INDRAPRASTHA INSTITUTE OF INFORMATION TECHNOLOGY, DELHI

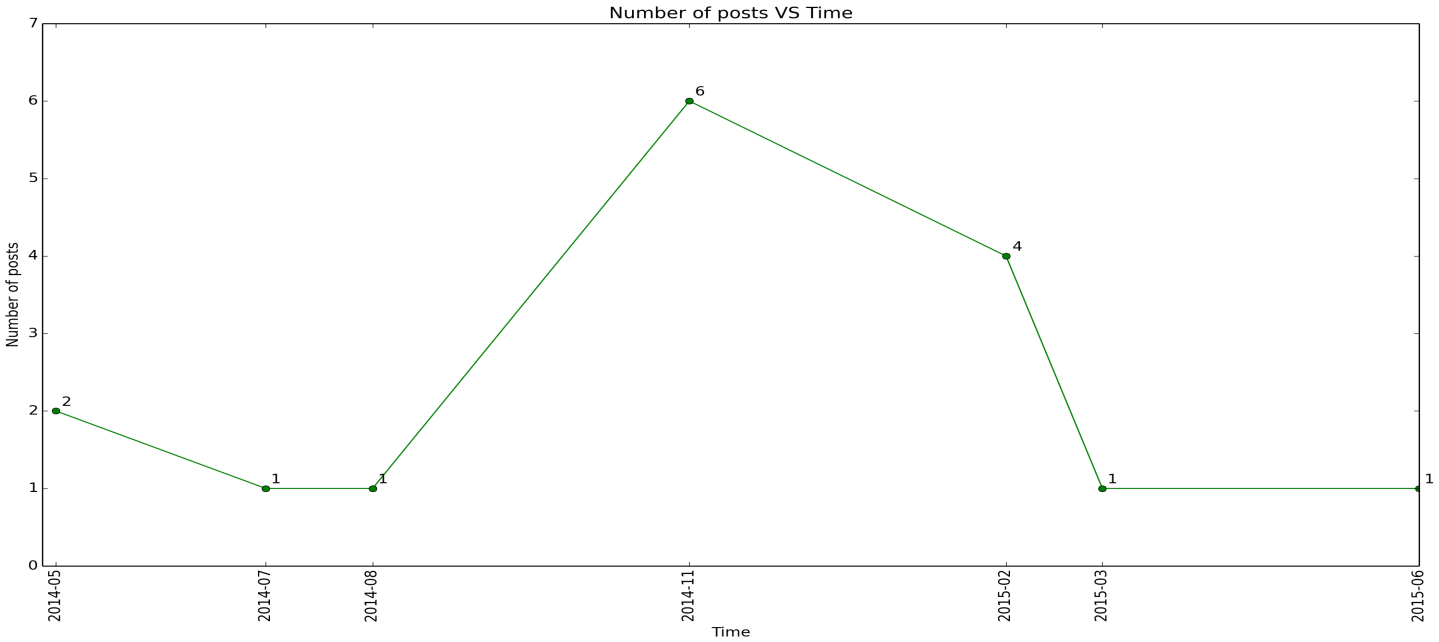
Homework 1 : Twitter and FB data analysis

Sagar Verma

August 24, 2015

0.1 CERCatIIITD #posts vs time

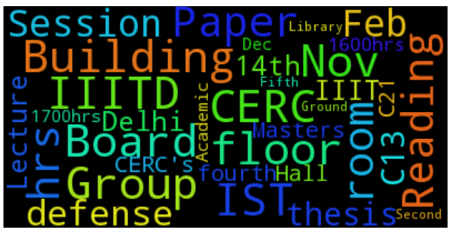
Below graph shows Number of Posts for each month from creating of page till the date data was collected. 21 posts were obtained from CERCatIIITD page which contain What, Where and When in thier description.



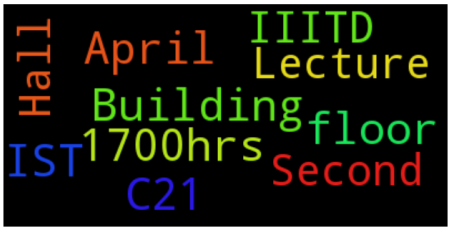
0.2 CERCatIIITD word cloud

Now, for the given graph we create a word cloud for the month page was created, for months which the number of posts are high and for the end of the month. Here they are "2014-05","2014-11","2015-03","2015-06".

