Assignment Report on MileStone project

Michale Gebrekiros

Wednesday, March 16, 2016

The main objective of the project is to develop a predicting Model. To be able to develop the text model, the following two preliminary analysis will be done

1. Understanding the distribution of the dataset(texts)
2. understanding the relationships between the words, tokens and pharases in the given dataset

```
setwd("M:/capestone_Project/final/en_US") # to the directory where the dataset is loaded
```

Load data into RStudio

For explanatory analysis and to understand the freqency distribution of words a sample for the given dataset will be taken as it reduces processing time.

Four folders(de_DE,en_US,fi_FI,ru_RU) containing three files each. One of news, one of blog information, and one from a twitter feed. Since the data sets are very large, a sample at random for each is taken to allow for reasonable processing time and to avoid computing resource issues and a dataset only for en_US is consiederd in this project.

```
news = readLines("M:/capestone_project/final/en_US/en_US.news.txt")
```

```
## Warning in readLines("M:/capestone_project/final/en_US/en_US.news.txt"):
## incomplete final line found on
## 'M:/capestone_project/final/en_US/en_US.news.txt'
```

```
blogs = readLines("M:/capestone_project/final/en_US/en_US.blogs.txt")
twitter = readLines("M:/capestone_project/final/en_US/en_US.twitter.txt")
```

```
## Warning in
## readLines("M:/capestone_project/final/en_US/en_US.twitter.txt"): line
## 167155 appears to contain an embedded nul
```

```
## Warning in
## readLines("M:/capestone_project/final/en_US/en_US.twitter.txt"): line
## 268547 appears to contain an embedded nul
```

```
## Warning in
## readLines("M:/capestone_project/final/en_US/en_US.twitter.txt"): line
## 1274086 appears to contain an embedded nul
```

```
## Warning in
## readLines("M:/capestone_project/final/en_US/en_US.twitter.txt"): line
## 1759032 appears to contain an embedded nul
```

Lets take a sample at random for each dataset

25% ,2.5% and 1% of the records were selected randomly for news, blog and twitter data sets respectively.

```
samplenews <- sample(news, length(news)*0.25)
sampleblogs <- sample(blogs, length(blogs)*0.025)
sampletwitter <- sample(twitter, length(twitter)*0.01)
```

Let change the data structure into a data frame as follows

```
snewsDF <- data.frame(charCount=nchar(samplenews), wordCount=sapply(strsplit(samplenews, " "), length))
sblogsDF <- data.frame(charCount=nchar(sampleblogs), wordCount=sapply(strsplit(sampleblogs, " "), lengt
stwitterDF <- data.frame(charCount=nchar(sampletwitter), wordCount=sapply(strsplit(sampletwitter, " "),
```

lets observe the heads of the news dataframe

```
head(snewsDF)
```

```
##   charCount wordCount
## 1       618       106
## 2       261        49
## 3       291        47
## 4       611       102
## 5       276        39
## 6       117        20
```

lets observe the heads of the blogs dataframe

```
head(sblogsDF)
```

```
##   charCount wordCount
## 1        54        11
## 2       310        56
## 3       522        88
## 4        17         3
## 5       617        93
## 6       519       102
```

lets observe the heads of the twitter dataframe

```
head(stwitterDF)
```

```
##   charCount wordCount
## 1        47         8
## 2        39         8
```

```
## 3        67        13
## 4       118        24
## 5        18         3
## 6        62        12
```

summary of the news,bolg and twitter sample datasets

```
#Summary for news,blogs and twitter
summary(snewsDF)
```

```
##    charCount        wordCount
##  Min.   :   3.0   Min.   :  1.00
##  1st Qu.: 109.0   1st Qu.: 18.00
##  Median : 185.0   Median : 31.00
##  Mean   : 201.8   Mean   : 34.04
##  3rd Qu.: 268.0   3rd Qu.: 45.00
##  Max.   :1779.0   Max.   :326.00
```

