A FINAL REPORT

ON

EVENT IDENTIFICATION FROM TEXT MESSAGES

Submitted in Partial Fulfilment of the
COMPUTER ORIENTED PROJECT

BY
KANIKA BAIJAL(2010B4A7554H)

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

HYDERABAD CAMPUS

APRIL, 2014

A FINAL REPORT

ON

EVENT IDENTIFICATION FROM TEXT MESSAGES

Submitted in Partial Fulfilment of the
COMPUTER ORIENTED PROJECT

BY
KANIKA BAIJAL(2010B4A7554H)

UNDER THE SUPERVISION OF

MR SURENDER SINGH SAMANT

DEPARTMENT OF CS/IS

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

HYDERABAD CAMPUS

APRIL, 2014

ACKNOWLEDGMENT

I would like to take the opportunity to thank to my project guide Mr.Surender Singh Samant for guiding me throughout the course of the project. I am deeply indebted to him for sharing his invaluable time and expertise throughout the span of my project work.

ABSTRACT

This project deals with extracting tweets from the social networking site 'Twitter' and preprocessing the obtained tweets. If there is much irrelevant and redundant information present or unreliable data, then using that data becomes difficult. Data pre-processing includes cleaning, normalization, transformation, feature extraction and selection, etc. After this step this final set of data is used for event identification using data clustering.ie. categorising the posts into clusters which may relate to certain topics based on a number of common words.

Contents

SOCIAL MEDIA

Social Media has recently evolved into a source of social, political and real time information. In addition to this it is also a great means of communication and marketing. People have been sharing information on social networks through the use of status updates , blogging, sharing multimedia content like images and videos as well as interacting together thereby forming groups and communities on social networks. Monitoring and analysing this information can lead to valuable insights that might otherwise be hard to get using conventional methods and media sources. The social networking sites such as Facebook,Twitter and Flickr provide a new way to share the information among them and get frequent updates. In addition to this, the sites also allow sharing of additional information which can be important in analysing the contents, e.g. location etc.

The social media has an advantage over conventional media sources as it is managed by the users. Conventional media only allowed users to gain information that was provided to them. The flow of information was only one-sided from the media to user. With social networks, however, the users now have the ability to respond to the news and events around them and provide their opinion on them as well as share them. This leads to the evolution of a multi-way mode of information dissemination in which the users post information along with other information like links, images and videos. As a result, a user generated model of information is generated.Since, the micro-blogging sites like Facebook, Twitter and Flickr allow users to share short messages and multimedia, they have become an instant source of information through which users from all around the world can remain connected and get to know about the information from several sources.

TWITTER

Twitter is a free social networking and micro-blogging service that enables its users to send and read messages knows as tweets.

Tweets are text-based posts of up to 140 characters displayed on the author's profile page and delivered to the author's profile page and delivered to the author's subscribers who are known as followers.

Twitter is an information network and communication mechanism that produces more than 200 million tweets a day.

Users can mention other users in their tweets by adding '@' to the username of another user in a tweet. A mention is a way to refer to some other user. Another popular concept of twitter is retweeting. A retweet is an event of sharing someone else's tweet to our followers. Retweet plays an important part in the dissemination of information on twitter. Users can also add a hashtag in their tweets by adding a '#' sign before relevant keywords. This is used to categorize those tweets to show more easily in twitter search. Very popular hash tags on twitter become trending topics on twitter.

The Twitter platform offers access to that corpus of data, via APIs. Each API represents a facet of Twitter, and allows developers to build upon and extend their applications in new and creative ways.

The Streaming APIs

Streaming API allows for large quantities of keywords to be specified and tracked, retreiving geo-tagged tweets from a certain region, or have the public statuses of a user set returned. This requires you to establish a long-lived HTTP connection and maintain that connection.

The set of streaming APIs offered by Twitter give developers low latency access to Twitter's global stream of Tweet data. A proper implementation of a streaming client will be pushed messages indicating Tweets and other events have occurred, without any of the overhead associated with polling a REST endpoint.

Twitter offers several streaming endpoints, each customized to certain use cases.

Public streams	Streams of the public data flowing through Twitter. Suitable for following specific users or topics, and data mining.
User streams	Single-user streams, containing roughly all of the data corresponding with a single user's view of Twitter.
Site streams	The multi-user version of user streams. Site streams are intended for servers which must connect to Twitter on behalf of many users.

PYTHON

Python is a general-purpose high-level programming language whose design philosophy emphasizes code readability. Python aims to combine "remarkable power with very clear syntax", and its standard library is large and comprehensive. Its use of indentation for block delimiters is unusual among popular programming languages. Python supports multiple programming paradigms, primarily but not limited

to object oriented, imperative and, to a lesser extent, functional programming styles. It features a fully dynamic type system and automatic memory management, similar to that of Scheme, Ruby, Perl, and Tcl. Like other dynamic languages, Python is often used as a scripting language, but is also used in a wide range of non-scripting contexts.

