

SFDC Topic Modeling and Sentiment

Ken Brooks

June 3, 2015

Adapted from Ted Kwartler (Ted@sportsanalytics.org (mailto:Ted@sportsanalytics.org)), Open Data Science Conference Workshop: Intro to Text Mining using R, 5-30-2015, v7.0 Topic Modeling and simple sentiment

```
#Set the working directory and import libraries
#setwd("~/Google Drive KB/Open Source Conf")

dataDir <- if(interactive()) 'data' else '../data'

#libraries
library(tm)
```

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

```
library(topicmodels)
#install.packages('topicmodels')
library(portfolio)
```

```
## Loading required package: grid
```

```
## Loading required package: lattice
```

```
## Loading required package: nlme
```

```
#install.packages("portfolio")
#library(ggplot2)
#library(ggthemes)
library(plyr)
library(stringr)
library(dplyr)
```

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

```
## The following objects are masked from 'package:plyr':  
##  
##      arrange, count, desc, failwith, id, mutate, rename, summarise,  
##      summarize
```

```
## The following object is masked from 'package:nlme':  
##  
##      collapse
```

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

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

Set options and defined functions

```
#options, functions  
options(stringsAsFactors = FALSE)  
Sys.setlocale('LC_ALL', 'C')
```

```
## [1] "C/C/C/C/C/en_US.UTF-8"
```

```

#try to lower function
tryTolower <- function(x){
  # return NA when there is an error
  y = NA
  # tryCatch error
  try_error = tryCatch(tolower(x), error = function(e) e)
  # if not an error
  if (!inherits(try_error, 'error'))
    y = tolower(x)
  return(y)
}

clean.corpus<-function(corpus){
  corpus <- tm_map(corpus, removePunctuation)
  corpus <- tm_map(corpus, stripWhitespace)
  corpus <- tm_map(corpus, removeNumbers)
  corpus <- tm_map(corpus, content_transformer(tryTolower))
  corpus <- tm_map(corpus, removeWords, custom.stopwords)
  return(corpus)
}

#Bigram token maker
bigram.tokenizer <-function(x)
  unlist(lapply(ngrams(words(x), 2), paste, collapse = " "), use.names = FALSE)

#Bring in subjective lexicons
pos <- readLines("positive_words.txt")
neg <- readLines("negative_words.txt")

#Simple sentiment subject word counter function, poached online
score.sentiment = function(sentences, pos.words, neg.words, .progress='none')
{
  scores = lapply(sentences, function(sentence, pos.words, neg.words) {
    word.list = str_split(sentence, '\\s+')
    words = unlist(word.list)
    # compare our words to the dictionaries of positive & negative terms
    pos.matches = match(words, pos.words)
    neg.matches = match(words, neg.words)
    pos.matches = !is.na(pos.matches)
    neg.matches = !is.na(neg.matches)
    #TRUE/FALSE will be treated as 1/0 by sum():
    score = sum(pos.matches) - sum(neg.matches)
    return(score)
  }, pos.words, neg.words, .progress=.progress )
  scores.df = data.frame(score=scores, text=sentences)
  return(scores.df)
}

```

Create custom stop words

```
#Create custom stop words
```

```
custom.stopwords <- c(stopwords('english'), 'lol', 'smh', 'learning curve', 'learning curves')
```

Import and clean text, build dtm

```
#bring in some text
```

```
#text <-read.csv (file.path dataDir, 'SFDC_Survey.csv', header=TRUE)
```

```
text <-readxl::read_excel('LP Spring 2016 Instructor Survey- 3_Free Responses.xlsx')
```

```
#Create a clean corpus
```

```
# Instructor:
```

```
coll = "What do you like most about LaunchPad?"
```

```
# coll = "How can LaunchPad be improved? Tell us one feature or function that could be added or improved to make your experience of LaunchPad better."
```

```
#Student:
```

```
# coll = "How can LaunchPad be improved? Tell us one feature or function that could be added or improved to make your LaunchPad experience better."
```

```
corpus <- Corpus(DataframeSource(data.frame(text[[coll]])))
```

```
corpus <- clean.corpus(corpus)
```

```
#Make a DTM
```

```
dtm<-DocumentTermMatrix(corpus, control=list(tokenize=bigram.tokenizer))
```

