Text Mining in R
Word Clouds

Manuel S. González Canché
msgc@email.arizona.edu
University of Arizona

Cecilia Rios Aguilar
Cecilia.Rios-Aguilar@cgu.edu
Claremont Graduate University

INCHER, University of Kassel

June 25, 2012

Contents

1 Motivation

Many times qualitative and quantitative researchers face a big problem: the impressive amount of text files (interviews, speeches, discourses, Facebook posts) that need to be coded and summarize. This is problematic because finding structure in unstructured data is a challenging process and summarizing hundreds or thousands of texts can be a task that is nearly impossible for the human being brain to be handle or that would take months of work.

With the advancement of computers and analytic algorithms we can rely on computers to ask them to find that hidden structure and report the outcomes back to us. The challenge in this case is to detect that structure, because by definition we have eliminate waste or noise in the data. The process to do this is called Text Mining (TM).

Text Mining is an interdisciplinary set of procedures involving âĂŞlinguistics, computational statistics, and computer scienceâĂŞ, that has become widely used in statistical and learning machine methods (Feinerer, Hornik, & Meyer, 2008). The main characteristic of text mining techniques is that unstructured text is the source of information. With a single text, (*i.e.* an interview or a discourse), the analytic properties are constrained to describing frequencies of words; however, when there is a collection of texts text mining can help to discover new facts and trends that are hidden in the text and are not apparent to naked eye (Hearst, 1999). These hidden structures would be virtually impossible to be captured without the help of computers, especially when analyzing thousands of texts.

One of the classical applications of text mining comes from the traditional data mining community focused on unsupervised learning techniques (ULT). ULTs try to find hidden structure in unstructured data (Duda, Hart, & Stork, 2001), a method that has been especially relevant in biology and epidemiology (Hastie, Tibshirani, & Friedman, 2009). This process, although computationally expensive, would render irrefutable evidence behind the hidden structure of the unstructured texts analyzed. The first step required for by TM involves âĂIJtransforming the text into a structured format based on term frequencies and subsequently applying standard data mining techniquesâĂİ (Feinerer, Hornik, & Meyer, 2008, p.1). Usually, this approach starts by removing irrelevant words from the texts, technically known as removing sparse terms. This first step is rather tedious because researchers need to be sure of not removing words that are synonyms of important concepts or that spelling errors are in the texts. A rather careful examination of the texts is necessary. Once the first step of removing irrelevant words is completed, the texts will be transformed into standard un-squared matrices, in which the rows will be each one of the texts to be analyzed and every column will correspond to the terms or words that each text contains. These matrices, data frames or tables will naturally account for co-occurrences of words. Obtaining this term-document matrix (Feinerer, et al., 2008) is a major milestone since this is what would allow researchers to perform classic data mining manipulations, such as finding correlation, clustering, and computing factors analysis to find the hidden structures that are important in the texts and are accounted for by the words that tended to be grouped or appeared together.

Two specific technique that we be implementing in this session is a method designed

to find associations for a given term, which corresponds to a category of techniques called count-based evaluation methods. We basically will quantify the frequency of words as they appeared in the texts. Using the wordcloud package in the statistical platform R we will be able to graphically represent the most important words weighted by their frequency of apparition. This technique goes beyond merely summarizing frequency of words using histograms or other basic graphical representations that are limited in their capabilities of accommodating hundreds of categories. In a wordcloud each word becomes a variable or category that can be plotted in a single wordcloud. Consequently, a wordcloud contains all the important words in the texts analyzed, yet it is powerful enough to emphasize the most important categories.