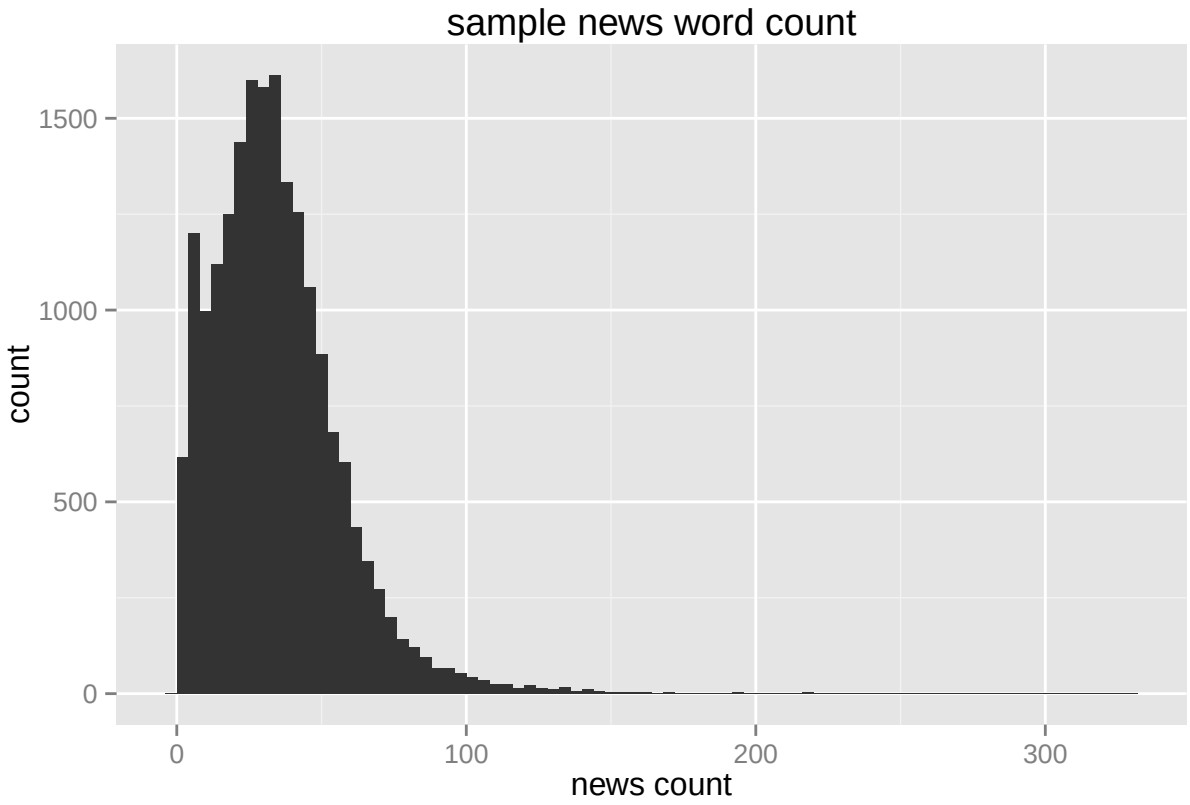
```
summary(sblogsDF)
```

```
##    charCount        wordCount
##  Min.   :   1.0   Min.   :   1.00
##  1st Qu.:  47.0   1st Qu.:   9.00
##  Median : 157.0   Median :  28.00
##  Mean   : 231.1   Mean   :  41.41
##  3rd Qu.: 327.0   3rd Qu.:  59.00
##  Max.   :5387.0   Max.   :1003.00
```

