANALYSIS **1**
FOR DICTIONARY: READ TWEETS JSON FORMAT 2
Install twitter package for Python 2
Obtaining Twitter apps development token 3
CREATING TWITTER #HASHTAG NETWORK 6
Getting the hashtags in each tweet 6
Hashtag Query Expansion by detecting hashtags in tweets 9
Building hashtag networks 11

For dictionary: Read Tweets JSON format

- 現今許多網路服務儲存資料或者回傳資料的格式均採用 JSON，而 Twitters 也不例外。在以下的例子中，將介紹如何使用第三者開發的函式庫來截取 Twitters 資料。而 JSON 格式對 PYTHON 而言可說是以 DICTIONARY 搭配 LIST 的階層式結構。

Install twitter package for Python

- 首先，先至 Canopy 的 Package Manager 搜尋「twitter」相關函式庫，並安裝「twitter 1.10.0」這個套件。



- 安裝後開啓 LiClipse，新增一個 PyDev 專案後，新增一個檔案（例如 tweetAnalysis.py）檔名不可以是 twitter.py（因為函式庫本身已經稱為 twitter，若檔名為 twitter.py，執行後會覆寫函式庫的物件導致無法使用物件的內容。）。
- 輸入 **`from twitter import *`**並執行。若執行成功的話，代表已經安裝成功；若不成功的話，請詳見本課程講義「C1_Installing Development Environments」該檔案中「Installing LiClipse/Problem shooting」的說明。
- 接下來上網查詢這個 Twitter API 要怎麼使用（這不是 Twitter 官方提供的 API，而是由某些程式設計者所開發）。因此輸入關鍵字「twitter 1.10」第一筆結果即是。

- 點選後觀察其說明，該函式庫可用以截取個人、朋友的訊息、並可以代發訊息，或者依照關鍵字進行搜尋。

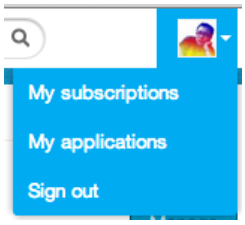
```
Examples::

```python
from twitter import *

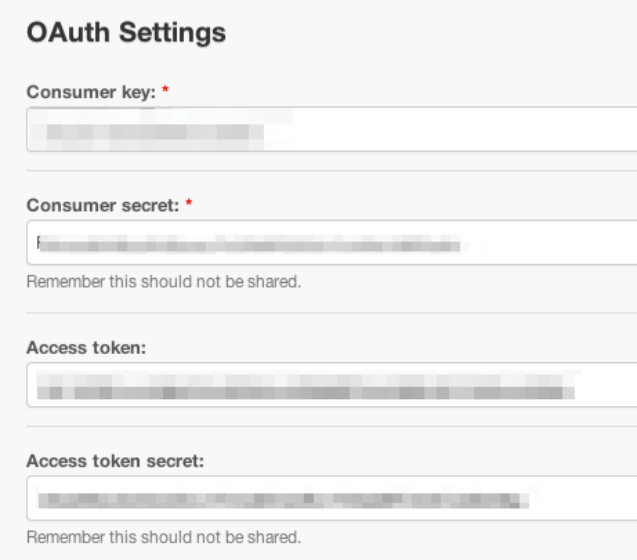
# see "Authentication" section below for tokens and keys
t = Twitter(
            auth=OAuth(OAUTH_TOKEN, OAUTH_SECRET,
                       CONSUMER_KEY, CONSUMER_SECRET)
           )

# Get your "home" timeline
t.statuses.home_timeline()


 Searching Twitter::

 ``` python
 # Search for the latest tweets about #pycon
 t.search.tweets(q="#pycon")
 ```
```

- 依照範例複製輸入以下程式碼。由於系統無法獲知「OAUTH_TOKEN, OAUTH_SECRET, CONSUMER_KEY, CONSUMER_SECRET」這四個變數為何，所以會出現 ERROR 的提醒。