OAUTH

Introducing Twitter OAuth

OAuth is a method for interacting with Twitter on behalf of users without requiring them to supply a password every time they want to use an application. It is an open protocol to facilitate a standard, secure authorization method for desktop, mobile, and Web applications. Twitter OAuth takes the form of the "Sign in with Twitter" service, which enables users to sign in to your website or application using their Twitter credentials.

- During registration, you are prompted with a series of fields that you must fill out, including application name, description, and website.

- You are also asked whether you are creating a desktop or browser application (because the authentication steps are slightly different)
- You are asked whether you require read/write or just read-only access to user data. Read-only access is just "pulling" data from Twitter, as you have seen with Twitter API accessor methods, and would include reading a user's updates, direct messages, or favorites. Read/write access includes pulling data from Twitter but also "pushing" data back. This includes updating a user's status, sending a direct message, or marking a favorite.

- The callback URL is a location where users are redirected after successfully authenticating your application. For unsuccessful attempts, a user is returned to the Twitter home page

Advantages to using OAuth :

- You can set a 'from myappname' which will appear in tweets posted
- More secure since you don't need the user's password
- Your app does not break if the user changes their password in the future

Implementing Twitter OAuth

After you have registered your application, you are ready to begin implementing Twitter OAuth. The Twitter API includes four OAuth methods:

- oauth/request_token
- oauth/authorize
- oauth/authenticate
- oauth/access_token

A simplified workflow for browser-based applications is as follows:

- 1. A user visits an application, and a request token is generated by Twitter by calling the oauth/request_token method and using the application's consumer key and consumer secret.
- 2. A request can be made to oauth/authorize by following a URL appended with the request token to request user authorization. The oauth/authenticate method is reserved for applications using the "Sign in with Twitter" feature, which can be used to provide "one-click" user authentication

- 3. Following the URL, the user is redirected to Twitter where the request token is verified. If not logged in to Twitter, the user is required to log in to grant access to the application. At this stage, the user is reminded of which application is requesting access by being shown its logo, description, and developer information, and is then prompted to allow or deny access to their protected resources. If access is denied, a prompt will be displayed by Twitter but the user will not be redirected back to the application.
- 4. If allowed, Twitter marks the request token as authorized and redirects the user back to the application using the callback URL together with the request token and other OAuth protocol parameters.
- 5. An access token is then generated by passing the request token and OAuth protocol parameters to the oauth/access_token method, which can then be stored alongside the token secret by the application.
- 6. Whenever applications want to access a user's protected resources, they use the access token and token secret along with their consumer key and consumer secret for each request.

Home → My applications

Create an application

Application Details

Name: *

My Test App

Your application name. This is used to attribute the source of a tweet and in user-facing authorization screens. 32 characters max.

Description: *

A set of Twitter tools for personal use

Your application description, which will be shown in user-facing authorization screens. Between 10 and 200 characters max.

Website: *

http://iag.me/

Your application's publicly accessible home page, where users can go to download, make use of, or find out more information about your application. This fully-qualified URL is used in the source attribution for tweets created by your application and will be shown in user-facing authorization screens.
(If you don't have a URL yet, just put a placeholder here but remember to change it later.)

Callback URL:

Where should we return after successfully authenticating? For @Anywhere applications, only the domain specified in the callback will be used. OAuth 1.0a applications should explicitly specify their `oauth_callback` URL on the request token step, regardless of the value given here. To restrict your application from using callbacks, leave this field blank.

OAuth settings

Your application's OAuth settings. Keep the "Consumer secret" a secret. This key should never be human-readable in your application.

Access level	Read-only About the application permission model
Consumer key	[redacted]
Consumer secret	[redacted]
Request token URL	`https://api.twitter.com/oauth/request_token`
Authorize URL	`https://api.twitter.com/oauth/authorize`
Access token URL	`https://api.twitter.com/oauth/access_token`
Callback URL	`None`
Sign in with Twitter	`No`

Your access token

Use the access token string as your "`oauth_token`" and the access token secret as your "`oauth_token_secret`" to sign requests with your own Twitter account. Do not share your `oauth_token_secret` with anyone.

