Milestone Report

Chao Liu

06/2016

Overview

This report contains basic summary information about the data collected by SwiftKey and Coursera. In the following sections, we will see the most popular words in three datasets and we will draw a nice wordcloud to visualize it. Moreover, we will also see the distribution and how many words are enough to cover 90% content of these three materials.

```
## Loading required package: NLP
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
## Loading required package: RColorBrewer
```

Getting Data

The following function takes a small part of the original data out as our training dataset

```
writeTestDocument <- function(twitter_lines, blogs_lines, news_lines){
    twitter <- character()
    con <- file("F://Data Science//Capstone//Coursera-SwiftKey//final//en_US//en_US.twit
ter.txt","r")
    twitter <- append(twitter, readLines(con, twitter_lines))
    write.table(twitter, file = "F://Data Science//Capstone//Coursera-SwiftKey//test//te
st_twitter.txt")
    close(con)

    blogs <- character()
    con <- file("F://Data Science//Capstone//Coursera-SwiftKey//final//en_US//en_US.blog
s.txt","r")
    blogs <- append(blogs, readLines(con, blogs_lines))
    write.table(blogs, file = "F://Data Science//Capstone//Coursera-SwiftKey//test//tes
t_blogs.txt")
    close(con)

    news <- character()
    con <- file("F://Data Science//Capstone//Coursera-SwiftKey//final//en_US//en_US.new
s.txt","r")
    news <- append(news, readLines(con, news_lines))
    write.table(news, file = "F://Data Science//Capstone//Coursera-SwiftKey//test//tes
t_news.txt")
    close(con)

    rm(con);
    rm(twitter);rm(blogs);rm(news);
}
```

Noticing the total lines can be evaluated as follows (we will only use the en_US data):

```
twitter <- character()
con <- file("F://Data Science//Capstone//Coursera-SwiftKey//final//en_US//en_US.twitte
r.txt","r")
while (length(readLines(con, 1)) != 0) {
    twitter <- append(twitter, readLines(con, 1000))
}
close(con)
twitter<-iconv(enc2utf8(twitter),sub="byte")
blogs <- character()
con <- file("F://Data Science//Capstone//Coursera-SwiftKey//final//en_US//en_US.blogs.tx
t","r")
while (length(readLines(con, 1)) != 0) {
    blogs <- append(blogs, readLines(con, 1000))
}
close(con)
blogs<-iconv(enc2utf8(blogs),sub="byte")
news <- character()
con <- file("F://Data Science//Capstone//Coursera-SwiftKey//final//en_US//en_US.news.tx
t","r")
while (length(readLines(con, 1)) != 0) {
    news <- append(news, readLines(con, 1000))
}
close(con)
news<-iconv(enc2utf8(news),sub="byte")
```

We will first take 30000 lines, 10000 lines and 10000 lines of twiiter, blogs and news data.

```
writeTestDocument(30000,10000,10000)
```

Finally, we will read the data as VCorpus provided by tm package:

```
rctol <- list(reader = readPlain, language = "en", load = TRUE)
dat <- VCorpus(x = DirSource("F://Data Science//Capstone//Coursera-SwiftKey//test"), rea
derControl = rctol)
rm(rctol)
```

Cleaning Data

Profane filtering

Profane words are hard to define since some potential profane word needs to be carefully evaluated as profane under certain circumstances according to different culture.

Hence, we will only exclude profane words according to Seven dirty words from Wikipedia, which is defined as profane word under almost every circumstances.

```
profane_word <- c("shit","piss","fuck","cunt","cocksucker","motherfucker","tits")
```

Text Stemming, Tokenization & Punctuation and Number Removal

We will transfer the raw data into Corpus that we can implement data analysis on. Hence, we will utilize the tm package to stem, tokenize the raw input, and then remove punctuation and numbers since we mainly focus on word prediction.

```
dat <- tm_map(dat, removeNumbers)
dat <- tm_map(dat, content_transformer(tolower))
dat <- tm_map(dat, removeWords, stopwords("english"))
dat <- tm_map(dat, stripWhitespace)
dat <- tm_map(dat, removePunctuation)
dat <- tm_map(dat, stemDocument)
dat <- tm_map(dat, removeWords, stopwords("english"))
dat <- tm_map(dat, removeWords, profane_word) # Profane filtering
```