```
6  tquery = Twitter(
7              auth=OAuth(OAUTH_TOKEN, OAUTH_SECRET,
8                         CONSUMER_KEY, CONSUMER_SECRET)
9             )
```

Obtaining Twitter apps development token

- 這四個變數應填入 Twitter 某個應用程式的開發者帳號。申請 Twitter 帳號後，登入 https://dev.twitter.com。
- 按右上方帳號的下拉式選單找到「My application」。



  - 按「Create New App」新增一個應用程式，請妥善填寫各項內容，勿隨便亂填。
  - 新增成功後進入該應用程式，選取「API Keys」這個分頁。然後拉至頁面最下方點選「Generate access token」。
  - 經過幾秒至數分鐘後，點選頁面右上方的「Test OAuth」按鈕，可以獲得以下畫面。注意，每一個欄位都應該有數值，若沒有「Access token」或「Access token
```

secret」請回到上一步稍微等候，或者點按「Regenerate」。



- 之後依序將這四個值，自程式中指定給「OAUTH_TOKEN, OAUTH_SECRET, CONSUMER_KEY, CONSUMER_SECRET」如下。填寫時注意相對應的欄位，不然會填錯。此時 ERROR 警訊應該已經消失了。

```
from twitter import *
OAUTH_TOKEN = "
OAUTH_SECRET = "
CONSUMER_KEY = "
CONSUMER_SECRET = "
tquery = Twitter(
            auth=OAuth(OAUTH_TOKEN, OAUTH_SECRET,
                       CONSUMER_KEY, CONSUMER_SECRET)
          )
```

- 接下來新增底下兩行程式碼進行查詢並列印出來。發現列印出來的 **tweets** 指到一個 DICTIONARY。

```
tquery = Twitter(
            auth=OAuth(OAUTH_TOKEN, OAUTH_SECRET,
                       CONSUMER_KEY, CONSUMER_SECRET)
          )

tweets = tquery.search.tweets(q="#CongressOccupied")
print tweets
```

```
{u'search_metadata': {u'count': 15, u'completed_in': 0.049,
u'max_id_str': u'450876365382774784', u'since_id_str': u'0',
u'next_results':
u'?max_id=450867883099561983&q=%23CongressOccupied&include_e
ntities=…
```

- 於是利用以下方式列印出該 DICTIONARY 的 KEY 和相對應的 VALUE。
- 該 DICTIONARY 有兩個項目，分別是**'search_metadata'**和**'statuses'**。
- **'search_metadata'**對應到的是一個與該查詢相關的資料，並且使用

DICTONARY 來儲存（例如 **count** 是資料筆數、**completed_in** 為執行時間等等）。

- **'statuses'**對應到的一個 LIST（請注意看！），該 LIST 中有非常多組 DICTIONARY，也就是非常多 DICTIONARY 所構成的 LIST。每個 DICTIONARY 中包含了例如**'text'**為內容等資訊。

```
11 tweets = tquery.search.tweets(q="#CongressOccupied")
12 for k in tweets:
13     print k, '\t', tweets[k]
```

```
search_metadata {u'count': 15, u'completed_in': 0.055, u'max_id_str': u'450876889964351488',
statuses        [{u'contributors': None, u'truncated': False, u'text': u"RT @ilmaurosacchi:
```

- 基於 **statuses** 對應到的為所需資料的 LIST，所以應把 FOR 迴圈改為 **for k in tweets['statuses']:**。基於 LIST 中每筆文章所需資料為 **text** 欄位，因此把 **print** 的內容改為 **print k['text']**。全部的程式碼與執行結果如下。