Access token	[redacted]
Access token secret	[redacted]
Access level	Read-only

Recreate my access token

INFORMATION AVAILABLE IN A TWEET

This is the available information in a tweet that can be obtained from twitter

- created_at:"Mon, 19 Nov 2012 20:19:21 +0000",
- from_user:"markweber",
- from_user_id:214343936,
- from_user_id_str:"214343936",
- from_user_name:"Mark Weber",
- geo:null,
- id:270622312510939137,
- id_str:"270622312510939137",
- iso_language_code:"en",
- metadata:{"result_type":"recent"},
- profile_image_url:"http:\/\/a0.twimg.com\/profile_images\/normal.jpeg",
- profile_image_url_https:"https:\/\/si0.twimg.com\/profile_images\/normal.jpeg",
- source:"<a href="http:\/\/twitter.com\/download\/iphone">Twitter for iPhone<\/a>",
- text:"RT @AlbertBreer: Gronkowski had screws inserted in his forearm at MGH this AM. He'll want to play during that Texans-Niners turn. That'd be awfully quick."
- to_user:null,
- to_user_id:0,
- to_user_id_str:"0",
- to_user_name:null

TWEETS ABOUT A TOPIC

The set of tweets that we collect are based on a set of keywords that describe a topic in real world. We collect tweets that contain a set of keywords:
For example:

Tweets about Arvind Kejriwal and AAP

KEYWORDS: Arvind Kejriwal, Kejriwal, AAP, Aam Aadmi Party.

These set of tweets allow us to achieve different goals. These set of tweets again serve the goal of modelling the spread of user interests around the world as well as the popularity of these topics at different points in time. This can again be used to model the rate of information flow on the internet. The collection of tweets from these keywords can also help these organisations to target different user groups in different places.

TWEET EXTRACTION

In the tweet extraction program we first use Oauth to get authorisation for collecting tweets.

auth = tweepy.OAuthHandler(consumer_key, consumer_secret)

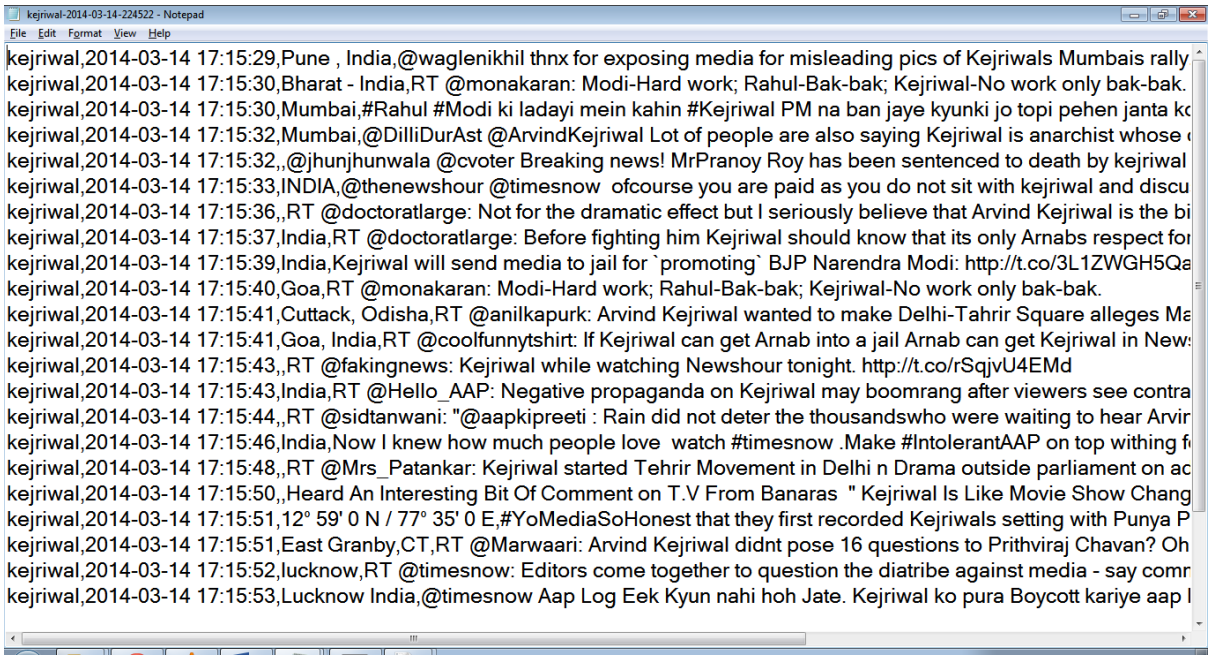
auth.set_access_token(access_token, access_secret)

Then the text of the tweets is cleaned.

status.text.replace('\'','').replace('&','').replace('>','').replace(',',',').replace("\n","')

The output is written to a text file.The fields in the output are the search key (in this case 'kejriwal'),the timestamp when the user tweeted,the location of the tweet and the cleaned tweet.The corresponding fields written to file are topicName, status.created_at, status.user.location and the cleaned code.

SAMPLE CAPTURE



PREPROCESSING OF DATA

Why preprocessing ?

1. Real world data are generally
    - Incomplete: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data
    - Noisy: containing errors or outliers
    - Inconsistent: containing discrepancies in codes or names

2. Tasks in data preprocessing
    - Data cleaning: fill in missing values, smooth noisy data, identify or remove outliers, and resolve inconsistencies.
    - Data integration: using multiple databases, data cubes, or files.
    - Data transformation: normalization and aggregation.
    - Data reduction: reducing the volume but producing the same or similar analytical results.
    - Data discretization: part of data reduction, replacing numerical attributes with nominal ones.

