University of Southampton

Text Mining: Tweets from British Airways

Consumer Sentiment Analysis

Jiaming Jiang

1/11/2020

EXCLUSIVE SUMMARY

With the advent of the data age, the data volume of Tweets has shown an explosive growth; meanwhile, text mining has become a new way for marketing research. This report primary using Support Vector Machine / Naive Bayes, Latent Semantic Analysis and Emotional Valence Analysis study on British Airways/ Virgin Atlantic Airways tweets in the UK. One key finding is "Forced Deportation" caused a significant decrease of BA's sentiment score. Ignoring this decline, both airlines scored the same. Another one is High-quality service offset flight delay on customer satisfaction. Which two findings contribute to enterprise further understands consumer psychology and behavior.

TABLE OF FIGURES

TABLE OF CONTENT

A. Introduction

With the development of Internet technology, the amount of data generated by people's social and shopping activities through the network is growing explosively. Increasingly users have expressed their opinions on twitter, which are usually spread by text. Mining users' sentiment information efficiently can provide the basis for the marketing strategy of enterprises. At present, the recognized research work of sentiment analysis began in Pang (2002), who used supervised learning to classify the sentiment polarity of film comment text; in the same year, Tumey(2002) proposed the method of unsupervised learning to analyze the sentiment tendency of the text. Wilson et al. (2005) believed that the main task of sentiment analysis is to identify the comment sentences Positive and negative views and sentiments. Vinodhini et al. (2016) think that sentiment analysis is to track sentences containing public sentiments, and Cambria et al. (2013) think that sentiment analysis is to classify sentiment polarity.

