Social Media Final Project: Airline Compliant Sentiment Analysis

Siqi Jiang

December 18 2019

1. Load data

```
library(DBI)
library(RMySQL)
library(tm)
require("NLP")
install.packages("NMF")
require("openNLP")
library("tm")
library("SnowballC")
library("RColorBrewer")
library("wordcloud")
library(NMF)
library(plyr) #weight sentiment
#install.packages("rJava")
#install.packages("Rwordseg")
library(rJava)
library(Rwordseg)
#access to text data
driver <- dbDriver("MySQL")
myhost <- "localhost"
mydb   <- "studb"
myacct <- "cis434"
mypwd  <- "LLhtFPbdwiJans8F@S207"
conn <- dbConnect(driver, host=myhost, dbname=mydb, myacct, mypwd)

#LOAD TEXT DATA
# rO(x6c15'Can
temp <- dbGetQuery(conn, "SELECT * FROM proj4final WHERE tag=\"rO(x6c15'Can\"")
dbDisconnect(conn)
```

2. Import pos/neg dictionary
    1) Assign weight “+1” to positive words and “-1” negative words

```
####load pos/neg dictionary "posneg"
# positive words ----label +1
pos <- read.csv("positive-words-dic.txt", header = T, sep = ",", stringsAsFactors = F)
weight <- rep(1, length(pos[,1]))
pos <- cbind(pos, weight)
names(pos) <- c("term", "weight")
#pos <- tolower(pos)
# negative words ----label -1
neg <- read.csv("negative-words-dic.txt", header = T, sep = ",", stringsAsFactors = F)
weight <- rep(-1, length(neg[,1]))
neg <- cbind(neg, weight)
names(neg) <- c("term", "weight")
```

3. Adjust dictionary (dictionary extension)

1) Read “x” that only contains 4555 tweets into corpus and convert to DocumentTermMatrix.

```
docs <- Corpus(DataframeSource(x))
mystop=c("now","dont","like","amp","http","love","have","I","get","got","will","can","just","been","flying")
dtm <- DocumentTermMatrix(docs, control=list(tolower=T, removePunctuation=T,
                                             removeNumbers=T, stripWhitespace=T,
                                             stopwords=c(mystop, stopwords("english"), stopwords("spanish"))))
dtm = removeSparseTerms(dtm,0.996)
idx <- rowSums(as.matrix(dtm))>0
newdocs <- docs[idx]
dtm = dtm[idx,]

```

2) Construct LDA model and topic modeling

```
lda.model=LDA(dtm,10)
myposterior=posterior(lda.model)
dices=myposterior$terms
coins=myposterior$topics
```

3) Adjust parameter ‘tid’ and see what words are frequently showing up with negative words

```
# adjusting parameter tid=2
tid <- 2
dice <- dices[tid,]
freqterms=sort(dice,decreasing=TRUE)
freqterms[1:10]
wordcloud(names(dice),dice, max.words = 20,colors=brewer.pal(6,"Dark2"), scale=c(0.8,1.5))
# find "lost"

# adjusting parameter tid=4
tid <- 4
dice <- dices[tid,]
freqterms=sort(dice,decreasing=TRUE)
freqterms[1:10]
wordcloud(names(dice),dice, max.words = 20,colors=brewer.pal(6,"Dark2"), scale=c(0.8,1.5))
# find "trying" and "cancel"

# adjusting parameter tid=5
tid <- 5
dice <- dices[tid,]
freqterms=sort(dice,decreasing=TRUE)
freqterms[1:10]
wordcloud(names(dice),dice, max.words = 20,colors=brewer.pal(6,"Dark2"), scale=c(0.8,1.5))
# find "wait"

```

(an example of picking out words)

4) By trying ten times, I found “lost”, “trying”, “wait”, “cancel”, which are not in my dictionary so that I add them into my dictionary.