SPELL CHECKING

We correct the spellings of commonly misspelled words and some common text messaging and chat abbreviations. For example:'gr8' is corrected to 'great', 'thnx' is corrected to thanks'.
A file containing misspellings and corrected words is used to implement this. The output is stored in the file 'spell_checked.txt'.

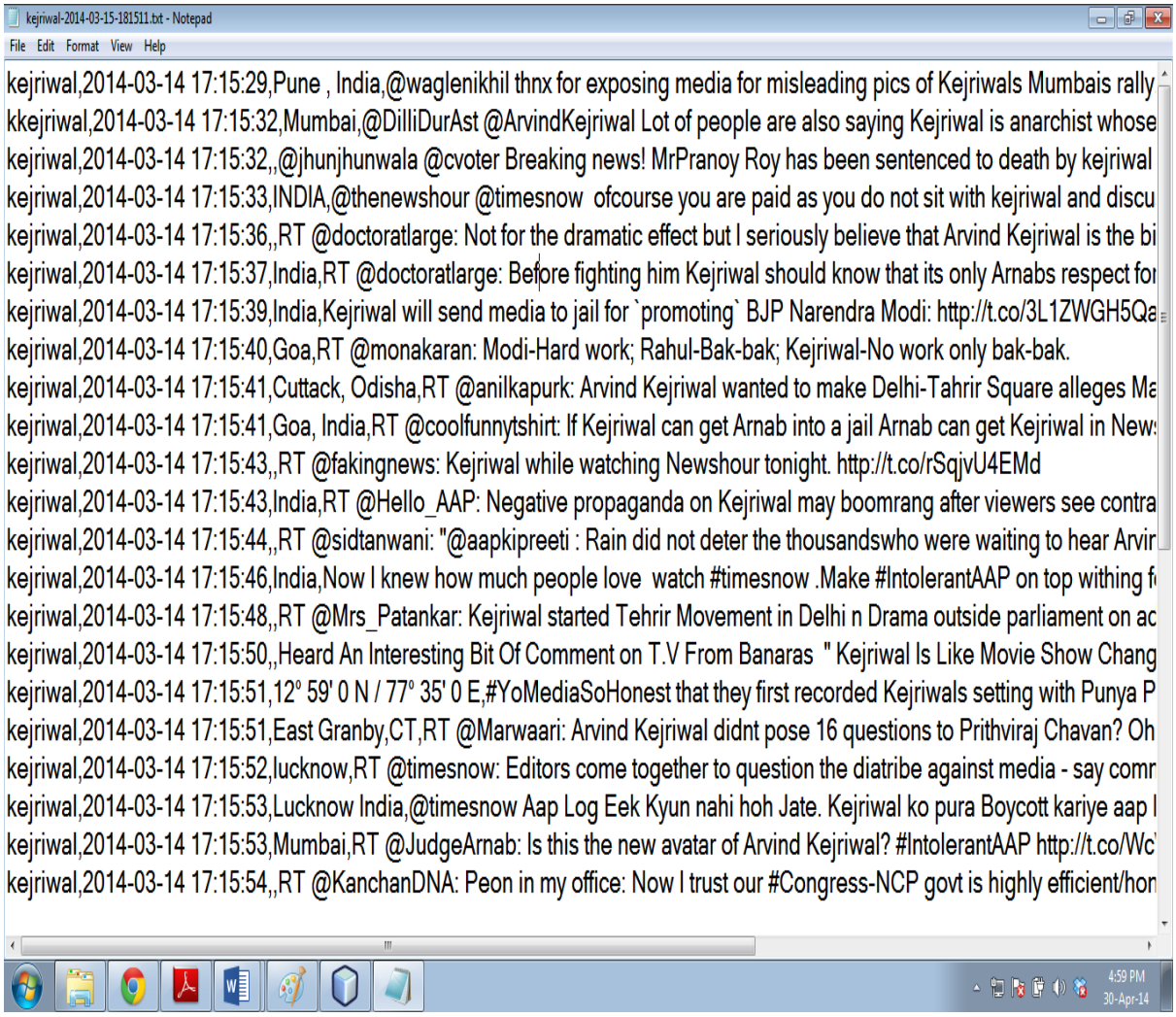
A file containing misspellings and corrected words is used to implement this. The output is stored in the file 'spell_checked.txt'.

INPUT: the input file is a set of tweets which contains tweets directly extracted from twitter based on a keyword.it also contains many words which are abbreviated.
Our aim is to replace these words with their original form.
A code in spellchecker.java was used to do the same.