Perform topic modeling

```

#In Topic Modeling, remove any docs with all zeros after removing stopwords
rowTotals <- apply(dtm , 1, sum)
dtm.new   <- dtm[rowTotals> 0, ]

#In Sentiment, to ensure the number of rows in the dtm.new and the sentiment data frame equal
text <- cbind(text, rowTotals)
text <- text[rowTotals> 0, ]

#Begin Topic Modeling; can use CTM or LDA
topic.model <- LDA(dtm.new, control = list(alpha = 0.1), k = 5)

#Topic Extraction
topics<-get_terms(topic.model, 5)
colnames(topics)<-c("topic1","topic2","topic3","topic4","topic5")
topics<-as.data.frame(topics)
t1<-paste(topics$topic1,collapse=' ')
t2<-paste(topics$topic2,collapse=' ')
t3<-paste(topics$topic3,collapse=' ')
t4<-paste(topics$topic4,collapse=' ')
t5<-paste(topics$topic5,collapse=' ')
topics

```

```

##          topic1          topic2          topic3
## 1      easy use      ease use      really like
## 2 students use      students like      user friendly
## 3 use students classes traditional activities videos
## 4      also like      easy use      also really
## 5      can assign encourage students      case studies
##
##          topic4          topic5
## 1      students can      ease use
## 2      auto grading      ability give
## 3          easy set      able assign
## 4          like idea adaptive learning
## 5 powerpoint presentations      ebook quizzing

```

Assign documents to topics

```
#Score each tweet's probability for the topic models then add the topic words to the df as headers
scoring<-posterior(topic.model)
scores<-scoring$topics
scores<-as.data.frame(scores)
colnames(scores)<-c(t1,t2,t3,t4,t5)

#The max probability of each tweet classifies the tweet document
topics.text<-as.data.frame(cbind(row.names(scores),apply(scores,1,function(x) name
s(scores)[which(x==max(x))]))))
```

Perform sentiment scoring

##

		0%
		1%
=		1%
=		2%
==		3%
==		4%
===		4%
===		5%
====		6%
====		7%
=====		7%
=====		8%
=====		9%
=====		10%
=====		10%
=====		11%
=====		12%
=====		12%
=====		13%
=====		13%
=====		14%
=====		15%
=====		15%
=====		16%

