STA 250 Final Project

Qing Lu

Introduction

The most popular web programmers are CSS,JavaScript and HTML, but they are very limited to second programmer users, such as R. For those who are interested in building their own web application, **Shiny**, a packages in R, is a great choice. It provides us a quick and easy way to create interactive web application with R.

The way to view a web page usually require a server (e.g your computer) to send a web page document written in HTML to your web browser. The HTML document will instruct your web browser to draw a web page. Shiny creates an interactive web application in the similar way. The different is that, when the viewer click the page, sender receive a signal to create a new web doc, and again sent the web browser for redrawing the page. To run App you need to create two document: sever.R and ui.R. The first one tells you what to draw , and the second one tells you how to draw it.

Shiny is favored by many web designers for its great flexibility. More and more statisticians use Shiny app as a way to interactively showcase and share statistical analyses and result, performed and generated in R on the web.In this project, I choose to analysis Twitter text data with different statistic skills, such as machine learning and parallel computing, and then present my result with Shiny app. The whole project will be designed as a interactive web page. Which means that viewers can read the article while exploring some interesting facts about the data via Shiny app.

keywords: interactive SVG, Shiny, R, machine learning. big data, parallel computing.

Dataset

The dataset is from University of Michigan Sentiment Analysis competition on Kaggle (Michigan, 2011). The training data contains 2037 sentences, which were labeled as 1 for positive sentiment and 0 for negative sentiment. The test data contains 1200 sentences that are unlabeled. The original dataset is in txt format, but viewers can check the “table” section to view the data. Viewer also can choose how many documents they want to see by inputting the observation numbers on Shiny sidebar.

Method

We could extract instant tweets from Twitter with **userTimeline()** in the **twitterR.** packages. Before that we need to get Twitter API authentication. In this case, I just download the data from Kaggle. Then I converted the original document to a term_document matrix before I performed clustering and classification for the data. Most of my result could be find from my 135 final project paper (Classification and Cluster analysis of Twitter data to predict sentiment.) However, the paper didn't provide any solution for the failure of classifying BigData with kNN due to the limited Computer memory. Therefore, I tried parallel computing to predict the sentiment of Twitter Data. I also created some Shinyapp to present my result and make them interactively with viewers.

The Text Mining is performed in the following procedure:
Step1: Transforming Text to term-document matrix
Step2: Find frequency and association, by construct a frequency table
Step3: Hierarchical and K-means.
Step4: Classification

Shiny Framework:
The whole report is distributed into several sections. Readers can choose different sections on the web sidebar panel, and then click the tab in the main panel to explore their interests part of the report. Under “Readme” set panel, it is the write up of the report. All the graph outcomes can be viewed in “Graph” section, and several dataset were presented in “Table” section. Viewers also can click the “code” tab to see the “R-code” for this project.

Transforming Text

Since the original data contains text, which cannot be processed directly with R, we need to transform the text into vectors. The Idea is that we create a dictionary which contains all the words occurred in all documents, and for each document we count the frequency for each word, then we built the term-document-matrix. Then the data will be ready for R to process.

The main structure for managing documents in R packages **tm** is "Corpus". After we converted our data to corpus, we need to clean the text by changing letters to lowercase, removing numbers, punctuations, and stop words. Finally we could transferred our document into term-document-matrix, where each row stands for a term and each column for a document, and the response are the frequency of the terms in each document.

Shiny hints:
Click “R-code” to see the Rscript of Transforming data
Click “Table” to view the Term and Document Matrix

