1. """
2. Use Twitter API and live tweet stream for writing tweets into csv file
3. """
4. # !usr/bin/python
5. import tweepy
6. import csv
7. #### Twitter Authentication ###
8. auth=tweepy.auth.OauthHandler('Consumer Key','Consumer Secret')
9. auth.set\_access\_token('Access Token','Access Token Secret')
10. api=tweepy.API(auth)
11. ### Open/Create a file to append data ###
12. csvFile=open('result.csv','a') ## result.csv --> path of destination csv file
13. csvWriter=csv.writer(csvFile)
14. ### Writing header row ###
15. csvWriter.writerow(['id' ,'id\_str' ,'created\_at' ,'text' ,'source' ,'truncated' ,'in\_reply\_to\_status\_id' ,'in\_reply\_to\_status\_id\_str' ,'in\_reply\_to\_user\_id' ,'in\_reply\_to\_user\_id\_str' ,'in\_reply\_to\_screen\_name' ,'user.id' ,'user.id\_str' ,'user.name' ,'user.screen\_name' ,'user.location' ,'user.url' ,'user.description' ,'user.protected' ,'user.followers\_count' ,'user.friends\_count' ,'user.listed\_count' ,'user.created\_at' ,'user.favourites\_count' ,'user.utc\_offset' ,'user.time\_zone' ,'user.geo\_enabled' ,'user.verified' ,'user.statuses\_count' ,'user.lang' ,'user.contributors\_enabled' ,'user.is\_translator' ,'user.profile\_background\_color' ,'user.profile\_background\_image\_url' ,'user.profile\_background\_image\_url\_https' ,'user.profile\_background\_tile' ,'user.profile\_image\_url' ,'user.profile\_image\_url\_https' ,'user.profile\_banner\_url' ,'user.profile\_link\_color' ,'user.profile\_sidebar\_border\_color' ,'user.profile\_sidebar\_fill\_color' ,'user.profile\_text\_color' ,'user.profile\_use\_background\_image' ,'user.default\_profile' ,'user.default\_profile\_image' ,'user.following' ,'user.follow\_request\_sent' ,'user.notifications' ,'geo' ,'coordinates' ,'place' ,'contributors' ,'retweet\_count' ,'favorite\_count' ,'favorited' ,'retweeted' ,'lang' ,'entities.hashtags[].text' ,'entities.hashtags[].indices[]' ,'entities.user\_mentions[].screen\_name' ,'entities.user\_mentions[].name' ,'entities.user\_mentions[].id' ,'entities.user\_mentions[].id\_str' ,'entities.user\_mentions[].indices[]' ,'entities.urls[].url' ,'entities.urls[].expanded\_url' ,'entities.urls[].display\_url' ,'entities.urls[].indices[]' ,'possibly\_sensitive' ,'geo' ,'coordinates' ,'place'])
16. ### Writing tweets using Cursor method of Twitter API ###
17. for tweet in tweepy.Cursor(api.search,q='dengue',lang='en').items(200000):
18. #Write a row to the csv file/ I use encode utf-8
19. csvWriter.writerow([tweet.id , tweet.id\_str , tweet.created\_at , tweet.text , tweet.source , tweet.truncated , tweet.in\_reply\_to\_status\_id , tweet.in\_reply\_to\_status\_id\_str , tweet.in\_reply\_to\_user\_id , tweet.in\_reply\_to\_user\_id\_str , tweet.in\_reply\_to\_screen\_name , tweet.user.id , tweet.user.id\_str , tweet.user.name , tweet.user.screen\_name , tweet.user.location , tweet.user.url , tweet.user.description , tweet.user.protected , tweet.user.followers\_count , tweet.user.friends\_count , tweet.user.listed\_count , tweet.user.created\_at , tweet.user.favourites\_count , tweet.user.utc\_offset , tweet.user.time\_zone , tweet.user.geo\_enabled , tweet.user.verified , tweet.user.statuses\_count , tweet.user.lang , tweet.user.contributors\_enabled , tweet.user.is\_translator , tweet.user.profile\_background\_color , tweet.user.profile\_background\_image\_url , tweet.user.profile\_background\_image\_url\_https , tweet.user.profile\_background\_tile , tweet.user.profile\_image\_url , tweet.user.profile\_image\_url\_https , tweet.user.profile\_link\_color , tweet.user.profile\_sidebar\_border\_color , tweet.user.profile\_sidebar\_fill\_color , tweet.user.profile\_text\_color , tweet.user.profile\_use\_background\_image , tweet.user.default\_profile , tweet.user.default\_profile\_image , tweet.user.following , tweet.user.follow\_request\_sent , tweet.user.notifications , tweet.geo , tweet.coordinates , tweet.place , tweet.contributors , tweet.retweet\_count , tweet.favorite\_count , tweet.favorited , tweet.retweeted , tweet.lang , [i['text'] for i in tweet.entities['hashtags']] , [i['indices'] for i in tweet.entities['hashtags']] , [i['screen\_name'] for i in tweet.entities['user\_mentions']] , [i['name'] for i in tweet.entities['user\_mentions']] , [i['id'] for i in tweet.entities['user\_mentions']] , [i['id\_str'] for i in tweet.entities['user\_mentions']] , [i['indices'] for i in tweet.entities['user\_mentions']] , [i['url'] for i in tweet.entities['urls']] , [i['expanded\_url'] for i in tweet.entities['urls']] , [i['display\_url'] for i in tweet.entities['urls']] , [i['indices'] for i in tweet.entities['urls']] , tweet.possibly\_sensitive , tweet.geo , tweet.coordinates , tweet.place])
20. print(tweet.text.encode('utf8'))
21. csvFile.close()
22. """
23. Extract tweets and write tweet.text in text file for futher processing
24. """
25. import sys, csv, re
26. tablefilename = '/home/deepak/nltk\_data/dengue\_fuzzy/data/tweets.csv'
27. outfilename = '/home/deepak/nltk\_data/dengue\_fuzzy/data/tweet.text.txt'
28. tablereader = csv.reader(open(tablefilename), delimiter=',', quotechar='"')
29. ### Read the headers in the table ###
30. headers = tablereader.next()
31. d = dict()
32. for index, item in enumerate(headers):
33. d[item] = index
34. ### Open the output file ###
35. output=open(outfilename, 'w')
36. ### Read through the table ###
37. for row in tablereader:
38. output.write(row[d['text']])
40. output.close()