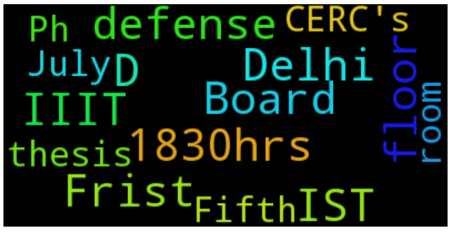
CERCatIIITD Words used during May, 2014



CERCatIIITD Words used during Nov, 2014



CERCatIIITD Words used during March, 2015

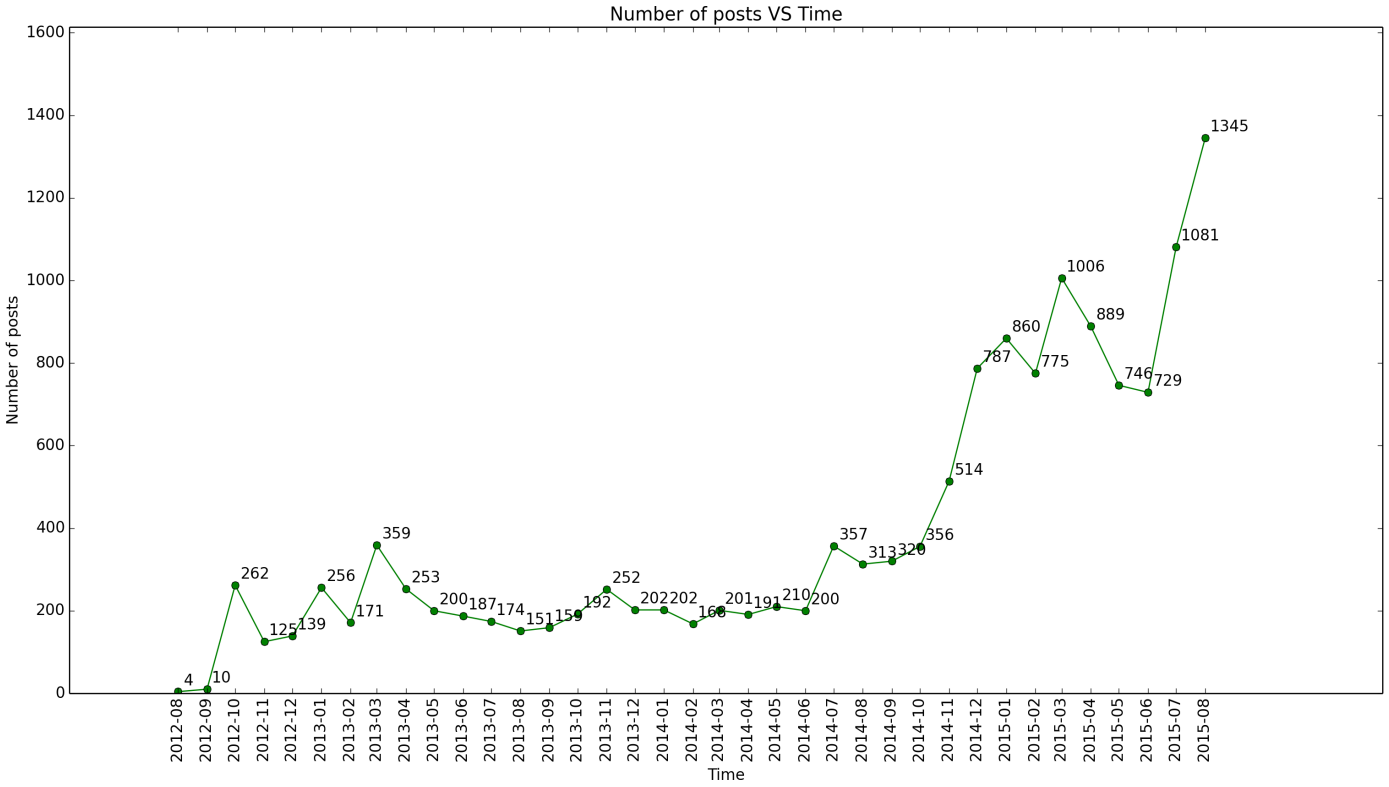


CERCatIIITD Words used during June, 2015

From word cloud we see that the page is talking mostly about CERC building, reading, thesis defense and symposium.

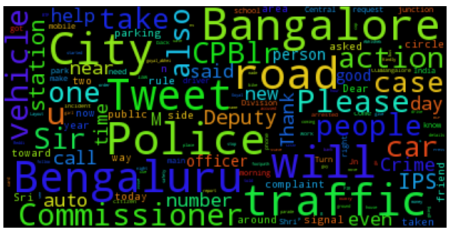
0.3 blrcitypolice #posts vs time

For Banglore city police facebook page, we draw word clouds for"2012-08","2015-01","2015-03","2015-08". 14348 posts were extracted from blrcitypolice page.



0.4 blrcitypolicei word cloud

blrcitypolice Words used during August, 2012



blrcitypolice Words used during Janurary, 2014

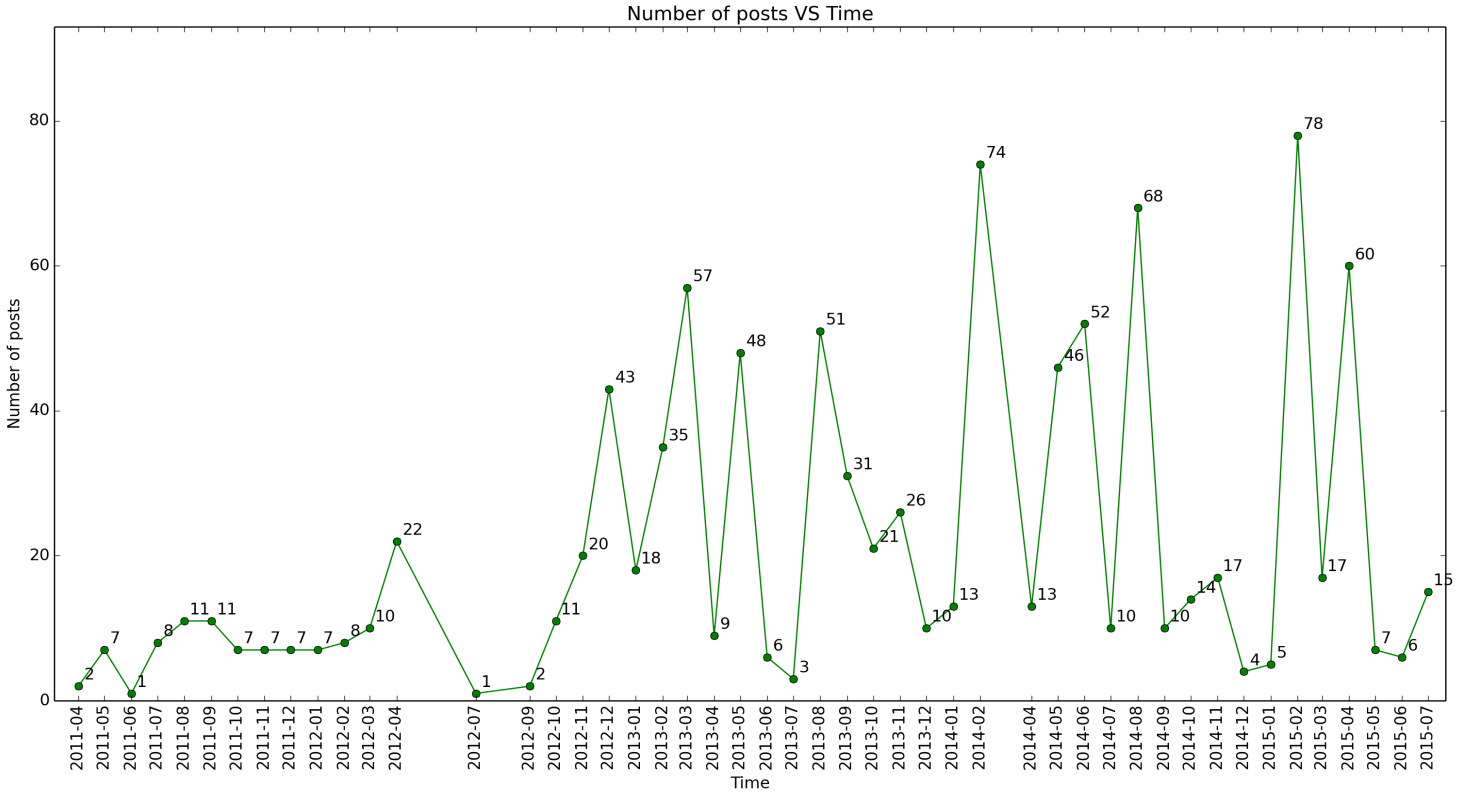
blrcitypolice Words used during March, 2015

blrcitypolice Words used during August, 2015

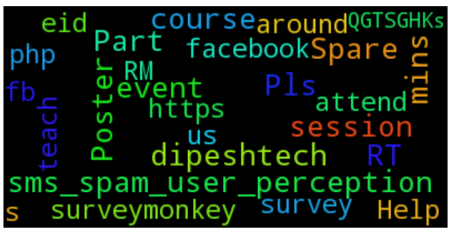
Most posts are about Bangalore road safety and traffics.
Most posts are by random people asking Commisoner of Police regarding road safety and other law and order related tasks.A post of real time crime.
On careful analysis of each date by manuaaly reading the twitter data of March-August 2015, the people are talking about IAS DK Ravi's case.

