IADB's Share of voice on twitter from Colombia
By Wladimir Llanos
General Assembly DC -DAT5

Introduction

The concept of "listening to the conversation" which is at the core of most social media programs is really just a pseudonym for tracking content your "customers" create based on their satisfaction, or dissatisfaction. It's often described as "share of voice" - The percentage of all online content and conversations about your company, compared to your competitors.

There are many paid tools to calculate the share of voice but the complex it and hight price is unnecessary for our organization. We want to know how many mentions do we have on twitter agains other organizations and API twitter has some good options to collect data at the moment.

Data Mining

Data Mining is an analytic process designed to explore data (usually large amounts of data - typically business or market related - also known as "big data") in search of consistent patterns and/or systematic relationships between variables, and then to validate the findings by applying the detected patterns to new subsets of data.

Tasks

I've been collecting data from April 20th to May 28th using the twitter's API. Python has a lot of options to connect and work with twitter, Tweepy was the library that I've chose.

Keyworks

The following words represent our organization and competitors:

@el_BID",'el bid','Banco Interamericano de Desarrollo','Banco Mundial','@BancoMundial','BancoMundial','el Banco Mundial','@AgendaCAF','AgendaCAF','banco de desarrollo de América Latina','CAF-Banco de Desarrollo de América Latina

The function to collect data in realtime that I used to filter twitter streaming and bring you tweets in that matched with your keywords is:

twitterStream.filter(track=[“keyworks”])

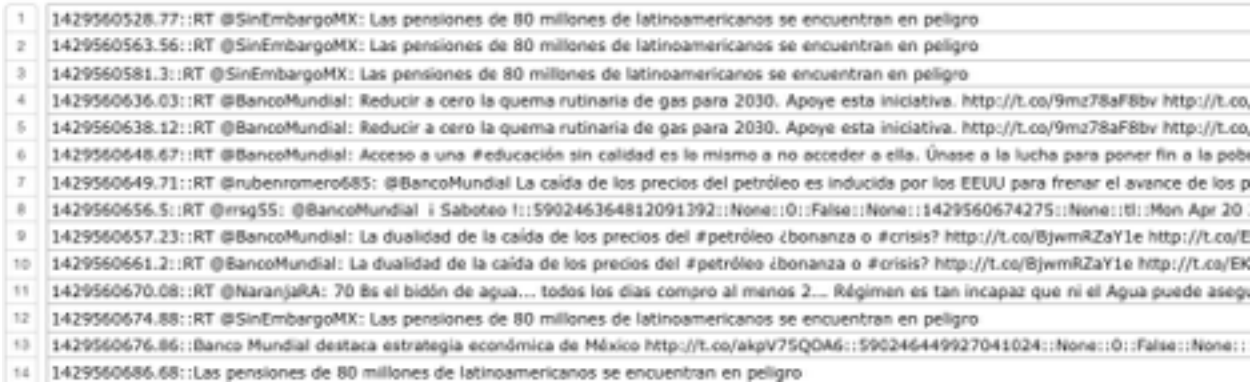
It returns an xml with all tweet data:


```
"created_at": "Fri Jan 23 23:57:36 +0000 2015",
"id": 558775589612437504,
"id_str": "558775589612437504",
"text": "Thanks, polar vortex: Attendance dips at major Chicago
"source": "\u003ca href=\"http:\/\/twitter.com\" rel=\"nofollow\
"truncated": false,
"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,
"user": {
  "id": 7313362,
  "id_str": "7313362",
  "name": "Chicago Tribune",
  "screen_name": "chicagotribune",
  "location": "Chicago, IL",
  "url": "http:\/\/www.chicagotribune.com\/",
  "description": "Chicago Tribune news, features and so much mor
  "protected": false,
  "verified": true,
  "followers_count": 321548,
  "friends_count": 523,
  "listed_count": 8074,
  "favourites_count": 34,
  "statuses_count": 47367,
  "created_at": "Sat Jul 07 14:10:07 +0000 2007",
  "utc_offset": -21600,
  "time_zone": "Central Time (US & Canada)",
  "geo_enabled": false,
  "lang": "en",
```


The next step was save the data in a csv file with 27 fields separated by “::”:

tweetText, tweetId,tweetInReplyStatusId,tweetFavoriteCount, tweetRetweeted
,tweetCoordinates,tweetTimestamp_ms,tweetGeo,tweetLang,tweetDate
,tweetUserScreen_name,tweetUserProfile_image_url_https
,tweetUserFollowers_count,tweetUserListed_count,tweetUserFollowing_count
,tweetUserLocation,tweetUserGeo_enabled,tweetUserName,tweetUserLang
,tweetUserUrl,tweetUserCreated_at,tweetUserTime_zone,tweetUserIs_translator

Example:



Data cleaning

Many lines in csv file had broken or corrupted data, I need to make a loop to check integrity and the results are below:

Row data	67,156
After cleaning	53,987

New fields:

Country

Every tweet data has informations but there is a problem with location, twitter allows the user to input text without specific format:

Bogot	**Colombia**
Bogot	
bogot	
Bogot - Colombia	
Bogot Colombia	
Bogot Colombia	
Bogot Colombia.	
Bogot D.C	
Bogot D.C - Colombia	

