Sentiment Analysis on Twitter with R

IE231 - Lecture Notes 12

May 9, 2017

In this tutorial, we are going to search for tweets about Ethereum and do sentiment analysis using `tidytext` package. First there is some menial work to get API keys from Twitter.

Setting Up Twitter and API keys

Step 0. If you do not have a Twitter account, create one.

Step 1.

Enter https://apps.twitter.com. Create a new app.

Fill the necessary information. You can ask for read and write permissions (but read is enough).

Step 2.

Find API keys in your application.

Also note the access token and secret.

Now that we got the necessary info, we can return to R. Just a word of caution. Twitter might block your account due to "suspicious use" of your app. Though it is only temporary. You should write them a paragraph explaining you are a student (if you are) and you are doing some experimental stuff (i.e. sentiment analysis) using your own API and own account. They will reopen your account quickly. (Yes, Twitter has a lot to improve.)

Warning! Never, ever write your API keys explicitly to a document.

Harnessing Twitter Information Using R

Setup

Start with loading required packages and setting up Twitter credentials.

Figure 1:

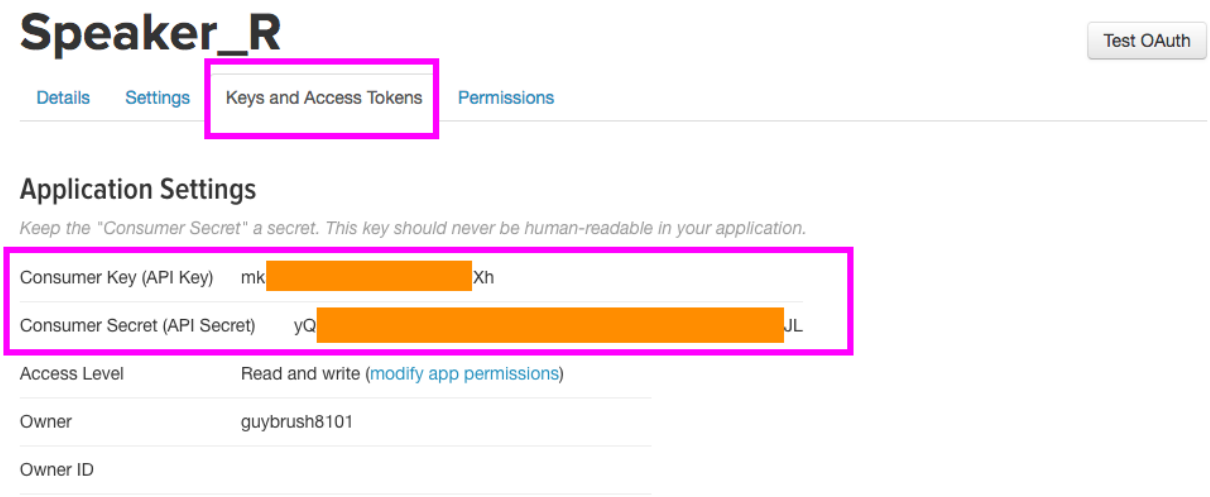


Figure 2:

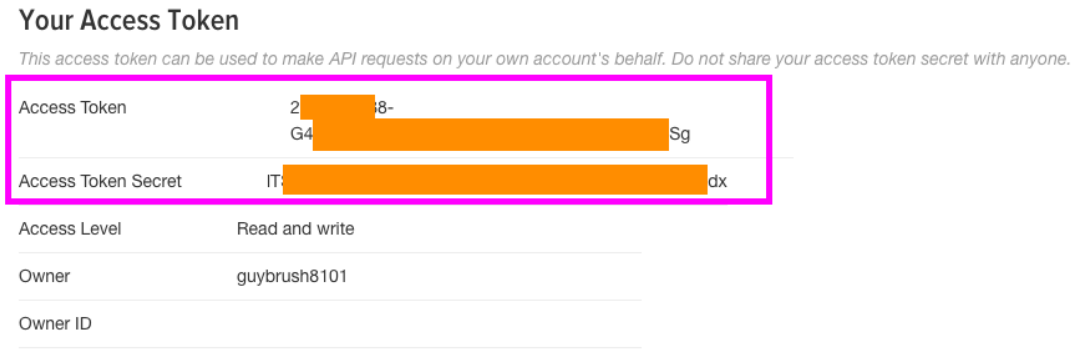


Figure 3:

```r
#Load required packages. Install before, if not installed.
#Package to get Tweets and lots of cool stuff
library(twitteR)
#Package to manipulate data sets
library(tidyverse)
#Package for text mining
library(tidytext)
```

```r
the_api_key <- "<YOUR APPLICATION API KEY HERE>"
the_api_secret <- "<YOUR APPLICATION API SECRET HERE>"
the_access_token <- "<YOUR ACCESS KEY HERE>"
the_access_secret <- "<YOUR ACCESS SECRET HERE>"
#This function will set your Twitter session with the provided keys.
setup_twitter_oauth(
    consumer_key=the_api_key,
    consumer_secret=the_api_secret,
    access_token=the_access_token,
    access_secret=the_access_secret
)
```

Congratulations! You can now start getting tweets.

