Python codes for Dynamic Topic Derivation
by: Robertus Nugroho (nugroho@unika.com)

Requirements:
- python 2.7
- MySQL database

There are two main codes that need to be run for the online topic derivation process:

- getTweets.py: To stream the tweets using streaming API and then save the data to MySQL database
      ==> all streaming API keys from Twitter must be set here
      ==> database connection details must be set

- streamDynamic.py: Code to execute the topic derivation process (read the tweets from MySQL database)
          ==> database connection details must be set
          ==> output folder must be set (raw output of the topic derivation process will be saved in this folder)
          ==> All required parameters must be set in this file

SQL to create the MySQL table structure is available from dbExample.sql

Other codes:
- cluster_count.py
  use to manually read the result file from the folder 'results' (streamDynamic will store the resulted topic derivation process in the 'results' folder).
  python cluster_count.py --rc ./results/the_result_file.csv

- Python library to be installed:
   - tweepy, mysql.connector, nltk, numpy

Suggested volume of tweets per batch (2000-5000) to adjust the interval parameters in streamDynamic.py

```
#!/usr/bin/env python

#Dynamic topic derivation in Twitter. The corpus is stored in MySQL database
#Robertus Nugroho (2017)

import mysql.connector
from mysql.connector import Error
from datetime import datetime
import NMijF as rob_nmf
import LDA as lda_cluster
import time
import re
from operator import mul    # or mul=lambda x,y:x*y
from fractions import Fraction
import math

def streamProcess():

    #Start Setting

    dbUser = 'root'                                 #database username
    dbPassword = ''                                 #database password
    dbHost = 'localhost'                            #database host
    dbName = 'sampleAll5'                           #database name
    table = 'tweetStream'                           #table in database
    #method = ['NMijF','TNMF','NMF','LDA']
    method = ['LDA']                               #method used to derive the
topics
    outputFolder = "./results/"                     #folder for the topic
derivation results
    alpha = 0.1                                     #alpha parameter (used in both
tNMijF and LDA)
    beta = 0.1                                      #beta parameter used in LDA
    K = 10                                          #number of topics
    iteration = 30                                  #number of iterations in
derivation process
    stopwords = True                                #True if stopwords need to be
removed
    df = 0
    limit = 0
    offset = 50
    interval = 60                                   #one interval in second
    maxNumInterval = 4                              #maximum number of interval
for every batch before topic derivation process started
    newsFactor = 2                                  #breaking news factor
    maxEmptyCount = 2*maxNumInterval                #maximum empty batch before
process stopped

    #End Setting

    timeProcessStart = datetime.now().strftime('%Y-%m-%d_%H-%M-%S')

    for i in range(len(method)):
```

```
47
48          #mysql database connection
49          conn = mysql.connector.connect(user=dbUser, password=dbPassword,
    database=dbName, host=dbHost)
50          cursorTop = conn.cursor()
51
52          startTimeStamp = ''
53          #get startTimeStamp
54          sqlT = "SELECT min(created_time) as startTimeStamp FROM "+table
55          cursorT = conn.cursor()
56          cursorT.execute(sqlT)
57          rowT = cursorT.fetchone()
58          while rowT is not None:
59              startTimeStamp = str(rowT[0])
60              rowT = cursorT.fetchone()
61          cursorT.close()
62
63          cursorTop.close()
64          conn.close()
65
66          #Hardcode the start time if you want to start from specific timestamp
67          #uncomment and follow this timestamp format to update the start time
68          #startTimeStamp = '2013-02-13 00:00:00'
69
70          #or use the line below if you want to start from now
71          #startTimeStamp = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
72
73          startTimeInterval = startTimeStamp      #variable for the startTime of