Frequency and Associations

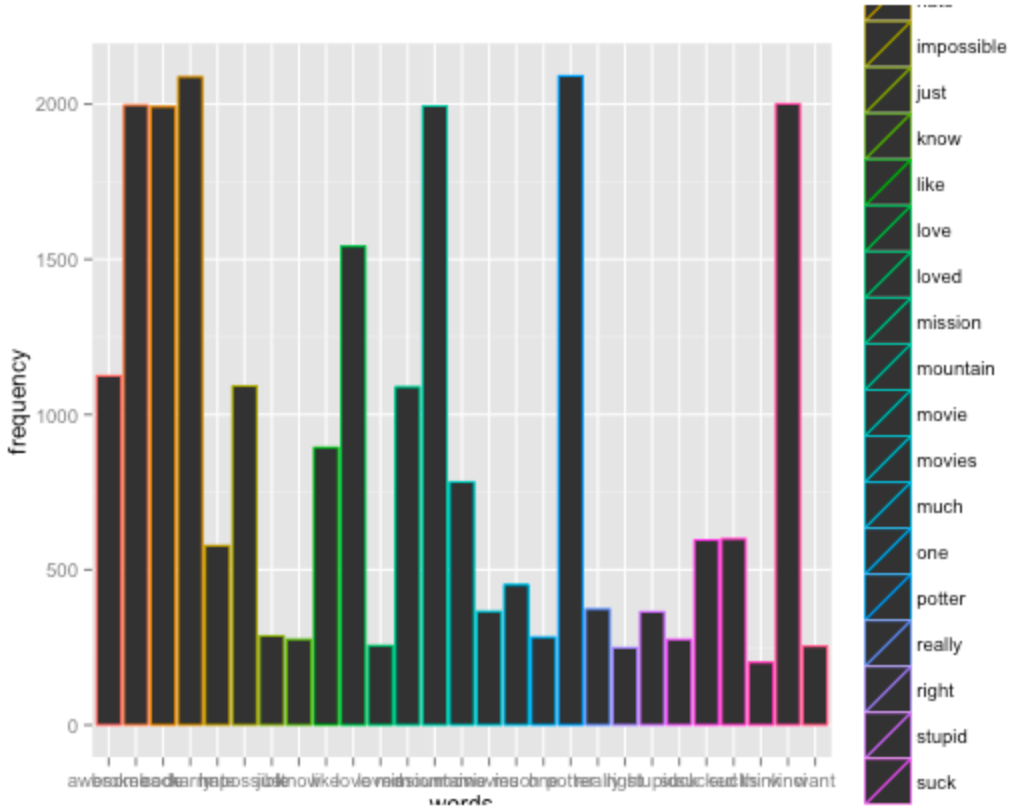
We are interested in the popularity of words and association between them. To display the top frequent words we could do bar plot for them with packages **ggplot2** in R. We also could use word cloud to depict the importance of words. The result can be viewed in plot section.

The words Awesome dominated the Twitter data set, and words from other major books and movies appear frequently as well. The words “movie” and “movies” even appear, indicating that these are topics that some users might utilize their account almost exclusively for. The high frequency appearance of “Brokeback Mountain”, the suspected meme phrase, is notable as it supports the previous hierarchical cluster. We also observe that many words indicating sentiment appear frequently. There seem to be a lot more variety of negative words such as “suck”, “hate”, “horrible,” but there are a few positive words like “love” and “awesome” that appear with much more frequency.

The word cloud shows that what are highly associated with a certain word. For example, I pick word "happy", the Word Cloud generated all the word which is associated with "happy", the most frequent words appear in the center of cloud, such as "harry potter", "love", and "vinci code".

Viewers can pick any term which they are interested, and the shiny web will show you the " word cloud" associated with the term.

Shiny hints:
In putting a word in “WordCloud” section of sidebar, and click “go” button to see what will happen in the main panel on under “Graph” part .

