Coursera Data Science Capstone Milestone report - Exploratory Data Analysis

Edward Lewis

https://rpubs.com/edlewis4/95047

Another very good example:https://rpubs.com/medwards188/LetMeFinishYourSentenceForYou

July 25, 2015

Executive Summary

The is a milestone report for the Coursera Data Science Capstone Project. The overall goal of the project is to build a predictive text application. An application that takes a phrase (multiple words) as input and after the user hits submit, predicts the next word.

This document will cover some exploratory data analysis on the datasets that will be used for this project.

Coursera-SwiftKey.zip dataset

Obtain the data

The first step is to download the data and unzip to local machine.

```
## Datafile
DATAFILE <-
"https://d396qusza40orc.cloudfront.net/dsscapstone/dataset/Coursera-
SwiftKey.zip"

## If file does not exist, download it
if(!file.exists("./Coursera-SwiftKey.zip")) { download.file(DATAFILE,
destfile="./Coursera-SwiftKey.zip", method="curl") }

## Unzip datafile
unzip("./Coursera-SwiftKey.zip")
```

After Unziping we end up with separate directories for German (de_DE), English (en_US), Finnish (fi_FI), and Russian (ru_RU) datafiles.

We will be using the English based files for our analysis and prediction modelling.

We will Read the en_US files for Twitter, Blogs, and News files into character vectors in R

```
con1 <- file("./final/en_US/en_US.twitter.txt", "r")
twitterbuf <- readLines(con1, encoding="UTF-8", skipNul=TRUE)
close(con1)

con2 <- file("./final/en_US/en_US.blogs.txt", "r")
```

```
blogsbuf <- readLines(con2, encoding="UTF-8", skipNul=TRUE)
close(con2)

con3 <- file("./final/en_US/en_US.news.txt", "r")
newsbuf <- readLines(con3, encoding="UTF-8", skipNul=TRUE)
close(con3)
```

Exploratory Data Analysis on the Data

Lets look into some basic stats about each of the data files The Twitter feed file contains `2360148` lines and `162096241` characters

```
length(twitterbuf)      ## Number of lines in file
## [1] 2360148
sum(nchar(twitterbuf))  ## Number of characters in the file
## [1] 162096241
```

Similarly, The Blog feed file contains `899288` lines and `206824505` characters
The News feed file contains `1010242` lines and `203223159` characters

Initial word counts of the files can easily be achieved by a linux system command

```
system("wc -w ./final/en_US/en_US.twitter.txt",intern=TRUE)
## [1] " 30374206 ./final/en_US/en_US.twitter.txt"
system("wc -w ./final/en_US/en_US.blogs.txt",intern=TRUE)
## [1] " 37334690 ./final/en_US/en_US.blogs.txt"
system("wc -w ./final/en_US/en_US.news.txt",intern=TRUE)
## [1] " 34372720 ./final/en_US/en_US.news.txt"
```

Observations about the data

Overall the data is large so we will use a smaller sample of the data (5-10%) as the basis (training data) to build out the prediction model. Also there are a lot of non-words in the data as well as other conditions that will need to be cleaned up to provide better prediction model. The cleanup is described further below.

Pre-processing work to clean up the data

Creating some 10% sample datasets to use for initial analysis

```
library(caTools)

set.seed(45324)
sample_indx1 <- sample.split(twitterbuf,SplitRatio= .1, group=NULL)
train_twit <- twitterbuf[sample_indx1]

sample_indx2 <- sample.split(blogsbuf,SplitRatio= .1, group=NULL)
train_blog <- blogsbuf[sample_indx2]
```

```
sample_indx3 <- sample.split(newsbuf,SplitRatio =.1, group=NULL)
train_news <- newsbuf[sample_indx3]
```

The R packages `tm` and `RWeka` have a number of helpful text mining functions that we will use to help clean up the data.

First we will convert each of the samples datasets to a Corpus of documents and then combine all 3 Corpora into a single `allCorpus` file

```
library(tm)
library(RWeka)
library(slam)

options(mc.cores=1)

## Convert character vectors into to Corpus
twitCorpus <- VCorpus(VectorSource(train_twit))
blogCorpus <- VCorpus(VectorSource(train_blog))
newsCorpus <- VCorpus(VectorSource(train_news))

## combine the 3 corpora into one
allCorpus <- c(twitCorpus,blogCorpus,newsCorpus)
```

There are a number of pre-process items we can do to ensure a better set of data to build our prediction model. Some of these will be completed as part of the implementation and not as part of this initial exploratory analysis.

These involve: Making all text lowercase Removing punctuation Removing special characters Removing Numbers Removing common words (e.g the, to, a, and, of, in, etc.) Removing Profanity Removing infrequent words Removing extra whitespace

We will make some transformations using the R `tm` package to the combined data to clean it up some.

```
allCorpus <- tm_map(allCorpus, content_transformer(tolower))
allCorpus <- tm_map(allCorpus, removePunctuation)
allCorpus <- tm_map(allCorpus, removeNumbers)
allCorpus <- tm_map(allCorpus,
removeWords,c(stopwords("english"),"the","you","and","for","that","with","you
r","have","be","this","are","can","but","what"))
allCorpus <- tm_map(allCorpus, stripWhitespace)

## will add Profanity, infrequent words, and other special situations later
for the final project
```

Next we will build a TermDocumentMatrix which will show the frequency of each term/word. For this analysis we will just look at unigram (single word) terms. For the final prediction algorithm we will look at 2,3,4 and possibly more n-gram word combinations.

```
library(tm)
library(RWeka)
```

```
unigramTokenizer <-function(x) { RWeka::NGramTokenizer(x,
RWeka::Weka_control(min = 1, max = 1)) }

tdm <- TermDocumentMatrix(allCorpus, control=list(tokenize =
unigramTokenizer))

## convert docmatrix to a dataframe with word and freq
df_words <- as.data.frame(slam::row_sums(tdm, na.rm=T))
colnames(df_words)<- "freq"
df_words <- cbind(word = rownames(df_words),df_words)
rownames(df_words) <- NULL

## sort the dataframe by most used words
library(plyr)
df_words_sorted <- arrange(df_words,desc(freq))
```

Here is a plot of the top 25 Words after doing some of the cleaning as described above.

```
library(ggplot2)
ggplot(df_words_sorted[1:25,],aes(word[1:25],freq[1:25]))+
        labs(x="Top 25 words",y="Frequency") +
        ggtitle("TOP 25 Words") +
        theme(axis.text.x=element_text(angle=60, size=18, vjust=0.5)) +
        geom_bar(stat="identity", fill="blue")
```

We can also view this in a wordcloud form for top 50 words

```
library(wordcloud)
wordcloud(df_words_sorted$word,df_words_sorted$freq,max.words=50,rot.per=.3,c
olors=brewer.pal(8,"Dark2"))
```

Prediction Model and building of the final App approach

Here are a few of the major tasks to finish the project: * Perform remaining pre-processing data cleanup + Remove Profanity + Remove infrequent words + Remove special characters + Determine if word Stemming is beneficial

- Create additional TermDocumentMatrices for 2-word, 3-word, 4-word groupings.
- Build a prediction model based on sample data
- Determine if sample size is representative enough
- Build a shiny web app to allow users to type in a phrase of words and hit submit.
- Create Slideshow presentation describing the application