B. Technology Road

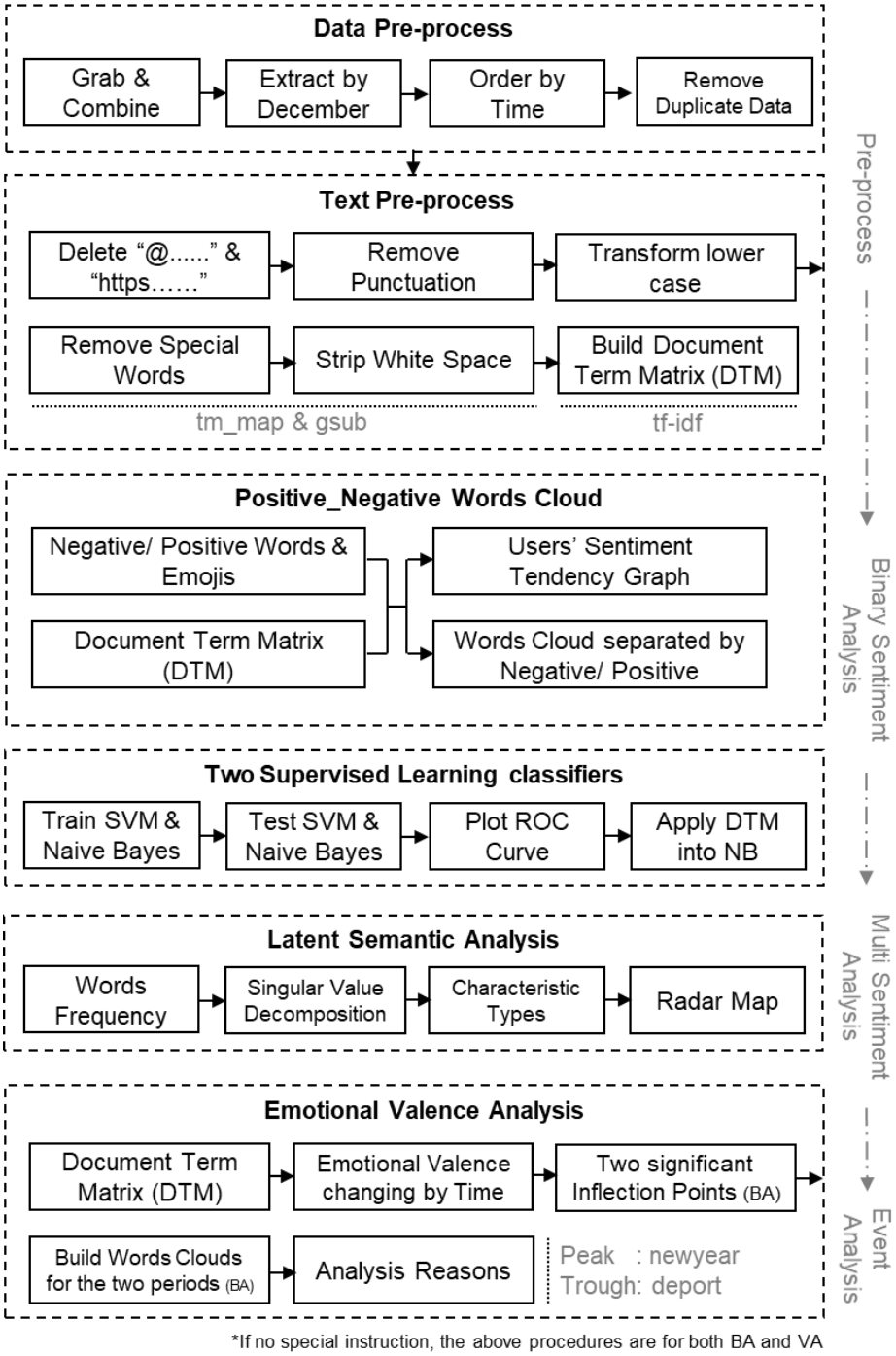


Figure 1 *Technology Roadmap*

I. Data Pre-process

First, as twitter developers, Capture and merge the tweets data. After that, extract data within December and delete duplicate data. For the convenience of the following steps, arrange the tweets in chronological order.

II. Text Pre-process

Remove spaces, punctuation, some special words and change the text to lowercase. It is worth mentioning, "@....." and "https……" also should be removed, otherwise, these words will affect the accuracy of the following steps. Now these clean texts can be used to build Document Term Matrix (DTM) by TF-IDF (term frequency–inverse document frequency), which algorithm is able to reasonably evaluate the importance of a word to a document set or one of the documents in a corpus. DTM will be used as data source in the next four steps.

III. Positive_Negative Words Cloud

In this step, Hu Liu's negative_positive dictionary(Hu and Liu, 2004) is introduced to determine the sentiment tendency of each sentence. I also created N/P emojis dictionary by myself, which cost me five hours. This is a simple sentiment analysis classifier, which can determine whether each sentence is positive or negative. The results of VA and Ba are shown in ***Figure 2*** The abscissa in the figure shows the sentiment score of each sentence. The higher the score, the more positive the user's attitude, and vice versa. For example, users with a score of 2 must be extremely happy when pushing this tweet, while users with a score of - 2 may have complained to us before sending the tweet. The ordinate represents the number of sentences in the score range. As shown in this figure, The sentiment scores of users are normally distributed. But this picture is just a by-product. In the fourth step, I will use a more accurate unsupervised learning classifier to analyze these tweets.