0.5 ponguru #posts vs time

For ponguru tweeter handle, we draw the word cloud for spikes present in the time series graph, i.e., "2011-04","2013-03","2014-02","2015-07". 1017 records were found for ponguru twitter handle.

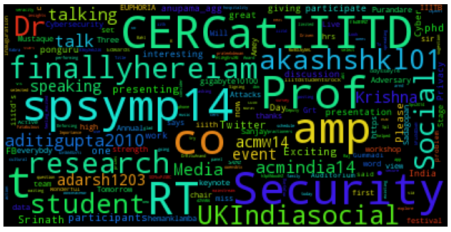


0.6 ponguru word cloud

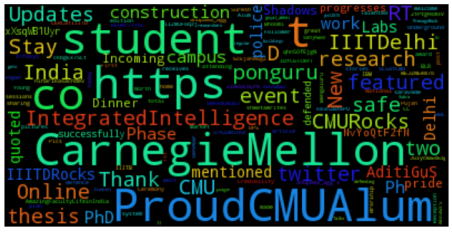


ponguru Words used during April, 2011

ponguru Words used during March, 2013



ponguru Words used during Feburary, 2014

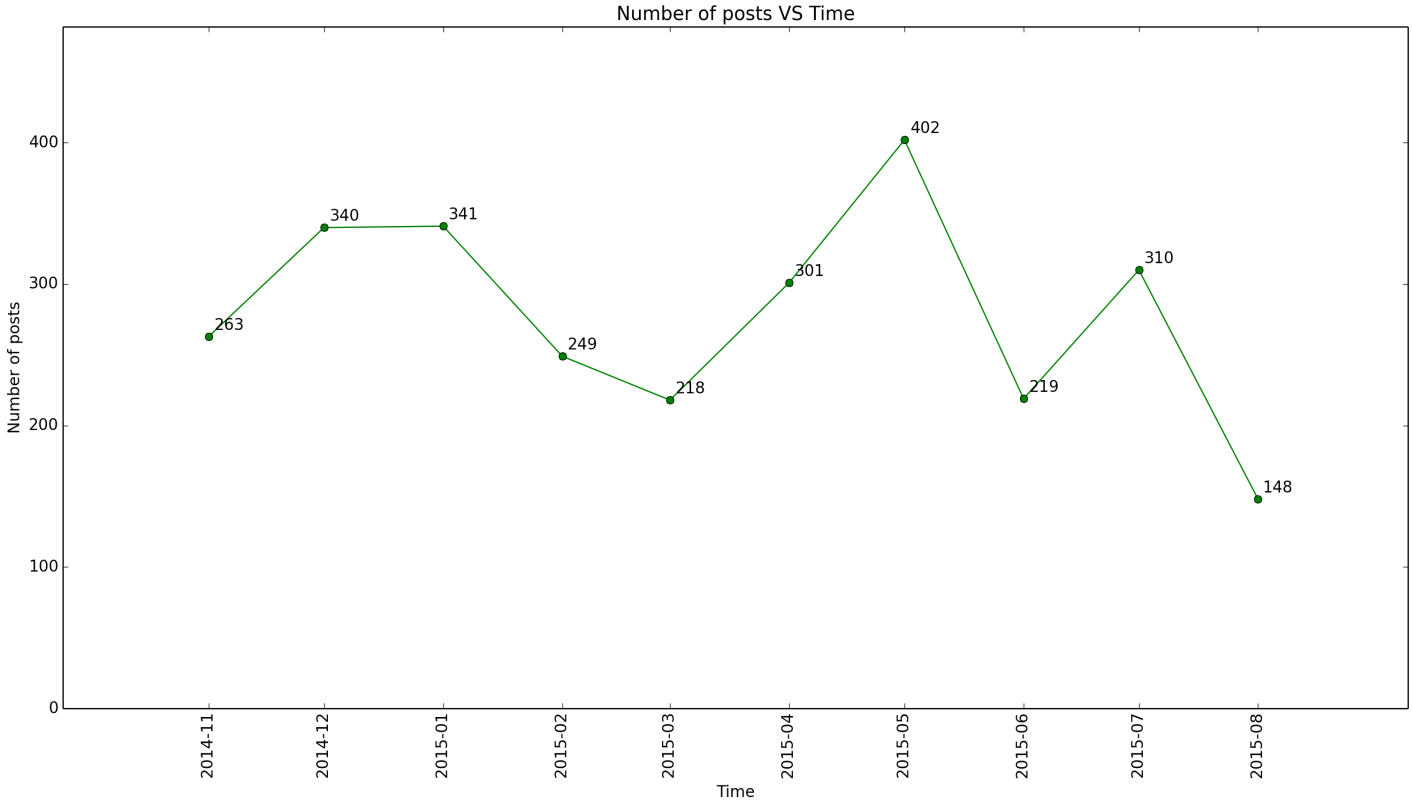


ponguru Words used during July, 2015

Tweets are about Security simposium 2013,2014 and 2015.

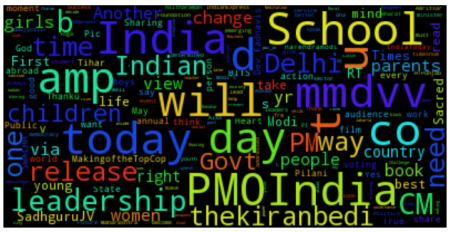
0.7 thekiranbedi #posts vs time

For thekiranbedi twitter handle, we make the word clouds for month ID was created, month data was collected and two months in between were there were maximum #posts, i.e. "2014-11","2015-05","2015-07","2015-08". 3247 tweets were extracted from thekiranbedi twitter handle.
Data limit for Facebook is 320 per request and one can get all posts of an ID and for Twitter it is arround 3300 per ID.

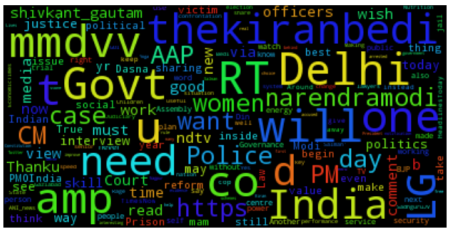


The graph for thekiranbedi handle is much denser then ponguru and CERCatIIITD, this shows that thekiranbedi's user is more active. blrcitypolice has maximum number of posts and about 80% of those are posts by page followers.

0.8 thekiranbedi word cloud



thekiranbedi Words used during November, 2014



thekiranbedi Words used during May, 2015

thekiranbedi Words used during July, 2015

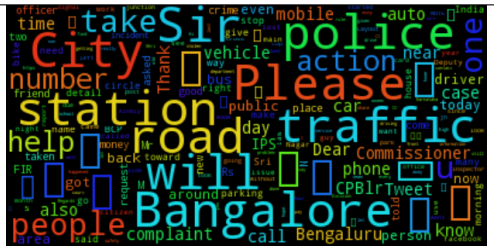
thekiranbedi Words used during August, 2015

Tweets are realated to AAP, BJP, PMO India and Women and Child welfare.