Basic summary statistics

1. Using the stemmed document to calculate the Term-Document-Matrix, which will be the core to our data analysis

```
my_TDM <- TermDocumentMatrix(dat)
```

2. Tranforming the TDM into new matrix for further analysis

```
test <- as.matrix(my_TDM)
test <- list(test, name  = rownames(test))
test <- tbl_df(as.data.frame(test))
test <- mutate(test, freq = test_blogs.txt + test_news.txt + test_twitter.txt)
test <- arrange(test, desc(freq)) #Rank the words by frequency
test$name <- as.character(test$name)
```

Let’s see the most 20 frequent words in our training dataset

```
## Source: local data frame [20 x 5]
##
##    test_blogs.txt test_news.txt test_twitter.txt  name  freq
##             (dbl)         (dbl)             (dbl) (chr) (dbl)
## 1            1254          1106              1249  will  3609
## 2            1140           545              1882  just  3567
## 3            1220           598              1658  like  3476
## 4            1024           597              1833   get  3454
## 5            1403           862              1122   one  3387
## 6             371          2478               239  said  3088
## 7            1238           653              1071  time  2962
## 8            1140           577              1177   can  2894
## 9             727           424              1410   day  2561
## 10            737           140              1571  love  2448
## 11            932           507               936  make  2375
## 12            678          1091               514  year  2283
## 13            593           325              1309  good  2227
## 14            619           671               864   new  2154
## 15            798           268              1078  know  2144
## 16            671           364              1059   now  2094
## 17            628           483               795  work  1906
## 18            171            41              1673 thank  1885
## 19            570           624               652   say  1846
## 20            614           281               927   see  1822
```

3. Words that can cover 90% content of the Corpus

The following functions can get the nearest number of the words that are able to constitute a dictionary to cover the corpus with the given coverage ratio

```
nearest_to_target <- function(freq, target){
    which.min(abs(freq-target))
}
cover_ratio <- function(freq,ratio){
    freq <- sort(freq, decreasing = TRUE)
    temp <- numeric()
    for(i in 1: length(freq)){
        temp[i] <- sum(freq[1:i])/sum(freq)
    }
    nearest_to_target(temp,ratio)
}
```

We will eliminate the words that only constitute 10% of all the content of the training corpus to see how many words are left to reduce the dimension.

```
uncommon_words <- as.character(test$name[cover_ratio(test$freq,0.9):dim(test)[1]])
```

So the result indicates that only **17.2636784** percent of words can be used to represent the corpus itself!

Reload the reduced data matrix

```
test <- test[1:cover_ratio(test$freq,0.9),]
```

4. Implement simple explantory data analysis

- Find associates
  For example, we can find the words that has high correlation with the word “love”