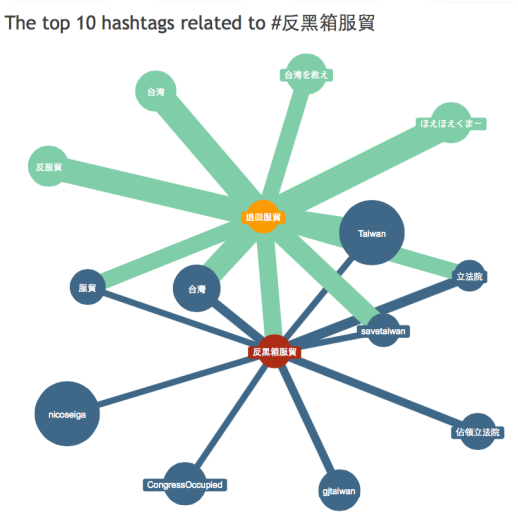
```
1  from twitter import *
2  OAUTH_TOKEN = "[redacted]"
3  OAUTH_SECRET = "[redacted]"
4  CONSUMER_KEY = "[redacted]"
5  CONSUMER_SECRET = "[redacted]"
6  tquery = Twitter(
7              auth=OAuth(OAUTH_TOKEN, OAUTH_SECRET,
8                         CONSUMER_KEY, CONSUMER_SECRET)
9             )
10
11 tweets = tquery.search.tweets(q="#CongressOccupied")
12 for k in tweets['statuses']:
13     print k['text']
```

```
<terminated> /Volumes/Data/Dropbox/Programming/python_src/pss2014/twitterViewer.py
Love from China! HaHa #CongressOccupied
 @nytimes &lt;3 http://t.co/KqkZmhrw6H
RT @amei22887: 今天 2014.3.30 我是50萬分之一 我以身為臺灣人為榮 RT @4amtw Democracy at 4pm. #CongressOc
#Taiwan #CongressOccupied #democracy #sunflowermovement http://t.co/pgnnSbHILL
RT @ilmaurosacchi: @mmbilal @AJStream  THANK YOU for covering Taiwan's historic, nonviolent revol
Pro-China group led by former felon Chang An-Le approached Legislature. Police is on high alert.
RT @gjtaiwan: #反服貿 #gjtaiwan #congressoccupied 集中兵力守議會前廣場 by @siegfy at http://t.co/mYlx
14點了，現場有動靜嗎？ #CongressOccupied
support #318 #congressoccupied @ 政治大學羅馬廣場 http://t.co/04JaTQA6yV
Check out "當電視淪陷時，我們該怎麼辦？" on Vimeo http://t.co/PVKgjDLgI8 #Vimeo #congressoccupied #tai
White wolf and gangsters go to parliament. #CongressOccupied @savetaiwan2014
RT @gjtaiwan: #反服貿 #gjtaiwan #congressoccupied 集中兵力守議會前廣場 by @siegfy at http://t.co/mYlx
Check out "當電視淪陷時，我們該怎麼辦？" on Vimeo http://t.co/9yDoEdFBtz #Vimeo #congressoccupied #tai
RT @gjtaiwan: #反服貿 #gjtaiwan #congressoccupied 集中兵力守議會前廣場 by @siegfy at http://t.co/mYlx
RT @gjtaiwan: #反服貿 #gjtaiwan #congressoccupied 集中兵力守議會前廣場 by @siegfy at http://t.co/mYlx
RT @gjtaiwan: #反服貿 #gjtaiwan #congressoccupied 集中兵力守議會前廣場 by @siegfy at http://t.co/mYlx
```

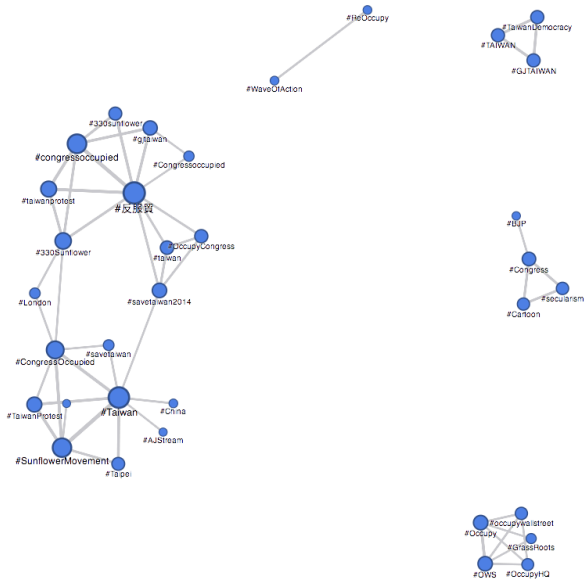
- 總結：上面的案例是讀取 Twitter API 所傳回的資料，該資料格式為 JSON，而 JSON 的資料儲存方式以 DICTIONARY 搭配 LIST 為主，所以需要一步一步查詢 JSON 中有哪些 KEY，並找到所需要的 VALUE 再一一列印出來。