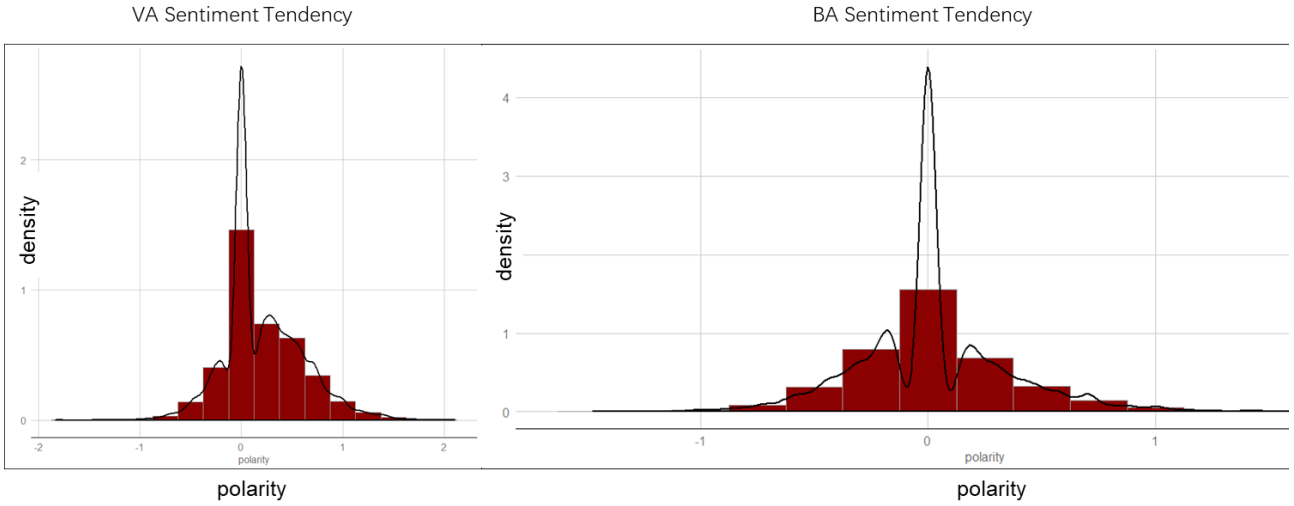


Figure 2 *Sentiment Tendency Histogram*

In fact, negative_positive words cloud is the main part of this step. ***Figure 3*** shows a comparison of Ba and VA's words cloud in December. It is not hard to see that there is no significant difference between them; therefore, it is not significant to study these words cloud. The reason why there is no significant difference may be that the time line of a whole month is too long, which leads to the convergence of texts and I will improve this in step 6.

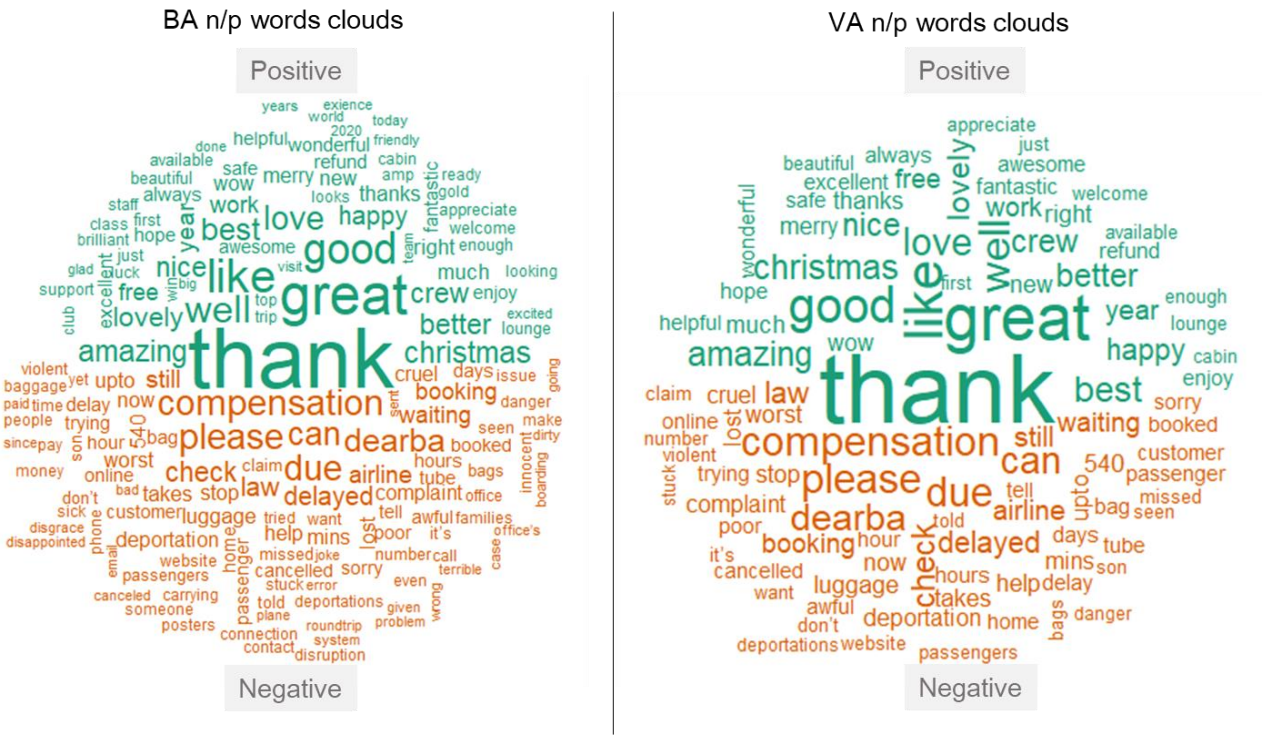


Figure 3 *Negative_Positive Words Clouds*

IV. Two Supervised Learning classifiers

In this step, I trained SVM and naive Bayes. The accuracy of the original Bayesian classifier is less than 15%, with such a classifier, if we want to get the right answer, we may need to reverse analyze the results. Therefore, I improved it. The ROC curve of SVM and improved Naive Bayes is shown in ***Figure 4*** Their accuracy was 80.3% and 85.5% respectively.

