IST687 - Text Mining HW

John Fields

6/10/2019

Introduction

Now that we are doing text mining, we will be creating our own termDocMatrix.

This was also done in class, when we analyzed the structure of the "I have a dream" speech – in terms of the use of positive and negative words. However, in that effort, we treated all positive words the same (ex. good is the same as great). This might not be appropriate – maybe we should count more positive (and negative) words more than other words. For example "I loved the movie" might be stronger than "I liked the movie".

There is a different word file that ranks each word on a scale of -5 to 5 (negative to positive). It is known as the AFINN word list.

Assignment - Part 1

Your task for this homework is to adapt the lab that we did in class, to compute the score for the MLK speech using the AFINN word list (as opposed to the positive and negative word lists). 1. First read in the AFINN word list. Note that each line is both a word and a score (between -5 and 5). You will need to split the line and create two vectors (one for words and one for scores).

```
#read in afinn word list
afinnRaw <- "/Users/johnfields/Desktop/AFINN-en-165.txt"
require(tm)
```

```
## Loading required package: tm
```

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

```
require(stringr)
```

```
## Loading required package: stringr
```

```
input <- readLines(afinnRaw)
#input
afinnTxt <- gsub("\t","",input)
afinnTxt2 <- gsub("-","",afinnTxt)
#separate the word and store in a new vector
afinnWord <- gsub("[[:digit:]]","",afinnTxt2)
#function to separate the score and store in a new vector
#Note: code found at http://stla.github.io/stlapblog/posts/Numextract.html
     numextract <- function(string)
     {
      str_extract(string, "\\-*\\d+\\.*\\d*")
     }
     afinnScore <- numextract(afinnTxt)
afinnCombined <- data.frame(afinnWord,as.numeric(afinnScore))
colnames(afinnCombined) <- c("Word", "Score")
#afinnCombined
afinnCombined[1,]
```

```
##      Word Score
## 1 abandon    -2
```

```
str(afinnCombined)
```

```
## 'data.frame':    3383 obs. of  2 variables:
##  $ Word : Factor w/ 3383 levels "","abandon","abandoned",..: 2 3 4 5 6 7 8 9 10 11 ...
##  $ Score: num  -2 -2 -2 -2 -2 -2 -3 -3 -3 -3 ...
```

```
#View(afinnCombined)
```

Assignment - Part 2

2. Compute the overall score for the MLK speech using the AFINN word list (as opposed to the positive and negative word lists).

```
library(tm)

#read in the MLK speech
mlktxt <- "/Users/johnfields/Desktop/I have a dream.txt"
mlk <- readLines(mlktxt)

words.vec <- VectorSource(mlk)
words.corpus <- Corpus(words.vec)
words.corpus
```

```
## <<SimpleCorpus>>
## Metadata:  corpus specific: 1, document level (indexed): 0
## Content:  documents: 28
```

```
#four transformations
words.corpus <- tm_map(words.corpus,content_transformer(tolower))
```

```
## Warning in tm_map.SimpleCorpus(words.corpus, content_transformer(tolower)):
## transformation drops documents
```

```
words.corpus <- tm_map(words.corpus,removePunctuation)
```

```
## Warning in tm_map.SimpleCorpus(words.corpus, removePunctuation):
## transformation drops documents
```

```
words.corpus <- tm_map(words.corpus,removeNumbers)
```

```
## Warning in tm_map.SimpleCorpus(words.corpus, removeNumbers): transformation
## drops documents
```

```
words.corpus <- tm_map(words.corpus,removeWords,stopwords("english"))
```

```
## Warning in tm_map.SimpleCorpus(words.corpus, removeWords,
## stopwords("english")): transformation drops documents
```

```
#creating a termdocumentmatrix
tdm <- TermDocumentMatrix(words.corpus)
tdm
```

```
## <<TermDocumentMatrix (terms: 445, documents: 28)>>
## Non-/sparse entries: 653/11807
## Sparsity           : 95%
## Maximal term length: 14
```

```
## Weighting          : term frequency (tf)
```

```
m <- as.matrix(tdm)
wordCounts <- rowSums(m)
wordCounts <- sort(wordCounts,decreasing=TRUE)
head(wordCounts)
```

```
##    will freedom     let   negro     one    ring
##      26      19      14      13      12      12
```

```
#calculate the total number of words
totalWords <- sum(wordCounts)
totalWords
```

```
## [1] 793
```

```
#have a vector that just has all the words
words <- names(wordCounts)

# word counts
matchedWords <- match(words,afinnCombined$Word,nomatch=0)
head(matchedWords)
```

```
## [1]    0 1358    0    0    0    0
```

```
matchedScores <- as.numeric(afinnCombined$Score[matchedWords])
matchedCount <- wordCounts[which(matchedWords!=0)]
sum(matchedCount*matchedScores)
```

```
## [1] 96
```

