**Context Filter and Analysis**

As previous report illustrated, it need to make use of the captured tweets under the protected hashtag from internet to build a database. And analyze every tweet and rank its risk level. For a new tweet, compare and analyze it with the database to rank its risk level and make alert to warn the protected organization.

**Implementation**

The implementation includes two main steps: hashtag dictionary setup and new tweet analysis. The following graph is the specific steps.

Hashtag Dictionary Setup

New Tweet Analysis

The entire program is aimed to giving a threat warning to the protected organization when there is a new tweet. Therefore, build a hashtag dictionary is the first step. The program uses the Twitter Search API, which allows queries against the indices of recent or popular Tweets. It uses hashtag as key word and gets the relevant tweets under the hashtag.

new\_tweets = api.search(search\_text, count = count, max\_id = str(last\_id -1))

**for** msg **in** new\_tweets:

msg = msg.text.encode(**"utf-8"**)

The search\_number for the total tweets number is assigned by user input. When it gets all the tweets, change the encoding type and filter the useless information. The following is the steps to filter.

1. Filter emoji.
2. Filter special characters.
3. Filter address, @, RT, &.
4. Filter tweets will hashtag #Ad.
5. Filter the top 300 of most common characters using Project Gutenberg digital library.
6. Only keep characters use regular expression [^a-zA-Z0-9\-\# ].

**msg\_filters\_format = ["https://", "&", "RT", "@"]**

**msg\_filters\_out = ["#Ad"]**

**msg\_filters\_words = ["https", "rt"]**

**……**

**def takeout(msg, filter\_format = msg\_filters\_format, filter\_words = msg\_filters\_words):**

**msg = msg.lower()**

**for char in filter\_format:**

**buffer = ""**

**for word in msg.split():**

**if char not in word:**

**buffer = buffer + word + " "**

**msg = buffer**

**msg = msg[:-1]**

**for words in filter\_words:**

**words = words.lower()**

**#print char**

**buffer = ""**

**for word in msg.split():**

**#print "word:", word**

**if words != word:**

**buffer = buffer + word + " "**

**msg = buffer**

**msg = re.sub('[^a-zA-Z0-9\-\# ]+', '', msg)**

**return msg[:-1]**

After filtering, it will build two files: hashtag\_name.txt, which is context after filtering and hashtag\_name\_bak.txt, which contains original tweets and amis for debug. In the files, every tweet in is one line.

f = open(hashtag + **'.txt'**, **"w+"**)  
 g = open(hashtag + **'\_bak.txt'**, **"w+"**)

The next step is analyzing the value of tweets. It adopts the deformation of TF-IDF algorithm. First, it will split every tweet into words and count the number of tweets **cnt[word]** which include this word. The total number of tweets is **search\_number** what we set. So the value of one word is:

**Value = search\_number / cnt[word]**

The sum of every word in this tweet **sum** and the number of word in this tweet is **num**, the value of this tweet is:

**Value = sum /num**

**with** open (hashtag + **'.txt'**) **as** f:  
 **for** line **in** f:  
 line\_count += 1**for** words **in** line.lower().split():  
 **if** word == words:  
 cnt\_idf[word] += 1  
 **break**

**with** open (hashtag + **'.txt'**) **as** f:  
 **for** line **in** f:  
 **for** word **in** line.lower().split():  
 **if** word **not in** filter\_words:  
 **print** word, cnt[word], cnt\_idf[word]  
 sum = sum + float(search\_number)/cnt\_idf[word]  
 num += 1sum = sum/num

After calculating the value of each tweet, next step is sorting them from largest value to lowest. And then rank the risk level of them. The rule for risk level is as following table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Level Class | Level 1 | Level 2 | Level 3 | Level 4 |
| Ranking rate (%) | 0 ~ 0.1 | 0.1 ~ 0.5 | 0.5 ~ 2 | 2 ~ 5 |
| Threat Degree | Very high | High | Medium | Small |

tweet\_level = sorted(tweet\_level, key = float, reverse = True)  
  
level.append(tweet\_level[int(search\_number\*0.025)])  
level.append(tweet\_level[int(search\_number\*0.075)])  
level.append(tweet\_level[int(search\_number\*0.175)])  
level.append(tweet\_level[int(search\_number\*0.375)])

Through the above steps, the hashtag dictionary is finished. It is used as a database when we have a new tweet. When a new tweet is captured, first, it will do the filtering and keep the useful words. Second, calculate the value of the tweet. It is a little different from the above formula because the total number is changed.

**Value = (search\_number +1) / (cnt[word]+1)**

After calculating the value, then compare with tweets in the dictionary and rank its risk level. Finally, it will make alert to warn the protected organizations.