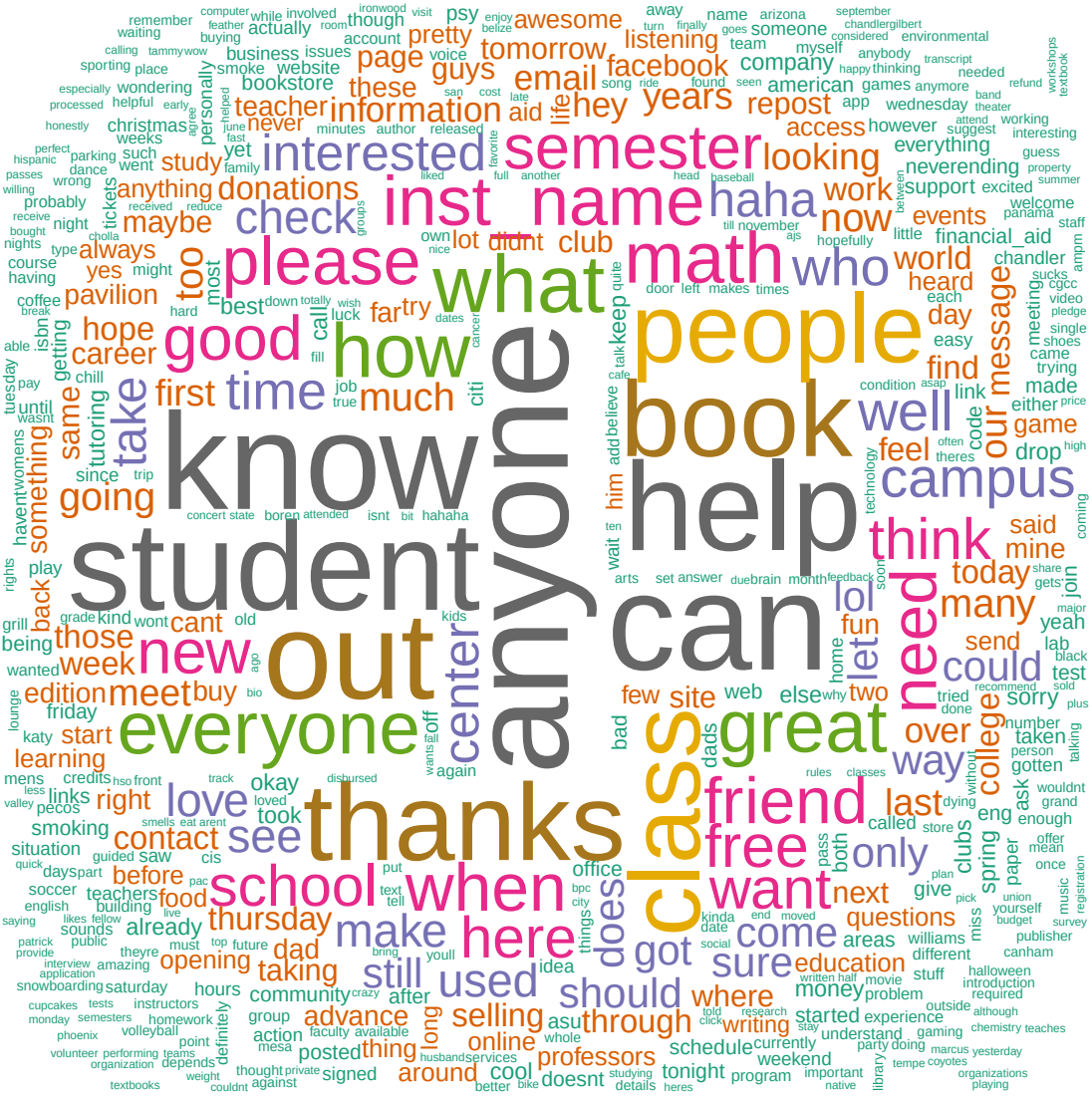

Figure 1: Word Cloud summarizing text from 1,272 texts

1.1 Purpose

To understand and replicate the Text Mining mechanisms required to successfully plot a word cloud in R as show in Figure 1

2 R functions

In this section we present the R codes to prepare the data to replicate Figure 1.

2.1 Code to clean the data

Depending on where you have placed the dataset the path will need to be modified accordingly. Also, Mac and Pc will have small differences in terms of how to write paths, for instance:

Possible paths:

```
#Sets workding directory
setwd("C:/Users/MSGC/Documents/Ph. D/Dropbox/Germany PHUDCFILY
/TricDataWorkshop") #Pc

setwd("/Users/msgc/Dropbox/Dropbox/Germany PHUDCFILY
/TricDataWorkshop") #mac
```

Please follow the next directions:

1. Open R.
2. Remember to modify the path based on your computer's configuration.
3. Copy and paste one of the previous paths (depending on whether you are using a mac or Pc).
4. Once you are sure about having correctly modified the path, paste it in R and hit enter. This command told R where to read the data.

The following command will import our previously modified *.csv into the R environment. Remember, this dataset needs to be located in the folder indicated by our previous path.

We start by getting the data ready

1. Just in case, setting working directory
   `setwd("C:/Users/MSGC/Documents/Ph. D/Dropbox/Germany PHUDCFILY /TricDataWorkshop") #Pc`
2. To read the data from csv format located in our working directory
   `data <- read.csv(""cgcc_feed_items.csv",", header = TRUE, sep = ",", skip = 0,row.names = NULL)`
3. Let's see how many texts we are dealing with
   `length(data$user_id)`

4. Installing our package to handle the process
   `install.packages("tm")`

5. Loading the package
   `library(tm)`

6. Creating our collection of texts, called corpus
   `txt<-Corpus(VectorSource(data$text_body),`
   `readerControl = list(language = "en"))`