Creating Twitter #hashtag network

- 接下來我們希望能夠做到類似這個網站所提供的效果，若要實作互動性的話必須要建立網路服務，因此我們會建立一個動態網絡界面，但不提供「與資料互動」的功能。



- 大致上希望建置出來的界面如下圖。



Getting the hashtags in each tweet

- 在之前我們已經可以用 **tquery.search.tweets(q="#CongressOccupied", lang = 'en', count = 100)**的指令來下查詢。會得到非常多的 tweet 文章列表如下。仔細觀察，會發現每一篇文章都有自己的 Hashtag，為#字記號開頭。所以我希望偵測文章裡面的#字記號開頭文字，來觀察 Hashtag 與 Hashtag 間的關係（還記得已經教過關鍵字的共現關係網絡）。
- 想法大致如下。
  - 先建立一個 DICTIONARY 來存放文章（利用 **id**）與其所包含的 Hashtag 的對應（**tweetdict = {}**），所建置的對應即為 **id** 對應至多個 Hashtags。
  - 就前面的查詢，我去讀取查詢結果的每一則 tweet 文章（**for tweet in tweets['statuses']:**）。如果某則文章的 id 沒有在 tweetdict 中時，那麼就用 **tweetdict.setdefault(id, initial_value)**建立一個 key 與初始值的對應。
  - 建立完後，進行斷句、斷字，並用 **word.startswith()**找到開頭為#的文字，一一附加（**append**）在 **hashtags** 這一個 LIST 中。
  - 建立 **id** 對 **hashtags** 的對應

```
def hashtagquery(query):
    print "current tag:\t%s"%query
    tweets = tquery.search.tweets(q=query, count = 100)
    for tweet in tweets['statuses']:
        if tweet['id'] not in tweetdict:
            hashtags = []
            for word in tweet['text'].split():
                if word.startswith('#'):
                    hashtags.append(word)
            tweetdict[tweet['id']] = hashtags
```

- 同時我希望建立一個輔助參考用的 DICTIONARY（**tagdict = {}**），存放所有曾經出現過的 Hashtag 及其出現次數。這樣我就可以列舉所有出現過的 Hashtag。因此在上述的程式後加入這些程式碼。
  - 下圖紅色為上述說明的程式碼。請注意到藍色的部分，比較前一個程式碼的範例，應可發現這行程式碼大大精簡了原本的程式碼。這樣的做法在 Python 中稱為「List Comprehension」。如果你對這種撰寫方法感興趣的話，可以上網用 google 查閱相關資訊。

```
tweetdict = {}
tagdict = {}
tagpair = {}
def hashtagquery(query):
    print "current tag:\t%s"%query
    tweets = tquery.search.tweets(q=query, count = 100)
    for tweet in tweets['statuses']:
        if tweet['id'] not in tweetdict:
            hashtags = [word for word in tweet['text'].split()
            if word.startswith('#')]
            tweetdict[tweet['id']] = hashtags
            for tag in hashtags:
                if tag not in tagdict:
                    tagdict[tag] = 1
                else:
                    tagdict[tag] += 1
```