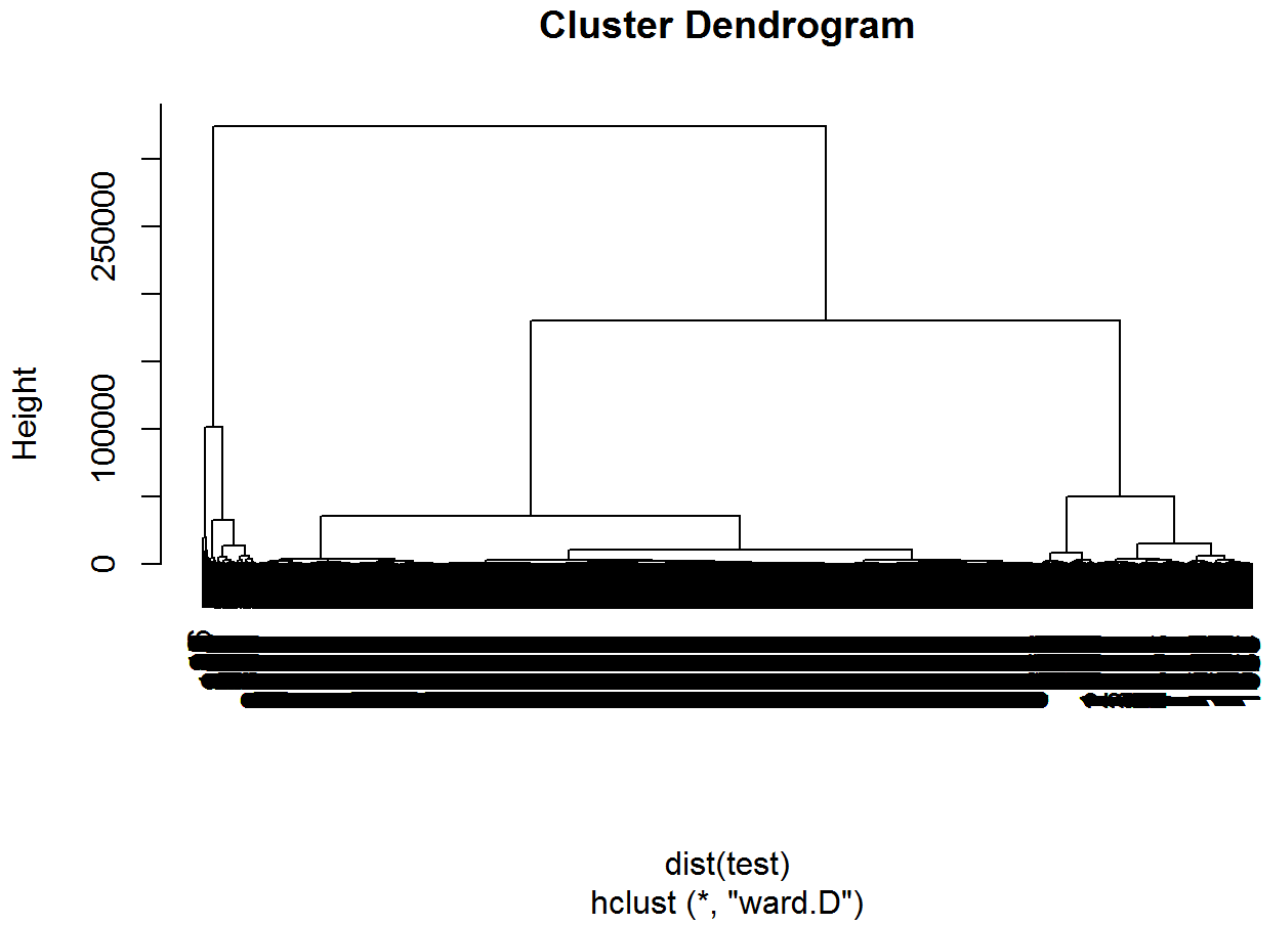
```
findAssocs(my_TDM, "love", 0.9999) # Find associates with "love"
```

```
## $love
##  alreadi    bacon     blue  boredom  cracker      dog    drink     drum
##        1        1        1        1        1        1        1        1
##  everyon  fantast     help      hot     kati    lyric  nervous   pardon
##        1        1        1        1        1        1        1        1
##    pasta     rock    scari  seminar  serious   someon     sxsw valentin
##        1        1        1        1        1        1        1        1
##    waffl     weed
##        1        1
```

- Word clustering

```
## The "ward" method has been renamed to "ward.D"; note new "ward.D2"
```

```
## Warning in dist(test): 强制改变过程中产生了NA
```



