

ANALYSIS WITH PROGRAMMING

cout << "let's do some analysis and programming" << endl;

<http://alstatr.blogspot.com/>

TEXT MINING ON TWITTER #PRAYFORMH370

MALAYSIA AIRLINES

R Programming

21 of March 2014

Al-Ahmadgaid B. Asaad

alstated@gmail.com

It's been two weeks for search and rescue operations of the Malaysia Airlines Flight MH370, after it vanished from the radar on March 8, 2014. And wherever they are, we hope and pray for them.

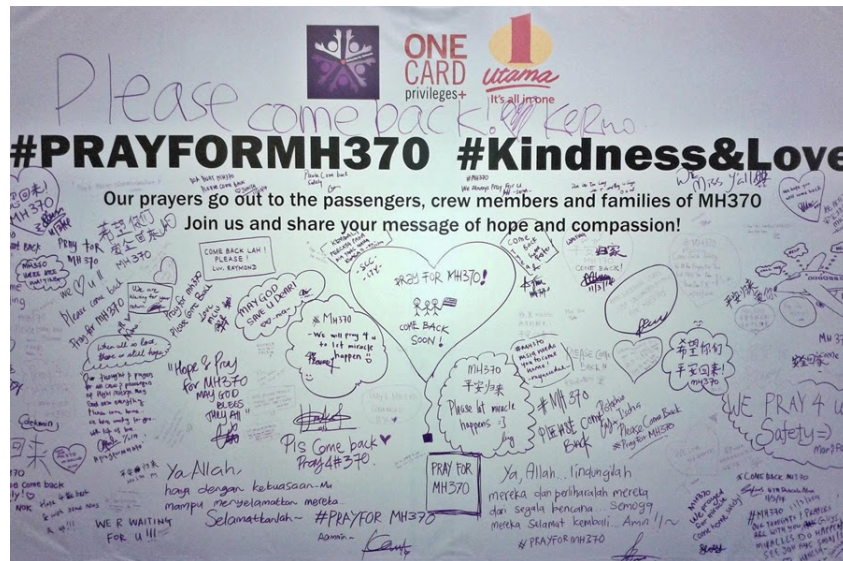
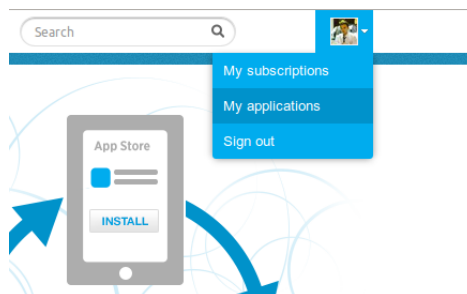


Photo from VENUS - Wall of Hope & Prayers for MH370

In this post, we are going to do text data mining on Twitter tweets containing #PrayForMH370 from March 8, to March 20, 2014 using Twitter API. First, we need to have an authentication on the Twitter API, to obtain the data. In the proceeding tutorial, the idea and codes for Twitter authentication were based from Julianhi's amazing blog, and I am going to replicate his code to save a copy of it.

Go to <https://dev.twitter.com/>, and sign in with your twitter account. When you're done, click on your profile picture on the top right corner like the one below.



Click on **My applications**, then **Create New App**. Fill in what's necessary (i.e. Name, Description and Website).

Create an application

Application details

Name *

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

Description *

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

Website *

Your application's publicly accessible home page, where users can go to download, make use of, or 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? [OAuth 1.0a](#) applications should explicitly

Julianhi recommends `http://test.de/` for valid Website, and name it anything you want (same case for description). To finish this, check **Yes, I agree** on Developer Rules of the Road, and click on **Create your Twitter application**. You will have something like this,

MostAwesomeApp

Details Settings API Keys Permissions

 The Awesome App that you shouldn't use.
<http://test.de/>

Organization

Information about the organization or company associated with your application. This information is optional.

Organization	None
Organization website	None

« »

Keep this open on your browser, and go to your R script. Copy and paste the following code,

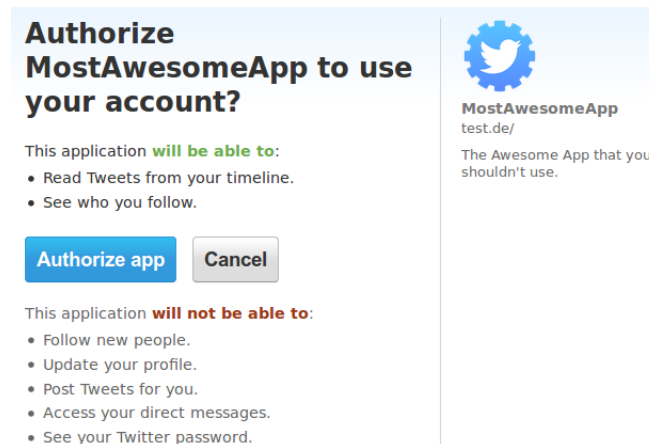
```
> library(RCurl)
> # Set SSL certs globally
> options(RCurlOptions = list(cainfo = system.file("CurlSSL",
+         "cacert.pem", package = "RCurl")))
>
> library(twitterR)
> reqURL <- "https://api.twitter.com/oauth/request_token"
> accessURL <- "https://api.twitter.com/oauth/access_token"
```

```
> authURL <- "https://api.twitter.com/oauth/authorize"
> apiKey <- "xxxxxxxxxxxxxxxxxxxxxxxx"
> apiSecret <- "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
>
> twitCred <- OAuthFactory$new(
+   consumerKey = apiKey,
+   consumerSecret = apiSecret,
+   requestURL = reqURL,
+   accessURL = accessURL,
+   authURL = authURL
+ )
>
> twitCred$handshake(
+   cainfo = system.file("CurlSSL", "cacert.pem", package = "RCurl")
+ )
>
> registerTwitterOAuth(twitCred)
```