- 在 **if __name__=="__main__":**的地方撰寫程式碼來測試上述函式。
  - 注意，在這邊完全沒有傳回值，因為這個程式把「**tweetdict = {}**與**tagdict = {}**」宣告為「全域變數（Global Variable）」，而且又宣告在最前面，所以後續程式碼都可以讀寫這兩個 DICTIONARY（為什麼這麼做有後續用途）。

```
from twitter import *
OAUTH_TOKEN = "YOURS"
OAUTH_SECRET = "YOURS"
CONSUMER_KEY = "YOURS"
CONSUMER_SECRET = "YOURS"
tquery = Twitter(auth=OAuth(OAUTH_TOKEN, OAUTH_SECRET,
                        CONSUMER_KEY, CONSUMER_SECRET))
tweetdict = {}
tagdict = {}
tagpair = {}

def hashtagquery(query):
    …
if __name__=="__main__":
    hashtagquery("#CongressOccupied")
    for k in sorted(tagdict, key = tagdict.get,
    reverse = True)[:100]:
        print k, '\t', tagdict[k]
```

- 執行查詢的結果如下。

```
#CongressOccupied        58
#congressoccupied        31
#Taiwan 15
#反服貿  12
#osdctw 12
#SunflowerMovement       8
#SunflowerMvmt   7
#g0v     7
#捍衛民主          6
#中天     6
#中國時報          6
#318學運           6
#Sunflower       6
#fb      6
```

Hashtag Query Expansion by detecting hashtags in tweets

- 上面的範例所展示的是，用#hashtag 送至 Twitter 進行查詢，把查詢的結果，也就是一則則的 Tweet 文章傳回來。傳回來的同時，我嘗試抓出每一則文章裡面的 Hashtags。
- 接下來則可以做兩件事：
  - 其一，既然 **`hashtagquery(query)`**可以查詢某個 Hashtag 的相關文章並擷取出相關關鍵字。那麼，多延伸一步，也可以將抓出來的 Hashtags 再一一傳入 **`hashtagquery(query)`**函式，就可以找到更多的 Hashtags。
  - 其二，既然抓出的每篇文章都有自己的 Hashtags，那就可以建立出早先 Web of Sciences 範例中的「關鍵字共現網絡（Co-existing keywords）」。
- 關於其一，這樣的操作在資訊擷取（Information Retrievals）領域稱為「查詢擴展」。亦即利用查詢出來的結果（在這個範例中，以 hashtag 進行查詢，查詢到有包含這個 hashtag 的 tweets，查詢的結果即指這些 tweets），再進行剖析以查詢，嘗試擴展原本只就單一關鍵字查詢，所能夠獲得的結果。並且在查詢擴展後，甚至能夠提供原本查詢關鍵字的相關關鍵字給使用者，當使用者原本所建立的「查詢關鍵字」不見得能夠完全符合其所需時（也就是不會下關鍵字的時候），可以協助使用者找到對的關鍵字。
- 關於其一的做法，想法如下：如果在 **`hashtagquery(query)`**函式裡面，把找到的關鍵字立刻再丟入 **`hashtagquery(query)`**函式來查詢，只要文章還有新的 Hashtag，可以想見這個程式是不會停止的。停止的條件是，程式因超過記憶體界限當掉了，網路不穩，或者是因為被 Twitter 偵測大量抓取的行為而被擋掉了。因此，要設定一個 Level，例如 Level 為 3 的話，就往外抓 3 層。程式如下。
  - 黃色的凸顯部分為與前述範例不同的部分。**`lv`** 用以控制要多找幾圈，因故每次往外展一圈，就要減 1，避免產生無窮的查詢。而最底下的黃色部分則是把查詢出來的 **`tag`**，再一一傳入 **`hashtagquery(query, lv)`** 進行查詢。
  - 註解以下的第二部分前面也有出現過，在此用文字進行解釋。該段程式碼的用意在於，對於每篇 Tweet 所出現的每一個 hashtag，都做以下的操作。如果該 hashtag 已經出現在 tagdict 中，那麼代表我已經把它傳入 **`hashtagquery(query)`**做進一步查詢過了。所以只需要把這個 hashtag 的出現次數遞增 1 即可。但如果尚未出現過的話，就要在 tagdict 中產生這個 tag 的 key 與 VALUE（VALUE 為 1，為什麼不是 0？），並把該 tag 送入進行