0.9 Word clouds of all the IDs showing frequent words used by IDs

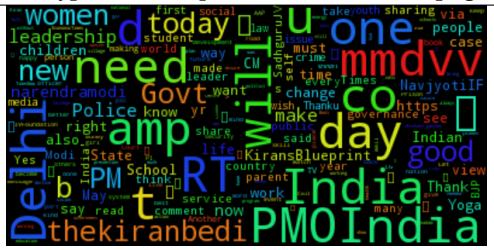
Word cloud showing what each ID's have talked about most from the time ID was created till last time data was collected.

CERCatIIITD : Frequent words in whole page.



blrcitypolice : Frequent words in whole page.

ponguru : Frequent words in all the tweets.

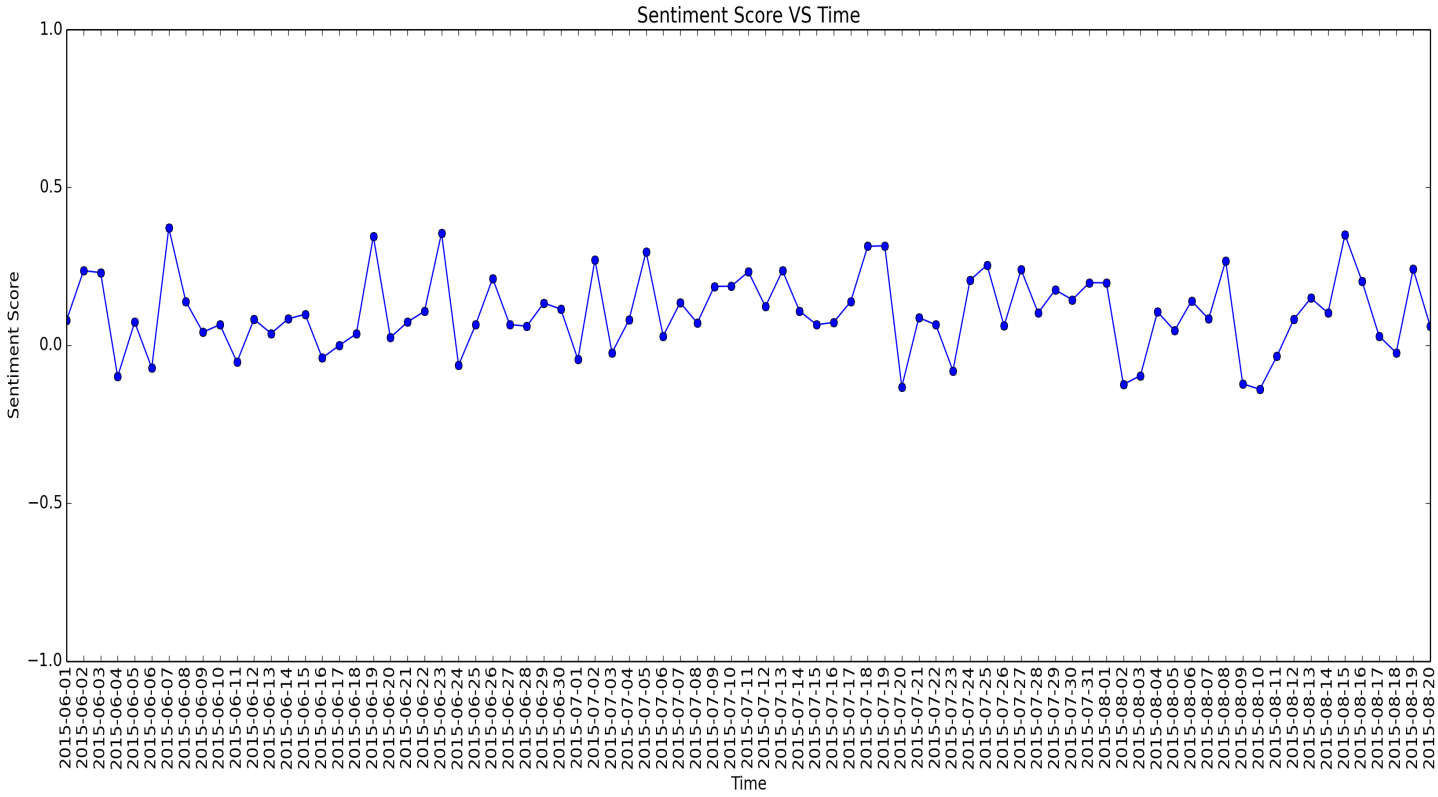


thekiranbedi : Frequent word in all the tweets.

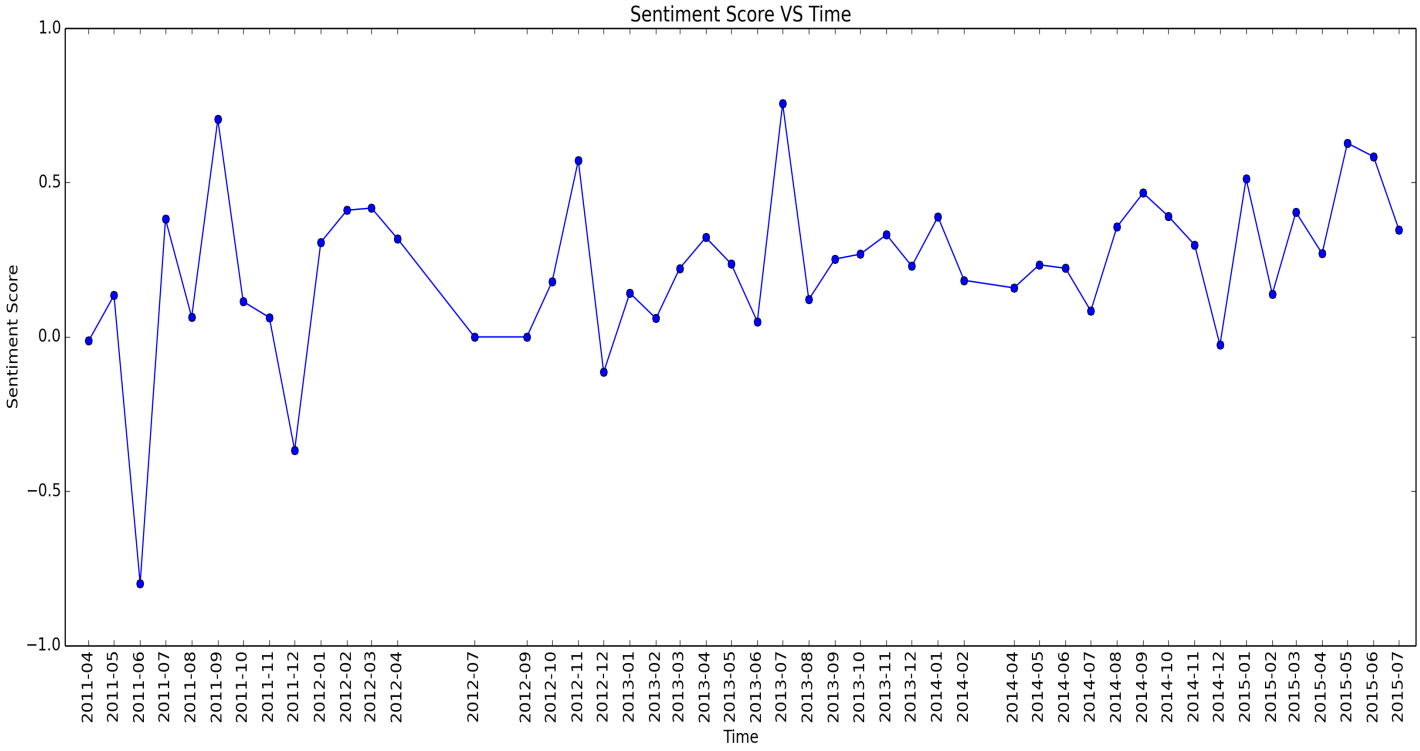
0.10 Sentiment scores vs time for all IDs