Assignment - Part 3

3. Then, just as in class, compute the sentiment score for each quarter (25%) of the speech to see how this sentiment analysis is the same or different than what was computing with just the positive and negative word files. Note that since you will be doing almost the exact same thing 4 times (once for each quarter of the speech), you should create a function to do most of the work, and call it 4 times.

```
#split the speech into 4 equal parts
MLK1Q <- words.corpus[1:7]$content
MLK2Q <- words.corpus[8:14]$content
MLK3Q <- words.corpus[15:21]$content
MLK4Q <- words.corpus[21:28]$content

#create a function to run each quarter
#This section developed with help from this site --> https://github.com/jashmehta89/IST-687-Applied-Dat
sentScore <- function(quarter)
{
   q.vector <- VectorSource(quarter)
   q.corpus <- Corpus(q.vector)
   q.TDM <- TermDocumentMatrix(q.corpus)
   q.matrix <- as.matrix(q.TDM)
   q.WordCount <- rowSums(q.matrix)
   q.Words <- names(q.WordCount)
   q.Matched <- match(q.Words,afinnCombined$Word,nomatch=0)
   q.MatchScore <- as.numeric(afinnCombined$Score[q.Matched])
   q.MatchCount <- q.WordCount[which(q.Matched!=0)]
```

```
   q.Score <- sum(q.MatchCount*q.MatchScore)
   return (q.Score)
}

#run the function for each quarter
first <- sentScore(MLK1Q)
first
```

```
## [1] 16
```

```
second <- sentScore(MLK2Q)
second
```

```
## [1] 28
```

```
third <- sentScore(MLK3Q)
third
```

```
## [1] 20
```

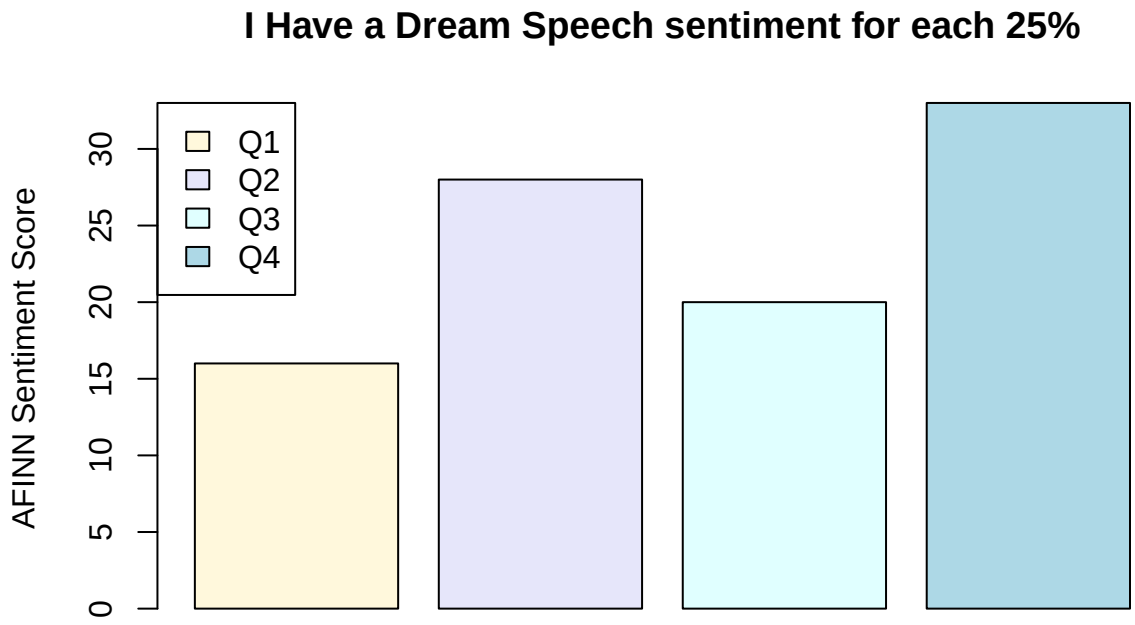
```
fourth <- sentScore(MLK4Q)
fourth
```

```
## [1] 33
```

Assignment - Part 4

4. Finally, plot the results (i.e, 4 numbers) via a bar chart.

```
mlkAllQ <- c(first,second,third,fourth)
barplot(mlkAllQ, col = c("cornsilk", "lavender", "lightcyan","lightblue"), legend.text=c("Q1","Q2","Q3"
```



Learning Goals for this activity:

A. Consider how a simple text mining technique can be applied to a variety of kinds of source data. B. Provide practice in conditioning data to prepare for analysis. C. Develop skill in the setup, execution, and

interpretation of text mining analytics. D. Increase familiarity with bringing external data sets into R.