"""

Extracting Tokens from tweets and filtering stopwords in tokenized sentences

"""

outfilename = '/home/deepak/nltk\_data/dengue\_fuzzy/data/tweet.text.txt'

### Open the output file ###

outputfile=open(outfilename)

### NLTK Regexp Tokenizer ###

from nltk.tokenize import RegexpTokenizer

tokenizer = RegexpTokenizer('\s+', gaps=True)

tokens=[]

### Read through the output\_file ###

for row in outputfile:

tokens+=tokenizer.tokenize(row)

outputfile.close()

### Filtering Stopwords ###

from nltk.corpus import stopwords

import string

english\_stops = set( stopwords.words('english') + list(string.punctuation) )

tokens=[str.lower(word) for word in tokens if str.lower(word) not in english\_stops]

from nltk.probability import FreqDist, ConditionalFreqDist

fd = FreqDist(tokens)

#cfd = ConditionalFreqDist(tagged\_words)

most\_freq = (word for word, count in fd.most\_common(50))

#return dict((word, cfd[word].max()) for word in most\_freq)

import csv

outfilename = '/home/deepak/nltk\_data/dengue\_fuzzy/data/hash.tag.count.csv'

# open the output file and write headers

csvfile=open(outfilename, 'w')

writer = csv.writer(csvfile, delimiter=',')

writer.writerow(['hash.tag','count'])

for word,count in fd.most\_common():

writer.writerow([word,count])

csvfile.close()

### Most Frequenly Occuring Words ###

>>> list(most\_freq)

['dengue', 'fever', 'virus', 'news:', 'tests', 'cases', 'vaccines', 'sao', 'clinical', 'several', 'carried', 'dr', 'hilmi', 'general', 'paulo', 'fighting', 'outbreak', '#dengue', 'brazilian', 'yellow', '#ondox', 'brazil', 'fight', 'ebola', 'samples', 'update:', 'negative', 'for:', 'authorities', 'valley', 'rift', 'city', 'lassa', 'yahaya:\xe2\x80\xa6', '@ebolaalert:', 'army', '2015', 'calls', 'new', 'latest', 'news', 'test', 'yahay...', 'viru\xe2\x80\xa6rt', 'like', 'teams', 'mosquito', 'vigilant', 'http://t.co/d4nvs0rwqu', 'help']

### Top 50 Hash Tags and their count ###

dengue 3889 fever 1635

virus 842 news: 807

tests 682 cases 655

vaccines 463 sao 457

clinical 451 several 442

carried 442 dr 435

hilmi 435 general 397

paulo 341 fighting 332

outbreak 305 #dengue 303

brazilian 295 yellow 272

#ondox 253 brazil 247

fight 246 ebola 240

samples 236 update: 235

negative 235 for: 235

authorities 235 valley 234

rift 233 city 232

lassa 232 yahaya:… 222

@ebolaalert: 222 army 221

2015 208 calls 200

new 191 latest 187

news 177 test 174

yahay... 172 viru…rt 169

like 168 teams 164

mosquito 158 vigilant 158

http://t.co/d4nvs0rwqu 156 help 151

"""

Read in a .csv file containing twitter data and output the network hashtag groups

"""

import sys, csv, re

# Remove duplicates without preserving order

def remove\_duplicates(seq):

# Not order preserving

keys = {}

for e in seq:

keys[e] = 1

return keys.keys()

filename = '/home/deepak/nltk\_data/dengue\_fuzzy/data/tweets.csv'

reader = csv.reader(open(filename), delimiter=',', quotechar='"')