The following graphs show sentiments of IDs with time. Since API provides very less data daily only few data was collected. If we carefully analyse the tweets for which the sentiment score is very high -ve or +ve we can find the reason behind the score.

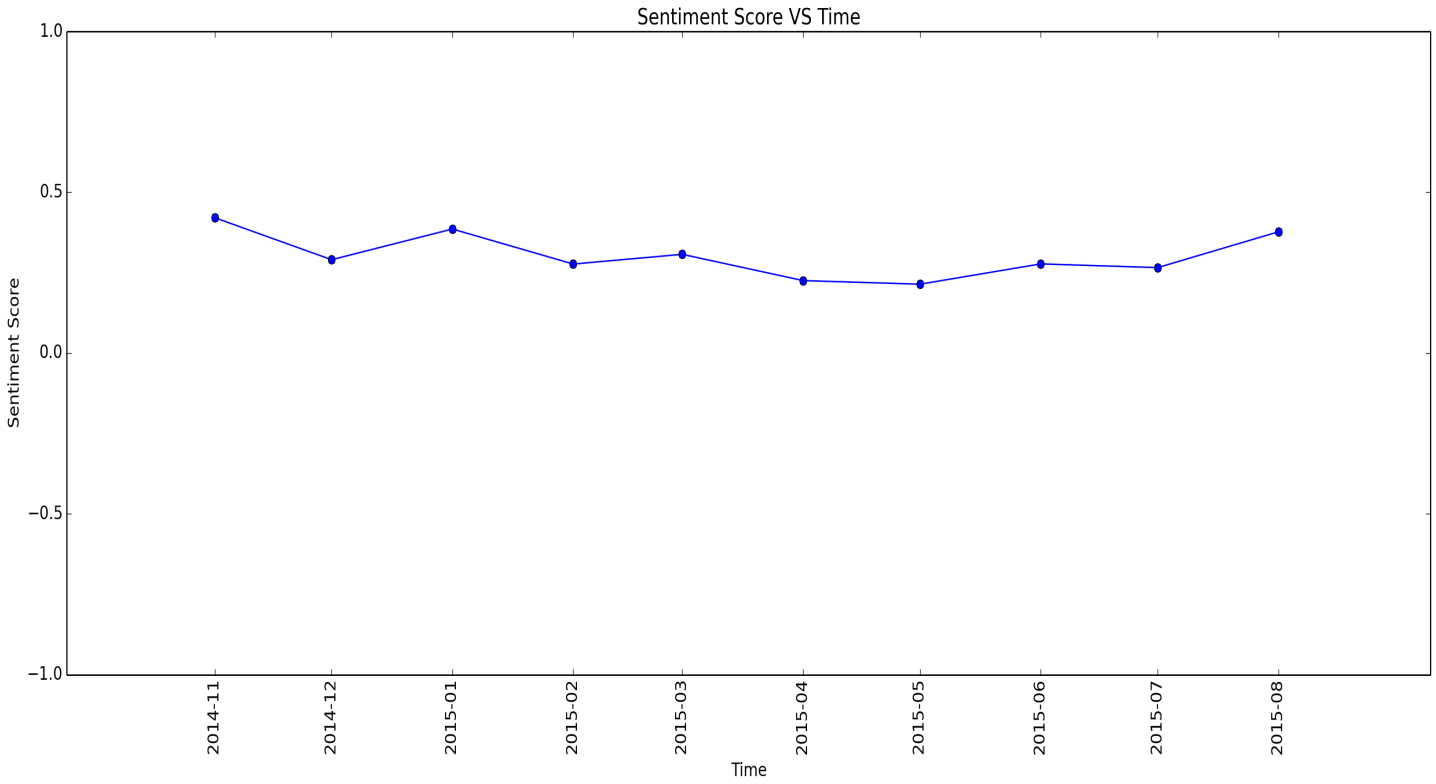
blrcitypolice(Banglore City Police)



ponguru(PK)

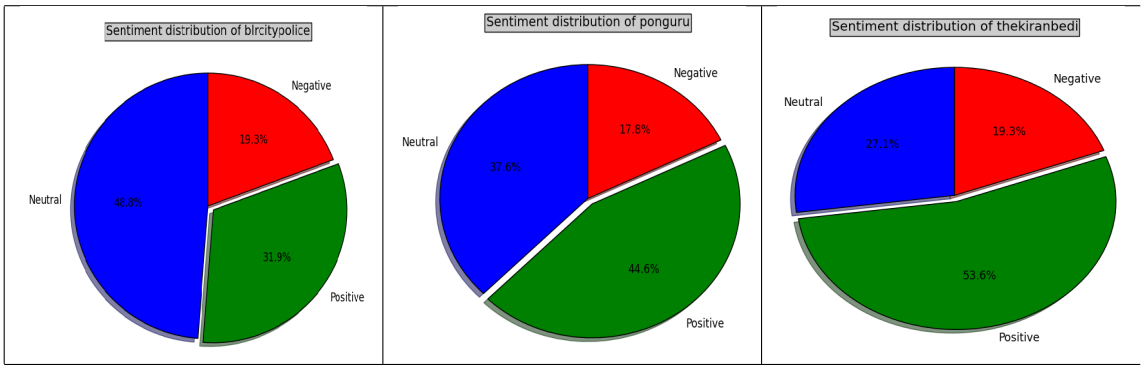

Sentiment Score VS Time
Sentiment Score
1.0
0.5
0.0
−0.5
−1.0
2011-04
2011-05
2011-06
2011-07
2011-08
2011-09
2011-10
2011-11
2011-12
2012-01
2012-02
2012-03
2012-04
2012-07
2012-09
2012-10
2012-11
2012-12
2013-01
2013-02
2013-03
2013-04
2013-05
2013-06
2013-07
2013-08
2013-09
2013-10
2013-11
2013-12
2014-01
2014-02
2014-04
2014-05
2014-06
2014-07
2014-08
2014-09
2014-10
2014-11
2014-12
2015-01
2015-02
2015-03
2015-04
2015-05
2015-06
2015-07
Time

thekiranbedi(Kiran Bedi)



0.11 Sentiment composition of tweets/posts of ID's

Pie charts showing total sentiment composition of each ID's.