Searching for Ethereum

`twitteR` package provides us with a simple wrapper function to get tweets from Twitter Search. There are other parameters (check `?searchTwitter`) but we are going to use only some of them.

`searchString` is our query parameter. We are going to write "Ethereum" there to find results related to Ethereum (it is quite a unique name, so no worries). We will also add `-filter:retweets` to remove RTs from the results.

`n` is the number of results we desire. Remember, Twitter placed rate limits to its API. So, you might not get 100,000 tweets about the query. Though, you can put a number on `retryOnRateLimit` parameter to retry the query. It takes time but eventually you will be building a tweet base.

`lang` is the language of Tweets, based on the account's preferred language. We set it to "en" to get English tweets.

With `resultType`, you can get "popular", "recent" or "mixed" type tweets. Definition of popular comes from Twitter.

```r
ethereum_tweets <-
    searchTwitter(
        searchString="Ethereum -filter:retweets",
        n=1000,
        retryOnRateLimit=120,
        lang="en",
        resultType="mixed",

    )
```

Let's see the results. Needless to say, you will get different results from Twitter since the time of your query is different.

```r
print(head(ethereum_tweets))
```

```
## [[1]]
```

```
## [1] "coindesk: Ethereum has provided details on how a major change called proof-of-stake will be depl
##
## [[2]]
## [1] "Excellion: Agree with Luke. Plus no one should listen to Vitalik's advice on hard-forks. Ethereu
##
## [[3]]
## [1] "Excellion: Yep. #oldjeffgarzik was smart. The new Garzik needs to sell Ethereum though, so we mu
##
## [[4]]
## [1] "StakepoolCom: 'Ethereum' article is now on main stream and media in Korea Check it out! https:/
##
## [[5]]
## [1] "MurphyAnalyst: #ethereum is in an upward momentum right now, making new highs almost everyday."
##
## [[6]]
## [1] "BlockChannel: nickjohnson: Can you link to the transaction hash? https://t.co/vlPajmAtf0"
```

The output is a different object format (similar to list). Let's peek at the structure of the object.

```
print(str(ethereum_tweets[[1]]))
```

```
## Reference class 'status' [package "twitteR"] with 17 fields
##  $ text         : chr "Ethereum has provided details on how a major change called proof-of-stake wil
##  $ favorited    : logi FALSE
##  $ favoriteCount: num 75
##  $ replyToSN    : chr(0)
##  $ created      : POSIXct[1:1], format: "2017-05-06 15:00:06"
##  $ truncated    : logi FALSE
##  $ replyToSID   : chr(0)
##  $ id           : chr "860871802364579841"
##  $ replyToUID   : chr(0)
##  $ statusSource : chr "<a href=\"http://bufferapp.com\" rel=\"nofollow\">Buffer</a>"
##  $ screenName   : chr "coindesk"
##  $ retweetCount : num 57
##  $ isRetweet    : logi FALSE
##  $ retweeted    : logi FALSE
##  $ longitude    : chr(0)
##  $ latitude     : chr(0)
##  $ urls         :'data.frame':   1 obs. of  5 variables:
##   ..$ url         : chr "https://t.co/xoAwZPMZAv"
##   ..$ expanded_url: chr "http://www.coindesk.com/ethereums-big-switch-the-new-roadmap-to-proof-of-sta
##   ..$ display_url : chr "coindesk.com/ethereums-big-..."
##   ..$ start_index : num 106
##   ..$ stop_index  : num 129
##  and 53 methods, of which 39 are  possibly relevant:
##    getCreated, getFavoriteCount, getFavorited, getId, getIsRetweet,
##    getLatitude, getLongitude, getReplyToSID, getReplyToSN, getReplyToUID,
##    getRetweetCount, getRetweeted, getRetweeters, getRetweets,
##    getScreenName, getStatusSource, getText, getTruncated, getUrls,
##    initialize, setCreated, setFavoriteCount, setFavorited, setId,
##    setIsRetweet, setLatitude, setLongitude, setReplyToSID, setReplyToSN,
##    setReplyToUID, setRetweetCount, setRetweeted, setScreenName,
##    setStatusSource, setText, setTruncated, setUrls, toDataFrame,
##    toDataFrame#twitterObj
## NULL
```