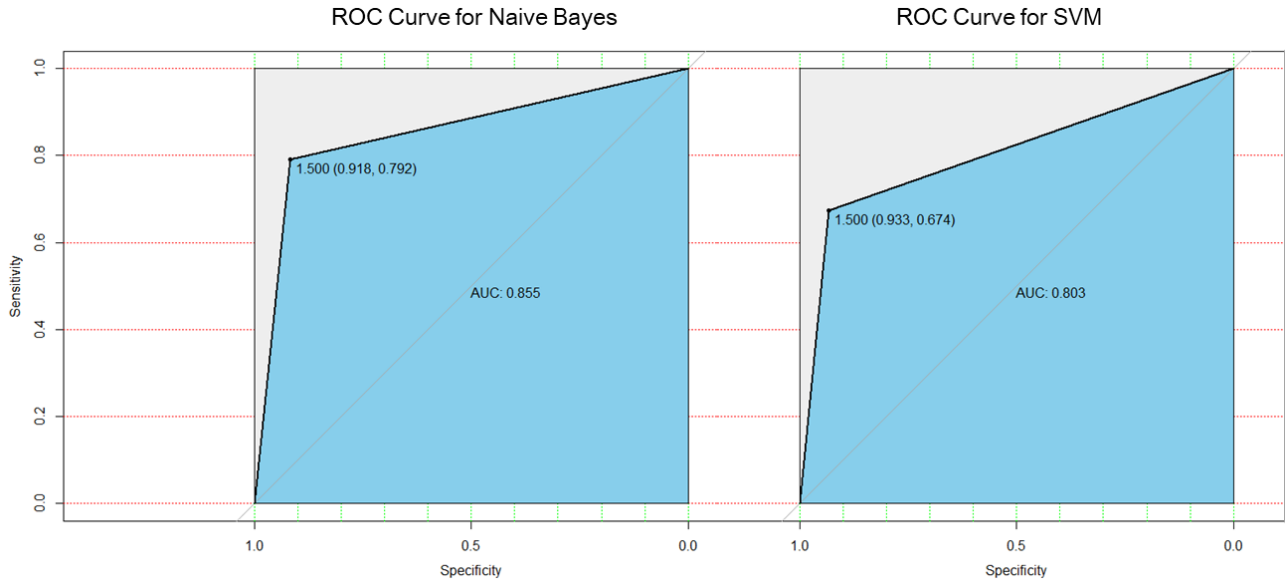


Figure 4 *ROC Curve for Naive Bayes and SVM*

Because the accuracy of naive Bayes is higher, I substituted the data of Ba and VA to get the results shown in the following table. This result is not evident different from the histogram obtained in the third step, so to a high degree, the result is acceptable.

Table 1 *Naive Bayes Prediction*

	0	1
BA	11939	7445
VA	2480	4567

V. Latent Semantic Analysis

Input the matrix form of word frequency, and then do Singular Value Decomposition. According to the output results, the top 60 words in the word frequency map are divided into five main types: Time, Travel, Holiday, Home and Service.

VI. Sentiment Valence Analysis

Using DTM, get users' Sentiment Valence changing by Time. In this graph, there are Two significant Inflection Points for BA. After build ana analyze the Words Clouds for these two periods. I found the Peak and Trough are most likely influenced by the coming of new year and "Forced Deportation" event.

C. Event Analysis

In ***Figure 5*** This Blue Smooth Curve depicts BA users' Emotional Valence changing by Time, and orange for VA. Abscissa represents time, starting from the initial time (1/12/2019 00:00:00) and ending time (31/12/2019 23:59:59). The area under the curve in the figure shows the emotional score of consumers. Sentiment score of BA is less than VA, but the main reason is affected by its Trough Event (Forced Deportations). If the impact of this event is eliminated, the sentiment score of BA users is as same as that of VA (as shown in the dotted line in the figure).