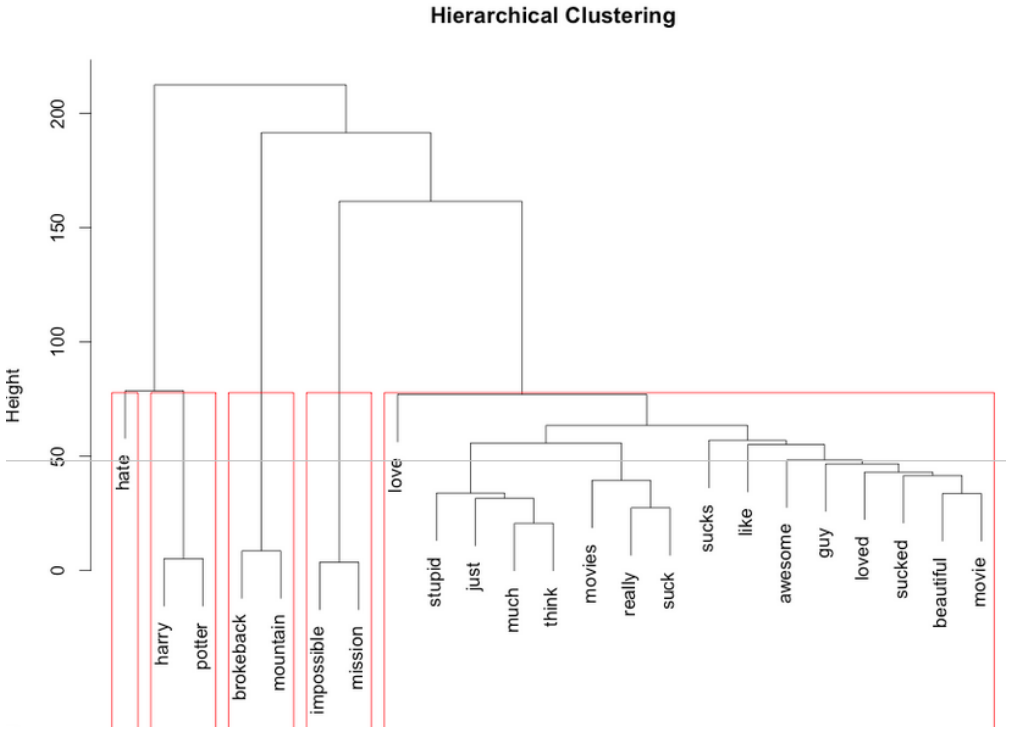


Clustering

Hierarchical Clustering

A hierarchical clustering analysis on the data set is displayed in the "Graph" section. Only frequently used terms are displayed for clarity, so that the dendrogram is not crowded with too many words. In order to view a clear word, we have to removed some sparse terms. Then we calculate the distances between terms. We could use single linkage, complete linkage, average linkage, median and centroid to do the clustering, but in this case, I set the method to "ward", such that the variance were increased when we merged two clusters.

As you can see in the result, there were clustered words for two popular movie franchises, “Harry Potter” and “Mission Impossible”. The appearance of the “Brokeback Mountain” cluster is interesting. While it might be easy to surmise that this is simply a popular movie among Twitter users and is widely discussed. The remaining words were all clustered together as a single, large group, including a variety of words like “love”, “stupid”, “awesome”, and “suck”. This seems to be a limitation of the clustering mechanism. It would have been nice to see words of similar sentiment clustered together, or even words of certain sentiments paired with specific media references. Instead, we have a seemingly random mix of words that do not seem predictive of any particular sentiment in the tweets.



k-Means Clustering

As an alternative to the hierarchical model, we completed a k-means cluster model on the same dataset.We need to transpose the term-document matrix to a document-term matrix. with function **kmeans()** we got five clusters, and is displayed in the "table " section. The initial table is a display of the top three words in each cluster, which suggests that there are some themes we might be able to pick up using this model. For example cluster 1 below has the words “harry potter love”, a media subject and a positive sentiment. We also have “vinci code awesome” in this analysis. However, because this only displays the top three words of each cluster, it is possible that there are a variety of topics or sentiments mixed in to the clusters.Viewers could choose how many top words they want to see by play with the sidebar app on Shiny.

```
cluster1: harry potter love
cluster2: brokeback mountain potter
cluster3: impossible mission harry
cluster4: brokeback mountain movie
cluster5: vinci code awesome
```

Shiny hints:

Use Shiny widgets to choose how many top words you want to see in the K-means Clustering result under “Table” set.

Classification

K-nearest Neighbor Classification

The original training data were classified to two group, group one has positive sentiment, labeled as "1", and group two has negative sentiment with label "0". We randomly split the training data in to two subsets: training(70%) and Validation (30%). We choose K=1. We could use **Knn()** in R to predict the Validation dataset, and conduct a confusion and compared them to observed result. The table is presented on Table section. Among 612 tweets, 306 of them have negative sentiment, and 306 of them have positive sentiment. Our model successfully predicted 298 negative sentiment and 305 positive sentiment. The misclassification rate is 1.9%.