Replace the `apiKey` and `apiSecret` with the one on the API Keys tab of your Twitter app (see above photo – refer back to your browser), then run the above code. If you encounter error that says **Error: Unauthorized**, make sure to remove the spaces (if any) both at the end of your `apiKey` or `apiSecret` between the quotations, i.e.

```
> apiKey <- "(no space here)xxxx....xx(no space here)"
> apiSecret <- "(no space here)xxxx....xx(no space here)"
```

This happens especially when you do copy and paste (I'm guilty of that). But if there is no error, `twitCred$handshake` will return a link. Copy and paste this link to your browser, then authorize the app,



After clicking **Authorize app**, take note of the PIN and enter it to the R Console; finally, run the last function (line 24 above). Now that we have an access, we can extract data then. For this post, we will make a word cloud for #PrayforMH370 and #MH370 (bonus) on the said inclusive dates. Here it is, run the following:

```
> library(twitter)
> library(tm)
> library(wordcloud)
> library(RColorBrewer)
>
> mh370 <- searchTwitter("#PrayForMH370", since = "2014-03-08",
+                               until = "2014-03-20", n = 1000)
> mh370_text = sapply(mh370, function(x) x$getText())
> mh370_corpus = Corpus(VectorSource(mh370_text))
>
> tdm = TermDocumentMatrix(
+   mh370_corpus,
+   control = list(
+     removePunctuation = TRUE,
+     stopwords = c("prayformh370", "prayformh", stopwords("english")),
+     removeNumbers = TRUE, tolower = TRUE)
+   )
>
> m = as.matrix(tdm)
> # get word counts in decreasing order
> word_freqs = sort(rowSums(m), decreasing = TRUE)
> # create a data frame with words and their frequencies
> dm = data.frame(word = names(word_freqs), freq = word_freqs)
>
> wordcloud(dm$word, dm$freq, random.order = FALSE,
+           colors = brewer.pal(8, "Dark2"))
```

The above code for creating word cloud is originally from Mining twitter with R site, and below is the output of that,



If you notice in the code, line 6 above, I set `n = 1000`. This line will likely to give you a warning that says,

Warning message:

```
In doRppAPICall("search/tweets", n, params = params,
  retryOnRateLimit = retryOnRateLimit, : 1000 tweets were
  requested but the API can only return 599
```

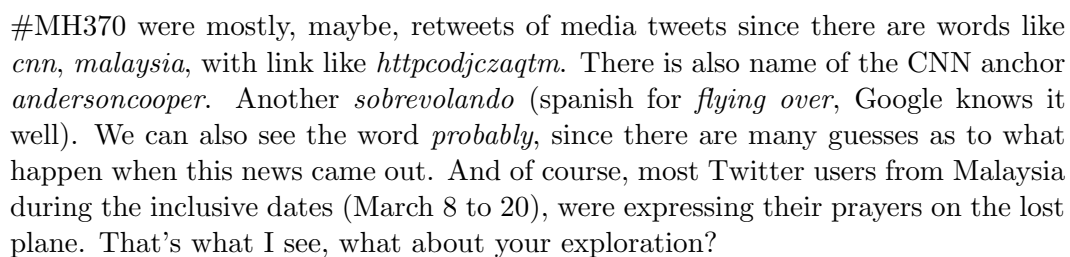
Just ignore this, as this simply mean that there are only 599 tweets in total that use #PrayforMH370 during the inclusive dates. Now, let's investigate the above word cloud. I know very little about Bahasa Melayu (Malay Language), but I can tell the words in high scale are likely to be Malay prepositions, or in computing term, Malay stop words. This is because we exclude Malay stop words (only 'prayformh370', 'prayformh' and stopwords('english') – English stop words) in the code as it is not yet supported by the stopwords function. Hence, not a good word cloud for tweets exploratory. So I look for a list of Malay stop words on the web, and led me to this site. Since this is what we are looking for, thus we import the list using the XML package.

```
> library(XML)
> df1 <- readHTMLTable('http://blog.kerul.net/2014/01/
+                       list-of-malay-stop-words.html')
> df1 <- df1[[1]]
> malaystopwords <- as.character(unlist(df1))[-c(320, 321)]
> head(malaystopwords)
# OUTPUT
[1] "ada" "adakah" "adakan" "adalah" "adanya" "adapun"
```

Supplying the previous code with the Malay stopwords, thus we have

```
> tdm = TermDocumentMatrix(
+   mh370_corpus,
+   control = list(
+     removePunctuation = TRUE,
+     stopwords = c("prayformh370", "prayformh",
+                   malaystopwords, stopwords("english")),
+   removeNumbers = TRUE, tolower = TRUE)
+ )
> m = as.matrix(tdm)
> # get word counts in decreasing order
> word_freqs = sort(rowSums(m), decreasing = TRUE)
> # create a data frame with words and their frequencies
> dm = data.frame(word = names(word_freqs),
+                 freq = word_freqs)
> wordcloud(dm$word, dm$freq, random.order = FALSE,
+           colors = brewer.pal(8, "Dark2"))
```

There is the difference, *dengannya*, *telah* and *bagi* were actually stop words. And base from this multilingual guy (Google Translate), the “*waktu solat masuk*” means “*prayer time in*”. You can play with the words above. For #MH370, using the same codes with Malay and all available stop words from stopwords function, we have And why is it in square? and not circle? Well that is due to the small screen I have and large number of dataset (1000 data points). So, from the two plots we have (#PrayForMH370 and #MH370), we can say subjectively, that tweets under



Labels

Data Mining, R, Tutorial,