0.12 Python program to get data from facebook for CERCatIIITD page. This program parses description in to What, Where and When components.

```
import requests
import re
from pymongo import MongoClient
import csv


def remove_chars_re(subj, chars):
    return re.sub('[' + re.escape(''.join(chars)) + ']', '', subj)


client = MongoClient()
#client.database_names() shows databases names

cerc_db = client['cerc_db']
cerc_colle = cerc_db['cerc_collection']
posts = cerc_db['posts']


fin = open("fb-data.list", "r")
fout = open("five-w.csv", "w")

f = open('cercwww.csv','wb')
writer = csv.writer(f)
writer.writerow(["created_time","what","when","where"])

t = 1
while t:
    url = fin.readline()[:-1]
    ACCESS_TOKEN = fin.readline()[:-1]
    VERSION = fin.readline()[:-1]
    LIMIT = fin.readline()[:-1]
    ID = fin.readline()[:-1]
    fin.readline()[:-1]

    out = ""

    tot_url = url+VERSION+"/"+ID+"?access_token="+ACCESS_TOKEN+"&limit="+LIMIT

    #print tot_url

    response = requests.get(tot_url)

    feeds = response.json()
    print feeds
    count = 0
```

```
while 'data' in feeds and len(feeds['data']) != 0:
    feeds_in_this_page = len(feeds['data'])
    for i in range(feeds_in_this_page):
        if 'description' in feeds['data'][i]:
            description = feeds['data'][i]['description']
            description = description.encode('ascii', 'ignore')
            name = feeds['data'][i]['name'].encode('ascii', 'ignore')

            what_pos = description.find("WHAT:")
            what_ends = description[what_pos + 5:].find(":")
            what_cont = description[what_pos:what_pos + 5 +what_ends -
                (description[what_pos:what_pos + 5
                +what_ends])[::-1].find("\n")]
            what_cont = remove_chars_re(what_cont, ['\n', '+'])

            when_pos = description.find("WHEN:")
            when_ends = description[when_pos + 5:].find(":")
            when_cont = description[when_pos:when_pos + 5 +when_ends -
                (description[when_pos:when_pos + 5
                +when_ends])[::-1].find("\n")]
            when_cont = remove_chars_re(when_cont, ['\n', '+'])

            where_pos = description.find("WHERE:")
            where_ends = description[where_pos + 6:].find(":")
            where_cont = description[where_pos:where_pos + 6 +where_ends -
                (description[where_pos:where_pos + 6
                +where_ends])[::-1].find("\n")]
            where_cont = remove_chars_re(where_cont, ['\n', '+'])

            if len(what_cont) > 1 or len(where_cont) > 1 or len(when_cont) >
                1 or name:
                #print
                    feeds['data'][i]['id'],"\n",feeds['data'][i]['created_time'],"\n",what_cont
                #post_id =
                    posts.insert_one({'feed_id':feeds['data'][i]['id'],'created_time':feeds['da
                #print post_id
                writer.writerow([feeds['data'][i]['created_time'],what_cont.replace(",","")[6:]
                count += 1
            #break


    tot_url = feeds['paging']['next']
    response = requests.get(tot_url)
    feeds = response.json()
    #break
print count
#print feeds
break
t -= 1
```

0.13 Python program to get data from facebook for theblrcitypolice page. Store time, id, message and description in to mongo db for latter use.

```
import requests
import re
from pymongo import MongoClient


def remove_chars_re(subj, chars):
    return re.sub('[' + re.escape(''.join(chars)) + ']', '', subj)


client = MongoClient()
#client.database_names() shows databases names

blrcitypolice_db = client['blrcitypolice_db']
blrcitypolice_colle = blrcitypolice_db['blrcitypolice_collection']
posts = blrcitypolice_db['posts']


fin = open("fb-data.list", "r")
fout = open("five-w.csv", "w")


url = fin.readline()[:-1]
ACCESS_TOKEN = fin.readline()[:-1]
VERSION = fin.readline()[:-1]
LIMIT = fin.readline()
ID = fin.readline()[:-1]

out = ""

tot_url = url+VERSION+"/"+ID+"?access_token="+ACCESS_TOKEN+"&limit="+LIMIT

#print tot_url

response = requests.get(tot_url)

feeds = response.json()
#print feeds
count = 0
while 'data' in feeds and len(feeds['data']) != 0:
    feeds_in_this_page = len(feeds['data'])
    for i in range(feeds_in_this_page):
            id = ""
            message = ""
            created_time = ""
            from_name = ""
```

```
            to_name = ""
            description = ""

            id = feeds['data'][i]['id']
            if 'message' in feeds['data'][i]:
                message = feeds['data'][i]['message']
            if 'description' in feeds['data'][i]:
                description = feeds['data'][i]['description']
            created_time = feeds['data'][i]['created_time']
            from_name = feeds['data'][i]['from']['name']
            if 'to' in feeds['data'][i]:
                to_name = feeds['data'][i]['to']['data'][0]['name']

            print id

            post_id = posts.insert_one({'post_id':
                id,'message':message,'created_time':created_time,'from_name':from_name,'to_name':t
            print post_id
            count += 1

    tot_url = feeds['paging']['next']
    response = requests.get(tot_url)
    feeds = response.json()

print count
#print feeds
```

0.14 A list file to be read by above two programs to get parameters for facebook graph API.

```
https://graph.facebook.com/
CAACEdEose0cBANFhi44OULtO5kajiNKRK3OrjaLGNGnwdIegJkV4ObNGnVdr3GaaTdDkF9MOa8NNnXyZBdZBJuLDOYeNrVfU
v2.0
318
CERCatIIITD/feed
blrcitypolice/feed
```