5) I adjust weights to -100 on some words that frequently show in the comments that deliver negative message:

“bad”; “delay”; “delayed”; “lost”; “last”; “trying”; “sucks”; “try”; “late”; “rudest”; “stuck”; “wait”; “waiting”; “cancel”; “cancelled”; “cancellation”.

```
# assign special weights to the negative words that I found from LDA
#They are: bad;delay;delayed;lost; last;trying; sucks; try; sucks; late;
###########rudest;stuck;wait;waiting;cancel;cancelled;shitty; stupid; ashamed
neg$weight[which(neg$term =="bad")]=-100
neg$weight[which(neg$term =="delay")]=-100
neg$weight[which(neg$term =="delayed")]=-100
neg$weight[which(neg$term =="lost")]=-100
neg$weight[which(neg$term =="last")]=-100
neg$weight[which(neg$term =="trying")]=-100
neg$weight[which(neg$term =="sucks")]=-100
neg$weight[which(neg$term =="try")]=-100
neg$weight[which(neg$term =="late")]=-100
neg$weight[which(neg$term =="rudest")]=-100
neg$weight[which(neg$term =="stuck")]=-100
neg$weight[which(neg$term =="wait")]=-100
neg$weight[which(neg$term =="Waiting")]=-100
neg$weight[which(neg$term =="waiting")]=-100
neg$weight[which(neg$term =="cancel")]=-100
neg$weight[which(neg$term =="cancelled")]=-100
neg$weight[which(neg$term =="Cancel")]=-100
neg$weight[which(neg$term =="cancellation")]=-100
neg$weight[which(neg$term =="Thieves")]=-100
neg$weight[which(neg$term =="ashamed")]=-100
neg$weight[which(neg$term =="shitty")]=-100
neg$weight[which(neg$term =="stupid")]=-100

```

I did this to make sure that if the sentence contains these words, it should be weighted out, which means that it will show up in my non-negative csv.

4. Combine positive and negative vocabulary and its weights to a dictionary.

```
# combine pos and neg words
posneg <- rbind(pos, neg)
posneg <- posneg[!duplicated(posneg$term), ]

#`duplicated` is similar to `unique`, but it can return to its own number
```

5. Clean text
    1) Convert tweets to a vector by using “as.vector”
    2) Clean text: remove punctuations, digits, and lower case

```
86  ########divide words into vectors in each tweet
87  #make them to each string
88  sentence <- as.vector(temp$tweet)
89  #clean text
90  sentence= gsub("[[:punct:]]", "", sentence )
91  sentence = gsub("[[:digit:]]", "", sentence )
92  sentence <- tolower(sentence)
93
```

6. Add ids to the terms
    1) Divide every word of each tweet to a single term by using “tokenize_words”
    2) Getting the number of terms by using “length” and replicate id with the same length.
    3) Unlist terms back to sentences and add with its assigned ids by using “cbind.”

```
95  #install.packages("tokenizers")
96  library("tokenizers")
97  x<-tokenize_words(sentence, strip_punct = FALSE)
98
99  term <- unlist(x)
100 temp0 <- lapply(x, length)
101 temp0 <- unlist(temp0)
102 id <- rep(temp$id, temp0)
103
104 #create a data frame that contains terms and their ids
105 testterm <- as.data.frame(cbind(id, term), stringsAsFactors = F)
106
```

7. Create a set of vocabulary of stop words and clean text with eliminating stop words.

```
109 # create a set of vocabulary and clean data with elimiating stop words
110
111 stopwords<-data.frame(stopwords(kind='en'))
112 names(stopwords) <- c('term')
113 stopwords <- data.frame(setdiff(stopwords$term,posneg$term))
114 names(stopwords) <- c('term')
115 testterm <- testterm[!testterm$term %in% stopwords,]
116
```

8. Weight each review with each word that contains weight from the dictionary that I made earlier and compute sentiment index

```
#weight each review and weight with pos/neg dictionary
library(plyr)

testterm <- merge(testterm, posneg)
testterm <- testterm[!is.na(testterm$weight), ]
#head(testterm)
#computing sentiment index
dictresult <- aggregate(weight ~ id, data = testterm, sum)
dictlabel <- rep(-1, length(dictresult[, 1]))