    current batch
74
75          runStatus = 1                           #1 means online processing is
    running, 0 means stop
76          currInterval = 0                        #counter of interval. Reset if
    reach maxNumInterval
77          prevTweets = 0                          #number of tweets from
    previous batch
78          secondPrevTweets = 0                    #number of tweets from the
    previous of previous batch
79          emptyCount = 0
80          currTotalTweets = 0
81
82          endTimeStamp = startTimeStamp
83
84          while runStatus == 1:
85              currRate = 0
86
87              print "Checking interval: "+startTimeStamp+", waiting for
    "+str(round(interval/60.0,2))+" minutes"
88
89              #delay for real world implementation, not used for simulation or
    offline implementation
90              #time.sleep(interval)
91
```

```
92              conn = mysql.connector.connect(user=dbUser, password=dbPassword,
… database=dbName, host=dbHost)
93              cursorTop = conn.cursor()
94
95              sqlTop = ""
96
97              if (currTotalTweets > 0 or currInterval == 0):
98                  sqlTop = "SELECT count(str_id) as numOfTweets,
… '"+startTimeStamp+"' + Interval "+str(interval)+" second FROM "+table+" where
… created_time between '"+startTimeStamp+"' and '"+startTimeStamp+"' + Interval
… "+str(interval)+ " second"
99              else:
100                 sqlTop = "SELECT count(str_id) as numOfTweets,
… '"+endTimeStamp+"' + Interval "+str(interval)+" second FROM "+table+" where
… created_time between '"+startTimeStamp+"' and '"+endTimeStamp+"' + Interval
… "+str(interval)+" second"
101
102             cursorTop.execute(sqlTop)
103             rowTop = cursorTop.fetchone()
104
105             filename =
… outputFolder+table+"_"+method[i]+"_"+str(timeProcessStart)+".csv"
106
107             currInterval = currInterval + 1
108
109             #note the start time of processing window
110             if currInterval == 1:
111                 startTimeInterval = startTimeStamp
112                 currTotalTweets = 0
113
114             #check rates dynamically and compare with previous rates
115             if rowTop is not None:
116                 currRate = rowTop[0]
117                 endTimeStamp = rowTop[1]
118
119                 #increment empty counter if no tweets in this checking
… interval
120                 if currRate == 0:
121                     emptyCount = emptyCount + 1
122                 else:
123                     #reset the counter if not empty
124                     emptyCount = 0
125
126                 #dynamic rate checking
127                 if (prevTweets !=0 and secondPrevTweets !=0):
128                     if (currRate >= (secondPrevTweets * newsFactor) or
… currRate >= (prevTweets * newsFactor)):
129                         currInterval = maxNumInterval
130
131                 secondPrevTweets = prevTweets
132                 prevTweets = currRate
133
134                 currTotalTweets = currTotalTweets + currRate
```

```
135
136             cursorTop.close()
137             conn.close()
138
139             if currInterval >= maxNumInterval and currTotalTweets > 0:
140                 print "==> Topic derivation for processing window:
…"+startTimeInterval+" to "+str(endTimeStamp)
141
142                 #process topic derivation for current processing window
143                 strResult = ''
144                 if(method[i]!="LDA"):
145                     startTime = time.clock()
146                     strResult = rob_nmf.deriveTopics(method[i], table, dbUser,
…dbPassword, dbHost, dbName, startTimeInterval, endTimeStamp, limit, offset,
…df, stopwords, alpha, beta, iteration, K)
147                     totalTime = time.clock() - startTime
148                 else:
149                     startTime = time.clock()
150                     strResult = lda_cluster.deriveTopics(method[i], table,
…dbUser, dbPassword, dbHost, dbName, startTimeInterval, endTimeStamp, limit,
…offset, df, stopwords, alpha, beta, 50, K)
151                     totalTime = time.clock() - startTime
152
153                 strResult =
…method[i]+";"+str(totalTime)+";"+startTimeStamp+";"+str(endTimeStamp)+";"+str(
…K)+";"+strResult+"\n"