0.15 Python program to get data from twitter using Tweepy python library and twitter API and store it in mongo DB.

```
import tweepy
import csv
import json
from pymongo import MongoClient

client = MongoClient()


#Twitter API credentials
consumer_key = "IIJVagCrIuipiZQRNxiX8okOO"
consumer_secret = "TwigYDzho9YGUOlGIPvNQJXhZt0aY5r33YbaA6ijifPKNOJvPr"
access_key = "1298538012-Td1042ReUAagHzSDDPlCa9eOpbJmwc6uB3BmYOd"
access_secret = "qufnJEjTDgXTicuyl3vsAbipHYAoDYUhJOKtmrr9p6w5i"


def get_all_tweets(screen_name):
    t_db = client[screen_name+'_db']
    t_colle = t_db[screen_name+'_collection']
    posts = t_db['posts']
    #Twitter only allows access to a users most recent 3240 tweets with this
        method

    #authorize twitter, initialize tweepy
    auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_key, access_secret)
    api = tweepy.API(auth)

    #initialize a list to hold all the tweepy Tweets
    alltweets = []

    #make initial request for most recent tweets (200 is the maximum allowed
        count)
    new_tweets = api.user_timeline(screen_name = screen_name,count=200)

    #save most recent tweets
    alltweets.extend(new_tweets)

    #save the id of the oldest tweet less one
    oldest = alltweets[-1].id - 1

    #keep grabbing tweets until there are no tweets left to grab

    while len(new_tweets) > 0:
        print "getting tweets before %s" % (oldest)

        #all subsiquent requests use the max_id param to prevent duplicates
```

```
        new_tweets = api.user_timeline(screen_name =
            screen_name,count=200,max_id=oldest)

        #save most recent tweets
        alltweets.extend(new_tweets)

        #update the id of the oldest tweet less one
        oldest = alltweets[-1].id - 1

        print "...%s tweets downloaded so far" % (len(alltweets))

    #transform the tweepy tweets into a 2D array that will populate the csv

    #outtweets = [[tweet.id_str, tweet.created_at, tweet.text.encode("utf-8"),
        tweet.coordinates, tweet.geo, tweet.place] for tweet in alltweets]

    for tweet in alltweets:
        post_id =
            posts.insert_one({'id_str':tweet.id_str,'created_at':tweet.created_at,
            'text':tweet.text.encode("utf-8"), 'coordinates':tweet.coordinates,
            'geo':tweet.geo})
        print post_id
    #write the csv
    """
    with open('%s_tweets.csv' % screen_name, 'wb') as f:
        writer = csv.writer(f)
        writer.writerow(["id","created_at","text"])
        writer.writerows(outtweets)
    """
    pass


if __name__ == '__main__':
    #pass in the username of the account you want to download
    get_all_tweets("ponguru")
    get_all_tweets("thekiranbedi")
```

0.16 Python program to generate graph of #posts with time for all four IDs.

```
from pymongo import MongoClient
import datetime as DT
from matplotlib import pyplot as plt
from matplotlib.dates import date2num
from operator import itemgetter

def timeAndPosts(db_name, val, format):
    client = MongoClient()
    db = client[db_name + '_db']
    collection = db[db_name + '_collection']
    posts = db['posts']

    feeds = []
    date_freq = {}
    for post in posts.find():
        dt = None
        if 'created_time' in post:
            dt = DT.datetime.strptime(post['created_time'][:val], format)
            feeds.append(dt)

            if dt not in date_freq:
                date_freq[dt] = 1
            else:
                date_freq[dt] += 1

        if 'created_at' in post:
            dt =
                DT.datetime.strptime((str(post['created_at'].year)+"-"+str(post['created_at'].mont
                format)
            feeds.append(dt)
            if dt not in date_freq:
                date_freq[dt] = 1
            else:
                date_freq[dt] += 1


    od = sorted(date_freq.iteritems(), key=itemgetter(0))[2:]
    x = [date2num(date) for (date, no_of_posts) in od]
    y = [no_of_posts for (date, no_of_posts) in od]

    print od
    fig = plt.figure()
    fig.set_size_inches(18.5,11)

    graph = fig.add_subplot(110)
```

```
    # Plot the data as a red line with round markers
    graph.plot(x,y,'g-o')
    plt.gcf().subplots_adjust(bottom=0.25)
    plt.xticks(rotation='vertical')
    plt.ylabel('Number of posts')
    plt.xlabel('Time')
    plt.title('Number of posts VS Time')
    for i,j in zip(x,y):
        graph.annotate('%s' %j, xy=(i,j), xytext=(5,5), textcoords='offset
            points')
        #graph.annotate('(%s,' %i, xy=(i,j))
    axes = plt.gca()
    axes.set_ylim([0,max(y)+max(y)/5])
    # Set the xtick locations to correspond to just the dates you entered.
    graph.set_xticks(x)

    # Set the xtick labels to correspond to just the dates you entered.
    graph.set_xticklabels([date.strftime(format) for (date, value) in od])
    #plt.grid(True)
    plt.savefig(db_name+"_year.png",format="png", dpi=200)

timeAndPosts('cerc',7,"%Y-%m")
timeAndPosts('blrcitypolice',7,'%Y-%m')
timeAndPosts('ponguru',7,'%Y-%m')
timeAndPosts('thekiranbedi',7,'%Y-%m')
```

0.17 Python program to generate word clouds for the spikes and start month and end month of all the time series graph.

```
from pymongo import MongoClient
import matplotlib.pyplot as plt
from wordcloud import WordCloud
import datetime as DT


def word_cloud(db_name, dates):
    client = MongoClient()
    db = client[db_name + '_db']
    collection = db[db_name + '_collection']
    posts = db['posts']

    count = 0
    for date in dates:
        all_tweets = ""
        for post in posts.find():
            message = ""
            #time = ""
            #id = ""

            dt = ""
            if 'created_time' in post:
                dt = post['created_time'][:7]
            elif 'created_at' in post:
                dt = str(post['created_at'].year)+"-"
                temp = str(post['created_at'].month)
                if len(temp) == 1:
                    dt += "0"+temp
                else:
                    dt += temp
                #print dt
            if dt == date:
                if db_name == "blrcitypolice":
                    if 'message' in post:
                        message = post['message'].encode("ascii","ignore")
                    if 'description' in post:
                        message += " " +
                            post['description'].encode("ascii","ignore")

                elif db_name == "cerc":
                    if 'what' in post:
                        message += post['what'].encode("ascii","ignore")
                    if 'where' in post:
                        message += " " + post['where'].encode("ascii","ignore")
                    if 'when' in post:
                        message += " " + post['when'].encode("ascii","ignore")
```

```
            else:
                message = post['text']

            all_tweets += message + "\n"

        all_tweets = all_tweets.encode("ascii","ignore")

        fout = open(db_name+"_"+date+".result","w")
        fout.write(all_tweets)
        fout.close()
        # take relative word frequencies into account, lower max_font_size
        wordcloud = WordCloud(max_font_size=40).generate(all_tweets)
        plt.figure()
        plt.imshow(wordcloud)
        plt.axis("off")
        plt.savefig(db_name+date+"_word_cloud_rel.png",format="png",
            dpi=100,bbox_inches='tight')


word_cloud("cerc",["2014-05","2014-11","2015-03","2015-06"])
word_cloud("blrcitypolice",["2012-08","2015-01","2015-03","2015-08"])
word_cloud("ponguru",["2011-04","2013-03","2014-02","2015-07"])
word_cloud("thekiranbedi",["2014-11","2015-05","2015-07","2015-08"])
```