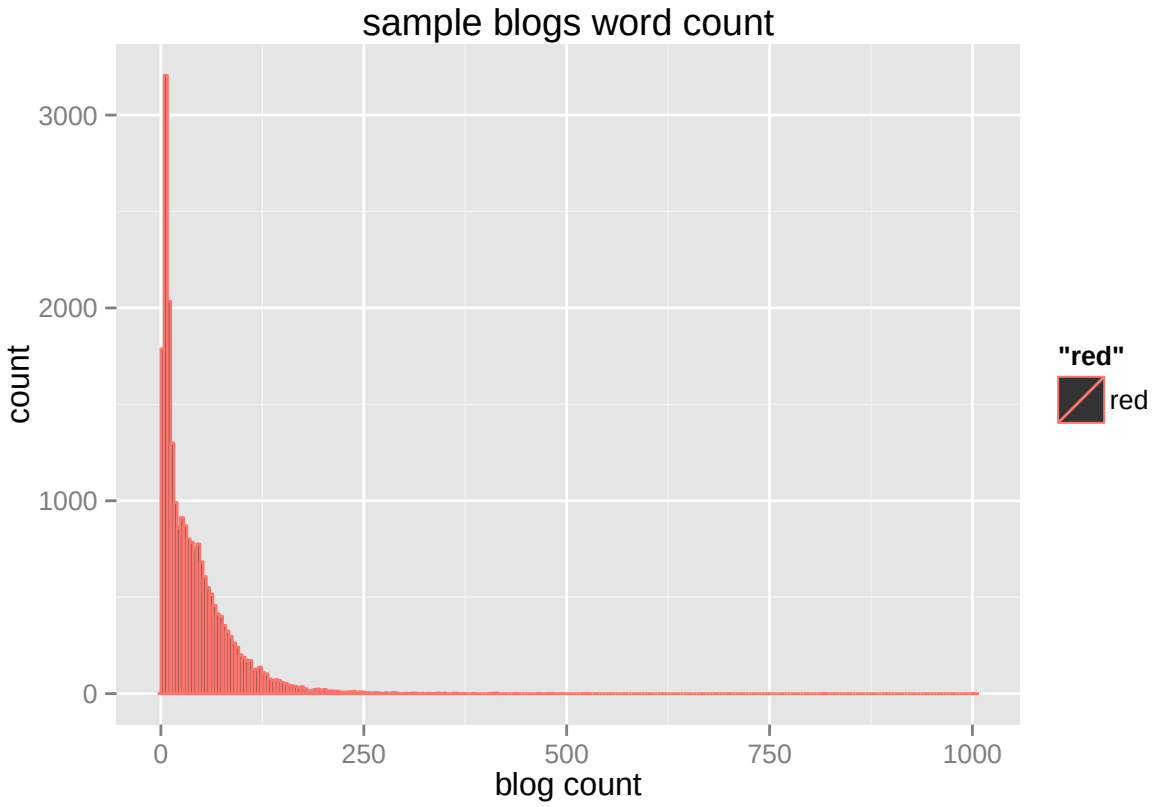
```
summary(stwitterDF)
```

```
##    charCount        wordCount
##  Min.   :  3.00   Min.   : 1.00
##  1st Qu.: 37.00   1st Qu.: 7.00
##  Median : 64.00   Median :12.00
##  Mean   : 69.06   Mean   :12.92
##  3rd Qu.:100.00   3rd Qu.:18.00
##  Max.   :148.00   Max.   :35.00
```