Now we need only text for this tutorial. There is a base R function for that `sapply`.

```
#Get only text from the objects
eth_text <- sapply(ethereum_tweets, "[[", "text")
head(eth_text)
```

```
## [1] "Ethereum has provided details on how a major change called proof-of-stake will be deployed on t
## [2] "Agree with Luke. Plus no one should listen to Vitalik's advice on hard-forks. Ethereum is 100% c
## [3] "Yep. #oldjeffgarzik was smart. The new Garzik needs to sell Ethereum though, so we must think c
## [4] "'Ethereum' article is now on main stream and media in Korea Check it out! https://t.co/v4da77hyc
## [5] "#ethereum is in an upward momentum right now, making new highs almost everyday."
## [6] "nickjohnson: Can you link to the transaction hash? https://t.co/vlPajmAtf0"
```

Congratulations! You just got tweets from Twitter into an R vector.

Text Mining

Now we are going to combine the powers of two packages tidyverse and tidytext to easily do sentiment analysis. You can also check some of the references here (1,2,3).

We are going to get the tweets from our tweet vector. We will also need to do some operations on the text to remove links, some special characters in order to get proper words and meanings. In this tutorial we are just going to work on single words, not n-grams (i.e. multi word phrases that may contain extra meaning).

```
eth_words <-
#Create a tibble (kind of data frame)
tibble(tweet=eth_text) %>%
    #Remove all links, RT, ampersand and some other special characters.
    mutate(tweet = stringr::str_replace_all(tweet,
        "https://t.co/[A-Za-z\\d]+|http://[A-Za-z\\d]+|&amp;|&lt;|&gt;|RT|https|\'|\"",
        "")) %>%
    #Separate each tweet into words
    #Keep hashtags(#) and (@) since they are special character to Twitter
    #Never mind the regex, it is always complicated
    unnest_tokens(word,tweet,
        token="regex",
        pattern="([^A-Za-z_\\d#@']|'(?![A-Za-z_\\d#@]))") %>%
    #Remove stop words such as; and, or, before, after etc.
    anti_join(stop_words,by="word") %>%
    #Remove numbers
    filter(stringr::str_detect(word, "[a-z]"))
```

Let's see the words with the highest frequency in tweets.

```
eth_words %>%
    count(word,sort=TRUE) %>%
    print(n=25)
```

```
## # A tibble: 1,722 × 2
##                word     n
##               <chr> <int>
## 1         ethereum   532
## 2        #ethereum   411
## 3         #bitcoin   215
## 4          bitcoin   159
## 5      #blockchain   141
```

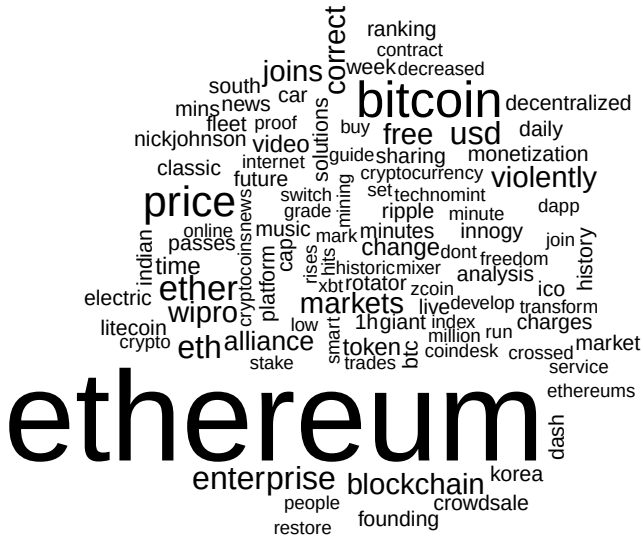
```
## 6            price   125
## 7            ether    89
## 8              usd    82
## 9             #eth    81
## 10      enterprise    76
## 11             eth    74
## 12            free    68
## 13         markets    68
## 14      blockchain    65
## 15         correct    62
## 16       violently    62
## 17           wipro    62
## 18           joins    57
## 19        alliance    56
## 20 #cryptocurrency    53
## 21            time    36
## 22          change    33
## 23           video    33
## 24           token    32
## 25         classic    31
## # ... with 1,697 more rows
```