7. If the data is just to much, we can select a portion of it, let's say the first 100 texts
   `#txt <- Corpus(VectorSource(data[c(1:100),c(3)]))` Sometimes we need to start the cleaning process early. For instance, we know that the data has a problem, certain words won't be recognized by the technique and won't allow the process to be completed. The tedious part is to detect those words. In this case we already detected the problem and successfully deleted these characters with the following function
   `txt <- gsub( "Ãğ Âğ ĂĄ Ãč ÂĄ Âŕ Ãę âĂŤ Âě Ãę ĹŞ Âń ÃĺÂŁĺ¿ Ãč ÂĄ ĹĎ Ãě ÂŁ ÃĺÆŠ Â¡ Ãč ÂĄ Âğ Ãč ÂĄ Âŕ Ãč ÂĄ ÂŁ Ãč ÂĄ âĂđ Ãč âĂŽ ËĘ" , "" , txt)`

8. Because of previous function, our corpus is now a character object, which in R is an dimensionless thing. Because of this we need to reapply the corpus function to obtain a data frame, and the to get our collection of texts (corpus) again
   `txt<-as.data.frame(txt)`
   `txt <- Corpus(DataframeSource(txt[1]))`

We succesfully have created our corpus, we then have to start with the real cleaning process in R

1. We can change all the words to lower font
   `txt <- tm_map(txt, tolower)`

2. We can remove all question marks, asterisks, and the like
   `txt <- tm_map(txt, removePunctuation)`

3. We may remove numbers is necessary
   `txt <- tm_map(txt, removeNumbers)`

4. We can remove URLs
   `removeURL <- function(x) gsub("http[[:alnum:]]*", "",x)`
   `txt <- tm_map(txt, removeURL)`

5. We can save a dictionary based on our own original data. This is done if we want to stem a document, which will only consider the roots of each word, but some roots will make no sense, for example, **students, student, study, studio** will only have **stud** after the stemming, in this case we need to let R know what word we want to represent stud, but first we need to create the dictionary
   `dictCorpus <- txt`

6. We apply the stemming algorithm which takes some time
```
txt <- tm_map(txt, stemDocument,language = "en")
```

7. We now apply the dictionary to replace the stemmed words with their most frequent complete word, for instance, study
```
txt <- tm_map(txt, stemCompletion, dictionary=dictCorpus)
```

8. A problem with stemDocument is that it will take the most frequent word within each of the thousands of texts, but if a text only contains students in stud, then it will take student as opossed to study as we wanted, we can address this problem with the following commands
```
txt <- gsub( " classes " , " class " , txt)
txt <- gsub( " students " , " student " , txt)
txt <- gsub( " books " , " book " , txt)
txt <- gsub( " cgcc" , " inst_name " , txt)
```

9. Because of the use of the function gsub we have a character object, so we need to repeat the data transformation
```
txt<-as.data.frame(txt)
txt <- Corpus(DataframeSource(txt[1]))
```

10. In addition, we can massively remove words that we do not consider important
```
newstopwords <-c("with","and", "for", "the", "to", "in", "then",
"he", "she","than", "not", "that", "thats", "dont", "whats","you",
"for", "are","but","have","your","this","was","like","just","all","its",
"get","they","from","one","about","some","would","more","them",
"really","had","been","also","because","did","has","which","were",
"other","very","into","her","their","even","his","say","ive","youre",
"youravoncomrobertdunn","ronaldczarneckimsncom","httpfacebook",
"httpwwwfacebookcomwethingtonphotography","i've","it's",":","i'm",
"i'll","--->","don't","and", "i'd","won't","aren't",
"wasn't","we'll", "ill""will","there")
```

11. After having defined the undesired words, we can remove them:
```
txt <- tm_map(txt, removeWords, newstopwords)
```

2.2 Code to replicate Figure 1

1. We need four packages
```
require(XML)
require(tm)
require(wordcloud)
require(RColorBrewer)
```

2. Creating a Term document matrix
```
ap.tdm <- TermDocumentMatrix(txt)
```

3. Now a matrix `ap.m <- as.matrix(ap.tdm)`

4. Now we can create a data frame
```
ap.v <- sort(rowSums(ap.m),decreasing=TRUE)
ap.d <- data.frame(word = names(ap.v),freq=ap.v)
```

5. We are finally ready
```
pal2 <- brewer.pal(8,"Dark2")
wordcloud(ap.d$word,ap.d$freq, scale=c(5,.25),min.freq=6,
max.words=Inf, random.order=FALSE, rot.per=.15, colors=pal2)
title(sub="Figure 6. Community College #6, Arizona",
col.main="Black", cex.main=1.5,font.sub=2)
```

Enjoy!

Evaluation and/or Feedback

June 27, 2012
University of Kassel, Germany
International Centre For Higher Education Research

Please, in order to help us improve this workshop, we are requesting your feedback: **In a scale of 0 to 10** (10 being maximum or positive), please respond to the following questions and explain the reason of your choice if possible:

1. How useful do you think the content was? _____
   why?

2. How much do you think you learned? _____
   why?

3. How well do you think the instructor knows the topic? _____
   why?

4. Would you be willing to attend to another session if possible? _____
   Why?

5. What would you recommend to make this workshop better? _____
   Why?

6. Is there a particular topic you wish it was covered? Yes ___ or No ___
   Why?

Thank you!