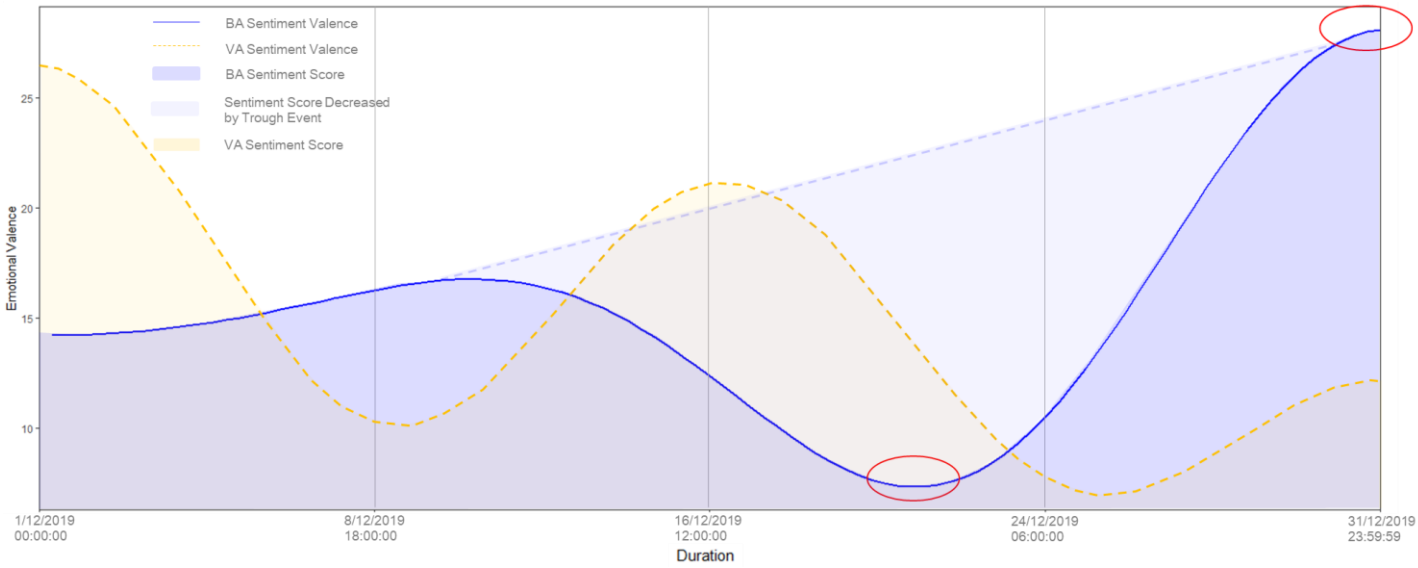


Figure 5 *Emotional Valence Curve*

The peak and Trough of this curve are obviously. I intercepted their time periods (Peak: 31/12/2019 00:00:00—1/1/2019 23:59:59 and Trough 18/12/2019 00:00:00—20/12/2020 23:59:59) and plot their general words cloud, as shown in ***Figure 6*** The third picture is the general words cloud of BA in the whole December, which is used for horizontal comparison with the first two pictures.

Figure 6 *BA Words Cloud in Critical time periods*

I. TROUGH Words Cloud

Special words in TROUGH Words Cloud are: **deport, dearba, stop, home, want, cruel, violent, tube**. The Trough is because of our assisting the government in deportations. Since VA stopped assisting the government in deportation in June 2018 without any punishment, netizens, especially LGSM, believed that Ba intentionally participated in this work rather than being forced to travel under national laws.

- words "**want**", "**stop**" and "**deport**" refer to users want us to stop doing forced deportations.
- "**tube**" refers to activist group Lesbians and Gays Supports the Migrants (LGSM) claimed to have hacked hundreds of Tube ads on International Migrants Day.
- "**dearba**" refers to "#Dear BA", which is a slogan set by LGSM to gain wide support from netizens. They took advantage of BA 100th birthday and its advertising campaign to launch their petition: 100 letters to British Airways.
- words "**cruel**" and "**violent**" refer to some users think Forced Deportations are cruel and violent. Newsman Alwakeel (2019)'s report confirms this.

> "Unite's national industrial sector committee for civil air transport, says there are safety risks and grim psychological consequences for staff forced to take part in or witness violent restraints of deportees. One British Airways cabin crew member, who spoke to HuffPost UK on condition of anonymity, said forced deportations could be traumatic for those on board. 'It can be violent,' he said. '[Deportees] can be spitting, shouting, saying that they are being taken against their will, that they are going to be killed when they land. And some of that could be true.'"