The following files were taken as input:

1. dictionary2.txt:
   this file contains all the keywords and their full forms
   Example: thnx thanks

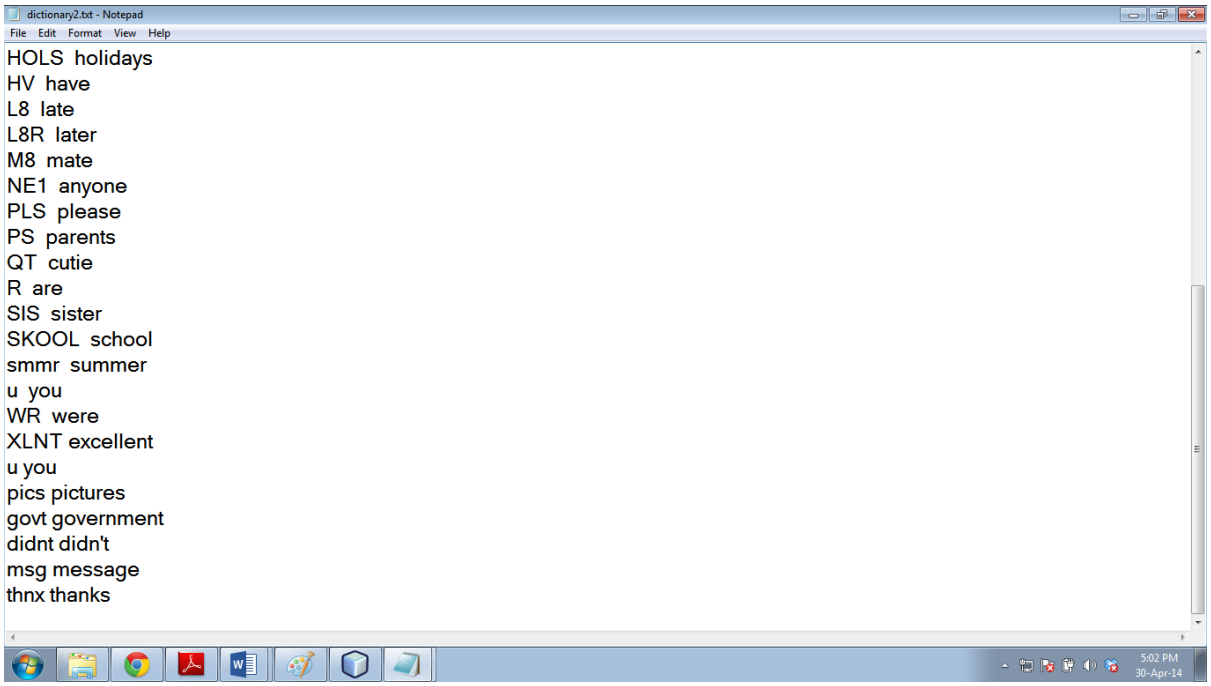
2. kejriwal-2014-03-15-181511.txt
   this is the input file given