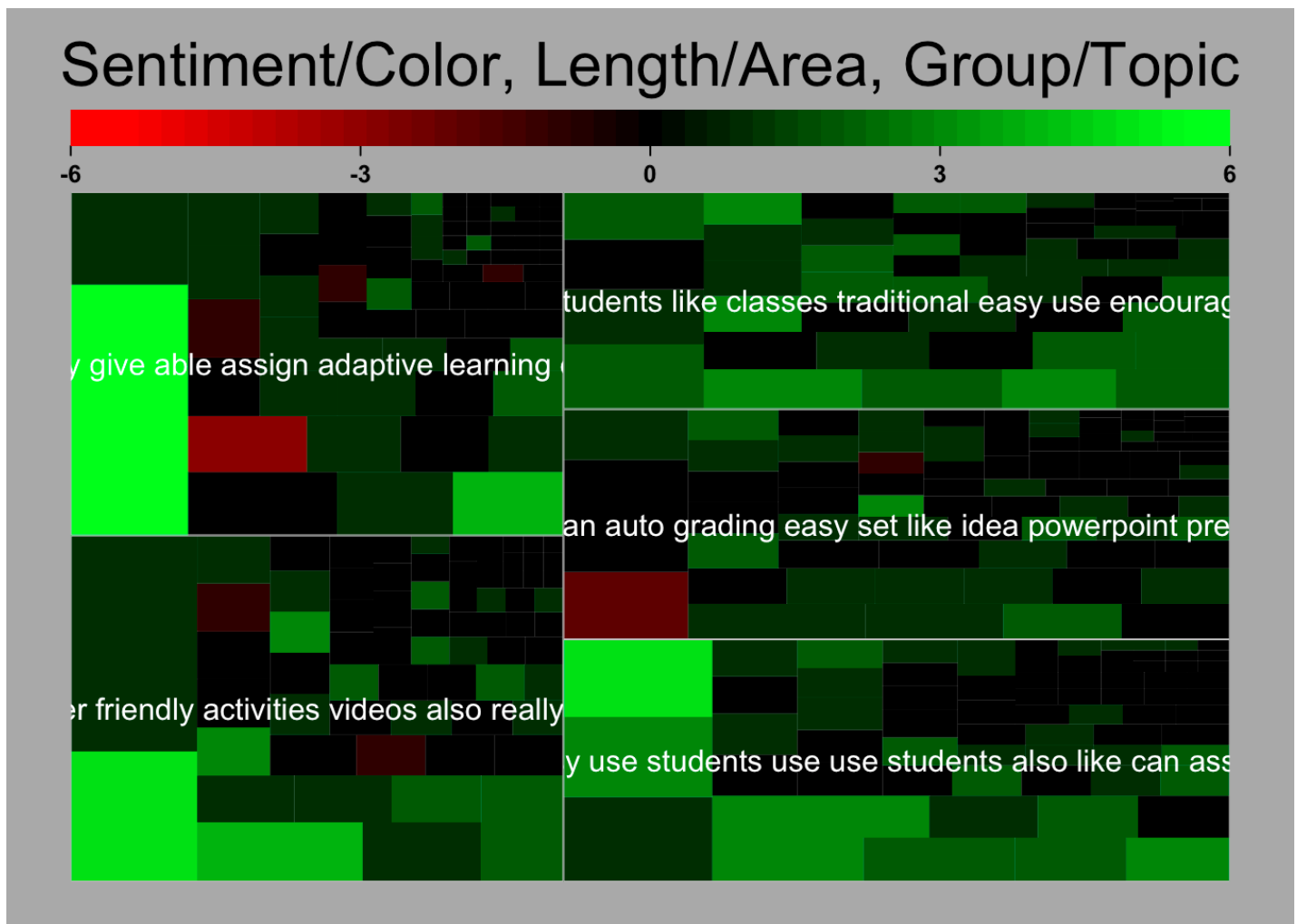
=====	16%
=====	17%
=====	18%
=====	19%
=====	19%
=====	20%
=====	21%
=====	21%
=====	22%
=====	22%
=====	23%
=====	24%
=====	24%
=====	25%
=====	26%
=====	27%
=====	27%
=====	28%
=====	29%
=====	30%
=====	30%
=====	31%
=====	32%
=====	33%
=====	33%
=====	34%

=====	35%
=====	35%
=====	36%
=====	36%
=====	37%
=====	38%
=====	38%
=====	39%
=====	39%
=====	40%
=====	41%
=====	41%
=====	42%
=====	42%
=====	43%
=====	44%
=====	44%
=====	45%
=====	46%
=====	47%
=====	47%
=====	48%
=====	49%
=====	50%
=====	50%

=====	51%
=====	52%
=====	53%
=====	53%
=====	54%
=====	55%
=====	56%
=====	56%
=====	57%
=====	58%
=====	58%
=====	59%
=====	59%
=====	60%
=====	61%
=====	61%
=====	62%
=====	62%
=====	63%
=====	64%
=====	64%
=====	65%
=====	65%
=====	66%
=====	67%
=====	67%