Word Clouds

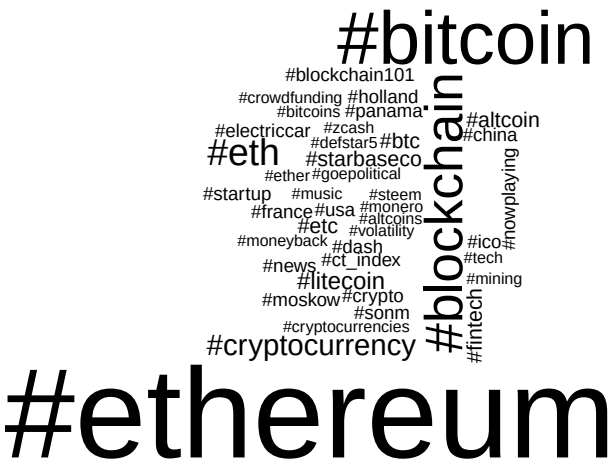
It is quite easy to create word clouds in R using the `wordcloud` package. Let's see the words, hashtags and users with the highest frequency.

```
#Install wordcloud if not installed
library(wordcloud)

#Plot the wordcloud with text only
eth_words %>%
    filter(!grepl("\\#|@",word)) %>%
    count(word,sort=TRUE) %>%
    with(wordcloud(word, n, max.words = 100))
```



```
#Plot the wordcloud of hashtags
eth_words %>%
    filter(grepl("\\#",word)) %>%
    count(word,sort=TRUE) %>%
    with(wordcloud(word, n, max.words = 100))
```




```
#Plot the wordcloud of user references
eth_words %>%
    filter(grepl("@",word)) %>%
    count(word,sort=TRUE) %>%
    with(wordcloud(word, n, max.words = 100))
```

Getting Sentiments

`tidytext` package contains three types of "sentiment dictionaries"; `afinn`, `bing` or `nrc`. Each word is associated with one or more sentiments. `bing` dictionary gives us positive/negative sentiments, `afinn` a scale between -5 and 5, and `nrc` provides more emotions such as anger, joy, fear etc. It is also possible to get `loughran` sentiment data set for finance specific sentiment analysis, but that version of the package is not on CRAN yet. See this post to learn how to load it.

We are going to use `bing` dictionary to get binary positive/negative results. See an example portion below.

```
get_sentiments("bing") %>% sample_n(10)
```

```
## # A tibble: 10 × 2
##            word sentiment
##           <chr>     <chr>
## 1   exuberantly  positive
## 2      conflict  negative
## 3        peeved  negative
## 4     unraveled  negative
## 5    stagnation  negative
## 6      fondness  positive
## 7  disgustingly  negative
## 8     reforming  positive
## 9        downer  negative
## 10     mangling  negative
```

We are going to associate `eth_words` with the sentiments data sets. There will be words without sentiments, so they will be removed from the data set.

```
#Get the sentiments
eth_bing_sentiments <-
eth_words %>%
    count(word,sort=TRUE) %>%
    inner_join(.,get_sentiments("bing"),by="word")

#See the data
print(eth_bing_sentiments)
```

```
## # A tibble: 140 × 3
##          word     n sentiment
##         <chr> <int>     <chr>
## 1        free    68  positive
## 2     correct    62  positive
## 3   violently    62  negative
## 4     classic    31  positive
## 5     freedom    15  positive
## 6       smart    12  positive
## 7    massacre     9  negative
## 8         led     8  positive
## 9  celebrated     6  positive
## 10     threat     6  negative
## # ... with 130 more rows
```

```
#Get the proportion of positive/negativeness
eth_bing_sentiments %>%
    group_by(sentiment) %>%
    summarise(occurence=sum(n)) %>%
    ungroup() %>%
    mutate(share=round(occurence/sum(occurence),2))
```

```
## # A tibble: 2 × 3
##   sentiment occurence share
##       <chr>     <int> <dbl>
## 1  negative       156  0.33
## 2  positive       321  0.67
```

It seems two-thirds of words in tweets contain positive sentiments about Ethereum.

Let's also put a wordcloud on sentiments. You will need `reshape2` package for this.

```
#install.packages(reshape2)
#Get the word cloud
eth_bing_sentiments %>%
    reshape2::acast(word ~ sentiment, value.var = "n", fill = 0) %>%
    comparison.cloud(colors = c("#F8766D", "#00BFC4"),
                     max.words = 100)
```

Final words

Now that you are familiar with Twitter API and sentiment analysis you can use and implement it in your predictions, models and reports. For instance, it might be an indicator of **Buy** if the positive sentiment is above a threshold and **Sell** if it is below. Advanced measures such as getting tweets from a curated list of users or using Loughran sentiment dictionary are also recommended.