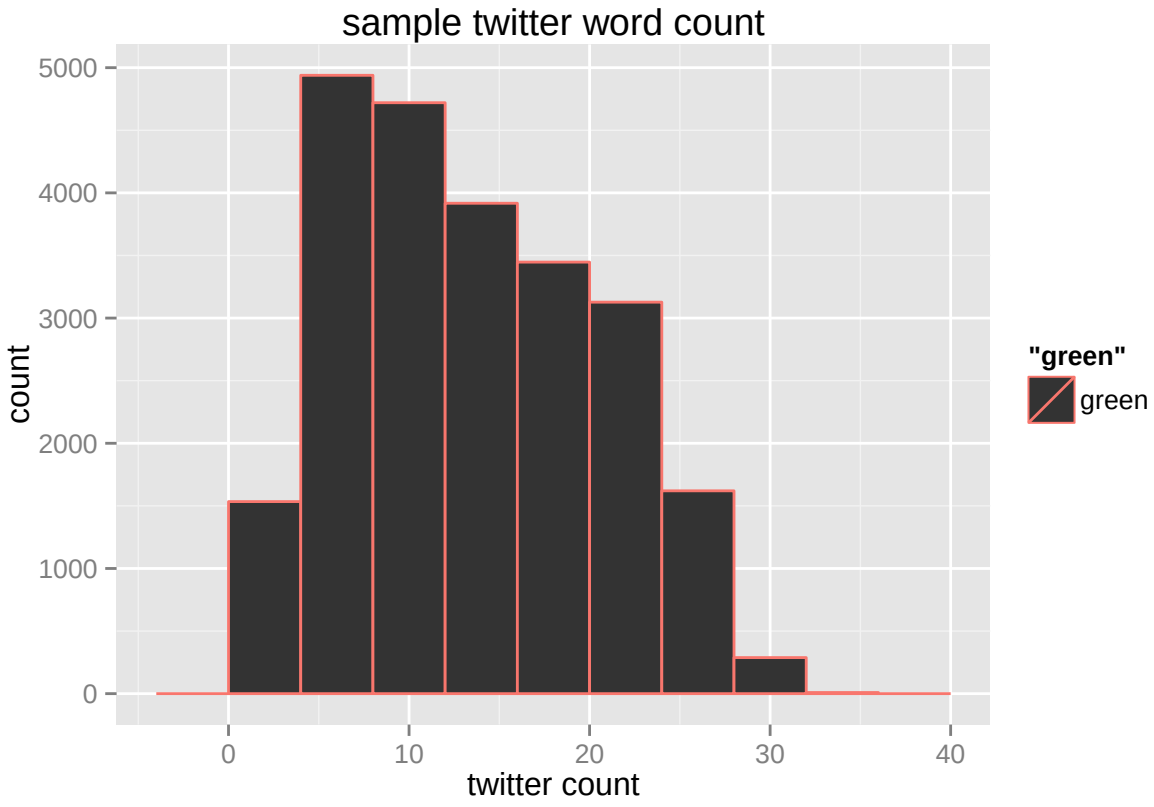
```
library("ggplot2")

qplot(snewsDF$wordCount,main="sample news word count",binwidth=4,xlab="news count")
```




```
qplot(sblogsDF$wordCount,col="red",main="sample blogs word count",binwidth=4,xlab="blog count")
```




```
qplot(stwitterDF$wordCount,col="green",main="sample twitter word count",binwidth=4,xlab="twitter count"
```



Frequency of the words in each texts

To analyze the actual word content, a corpus(a collection of writing) from the three data sets will be created so word frequency can be determined. Strip out numbers, punctuation, stem words, white space and special characters so just the words are available.

With the corpus, create a DocumentTermMatrix using the TM package. Strip sparse terms. Orient for the most frequent words, and list them for review to illustrate the counts of the most frequently found words in the corpus.

```
library("tm")
```

```
## Loading required package: NLP
##
## Attaching package: 'NLP'
##
## The following object is masked from 'package:ggplot2':
##
##     annotate
```

```
corpus <- c(samplenews,sampleblogs,sampletwitter)
corpus <- Corpus(VectorSource(corpus))
```

Building the corpus, removing numbers, whitespaces, special characters and lowercasing all contents.

```
corpus  <- tm_map(corpus,tolower)
corpus  <- tm_map(corpus,removePunctuation)
corpus  <- tm_map(corpus,removeNumbers)
corpus  <- tm_map(corpus,stemDocument)
corpus  <- tm_map(corpus,stripWhitespace)
corpus  <- tm_map(corpus, PlainTextDocument)
```

Tokenizing corpus into matrix by using DocumentTermMatrix.

```
dtm_samples = DocumentTermMatrix(corpus)
```