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 298   5
         1   7 305
```

Shiny hints:
Use Shiny widget to choose “KNN” section, the confusion table will be shown inside the “Table” set.

Discussion

In my 135 project, we also tried to predict the sentiment for a big training dataset with 1,578,627 tweets in English. However, due to the limited memory we failed to carry out KNN in a traditional way. I believed that predictive modeling of Big Data with Limited Memory required us to use parallel computing. As the training dataset was too big and not easy to build decision trees with R. What we could do is to draw multiple subsets from original training data by random sampling, and to build a decision tree on each of them. Record the variables appeared on those trees, and draw the same variables from original dataset for reducing the dimension. R supports several levels of parallel execution, which allow us to build **rpart** decision trees in parallel. The packages we could use here are **wsrpart() and parallel() .** wsrpart () would build a decision tree, which selects a different random training dataset and a different random choice of variables from original dataset. The processing for each call to the function to build the decision tree will be distributed across multiple cores or servers.

References

Ahmad, I. (2014). *30+ Statistics how social media influence purchasing decisions.* Retrieved from Social Media Today: http://socialmediatoday.com/irfan-ahmad/2108426/30-statistics-how-social-media-influence-purchasing-decisions-infographic

Choy, M. (2012). Effective Listings of Function Stop words for Twitter. *International Journal of Advance Computer Science and Application, 3*(6).

Indvik, L. (2011). *East coasters turn to Twitter during Virginia earthquake.* Retrieved from Mashable: http://mashable.com/2011/08/23/virginia-earthquake/

Manning, C., Raghavan, P., & Schutze, H. (2009). An Introduction to Information Retrieval. Cambridge: Cambridge University Press.

Michigan, U. o. (2011). *UMICH SI650 - Sentiment Classification.* Retrieved from Kaggle: http://inclass.kaggle.com/c/si650winter11

Plumer, B. (2011). *Tweets move faster than earthquakes.* Retrieved from The Washington Post [Online]: http://www.washingtonpost.com/blogs/wonkblog/post/tweets-move-faster-than-earthquakes/2011/08/25/gIQA4iWHeJ_blog.html

Sanders, N. (2011). *Twitter Sentiment Corpus.* Retrieved from Sanders Analytics: http://www.sananalytics.com/lab/twitter-sentiment/

Wikipedia. (2014). *Internet Meme.* Retrieved from Wikipedia: http://en.wikipedia.org/wiki/Internet_meme

Wikipedia. (2014). *Twitter*. Retrieved from Wikipedia: http://en.wikipedia.org/wiki/Twitter


```
R-code:
library("shiny")
library(shinyapps)
library("e1071")
install.packages("class")
library("RColorBrewer")
source("helpers.R")
library("class")
require("class")
#########sever.R##########

shinyServer(function(input, output) {

output$readme <- renderUI({
```

```
filename <- normalizePath(file.path('./Readme',
                                    paste('read', input$obs, '.txt', sep='')))
   list(src = filename,       alt = paste("Txt number", input$obs))
  })


  output$help=renderUI({
    paste0("http://http://anson.ucdavis.edu/~cclu2012/HW4")
  })


TableInput <- reactive({
  switch(input$select,
         "Intro" = ,
         "Dataset" = Dataset1,
         "Transforming Text"=Displaymatrix,
         "Frequency and Associations"=Displayfreq)
})

PlotInput=reactive({
  switch(input$select,
         "Introduction",
         "Dataset",
         "Method",
         "Transforming Text",

