

STAT 523 Final Project

Team TBA: Jie Sun, Leonardo Shu, Shijia Bian, Yang Chen, Ziyue Zeng

December 11, 2015

Twitter Mysteries User Manual

Introduction (Basic Functionality)

Our final project consists of programming an R Shiny application that performs various types of analysis on data from Twitter. Users will choose between a particular topic (or in Twitter terms, hashtags #) or a specific Twitter handle. If a topic is chosen then the user can write whatever topic comes to mind which also exists on Twitter. The app will then collect a user-selected amount of random tweets associated with that hashtag in order to build a set of words to analyze or display. Conversely, if a handle is selected then the user can write out the exact handle and the app will again collect a particular amount (set by the user, or however many tweets that handle has if less than the specified number) of the most recent tweets to analyze. After the topic or handle have been selected, the app will run word cloud, word frequency association & clustering algorithms. The specifics of these will be discussed separately alongside with the options available to the user.

Scraping from Twitter

Word Cloud

The word cloud is analyzed by using the `twitterR` package. The goal of this section is to create a visualization of the most frequent vocab associated with the user's input string. For example, if the user enters "Christmas", this part will collect all the twitts that are associated with the "Christmas", retrun the frequency of the vocab that are associated with "Christmas" and display the word clout.

First, we need set up the authorization for the `twitterR` by giving the `consumer_key`, `consumer_secret`, `access_token` and all other authorization information.

```
consumer_key<-"zSeAWHNpaL5G7GpHuCO4zTffT"
consumer_secret<-"dZnarPgWiQCnb1bJOtvN3xFBmWVMRQCDDWl9UsFtkiTGpzztfg"
access_token<-"1071126529-DLvrKbaT9ju1yQsAHEBWZz5h3vHGEWwyWYeTHP4Z"
access_secret<-"h3XgPKhsKkF66ShXfbFPEnby2VUofGD6AYvu53CFvUFx7"
requestURL <- "https://api.twitter.com/oauth/request_token"
accessURL <- "https://api.twitter.com/oauth/access_token"
authURL <- "https://api.twitter.com/oauth/authorize"

setup_twitter_oauth(consumer_key, consumer_secret, access_token,access_secret)
```

The user will interact with the `twitterR` by entering the string, `rundel`, that he is interested in to know the most frequent vocab that are assocaited with it.

```
tweets = userTimeline("rundel",n = 1000)
```

All the twitts are treated as a whole. The `tm_map` will be applied to clean out the the text by removing punctuations, selecting preferred languages, excluding certain pattern of the words and setting up the stop words.

```

removeURL = function(x) gsub("http[[:alnum:]]*", "", x)
myStopwords = c(stopwords("english"), "isis", "rt", "amp", "twitter", "tweets", "tweet", "retweet", "tweet")

myCorpus = Corpus(VectorSource(tweets.df$text)) %>%
  tm_map(content_transformer(tolower)) %>%
  tm_map(content_transformer(removePunctuation)) %>%
  tm_map(content_transformer(removeNumbers)) %>%
  tm_map(content_transformer(removeURL)) %>%
  tm_map(removeWords, myStopwords) %>%
  tm_map(content_transformer(stemDocument))

```

The cleaned out words will be look like:

```

[[1]]winstonchang will im personally blown away nicely works shiny apps right box rprof mess
[[2]]winstonchang amazing work timely well saves teach students rprof later afternoon
[[3]]just attempt teaching shiny parallel programming r way bromans socks rabaaths tiny data
[[4]]perfect timing teaching sta students preserving vs simplifying subsetting today
[[5]] automated testing student rstats code github wercker now testthat

```

TermDocumentMatrix is applied to analyze the term frequency. The output will be treated as a matrix. The row name is the vocab and the colum list the number of times it appears in the document.

```

tdm <- TermDocumentMatrix(myCorpus, control = list(wordLengths = c(1, Inf)))
m = as.matrix(tdm)

```

The word cloud is generated by the package wordCloud. The ordered vocab in terms of the frequency will be passed into the wordcloud.

```

word.freq = sort(rowSums(m), decreasing = T)
wordcloud(words = names(word.freq), random.color = TRUE, colors=rainbow(10), freq = word.freq, min.freq = 1)

```

Here is an example of the word cloud associated with key words: *rundel*

word is github and the next few that follow are also terms related to things he has taught in 523 such as wercker, testthat, shiny, rgeos, hadleywickham. A word frequency association analysis shows that most of the words listed above are also very correlated. If we were to choose student, we see that testing and code have correlation of 1 so they are always paired together. rmd and rstats are also close with a correlation of 0.66. Finally, the cluster dendrogram is very interesting here since we can toggle a different number of cluster quite nicely. Again student, testig code are together but there is also a cluster with socks, parallel, programming, rabbaths & shiny which correctly refer to the SHiny-themed homework we had this semester. Hence we see that Professor Rundel's Twitter is a great blueprint for his teaching style & assignments.

Errors

Due to the random nature of many tweets and how very different ones can be grouped together by an specific topic or user, it could be the case that the app encounters some errors. For example, some users may not have many tweets and thus word association could be invalid. In addition, there may be a topic that doesn't has too many unique words and this overblows the capabilities of the clustering parameters. Issues like this may be solved by tuning the available options such as minimum frequency count or number of clusters to be set. We urge that a error screen does not mean the app cannot still work for other configurations.