0.18 Python program to generate word cloud for all the posts/tweets for all the IDs to get most frequent used words.

```
from pymongo import MongoClient
import csv
import matplotlib.pyplot as plt
from wordcloud import WordCloud


def word_cloud(db_name):
    client = MongoClient()
    db = client[db_name + '_db']
    collection = db[db_name + '_collection']
    posts = db['posts']

    all_tweets = ""
    count = 0
    for post in posts.find():
        message = ""
        #time = ""
        #id = ""

        if db_name == "blrcitypolice":
            if 'message' in post:
                message = post['message']
            if 'description' in post:
                message += " " + post['description']
            #time = post['created_time']
            #id = post['post_id']

        elif db_name == "cerc":
            if 'what' in post:
                message += post['what']
            if 'where' in post:
                message += " " + post['where']
            if 'when' in post:
                message += " " + post['when']

        else:
            message = post['text']
            #time = post['created_at']
            #id = post['id_str']

        all_tweets += message + "\n"

    #tags = make_tags(get_tag_counts(all_tweets), maxsize=120)
    print len(all_tweets)
    #create_tag_image(tags, db_name + '_cloud_large.png', size=(900, 600),
        fontname='Lobster')
```

```
    wordcloud = WordCloud().generate(all_tweets)
    # Open a plot of the generated image.
    plt.imshow(wordcloud)
    plt.axis("off")
    plt.show()

    # take relative word frequencies into account, lower max_font_size
    wordcloud = WordCloud(max_font_size=40).generate(all_tweets)
    plt.figure()
    plt.imshow(wordcloud)
    plt.axis("off")
    plt.savefig(db_name+"_word_cloud_rel.png",format="png", dpi=100)


word_cloud("cerc")
#word_cloud("blrcitypolice")
#word_cloud("ponguru")
#word_cloud("thekiranbedi")
```

0.19 Python program to get sentiment scores of posts/tweets in during a certain time period using Machpee community sentiment API and generate a CSV file.

```
from pymongo import MongoClient
import unirest
import csv


def sentiment_scores(db_name):
    client = MongoClient()
    db = client[db_name + '_db']
    collection = db[db_name + '_collection']
    posts = db['posts']

    senti_list = []
    count = 0
    f = open('%s_sentiments.csv' % db_name, 'wb')
    writer = csv.writer(f)
    writer.writerow(["id","time","sentiment","score"])
    for post in posts.find():
        message = ""
        time = ""
        id = ""

        if db_name == "blrcitypolice":
            if 'message' in post:
                message = post['message']
            elif 'description' in post:
                message = post['description']
            time = post['created_time']
            id = post['post_id']

        else:
            message = post['text']
            time = post['created_at']
            id = post['id_str']

        response =
            unirest.post("https://community-sentiment.p.mashape.com/text/",
            headers={
                "X-Mashape-Key":
                    "gHgfU3fRkJmshJIkZHwr69Z9YLYIp1gcCE9jsnN6Y9fYrjk59R",
                "Content-Type": "application/x-www-form-urlencoded",
                "Accept": "application/json"
                },
            params={
                "txt": message
                }
```

```
        )
        if response.code == 200:
            count += 1
            print count
            #print response.code
            #senti_list.append([id,time,response.body['result']['sentiment'],response.body['result
            writer.writerow([id,time,response.body['result']['sentiment'],response.body['result']['


    print count
    return senti_list

print sentiment_scores('thekiranbedi')[:10]
```

0.20 Python program to draw sentiment score for each day/month for all the IDs.

```
import csv
import datetime as DT
import matplotlib.pyplot as plt
from operator import itemgetter
from matplotlib.dates import date2num
import numpy as np

def sign(senti):
    if senti == "Neutral":
        return 0
    if senti == "Negative":
        return -1
    if senti == "Positive":
        return 1

def senti_time_series(csv_name, val, format):
    f = open(csv_name + '_sentiments.csv','rb')
    reader = csv.reader(f)

    senti_list = list(reader)[1:]

    i = 0


    date_freq = {}

    for lst in senti_list:
        #print lst[1]
        dt = DT.datetime.strptime(lst[1][:val], format)

        #if i == 30:
        if dt not in date_freq:
            date_freq[dt] = [sign(lst[2])*float(lst[3])/100.0,1]
            #print dt
        else:
            #print dt
            date_freq[dt][0] += sign(lst[2])*float(lst[3])/100.0
            date_freq[dt][1] += 1
        i = 0
        #elif i < 30:
        i += 1


    od = sorted(date_freq.iteritems(), key=itemgetter(0))[:]
    x = [date2num(date) for (date, pair) in od]
    y = [pair[0]/pair[1] for (date, pair) in od]
```

```
    xrng = np.arange(0.0, 1.1, 0.01)

    fig = plt.figure(1)
    fig.set_size_inches(20,20)

    graph = fig.add_subplot(211)

    # Plot the data as a red line with round markers
    graph.plot(x,y,'b-o')
    plt.gcf().subplots_adjust(bottom=0.25)
    plt.xticks(rotation='vertical')
    plt.ylabel('Sentiment Score')
    plt.xlabel('Time')
    plt.title('Sentiment Score VS Time')
    #for i,j in zip(x,y):
        #graph.annotate('%s' %j, xy=(i,j), xytext=(5,5), textcoords='offset
            points')
        #graph.annotate('(%s,' %i, xy=(i,j))
    axes = plt.gca()
    axes.set_ylim([-1.0,1.0])
    # Set the xtick locations to correspond to just the dates you entered.
    graph.set_xticks(x)

    # Set the xtick labels to correspond to just the dates you entered.
    graph.set_xticklabels([date.strftime(format) for (date, value) in od])
    #plt.grid(True)
    plt.savefig(csv_name+"_senti_time.png",format="png", dpi=200)

#senti_time_series('blrcitypolice',10,'%Y-%m-%d')
#senti_time_series('ponguru',10,'%Y-%m-%d')
senti_time_series('thekiranbedi',7,'%Y-%m')
```

0.21 Python program to generate pie chart for sentiment composition of all the IDs.

```
import csv
import matplotlib.pyplot as plt

def senti_pie(csv_name):
    f = open(csv_name + '_sentiments.csv','rb')
    reader = csv.reader(f)

    senti_list = list(reader)

    tot = len(senti_list)
    neutrals = 0
    positives = 0
    negatives = 0

    for lst in senti_list:
        if lst[2] == 'Neutral':
            neutrals += 1
        if lst[2] == 'Positive':
            positives += 1
        if lst[2] == 'Negative':
            negatives += 1

    labels = 'Neutral','Positive','Negative'
    sizes = [neutrals/(tot * 1.0) * 100, positives/(tot * 1.0) * 100,
        negatives/(tot * 1.0) * 100]
    explode=(0,0.05,0)

    plt.pie(sizes, explode=explode, labels=labels,
        autopct='%1.1f%%', shadow=True, startangle=90)
    plt.title('Sentiment distribution of ' + csv_name, bbox={'facecolor':'0.8',
        'pad':5})

    #show()
    plt.savefig(csv_name+"_sentiment_dist.png",format="png", dpi=100)

#senti_pie('ponguru')
#senti_pie('blrcitypolice')
senti_pie('thekiranbedi')
```