The best option to match the county was make a csv file with all Colombia's cities, states and counties. Regular expressions simplify the work. I've got great results

```
terms = pd.read_csv('Terms.csv', parse_dates=True)
#accent removed
termsCleaned = []
for term in terms['TERM']:
    termsCleaned.append(term.encode("ascii", "ignore").strip())

#removing common places in other countris(the following list is irrelevant for our study)
#FLORENCIA, CÓRDOBA, AMAZONAS,LA PLATA,MADRID
termsCleaned.remove('FLORENCIA')
termsCleaned.remove('CRDOBA')
termsCleaned.remove('AMAZONAS')
termsCleaned.remove('LA PLATA')
termsCleaned.remove('FLORIDA')
termsCleaned.remove('MADRID')

#mapping twData.tweetUserLocation with terms cleaned

for location in tweetUserLocation:
    flag = 0
    op1 = '''\\b'''
    op2 = '''\\b'''

    for place in termsCleaned:
        if not (re.search(r""+op1+place+op2,location,flags=re.IGNORECASE)) is None:
            tweetisColombia.append("1")
            flag = 1
            break
    if flag == 0:
        tweetisColombia.append("0")
```

Brand

To tag the brand I've defined 4 names and proceeded to match every tweet with a brand:

BID	BANCO MUNDIAL	CAF	UNDEFINED

Example:

BID

RT **@el_BID** 5 maneras de usar la #tecnología para mejorar la #movilidad urbana http://t.co/UXvUw703N6 @BID_Ciudades http://t.co/VZB1bvpAEi

RT @EFEnoticias: **El BID** aconseja reformas para salvar la pensión de 80 millones de personas en A.Latina http://t.co/NKORj8qlzD

BANCO MUNDIAL

Banco Mundial destaca estrategia económica de México http://t.co/zVPds2xkUu:: 590246511759470592

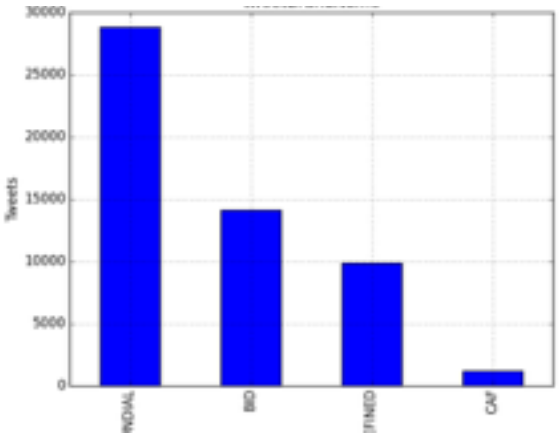
@BancoMundial Las razones en la caída de los precios del crudo son diversas

I've used regular expression again:

re.search(r"**\b**@el_BID**\b**",row.split('::')[1],flags=re.IGNORECASE)

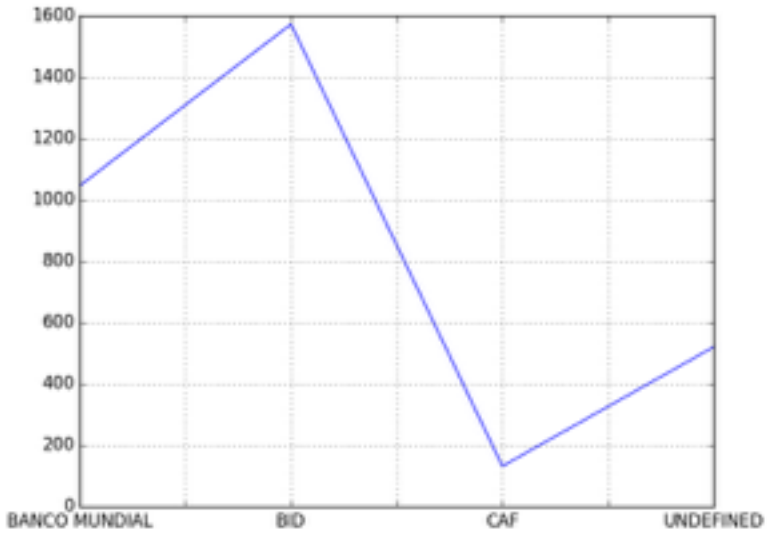
Data exploration

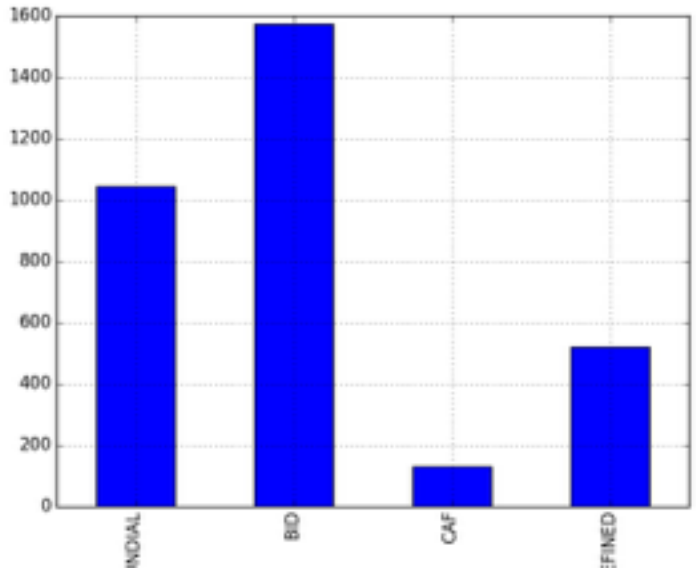
Not filtered by location



		TWEETS	**%**
1	**BANCO MUNDIAL**	28797	53
2	**BID**	14153	26
3	**CAF**	1190	2
	UNDEFINED	9847	18
	TOTAL:	53987	100

By country





		TWEETS	%
1	**BID**	1572	48
2	**BANCO MUNDIAL**	1045	32
3	**CAF**	131	4
	UNDEFINED	520	16
	TOTAL:	3268	100

Conclusion

Collecting data with python is the best and faster option. Performance and usability is easier than other languages like php, c#, c++.
The challenge is to build the data-frame. Once that is complete you only have to analyze the data, make connections and write the code to look for matches.