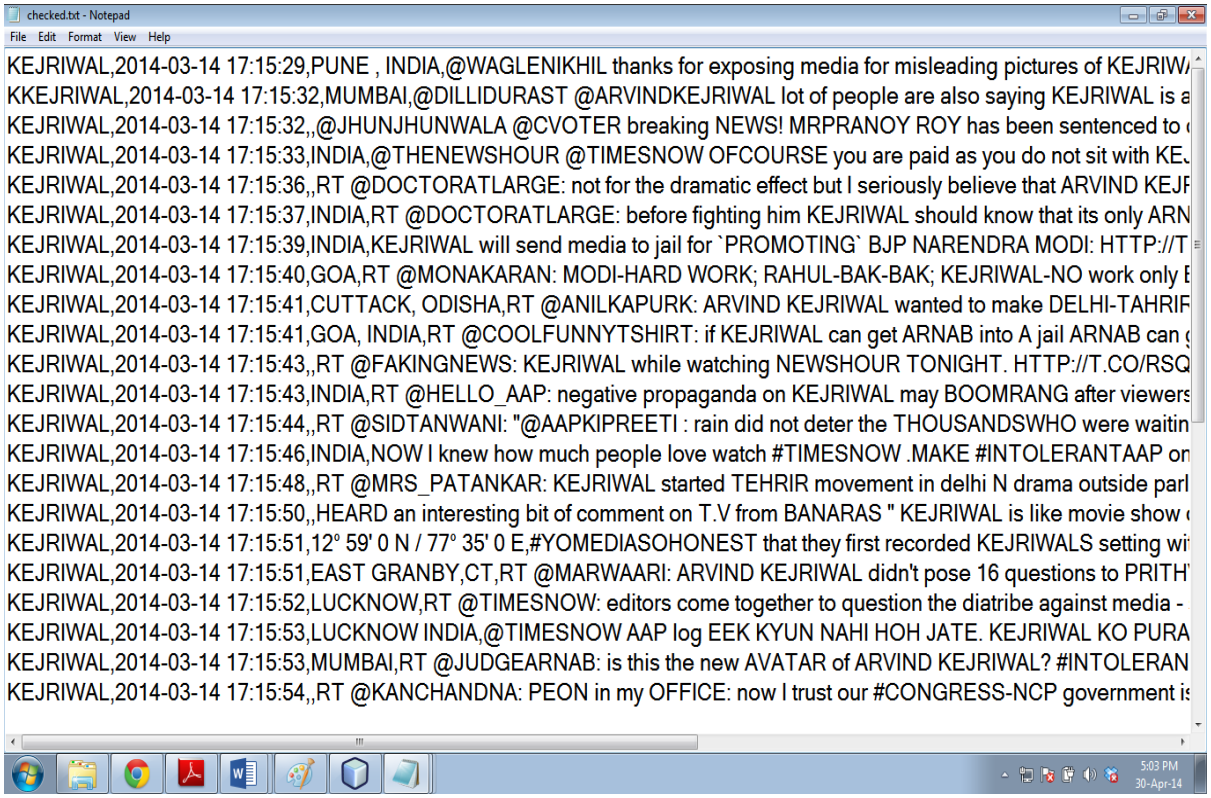
kejriwal-2014-03-15-181511.txt

This file contains the words ‘thnx; and govt which have to be replaced

dictionary2.txt:



output file:



EVENT IDENTIFICATION

User-contributed messages on social media sites such as Twitter have emerged as powerful, real-time means of information sharing on the Web. Approaches are explored for analyzing the stream of Twitter messages to distinguish between messages about real-world events and non-event messages.
The approach relies on a rich family of aggregate statistics of topically similar message clusters, including temporal, social, topical, and Twitter-centric features.

We define an event as a real-world occurrence e with
(1) An associated time period Te and
(2) A time-ordered stream of Twitter messages Me, of substantial volume, discussing the occurrence and published during time Te.

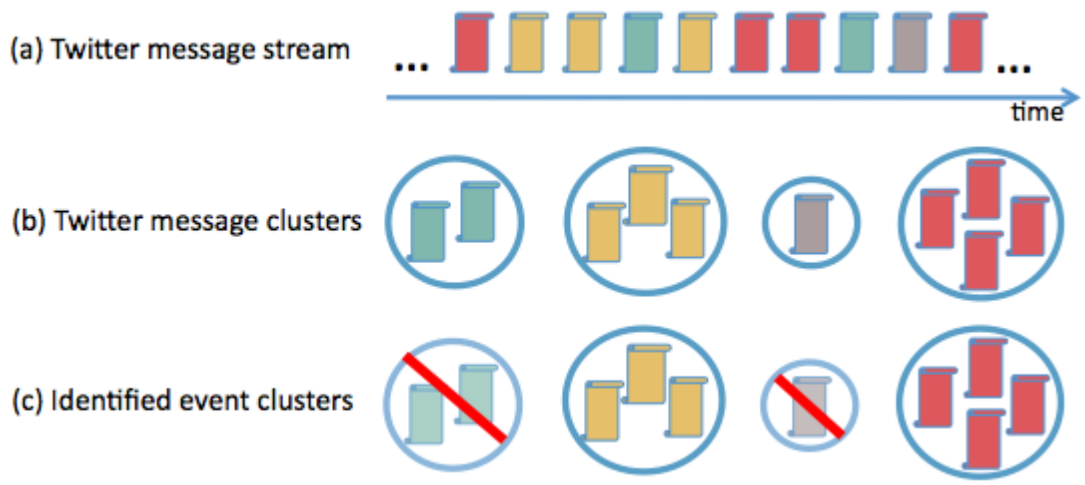
According to this definition, events on Twitter include widely known occurrences such as the presidential inauguration, and also local or community-specific events such as a high-school homecoming game or the ICWSM conference. Non-event content, of course, is prominent on Twitter and similar systems where people share various types of content such as personal updates, random thoughts and musings, opinions, and information

As a challenge, non-event content also includes forms of Twitter activity
that trigger substantial message volume over specific time periods which is a common characteristic of event content. Examples of such non-event activity are Twitter-specific conversation topics or memes (e.g., using the hashtag #thingsparentssay), and retweet activities,

characterized by a \storm" of retweets of popular Twitter users (e.g., an inspiring comment by a famous movie star).

The goal is to differentiate between messages about real-world events and non-event messages, where non-event messages include those for \trending" activities that are Twitter-centric but do not reflect any real-world occurrences

Consider a time-ordered stream of Twitter messages M. At any point in time t, our goal is to identify real-world events and their associated Twitter messages present in M and published before time t. Furthermore, we assume an online setting for our problem, where we only have access to messages posted before time t.