II. PEAK Words Cloud

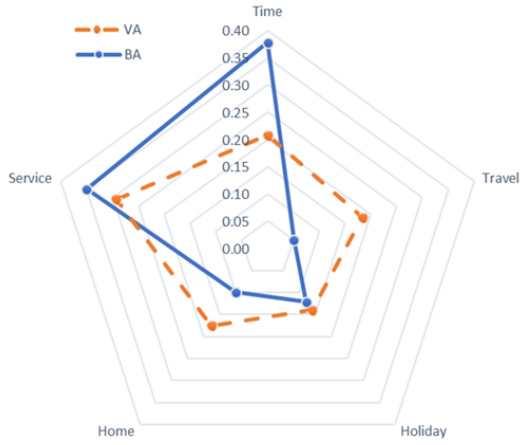
Compared with Trough words cloud, this one is much simpler to explain. Obviously, everyone is very happy in the new year, and users are satisfied with our new resolution in 2020. In addition, words "**crew**" refer to three British Airways cabin crew members were killed in a car crash on New Year's Eve, and users are concerned about this accident at Heathrow airport.

Sentiment Valence Analysis can conclude:

"Forced Deportation" caused a significant decrease of BA's sentiment score. Ignoring this decline, both airlines scored the same. ***Finding 1***

D. Delay and Services

The radar map in **Figure 7** shows the frequency of each characteristic type of words are mentioned by users, that is, the user's Consumption Pain Point. The higher the weight of type, the more obvious the pain point effect. The right table shows the main words and their frequency in each part of speech.



Type	Characteristic Words (VA)	Weight (VA)	Characteristic Words (BA)	Weight (BA)
Time	time 580, book 456,	0.21	book 1505, time 1366, hour 897, delay 760, wait 745, cancel 639	0.38
Travel	travel 390, trip 281, journey 247	0.18	travel 798	0.05
Holiday	newyear 352, christma 348,	0.14	newyear 775, christma 644, holiday 483	0.12
Home	home 339, famili 187, back 352	0.18	back 900, home 658	0.10
Service	safe 569, seat 338, servic 290, help 263	0.29	servic 1355, seat 1259, help 874, bag 847, staff 614, luggag 537	0.35

Figure 7 *Consumption Pain Point Radar Map*

As shown in ***Figure 7.***, compared with VA, BA's **Travel** index is low, so it is speculated that most of BA's core customers are business travel and most of VA's core customers are tourists. BA's indexes for **Time** and **Service** are significantly high. Compared with words in the table, the words about **Time** are mostly negative words, while that about **Service** are mostly positive words. Based on ***Finding 1*** of the Emotional Value Curve that the total emotional scores of BA and VA are the same after ignoring the impact of Forced Repatriation event. In case of flight delay, BA will provide satisfactory service. Specifically, BA will take operational measures to modify the flight schedule, so as to avoid the delay of arrival time. Also when BA gets the information of flight delay, it will inform users immediately. If one flight is delayed for more than 5 hours, the user will be able to choose other flights of BA and other airlines or get full refund and other compensation measures in Clause 251/2004 (British Airways, 2015). Therefore, it is concluded that

High-quality service offset flight delay on customer satisfaction. ***Finding 2***

E. Customer Satisfaction Survey VS Mining Twitter Data

Text mining technology has become increasing significant in modern business intelligence analysis, but can twitter data mining replace the traditional customer satisfaction survey? I summarize the advantages and disadvantages of this decision as shown in the following table by comparing this report (representative mining twitter data article) with Efthymiou et al.'s article (representative customer satisfaction survey article): The Impact of Delays on Customers' Satisfaction: an Empirical Analysis of the British Airways On-Time Performance at Heathrow Airport. This article was published in December 2019, and the data source of this report is also tweets data in December 2019. The two articles have made the same core findings for the same research object (BA) through two different methods (customer satisfaction survey and mining twitter data); therefore, there has a strong comparative value. Comparison of these two articles are shown in ***Table 2.***; meanwhile, the pros and cons of Mining Twitter Data Replace Customer Satisfaction Survey are in ***Table 3.***