hashtagquery(query, lv)擴展查詢。

```
def hashtagquery(query, lv):
    print "current tag:\t%s"%query
    if lv > 0:
        lv -= 1
        tweets = tquery.search.tweets(q=query, count = 100)
        for tweet in tweets['statuses']:
            if tweet['id'] not in tweetdict:
                hashtags = []
                for word in tweet['text'].split():
                    if word.startswith('#'):
                        hashtags.append(word)
                tweetdict[tweet['id']] = hashtags
//Build tagdict for storing the tags crossing tweets
//Expand the query to hashtags in tweets
                for tag in hashtags:
                    if tag not in tagdict:
                        tagdict[tag] = 1
                        hashtagquery(tag, lv)
                    else:
                        tagdict[tag] += 1
```

- 在主函式中執行該函式，並做 lv == 2 的查詢擴展（lv==2 的意思是向外展一層，為什麼？）。

```
if __name__=="__main__":
    reportf = open("report.txt", 'w')
    sys.stdout = Tee(sys.stdout, reportf)
    hashtagquery("#CongressOccupied", 2)
    for k in sorted(tagdict, key = tagdict.get)[:100]:
        print k, '\t', tagdict[k]
```

- 觀察以下查詢擴展的結果，和未進行查詢擴展的結果相比較，請問你覺得有什麼差別？你認為結果有比較好嗎？有的話，請指出哪些 hashtag 在查詢擴展中被找出來？沒有的話，請指出哪些 hashtag 是不太必要的？查詢擴展就邏輯上來說應該非常有效的，請問，為什麼在這個議題，會產生這樣的結果？

```
#sunflower      77
#CongressOccupied       58
#congressoccupied       31
#Sunflower      19
#Taiwan 15
#photography    13
#反服貿  12
#osdctw 12
#SunflowerMovement      8
#g0v    7
#art    7
#flowers        7
```

Building hashtag networks

- 接下來要做的是建立 Hashtag 共現網絡。觀察前面在取得文章的 Hashtag 資料時，記錄下了這個 DICTIONARY：`tweetdict[tweet['id']] = hashtags`。亦即，在 tweetdict 中的 KEY 與 VALUE 分別是每篇 tweet 的 id 以及「共存於」該文章中的 hashtags。因此，我只需要取出 VALUE 的部分，然後和 Web Of Science 的例子一樣，兩兩配對，建置網路即可。
  - 首先，先建立一個 DICTIONARY（`tagpair = {}`）以存放（#hashtag, #hashtag）: Number of Occurrence 的對應關係。其餘配對的程式碼如下。
  - 取出每一篇 Tweet 的 tags 後（在下面的 `v`），以 `x, y` 分別代表一對取自 `v` 的 tags，進行建置網絡的操作。

```
tweetdict = {}
tagdict = {}
tagpair = {}

def hashtagquery(query, lv):
  ##省略##
if __name__=="__main__":
    hashtagquery("#CongressOccupied", 2)
    for v in tweetdict.values():
        if len(v) > 1:
            for x in v:
                for y in v:
                    if x != y:
                        tagpair[x,y] = tagpair.get((x,y), 0) + 1
                        tagpair[x,y] = tagpair.get((x,y), 0) + 1
    for k in sorted(tagpair, key=tagpair.get, reverse=True):
        print k[0], '\t', k[1], '\t', tagpair[k]
    print "NUMBER OF TAGS: ", len(tagdict)
    print "NUMBER OF TWEETS: ", len(tweetdict)
    print "NUMBER OF TAGPAIR: ", len(tagpair)
```

- 之後即可把成對的標籤（也就是網絡的一個連結）及其權重用複製貼上的方法貼至 Google Spreadsheet 中，再利用 Google Fusion Table 即可進行網絡視覺化。

```
#photography     #sunflower      26
#sunflower      #photography     26
#CongressOccupied       #osdctw 24
#osdctw #CongressOccupied        24
#反服貿  #CongressOccupied        20
#CongressOccupied       #反服貿  20
#CongressOccupied       #Taiwan 18
#Taiwan #CongressOccupied        18
#art    #sunflower      14
#flowers        #sunflower      14
#sunflower      #art    14
```