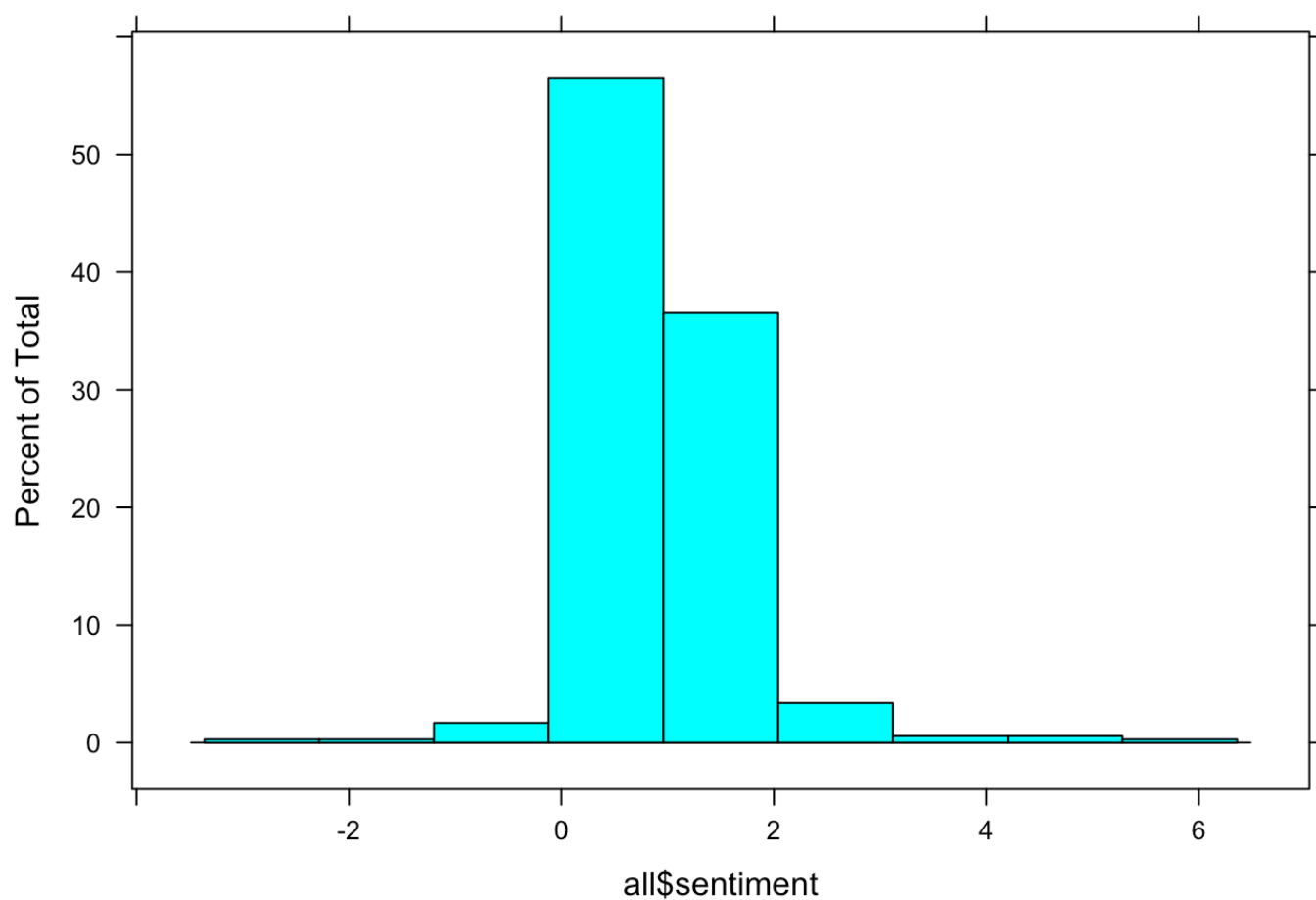
=====	68%
=====	69%
=====	70%
=====	70%
=====	71%
=====	72%
=====	73%
=====	73%
=====	74%
=====	75%
=====	76%
=====	76%
=====	77%
=====	78%
=====	78%
=====	79%
=====	79%
=====	80%
=====	81%
=====	81%
=====	82%
=====	83%
=====	84%
=====	84%
=====	85%

6/29/2016	SFDC Topic Modeling and Sentiment	
	=====	85%
	=====	86%
	=====	87%
	=====	87%
	=====	88%
	=====	88%
	=====	89%
	=====	90%
	=====	90%
	=====	91%
	=====	92%
	=====	93%
	=====	93%
	=====	94%
	=====	95%
	=====	96%
	=====	96%
	=====	97%
	=====	98%
	=====	99%
	=====	99%
	=====	100%



Sort comments with most negative on top and print them

```
histogram(all$sentiment)
```



```
sent.limit = -5
```

```
all %>% filter(sentiment <= sent.limit) %>% arrange(desc(sentiment)) %>% select(text)
```

```
## [1] text
## <0 rows> (or 0-length row.names)
```

Plot sentiment over time

- Make sure data frame is in date order
- aggregate by week?
- plot time series
- add a loess trend line