Table 2 *Compare two articles*

	This Report	**Comparative Literature**
Research	The Impact of Delays on Customers' Satisfaction	The Impact of Delays on Customers' Satisfaction
Objectives	BA	BA(LHA)
Method	Mining Twitter Data	Customer Satisfaction Survey
Result	High quality service can largely offset the impact of flight delay on customer satisfaction	High quality service can largely offset the impact of flight delay on customer satisfaction
Data Source	twitter developer	interview of 160 BA passengers and 4 BA airline staff
By-product	Emotional Valence Graph	
Limitation	Technical Reasons	Sample Reasons
Applicability	Most Airlines	BA (LHA)
Measuring Users' Expectations	Words Cloud	Means of 5-point Likert Type

Table 3 *Mining Twitter Data Replace Customer Satisfaction Survey*

PROS	CONS
Once formed, text mining system can save huge time of market research.	The pre-process of text mining is troublesome and requires more professional knowledge.
Text mining system's result is not always correct and interpretable. Once the system gives an incorrect result, it is difficult to detect and repair the problem later.	Only mining twitter data has certain deviation. If we can integrate the whole network text data for mining results with higher reliability, we need to equip format converter to deal with the various formats of the target document.
Compared with the data from customer satisfaction survey, the tweets published by users are more realness.	With the same amount of data, text mining needs more computer configuration, which is a huge expense for enterprises.
The timeliness of text mining is stronger, while the research on customer satisfaction is relatively delayed.	Since 1960s, many scholars have done huge research on customer satisfaction model. However, text mining began in this century, and needs to be improved compared with mature customer satisfaction survey.
Text mining can expand the number of research objects and establish a more accurate and comprehensive index system.	The adaptability of text mining technology to cross language customer satisfaction survey is low.
	The rusults from text mining are mostly conjectures, which need further empirical research. Through customer satisfaction survey, we can draw a conclusion directly.

To sum up, the main advantages of the proposal are strong applicability of the system, strong learning ability and high accuracy of results; the main disadvantages are immature technology, excessive resource consumption and sample problems.

F. Limitation

- The Sentiment Dictionary in this paper is Hu Liu's general sentiment dictionary (Hu and Liu, 2004), rather than a professional Sentiment Dictionary for the aviation industry.
- Unsupervised learning classifier (e.g. Recurrent Neural Network and its special case Long Short-Term Memory) can achieve better classification results because of more than 8000 training samples.
- The Computation Speed will be quicker and result will be more accuracy if PCA or K-means are used before LSA.
- Lack of Region Analysis and Social Network Analysis.

(Word Count 2178)

REFERENCES

British Airways. (2015). General conditions of carriage for passengers and baggage. British Airways. Available from: http://www.britishairways.com/en-gb/information/legal/british-airways/general-conditions-of-carriage [Accessed 9th January 2020].

Cambria, E., Schuller, B., Xia, Y., & Havasi, C. (2013). New avenues in opinion mining and sentiment analysis. *IEEE Intelligent systems*, *28*(2), 15-21.

Efthymiou, M., Njoya, E. T., Lo, P. L., Papatheodorou, A., & Randall, D. (2019). The Impact of Delays on Customers' Satisfaction: an Empirical Analysis of the British Airways On-Time Performance at Heathrow Airport. *Journal of Aerospace Technology and Management*, *11*.

Hu, M. and Liu, B. (2004). Mining and summarizing customer reviews. Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining. Seattle, 168–177.

Pang, B., Lee, L., & Vaithyanathan, S. (2002, July). Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, 79-86.

Turney, P. D. (2002, July). Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th annual meeting on association for computational linguistics,* 417-424.

Vinodhini, G., & Chandrasekaran, R. M. (2017). A sampling based sentiment mining approach for e-commerce applications. *Information Processing & Management*, *53*(1), 223-236.

Wilson, T., Wiebe, J., & Hoffmann, P. (2005). Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*.