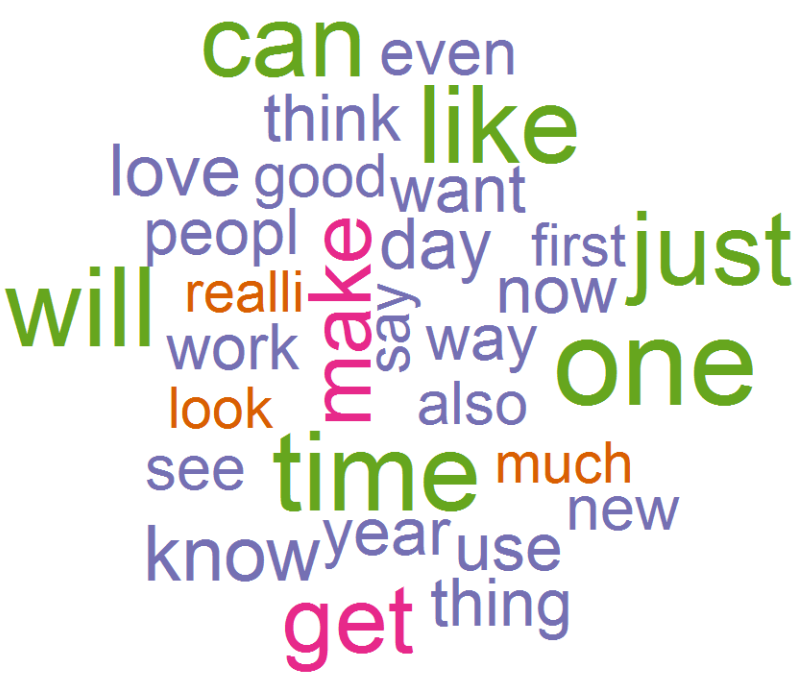
From the dendrogram above, we can roughly see the words can be split as two parts

- Wordcloud picture

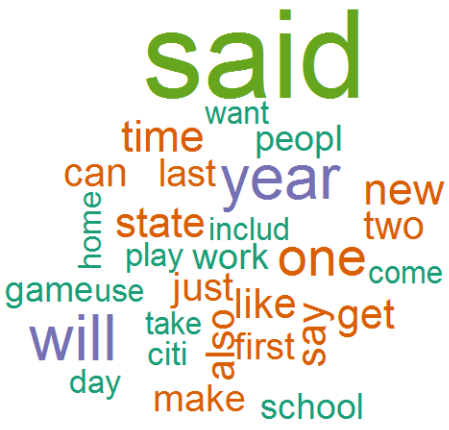
We will draw some nice wordcloud pictures to visualize the most frequent words in twitter, blogs and news dataset