# build the dictionary of headers

headers = reader.next()

d = dict()

for index, item in enumerate(headers):

d[item] = index

#d={'interval': 0, 'tweet\_id': 1, 'text': 2, 'user.screen\_name': 3}

# dictionary to map hashtags to the group of users

usersOfTag = dict()

# dictionary to map users to the hashtags they've used

tagsByUser = dict()

hash\_regex = re.compile(r'#[0-9a-zA-Z+\_]\*',re.IGNORECASE)

user\_regex = re.compile(r'@[0-9a-zA-Z+\_]\*',re.IGNORECASE)

for row in reader:

user = row[d['user.screen\_name']]

tags = tagsByUser[user] if user in tagsByUser else list()

# loop through all hashtags in this tweet

for ht in hash\_regex.findall(row[d['text']]):

tag = ht.lstrip('#').lower()

if tag is "":

continue

users = usersOfTag[tag] if tag in usersOfTag else list()

users.append(user)

usersOfTag[tag] = users

tags.append(tag)

if tags:

tagsByUser[user] = tags

#print the nodes

print "nodedef>name,mention\_count INT,member\_count INT"

for tag in usersOfTag.keys():

users = usersOfTag[tag]

print "%s,%d,%d" % (tag, len(users), len(remove\_duplicates(users)))

#print the edges

print ""

print "edgedef>node1 VARCHAR,node2 VARCHAR,weight INT"

edgeDict = dict()

for user in tagsByUser.keys():

tags = sorted(remove\_duplicates(tagsByUser[user]))

for i,val1 in enumerate(tags):

for j,val2 in enumerate(tags):

if j > i:

s = val1+','+val2

if s in edgeDict:

edgeDict[s] += 1

else:

edgeDict[s] = 1

#print "%s,%s,1" % (val1, val2)

for pair in edgeDict.keys():

print "%s,%d" % (pair,edgeDict[pair])

"""

Output data in the form => (time.interval, hash.tag, count)

"""

import sys, csv, re

top\_tags = ['dengue', 'fever', 'virus', 'news:', 'tests', 'cases', 'vaccines', 'sao', 'clinical', 'several', 'carried', 'dr', 'hilmi', 'general', 'paulo', 'fighting', 'outbreak', '#dengue', 'brazilian', 'yellow']

def reset\_tag\_counts():

tag\_counts = dict()

for tag in top\_tags:

tag\_counts[tag] = 0;

return tag\_counts

tablefilename = '/home/deepak/nltk\_data/dengue\_fuzzy/data/tweets.csv'

outfilename = '/home/deepak/nltk\_data/dengue\_fuzzy/data/hashtags\_over\_time\_factor\_cumul.csv'

tablereader = csv.reader(open(tablefilename), delimiter=',', quotechar='"')

# read the headers in the table

headers = tablereader.next()

d = dict()

for index, item in enumerate(headers):

d[item] = index

#d={'interval': 0, 'tweet\_id': 1, 'text': 2, 'user.screen\_name': 3}

# open the output file and write headers

writer = csv.writer(open(outfilename, 'w'), delimiter=',')

writer.writerow(['time.interval', 'hash.tag', 'count'])

# read through the table and for each interval calculate counts for each of the top tags

hash\_regex = re.compile(r'#[0-9a-zA-Z+\_]\*',re.IGNORECASE)

current\_interval = ''

counts = reset\_tag\_counts()

for row in tablereader:

row\_interval = row[d['interval']]

if row\_interval != current\_interval:

# write the row and reset the counters and current interval

if current\_interval != '' and current\_interval != 'NA':

for key in top\_tags:

writer.writerow([current\_interval, key, counts[key]])

#writer.writerow([current\_interval] + [counts[key] for key in top\_tags])

current\_interval = row\_interval

#counts = reset\_tag\_counts()

# loop through the hashtags and count up the top tags

for ht in hash\_regex.findall(row[d['text']]):

tag = ht.lstrip('#').lower()

if tag in top\_tags:

counts[tag] += 1

# write the last row

#writer.writerow([current\_interval] + [counts[key] for key in top\_tags])

for key in top\_tags:

writer.writerow([current\_interval, key, counts[key]])

### hashtags.R ####

### Top tag analysis ###

library(ggplot2)

hashtags <- read.csv("/home/deepak/nltk\_data/dengue\_fuzzy/data/hashtags\_over\_time\_factor\_cumul.csv", header=T)

#quartz()

top.hashtags <- subset(hashtags, hash.tag %in% c('dengue', 'fever', 'virus', 'news:', 'tests', 'cases', 'vaccines'))

ggplot(top.hashtags, aes(x=time.interval)) + geom\_freqpoly(aes(group=hash.tag, colour=hash.tag)) + theme(axis.text.x=element\_text(angle=-90, hjust=0, size=10))

ggplot.save()