154                 printTopic(strResult)
155
156                 with open(filename, "a") as text_file:
157
…#text_file.write(method[i]+";"+qualityEval+";"+str(totalTime)+";"+
…startTimeStamp+";"+str(endTimeStamp)+";"+str(K)+";\n")
158
…#text_file.write(method[i]+";"+str(totalTime)+";"+startTimeStamp+";"+str(
…endTimeStamp)+";"+str(K)+";"+strResult+"\n")
159                     text_file.write(strResult)
160
161
162                 #reset interval and empty counter after topic derivation for
…current batch
163                 currInterval = 0
164                 emptyCount = 0
165
166             if emptyCount >= maxEmptyCount:
167                 #stop the process as no more incoming tweets
168                 runStatus = 0
169
170             #set for next batch
171             if currTotalTweets > 0:
172                 startTimeStamp = endTimeStamp
173         print "No more tweets to be processed"
174
175 def printTopic(line):
```

```
176
177     allClusters=[]
178
179
180     thisLine = line.split(';')
181     clusters = []
182     thisLine[7] = re.sub(r'\n',"", thisLine[7])
183     txt = thisLine[6]
184     txt = txt.replace("'","")
185     txt = txt.replace(", ","|")
186     txt = txt.replace("]|[","&")
187     txt = txt.replace("[[[","[[")
188     txt = txt.replace("]]]","]]")
189     txt = txt.replace("]], [[","], [")
190     txt = txt.replace("[[","")
191     txt = txt.replace("]]","")
192     txt = txt.replace("&&","&")
193     txt = txt.replace("&&","&")
194     txt = re.sub(r'^&',"", txt)
195     txt = re.sub(r'&$',"", txt)
196     txt = txt.replace("&&","&")
197     #print txt
198     for stra in txt.split('&'):
199         #print stra
200         if(stra!=""):
201             clusters.append(map(int,stra.split('|')))
202
203     #the keywords
204     keywords = thisLine[7].split("&")
205
206
207     numMember=[]
208     for i in clusters:
209         numMember.append(len(i))
210
211
212     thisCluster = {}
213     thisCluster['keywords'] = keywords
214     thisCluster['counter'] = numMember
215     thisCluster['start'] = thisLine[2]
216     thisCluster['stop'] = thisLine[3]
217
218     allClusters.append(thisCluster)
219
220     #print allClusters
221
222     for i in xrange(len(allClusters)):
223         try:
224             print allClusters[i]['start']+" to "+allClusters[i]['stop']
225             totalTweets = 0
226             for a in xrange(len(allClusters[i]['keywords'])):
227                 #print (a+1),allClusters[i]['keywords'][a],
  … allClusters[i]['counter'][a]
```

```
228                 print "=>",allClusters[i]['keywords'][a],
  … allClusters[i]['counter'][a]
229                 totalTweets = totalTweets + allClusters[i]['counter'][a]
230             print "total tweets for period above: ",totalTweets
231             print "========================================="
232         except:
233             pass
234 
235 if __name__ == '__main__':
236     streamProcess()
237 
```

```
import sys
import tweepy
import os
import json

import re
import nltk
import collections
import math

import pytz
import datetime
import ast
import mysql.connector
import time

reload(sys)
sys.setdefaultencoding("utf8")

#Streaming API Key from Twitter
consumer_key="wVsd99kbxrMw9QOfDutrpg"
consumer_secret="AfdTq9TRRh2Cm88oRCOVZFrjFxG6YrFW16WBsNcskQ"
access_key = "2237500892-zXbMkpQ2h0HAqHHnmiAgB1ZR9tkCNiP6eK2jQgR"
access_secret = "TwwJPqXm0eXlg8lB9fCBBiXxL8qtEhniY1xaz86yEw2K0"

#database connection details:
dbUser = 'root'                                 #database username
dbPassword = ''                                 #database password
dbHost = 'localhost'                            #database host
dbName = 'sampleAll5'                           #database name
dbTable = 'tweetStream'                         #table in database

auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_key, access_secret)

api = tweepy.API(auth)

#Database connection details
cnx = mysql.connector.connect(user=dbUser, password=dbPassword,
database=dbName, host=dbHost)
cursor = cnx.cursor()


#function to preprocess the input.
#Currently just to remove ' and \
#complete preprocessing will be done in the topic derivation process
def preprocessWord(text):
    #text = text.lower()
    text = re.sub('\'',' ', text)
    text = re.sub(r'\\',' ', text)

    return text