CLUSTERING

Clustering as we know is grouping similar things. Data clustering is grouping of similar text. It is of three major types:-

1. Hierarchical Clustering
2. Partitional Clustering
    - Column Clustering
    - K-Means Clustering

An incremental, online clustering algorithm was used in order to effectively cluster a stream of Twitter messages in real time. For such a task, a clustering algorithm that is scalable, and that does not require a priori knowledge of the number of clusters was chosen, since Twitter messages are constantly evolving and new events get added to the stream over time.

Python code CLUSTERS.PY is used to cluster the data set

CODES

Spellchecker.java : used for spell checker

```
import java.io.*;
import java.util.StringTokenizer;
 import java.util.*;


 class SpellChecker {

   Hashtable dictionary;

   public static void main(String argv[]) {
      SpellChecker checker = new SpellChecker();
   }

    SpellChecker() {
      dictionary = new Hashtable(53);

      try {
         String s, token;

         // 2.a.  Read dictionary from file
         BufferedReader dictReader = new BufferedReader(new
FileReader("C:\\Users\\KANIKA\\Documents\\NetBeansProjects\\Jav
aApplication1\\src\\dictionary.txt"));

         while ((s = dictReader.readLine())!= null) {
            StringTokenizer st = new StringTokenizer(s);

            while (st.hasMoreTokens()) {
               token = st.nextToken();
               // key and value are identical
               dictionary.put(token, token);
            }
```

```
        }
        dictReader.close();

        //  2.b.  Read file with commonly misspelled words.
        BufferedReader misspellings = new BufferedReader(new
FileReader("C:\\Users\\KANIKA\\Documents\\NetBeansProjects\\Jav
aApplication1\\src\\dictionary2.txt"));
        String wrongSpelling, rightSpelling;

        while ((s = misspellings.readLine()) != null) {
           StringTokenizer st = new StringTokenizer(s);

           // should really check that both of these exist
           wrongSpelling = st.nextToken();
           rightSpelling = st.nextToken();

           // key is the wrong spelling, value is the correct spelling
           dictionary.put(wrongSpelling, rightSpelling);

        }
        misspellings.close();

        // 2.c-f. Read in the text file to be checked & output corrected
file.

        BufferedReader inputFile = new BufferedReader(new
FileReader("C:\\Users\\KANIKA\\Documents\\NetBeansProjects\\Jav
aApplication1\\src\\kejriwal-2014-03-15-181511.txt"));
        BufferedWriter outputFile = new BufferedWriter(new
FileWriter("C:\\Users\\KANIKA\\Documents\\NetBeansProjects\\Jav
aApplication1\\src\\checked.txt"));

        // Preserve original line breaks by reading in one line at a
time.  Note, however, that
```

```
        // we do not preserve other whitespace, nor do we handle
punctuation.
        while ((s = inputFile.readLine()) != null) {
           StringTokenizer st = new StringTokenizer(s);

           while (st.hasMoreTokens()) {
              String inputWord = st.nextToken();
              String outputWord = spellCheckWord(inputWord);
              outputFile.write(outputWord+" ");
           }
           outputFile.newLine();
        }

        inputFile.close();
        outputFile.close();
     }
     catch (IOException e) {
        System.out.println("Error -- " + e.toString());
        e.printStackTrace();
        System.exit(-1);
     }
  }

  public String spellCheckWord(String wordToCheck) {
     String lookup, uninflectedWord;
     String word = wordToCheck.toLowerCase();

     // if spelt correctly, output as is
     // if it is a common mispelling, output the corrected word
     if ((lookup = (String)dictionary.get(word)) != null)
        return lookup;

     // Remove inflections at end of word and try again ("es", "s",
"ing", "ed")
     int length = word.length();
```

```
// first check for final 's'.
if (length > 1 && word.substring(length - 1).equals("s")) {
   uninflectedWord = word.substring(0, length-1);

   if ((lookup = (String)dictionary.get(uninflectedWord)) != null)
      return lookup + "s";
   // don't fail yet.  fall through to 'es' check
}

if (length > 2 && word.substring(length-2).equals("es")) {
   uninflectedWord = word.substring(0, length-2);

   if ((lookup = (String)dictionary.get(uninflectedWord)) != null)
      return lookup + "es";
   else // not found
      return word.toUpperCase();
}

if (length > 3 && word.substring(length - 3).equals("ing")) {
   uninflectedWord = word.substring(0, length-3);

   if ((lookup = (String)dictionary.get(uninflectedWord)) != null)
      return lookup + "ing";
   else // not found
      return word.toUpperCase();
}

if (length > 2 && word.substring(length - 2).equals("ed")) {
   uninflectedWord = word.substring(0, length-2);

   if ((lookup = (String)dictionary.get(uninflectedWord)) != null)
      return lookup + "ed";
   else // not found
      return word.toUpperCase();
```

```
        }

        // word was not found, even after "uninflecting".  Assume it is
misspelt and return
        // it in ALL CAPS.
        return word.toUpperCase();
    }
```