1. Overall Wordcloud

2. Twitter Wordcloud



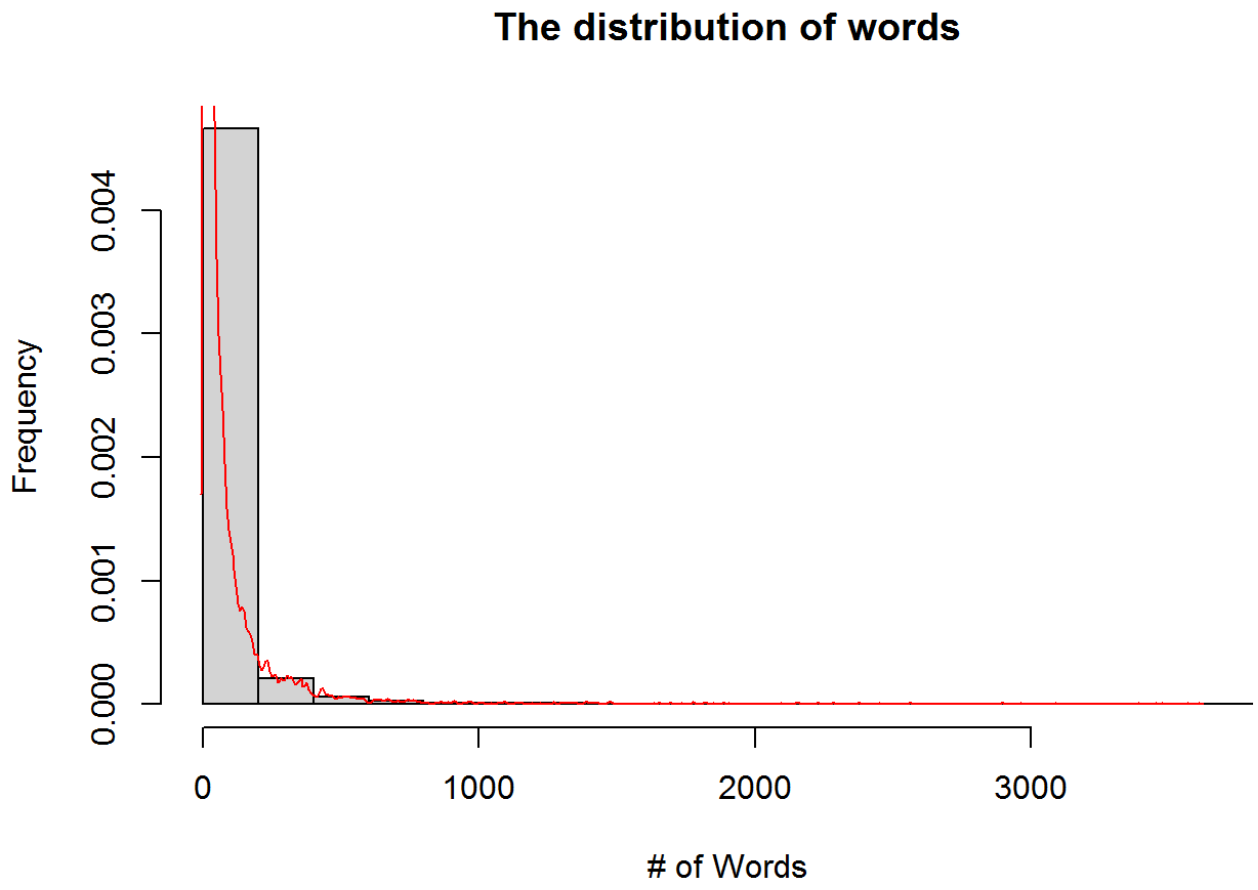
3. Blogs Wordcloud



4. News Wordcloud

Consequently, from the above wordclouds, we can see the distribution of different dataset.

- Histograms of frequency of words
  We can draw the histogram of frequencies of words of the reduced overall dataset:



As we expected, words are distributed as power distribution.

- Dissimilarity by cosine relationship We want to know the relation bewteen twitter, blogs and news data. Here, we will exploit the cosine relationship to measure the correlation:

```
dissimilarity <- function(x, y){
    sum(x*y)/(sqrt(sum(x^2))*sqrt(sum(y^2)))
}
correlation_matrix <- matrix(data = 0, nrow = 3, ncol = 3)
for(i in 1:3){
    for(j in 1:3){
        correlation_matrix[i,j] <- dissimilarity(test[,i],test[,j])
    }
}
rownames(correlation_matrix) <- names(test)[1:3]; colnames(correlation_matrix) <- name
s(test)[1:3]
correlation_matrix
```

```
##                  test_blogs.txt test_news.txt test_twitter.txt
## test_blogs.txt         1.000000     0.7830330        0.8590410
## test_news.txt          0.783033     1.0000000        0.6448335
## test_twitter.txt       0.859041     0.6448335        1.0000000
```

We can see from the table the datasets are actually highly correlated

- Bigram exploration For the further predictive model, we will first find the bigram pattern of our training data as follows:

```
ng <- ngram(concatenate(content(dat[[1]]),content(dat[[2]]),content(dat[[3]])), n = 2)
wordtable <- get.phrasetable(ng)
```

Ultimately, the 20 most common bigrams are:

```
##             ngrams freq
## 1       right now   307
## 2       look like   241
## 3       feel like   231
## 4       last year   228
## 5        new york   200
## 6      last night   200
## 7    look forward   193
## 8    thank follow   169
## 9        year ago   166
## 10    high school   160
## 11     first time   151
## 12      last week   146
## 13        can get   142
## 14      make sure   142
## 15      good morn   125
## 16 happi birthday   117
## 17       let know   111
## 18        can see   108
## 19       just got   108
## 20       year old   107
```