```

```
53
54 #function to insert into database
55 def insertToDB(jsonStr, dbTable, filename):
56     #homeTZ = pytz.timezone("Australia/Sydney")
57     # Instantiate time zone object
58     #utc = pytz.utc
59
60     try:
61         myStr = ast.literal_eval(str(jsonStr))
62     except:
63         return
64
65     #print myStr['lang']
66
67     #only English tweet
68     if (myStr['lang']=="en"):
69         #print myStr['text']
70         texT = preprocessWord(str(myStr['text']))
71         reply_id = str(myStr['in_reply_to_status_id'])
72         geoC = myStr['geo']
73         created_at = str(myStr['created_at'])
74         mentions= myStr['entities']['user_mentions']
75         hashtag = myStr['entities']['hashtags']
76         url = myStr['entities']['urls']
77         tweet_id = str(myStr['id'])
78         in_reply_to_user_id = str(myStr['in_reply_to_user_id'])
79         location = str(myStr['user']['location'])
80         location = re.sub('\'','', str(location))
81         user_id = str(myStr['user']['id'])
82         screen_name = str(myStr['user']['screen_name'])
83         time_zone = myStr['user']['time_zone']
84         time_zone = re.sub('u\'','', str(time_zone))
85         time_zone = re.sub('\'','', str(time_zone))
86
87         #print
… texT,reply_id,geo,created_at,mentions,hashtag,url,tweet_id,in_reply_to_user_id
… ,location,user_id,screen_name,time_zone
88
89         mention_id_list = ""
90         hashtag_list = ""
91         url_list = ""
92         geo = ""
93
94         if(mentions!=[]):
95             mention_id_list = '|'.join([re.sub('\D','', str(ml['id'])) for ml
… in mentions])
96
97         if(hashtag != []):
98             hashtag_list = '|'.join([str(ht['text']) for ht in hashtag])
99
100        if(url!=[]):
101            url_list = '|'.join([str(ul['url']) for ul in url])
102
```

```
103         if(geoC!=None):
104             geo = str(geoC['coordinates'])
105 
106         #if retweet
107         rt_texT =  ""
108         rt_reply_id = ""
109         rt_geoC = ""
110         rt_created_at = ""
111         rt_mentions = ""
112         rt_hashtag = ""
113         rt_url = ""
114         rt_tweet_id = ""
115         rt_in_reply_to_user_id = ""
116         rt_user_id = ""
117         rt_screen_name = ""
118         rt_location = ""
119         rt_time_zone = ""
120 
121         rt_aa  = ""
122         rt_mention_id_list =""
123         rt_hashtag_list = ""
124         rt_url_list = ""
125         rt_geo = ""
126 
127         if ('retweeted_status' in myStr):
128             #print "horeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee", filename
129             rt_texT = preprocessWord(str(myStr['retweeted_status']['text']))
130             rt_reply_id =
… str(myStr['retweeted_status']['in_reply_to_status_id'])
131             rt_geoC = myStr['retweeted_status']['geo']
132             rt_created_at = str(myStr['retweeted_status']['created_at'])
133             rt_mentions=
… myStr['retweeted_status']['entities']['user_mentions']
134             rt_hashtag = myStr['retweeted_status']['entities']['hashtags']
135             rt_url = myStr['retweeted_status']['entities']['urls']
136             rt_tweet_id = str(myStr['retweeted_status']['id'])
137             rt_in_reply_to_user_id =
… str(myStr['retweeted_status']['in_reply_to_user_id'])
138             rt_location = str(myStr['retweeted_status']['user']['location'])
139             rt_location = re.sub('\'','', str(rt_location))
140             rt_user_id = str(myStr['retweeted_status']['user']['id'])
141             rt_screen_name =
… str(myStr['retweeted_status']['user']['screen_name'])
142             rt_time_zone = myStr['retweeted_status']['user']['time_zone']
143             rt_time_zone = re.sub('u\'','', str(rt_time_zone))
144             rt_time_zone = re.sub('\'','', str(rt_time_zone))
145 
146             if(rt_mentions!=[]):
147                 rt_mention_id_list = '|'.join([re.sub('\D','', str(ml['id']))
… for ml in rt_mentions])
148 
149             if(rt_hashtag != []):
150                 rt_hashtag_list = '|'.join([str(ht['text']) for ht in