Clusters.py

This code takes data file as input and compares them to cluster basis on the similarity or common words.

```
from PIL import Image,ImageDraw

defreadfile(filename):
lines=[line for line in file(filename)]
colnames=lines[0].strip( ).split('\t')[1:]
rownames=[]
data=[]
for line in lines[1:]:
        p=line.strip( ).split('\t')
rownames.append(p[0])
data.append([float(x) for x in p[1:]])
returnrownames,colnames,data

from math import sqrt
defpearson(v1,v2):
```

```
   sum1=sum(v1)
   sum2=sum(v2)
   sum1Sq=sum([pow(v,2) for v in v1])
   sum2Sq=sum([pow(v,2) for v in v2])
pSum=sum([v1[i]*v2[i] for i in range(len(v1))])
num=pSum-(sum1*sum2/len(v1))
den=sqrt((sum1Sq-pow(sum1,2)/len(v1))*(sum2Sq-
pow(sum2,2)/len(v1)))
if den==0: return 0
return 1.0-num/den


classbicluster:
def __init__(self,vec,left=None,right=None,distance=0.0,id=None):
self.left=left
self.right=right
self.vec=vec
      self.id=id
self.distance=distance



defhcluster(rows):
distances={}
currentclustid=-1
clust=[bicluster(rows[i],id=i) for i in range(len(rows))]
whilelen(clust)>1:
lowestpair=(0,1)
closest=pearson(clust[0].vec,clust[1].vec)
for i in range(len(clust)):
for j in range(i+1,len(clust)):
if (clust[i].id,clust[j].id) not in distances:
distances[(clust[i].id,clust[j].id)]=pearson(clust[i].vec,clust[j].vec)
            d=distances[(clust[i].id,clust[j].id)]
if d<closest:
```

```
closest=d
lowestpair=(i,j)

mergevec=[
      (clust[lowestpair[0]].vec[i]+clust[lowestpair[1]].vec[i])/2.0
for i in range(len(clust[0].vec))]
newcluster=bicluster(mergevec,left=clust[lowestpair[0]],
right=clust[lowestpair[1]],
distance=closest,id=currentclustid)
currentclustid-=1
delclust[lowestpair[1]]
delclust[lowestpair[0]]
clust.append(newcluster)
returnclust[0]

defprintclust(clust,labels=None,n=0):
for i in range(n): print ' ',
if clust.id<0:
print '-'
else:
if labels==None: print clust.id
else: print labels[clust.id]
ifclust.left!=None: printclust(clust.left,labels=labels,n=n+1)
ifclust.right!=None: printclust(clust.right,labels=labels,n=n+1)

defgetheight(clust):
ifclust.left==None and clust.right==None: return 1
returngetheight(clust.left)+getheight(clust.right)

defgetdepth(clust):
ifclust.left==None and clust.right==None: return 0
returnmax(getdepth(clust.left),getdepth(clust.right))+clust.distance

defrotatematrix(data):
newdata=[]
```

```
for i in range(len(data[0])):
newrow=[data[j][i] for j in range(len(data))]
newdata.append(newrow)
returnnewdata


defdrawdendrogram(clust,labels,jpeg='clusters.jpg'):
h=getheight(clust)*20
   w=1200
depth=getdepth(clust)
scaling=float(w-150)/depth
img=Image.new('RGB',(w,h),(255,255,255))
draw=ImageDraw.Draw(img)
draw.line((0,h/2,10,h/2),fill=(255,0,0))
drawnode(draw,clust,10,(h/2),scaling,labels)
img.save(jpeg,'JPEG')

defdrawnode(draw,clust,x,y,scaling,labels):
if clust.id<0:
      h1=getheight(clust.left)*20
      h2=getheight(clust.right)*20
top=y-(h1+h2)/2
bottom=y+(h1+h2)/2
ll=clust.distance*scaling
draw.line((x,top+h1/2,x,bottom-h2/2),fill=(255,0,0))
draw.line((x,top+h1/2,x+ll,top+h1/2),fill=(255,0,0))
draw.line((x,bottom-h2/2,x+ll,bottom-h2/2),fill=(255,0,0))
drawnode(draw,clust.left,x+ll,top+h1/2,scaling,labels)
drawnode(draw,clust.right,x+ll,bottom-h2/2,scaling,labels)
else:
draw.text((x+5,y-7),labels[clust.id],(0,0,0))
```

REFERENCES

- https://apps.twitter.com/
- https://apps.twitter.com/app/5854468/show
- https://pypi.python.org/pypi/tweepy
- http://www.cs.ccsu.edu/~markov/ccsu_courses/DataMining-3.html