         "Frequency and Associations"=isolate("as.character(input$word)")
             )
})

textInput=reactive({
    switch(input$select,
           "Introduction"=1,"Dataset"= 2, "Method"= 3,
           "Transforming Text"=4,"Frequency and Associations"=5,
           "Clustering"=6,"KNN"=7,"Discussion"=8,"Reference"=9)
  })
output$text<- renderImage({
    filename <- normalizePath(file.path('./Doc',paste('doc',textInput(), '.png', sep='')))
    list(src = filename)
}, deleteFile = FALSE)

output$image <- renderImage({
filename <- normalizePath(file.path('./images',paste('image', textInput(), '.png', sep='')))
 list(src = filename)
}, deleteFile = FALSE)

output$code<- renderImage({
```

```
  filename <- normalizePath(file.path('./code',paste('code', textInput(), '.png', sep='')))
  list(src = filename)
}, deleteFile = FALSE)

output$kmeans=renderPrint({
  if(input$ktable){
    input$go
    ks(isolate(input$words))}
})

output$Plot=renderPlot({
  if(input$WordCloud)
  { input$goButton
    asso_cloud("isolate(as.character(input$word))")}
  else{invisible(1)}
})

output$table=renderTable({
  head(TableInput(),n=input$obs)
})

output$tabletext=renderPrint({
 if (input$select=="KNN"){(confusionMatrix(result1,testCl))}
})
})

#############ui.R#######
library("shiny")
library(shinyapps)
library("e1071")
library("RColorBrewer")
source("helpers.R")
library("class")
require("class")

shinyUI(pageWithSidebar(

  # Application title
  headerPanel("STA 250 Final Project" ),

    sidebarPanel(
    radioButtons("select", "Content:",
                 choices = c("Introduction", "Dataset","Method","Transforming Text","Frequency and
Associations","Clustering","KNN","Discussion","Reference")),
    numericInput("obs","Number of Obervations to view in Table",5),
    checkboxInput("WordCloud", "Word Cloud (view in Graph set)",TRUE),
    textInput("word","Enter a word here to see the Word Cloud", "happy"),
```

```
   actionButton("goButton", "Go"),
   checkboxInput("ktable", "K-means Clustering (view in Table set)"),
   numericInput("words","Choose the number of top frequent words you want to see ",3),
   actionButton("go", "next")
   ),

mainPanel(

tabsetPanel(id="select2",
            tabPanel("Readme", imageOutput("text")),
            tabPanel("Graph", imageOutput("image"),plotOutput("Plot")),
            tabPanel("Table",
tableOutput("table"),verbatimTextOutput("kmeans"),verbatimTextOutput("tabletext")),
            tabPanel("R-code", imageOutput("code"))))
))




#########helpers .R #######
load("Data/mydata.dtm.Rda")
load("Data/frequency.Rda")
load("Data/Displaymatrix.Rda")
load("Data/Displayfreq.Rda")
load("Data/Dataset2.Rda")
load("Data/Dataset1.Rda")
load("Data/Classdata.Rda")
load("Data/clusdata.rda")
load("Data/result1.Rda")
load("Data/testCl.Rda")

library(tm)
library("Rcpp")
library("class")
library("wordcloud")
library(ggplot2)
library("lattice")
library("caret")
library("class")
library("e1071")

install.packages("class")
asso_cloud=function(word){
  temp=findAssocs(mydata.dtm,'word',0.2)
  t1=rownames(temp)
  t2=rownames(mydata.dtm)
  i=t2%in%t1
```

```
  wcmydata=as.matrix(mydata.dtm)
  newdata=wcmydata[i,]
  wordFreq=sort(rowSums(newdata),decreasing=TRUE)
  wordcloud(word=names(wordFreq),freq=wordFreq, min.freq=50,
random.order=F,colors=rainbow("5"))
}

hplot=function(i){
  plot(hclus, main="Hierarchical Clustering",xlab="Terms")
  rect.hclust(hclus,k=i,border="red")
  groups=cutree(hclus,k=i)}

kclus=kmeans(clusdata,5)
ks=function(n){
kcenter=round(kclus$centers,digits=3)
  for(i in 1:5) {
    cat(paste("cluster", i, ": ", sep=""))
    s=sort(kclus$centers[i,], decreasing=T)
    cat(names(s) [1:n], "\n")}
}

confusionMatrix(result1,testCl)


######other-code#######
```