```

```
rt_hashtag])

            if(rt_url!=[]):
                rt_url_list = '|'.join([str(ul['url']) for ul in rt_url])

            if(rt_geoC!=None):
                rt_geo = str(rt_geoC['coordinates'])

            #print
rt_texT,rt_reply_id,rt_geoC,rt_created_at,rt_mentions,rt_hashtag,rt_url,
rt_tweet_id,rt_in_reply_to_user_id,rt_location,rt_user_id,rt_screen_name,
rt_time_zone


        sql = "insert IGNORE into " + dbTable + "
(str_id,tweet,created_at,created_time,reply_id,in_reply_to_user_id,user_id,
screen_name,mentions,hashtag,url,geo,location,timezone,rt_str_id,rt_tweet,
rt_created_at,rt_reply_id,rt_in_reply_to_user_id,rt_user_id,rt_screen_name,
rt_mentions,rt_hashtag,rt_url,rt_geo,rt_location,rt_timezone,filename) values
('"+str(tweet_id)+"','"+texT+"','"+created_at+"',STR_TO_DATE(replace('"+
created_at+"','+0000',''),'%a %b %d %H:%i:%s
%Y'),'"+str(reply_id)+"','"+str(in_reply_to_user_id)+"','"+str(user_id)+"','"+
screen_name+"','"+mention_id_list+"','"+hashtag_list+"','"+url_list+"','"+geo+
"','"+location+"','"+time_zone+"','"+str(rt_tweet_id)+"','"+rt_texT+"','"+
rt_created_at+"','"+str(rt_reply_id)+"','"+str(rt_in_reply_to_user_id)+"','"+
str(rt_user_id)+"','"+rt_screen_name+"','"+rt_mention_id_list+"','"+
rt_hashtag_list+"','"+rt_url_list+"','"+rt_geo+"','"+rt_location+"','"+
rt_time_zone+"','"+filename+"')"
        #print sql
        #with con:
            #cur = con.cursor()
            #cur.execute(sql)

        try:                    #insert into database
            cursor.execute(sql)
            cnx.commit()
        except Exception:
            return
    return


class CustomStreamListener(tweepy.StreamListener):
    FORMAT = '%Y%m%d%H%M%S'
    curDate = datetime.datetime.now()
    filename = 'tweet_'
    script_dir = os.path.dirname(os.path.abspath(__file__))
    #dest_dir = os.path.join(script_dir, 'TwitterData', 'LocationAus')
    dest_dir = os.path.join(script_dir, 'TwitterData')

    counter = 0

    try:
        os.makedirs(dest_dir)
```

```
187     except OSError:
188         pass # already exists
189     myPath = os.path.join(dest_dir, filename)
190 
191     def on_status(self, status):
192         #print status.user.screen_name,': ',status.text
193         #self.output.write(status + "\n")
194         #print status._json
195 
196         theStr = str(status._json) + "\n"
197         if (datetime.datetime.now() - self.curDate).total_seconds() > 60:
198             self.curDate = datetime.datetime.now()
199 
200         #write to file, uncomment these two lines below to save json output to
    … files
201         #f = open(self.myPath+self.curDate.strftime(self.FORMAT),"a")
202         #f.write(theStr)
203 
204 
205 
206         #insert into DB
207         insertToDB(theStr, dbTable,
    … self.myPath+self.curDate.strftime(self.FORMAT))
208 
209         #output status
210         timeNow = str(datetime.datetime.now().strftime('%Y-%m-%d_%H:%M:%S'))
211         self.counter = self.counter + 1
212         sys.stdout.write("\r  ==> %i tweets are downloaded. Current timestamp:
    … %s" % (self.counter, timeNow))
213         sys.stdout.flush()
214 
215         ##f.close
216 
217         return
218 
219     def on_error(self, status_code):
220         print >> sys.stderr, 'Encountered error with status code:',
    … status_code
221         return True # Don't kill the stream
222 
223     def on_timeout(self):
224         print >> sys.stderr, 'Timeout...'
225         return True # Don't kill the stream
226 
227 
228 sapi = tweepy.streaming.Stream(auth, CustomStreamListener())
229 
230 #change the filter here
231 #sapi.filter(track=['police','afp','traffic','accident','bus','train','beer','
    … raid','google','android','ios','apple','iphone','weekend','beach'],locations=[
    … 103.388729,-41.858809,155.945719,-9.175410])
232 #sapi.filter(locations=[103.388729,-41.858809,155.945719,-9.175410])
233 sapi.filter(track=['sydney','afp','cbd','lunch','police'])
```

```
234 #sapi.filter(track=['police','afp','traffic','accident','bus','train','google'
… ,'android','ios','apple','iphone'])
235
```