```

9. Get a data frame that only contains non-negative reviews by subset weight >0

```
#convert dictlabel which contains sign of sentiment to the data frame
dictlabel[dictresult$weight > 0] <- 1
dictresult <- as.data.frame(cbind(dictresult, dictlabel), stringsAsFactors = F)
###getting data frame only with non negative review
text <- join(dictresult, temp, by="id")
nonNeg <- subset(text, weight>0)
non_Negative <- nonNeg
```

10. Clean data frame by removing some columns that are not required in the homework.

```
#Finish non-negative data frame
non_Negative$weight <- NULL
non_Negative$dictlabel<-NULL
non_Negative$tag<-NULL
non_Negative$airline<-NULL
non_Negative$tid_not_to_be_used<-NULL
non_Negative$Evaluation <-1
non_Negative <- non_Negative[, c("id", "Evaluation", "tweet")]
```

11. Import complaint1700.csv to test my model
    1) Result: roughly 104 out of 1700 are non-complaint in the complaint1700.csv

```
> temp0 <- lapply(x, length)
> temp0 <- unlist(temp0)
> id <- rep(temp$id, temp0)
> #create a data frame that contains terms and their ids
> testterm <- as.data.frame(cbind(id, term), stringsAsFactors = F)
> stopwords<-data.frame(stopwords(kind='en'))
> names(stopwords) <- c('term')
> stopwords <- data.frame(setdiff(stopwords$term,posneg$term))
> names(stopwords) <- c('term')
> testterm <- testterm[!testterm$term %in% stopwords,]
> #weight each review and weight with pos/neg dictionary
> library(plyr)
> testterm <- merge(testterm, posneg)
> testterm <- testterm[!is.na(testterm$weight), ]
> #head(testterm)
> #computing sentiment index
> dictresult <- aggregate(weight ~ id, data = testterm, sum)
> dictlabel <- rep(-1, length(dictresult[, 1]))
> #convert dictlabel which contains sign of sentiment to the data frame
> dictlabel[dictresult$weight > 0] <- 1
> dictresult <- as.data.frame(cbind(dictresult, dictlabel), stringsAsFactors = F)
> ###getting data frame only with non negative review
> text <- join(dictresult, temp, by="id")
> nonNeg <- subset(text, weight>0)
> non_Negative <- nonNeg
> #Finish non-negative data frame
> non_Negative$weight <- NULL
> non_Negative$dictlabel<-NULL
> non_Negative$tag<-NULL
> non_Negative$airline<-NULL
> non_Negative$tid_not_to_be_used<-NULL
> non_Negative$Evaluation <-1
> non_Negative <- non_Negative[, c("id", "Evaluation", "tweet")]
> dim(non_Negative) #160 non negative reviews
[1] 104   3
```

12. Export to .cvs file

```
144  #export non_Negative
145
146  write.csv(non_Negative, "non_Negative.csv")
147  dim(non_Negative) #160 non negative reviews
148
```

13. Manually check

    After manually check, there are 127 out of 160 non-negative, and precision is 79.375%.

    There are still 33 reviews that contains obviously negative messages and sarcastic comments, which are hard to find by matching with dictionary and rating reviews. This is because these reviews that I did not find out are being sarcastic. These sarcastic reviews often contain positive words, such as “great”, “thanks”, and “nice job.” In addition, if the review is about praising one airline and scolding another, then I checked the raw data with its ID and respectively airline. If the one that is being praised belongs to the airline company under the same ID, I rated it as non-negative (1). If the review is negative but not directly toward the airline, I rated it as non-negative. If the sentiment of the review is being ambiguous, I rated it as non-negative.

Citation

Minqing Hu and Bing Liu. "Mining and Summarizing Customer Reviews." Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2004), Aug 22-25, 2004, Seattle, Washington, USA,