**Analysis of Outcome**

When we run this program and test the certain hashtag “#Trump”, the log of the system execution is followed below:

RT @ABCMundial: #Trump se enojó después de las críticas de China por hablar con Tsai Ing-wen

#Taiwan @Taiwan\_Argentin @ABCMundial…

---------------------------Analysis-------------------------------

word: se inverse document frequcney: 71.4642857143

word: enoj inverse document frequcney: 2001.0

word: despus inverse document frequcney: 2001.0

word: las inverse document frequcney: 2001.0

word: crticas inverse document frequcney: 2001.0

word: china inverse document frequcney: 45.4772727273

word: por inverse document frequcney: 222.333333333

word: hablar inverse document frequcney: 2001.0

word: con inverse document frequcney: 100.05

word: tsai inverse document frequcney: 2001.0

word: ing-wen inverse document frequcney: 2001.0

word: #taiwan inverse document frequcney: 71.4642857143

total value: 14517.7891775 effective words: 12

risk value: 1209.81576479

risk level break point: [1000.5, 500.25, 250.125, 166.75]

risk level: 1

--------------------------------------------------------------

RT @CoqResistant: En 2016 je suis 🇺🇸#Trump, je suis 🇬🇧#Brexit, aujourd'hui je suis 🇮🇹#referendumcostituzionale #RenziACasa

En 2017 j'espèr…

---------------------------Analysis-------------------------------

word: en inverse document frequcney: 24.4024390244

word: 2016 inverse document frequcney: 62.53125

word: je inverse document frequcney: 1000.5

word: suis inverse document frequcney: 2001.0

word: #trump inverse document frequcney: 7.32967032967

word: je inverse document frequcney: 1000.5

word: suis inverse document frequcney: 2001.0

word: #brexit inverse document frequcney: 10.5315789474

word: aujourdhui inverse document frequcney: 2001.0

word: je inverse document frequcney: 1000.5

word: suis inverse document frequcney: 2001.0

word: #referendumcostituzionale inverse document frequcney: 400.2

word: #renziacasa inverse document frequcney: 2001.0

word: en inverse document frequcney: 24.4024390244

word: 2017 inverse document frequcney: 181.909090909

word: jespr inverse document frequcney: 2001.0

total value: 15718.8064682 effective words: 16

risk value: 982.425404265

risk level break point: [1000.5, 500.25, 250.125, 166.75]

risk level: 2

--------------------------------------------------------------

RT @TheOneLadyEagle: #TRUMP, WE WANT AN AUDIT!!!!! WE WANT TO SEE THE RESULTS!!!!!! https://t.co/cgPAA6NHeX

---------------------------Analysis-------------------------------

word: #trump inverse document frequcney: 7.32967032967

word: audit inverse document frequcney: 1000.5

word: results inverse document frequcney: 500.25

total value: 1508.07967033 effective words: 3

risk value: 502.693223443

risk level break point: [1000.5, 500.25, 250.125, 166.75]

risk level: 2

--------------------------------------------------------------

RT @GoldStarMomTX55: #OpenBorders

EXTREME VETTING

#BuildTheWall #Trump

Daughter top EU official raped &amp; murdered Germany

Afghan migrant h…

---------------------------Analysis-------------------------------

word: #openborders inverse document frequcney: 1000.5

word: extreme inverse document frequcney: 1000.5

word: vetting inverse document frequcney: 1000.5

word: #buildthewall inverse document frequcney: 1000.5

word: daughter inverse document frequcney: 500.25

word: top inverse document frequcney: 333.5

word: eu inverse document frequcney: 60.6363636364

word: official inverse document frequcney: 333.5

word: raped inverse document frequcney: 1000.5

word: murdered inverse document frequcney: 1000.5

word: germany inverse document frequcney: 667.0

word: afghan inverse document frequcney: 1000.5

word: migrant inverse document frequcney: 1000.5

word: h inverse document frequcney: 16.4016393443

total value: 9915.28800298 effective words: 14

risk value: 708.234857356

risk level break point: [1000.5, 500.25, 250.125, 166.75]

risk level: 2

--------------------------------------------------------------

RT @UnsolvedRHYME: This is a GREAT CLIP from

one of our country's HEROES,

DICK Cheney. He let's #CNN

Reporterette know that #TRUMP

c…

---------------------------Analysis-------------------------------

word: clip inverse document frequcney: 2001.0

word: countrys inverse document frequcney: 2001.0

word: heroes inverse document frequcney: 2001.0

word: dick inverse document frequcney: 1000.5

word: cheney inverse document frequcney: 2001.0

word: lets inverse document frequcney: 400.2

word: #cnn inverse document frequcney: 400.2

word: reporterette inverse document frequcney: 2001.0

word: c inverse document frequcney: 181.909090909

total value: 11987.8090909 effective words: 9

risk value: 1331.97878788

risk level break point: [1000.5, 500.25, 250.125, 166.75]

risk level: 1

--------------------------------------------------------------