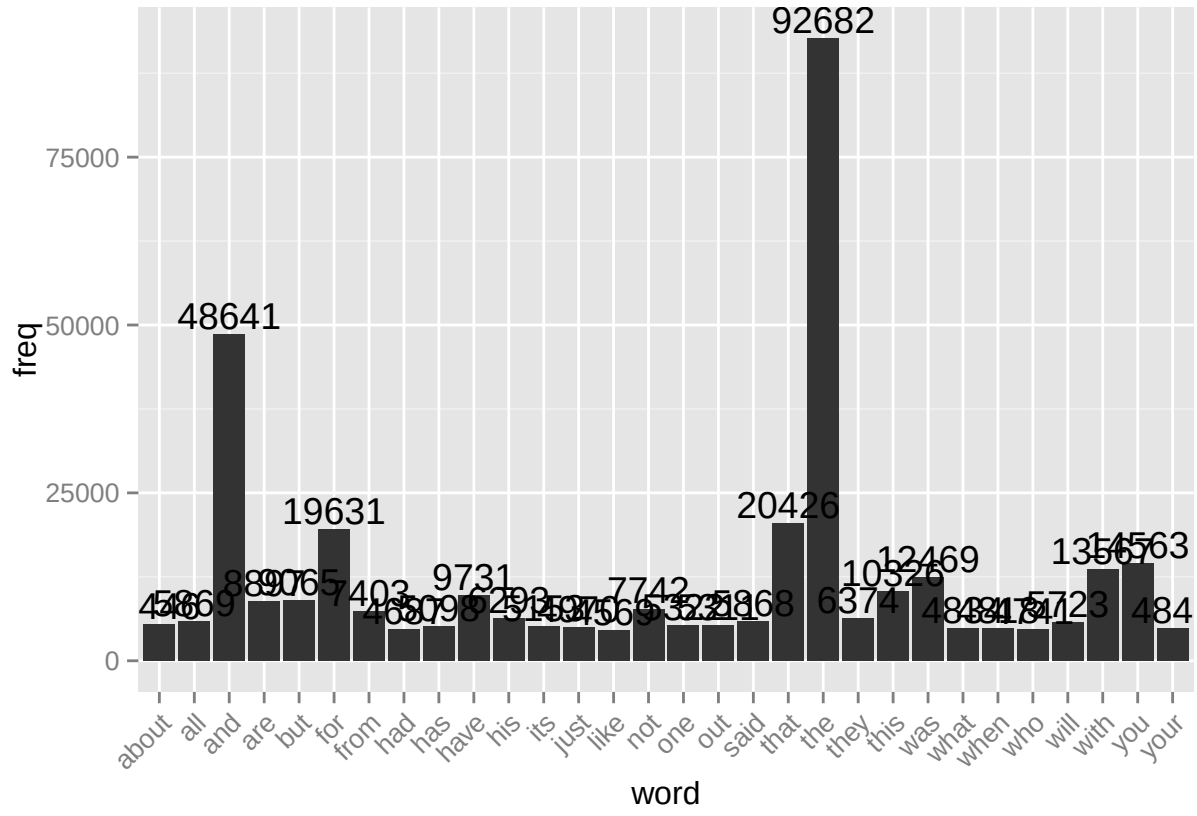
```
dtm_samples1 <- removeSparseTerms(dtm_samples, 0.98)

freq <- sort (colSums (as.matrix(dtm_samples1)), decreasing =TRUE)
wordsf <- data.frame (word=names(freq), freq=freq)

wordsf1 <- wordsf[1:30,]
wordsf1
```

```
##        word  freq
## the     the 92682
## and     and 48641
## that   that 20426
## for     for 19631
## you     you 14563
## with   with 13567
## was     was 12469
## this   this 10326
## have   have  9731
## but     but  9065
## are     are  8897
## not     not  7742
## from   from  7403
## they   they  6374
## his     his  6293
## all     all  5869
## said   said  5868
## will   will  5723
## about about  5446
## one     one  5322
## out     out  5311
## its     its  5153
## has     has  5098
## just   just  4970
## your   your  4843
## what   what  4834
## when   when  4818
## who     who  4741
## had     had  4687
## like   like  4569
```

The most frequent words

Illustrate the information by displaying a histogram (using qplot) of the 15 most frequently found words.

```
ggplot(wordsf1, aes(word, freq)) +  geom_bar(stat="identity") + theme(axis.text.x=element_text(angle=45
```



The Prediction Model

Developing on the frequency characteristics described above, the plan will be to extend the use of the TM library to build n-grams of 2 or 3 words, generated from the data. These n-grams will be used to develop the predictive model. The basic idea will be to use a entered word to find the most likely n-gram (of 2 or more) starting with that word.

Using the n-grams, the Shiny app will accept input and evaluate the input per the created n-grams in order to predict words most likely to follow the entered word using regular expresssions or similar coding techniques.

The Shiny App will display an text input box, allowing for data entry. The output will consist of a list of words per the evaluation of the n-grams.