APPENDIX

I. Pre-process

```
setwd("D:/R/R-profile/Text-mining Ass")
#load the library

library(wordcloud)
library(mgsub)
library(stopwords)
library(ggplot2)
library(tm)
library(stringr)
library(qdap)
library(lubridate)
library(caret)

###Read dataset and combination
setwd("D:/R/R-profile/Text-mining Ass")
ba = list.files("tweets.ba")
dir = paste("./tweets.ba/",ba,sep="")
n = length(dir)
ba.data = readRDS(file = dir[1])
for (i in 2:n){
   new.data = readRDS(file = dir[i])
   ba.data = rbind(ba.data,new.data)
}

library(data.table)
ba.data <- data.table(ba.data)

#delete repeat tweets
index <- duplicated(ba.data$status_id)
ba.data <- ba.data[!index,]

# subset data in Dec
ba.data$created_at<-
as.POSIXct(ba.data$created_at,format="%Y%m%d %
H:%M")
ba.data12<-subset(ba.data,created_at>=as.POSIXct('2019-
12-01 00:00:00')
                  &created_at<=as.POSIXct('2019-12-
31 23:59:59'))

#Order according to time
ba.data12.o <- ba.data12[order(ba.data12$created_at),]

#check data to see if there are missing values
length(which(!complete.cases(ba.data12.o$text)))

#have a look of top 10
head(ba.data12.o,n=10)
#tailor-made a few things

#A function changes all to lower case (and return NA stead
of error if it is a special character)
#Return NA instead of tolower error
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)
}

# create a pre-processing function using gsub
custom.gusb <- c('flight','can','get','fli','just',
                 'one','now','pleas','amp','take',
                 'plea','thank','hi','dont','tri',
                 'use','due','ba')
clean.gsub<-function(corpus){
   corpus$text <-gsub("@\\S*", "", corpus$text)
   corpus$text <-gsub("https\\S*", "", corpus$text)
   corpus$text <- gsub("[’]" , "",corpus$text)
   corpus$text <- gsub("[,]" , "",corpus$text)
   corpus$text <- gsub("[[:punct:]]" , "",corpus$text)
   return(corpus)
}
ba.data12.o <- clean.gsub(ba.data12.o)

#create my stop words list
custom.stopwords<-
c(stopwords('english'),'flight','can','get','fli','just',

'one','now','pleas','amp','take',

'plea','thank','hi','dont','tri',

'use','due','ba','paper 玫 <u+3e64> 愼 <u+3e30> 卵
<u+3e64>dnt')

#create a pre-processing function using tm functions and the
```

```
above two
clean.corpus<-function(corpus){
   corpus<-
tm_map(corpus,content_transformer(tryTolower))
   corpus<-
tm_map(corpus,removeWords,custom.stopwords)
   corpus<-tm_map(corpus,removePunctuation)
   corpus<-tm_map(corpus,stripWhitespace)
   corpus<-tm_map(corpus,removeNumbers)
   corpus<-tm_map(corpus,stemDocument,  language  =
"english")
   return(corpus)
}

#define the tweets object
#when measuring va using the following codes
# the.corpus <- VCorpus(VectorSource(va.data12.o$text))
the.corpus <- Corpus(VectorSource(ba.data12.o$text))

#clean the tweets with the function created earlier
the.corpus<-clean.gsub(the.corpus)
the.corpus<-clean.corpus(the.corpus)

the.corpus<-tm_map(the.corpus,removePunctuation)
the.corpus<-
tm_map(the.corpus,removeWords,c('flight','can','get','fli','ju
st',
'one', 'now', 'pleas', 'amp', 'take',
'plea', 'thank', 'hi', 'dont', 'tri',
'use','due','ba','paper  玫<u+3e64>愼<u+3e30>卯
<u+3e64>dnt'))

the.corpus <- Corpus(VectorSource(the.corpus))
head(the.corpus)
#Create the term document matrix
tdm                                                  <-
DocumentTermMatrix(the.corpus,control=list(weighting=
weightTf))

#remove sparse terms from a doucment if the sparsity is
more than 99%
tdm.n<-removeSparseTerms(tdm, 0.99)

#redefine it as matrix for easy to computation
tdm.tweets<-as.matrix(tdm.n)

#save the pre-processed document term matrix
saveRDS(tdm.tweets, file="matrix.tweets")

#check dimension of the tweets
dim(tdm.tweets)

#check term frequency
term.freq<-colSums(tdm.tweets)

#create a dataframe with the term and then the frequency as
the second column
freq.df<-
data.frame(word=names(term.freq),frequency=term.freq)
freq.df<-freq.df[order(freq.df[,2],decreasing=T),]
freq.df[1:20,]
```

II. positive_negative words cloud

```
setwd("D:/R/R-profile/text-mining/social network analysis")
#use the polarity function in the package qdap
#please install and load the package

library(sentimentr)
library(syuzhet)
library(lubridate)
library(reshape2)
library(dplyr)
library(zoo)
library(qdap)
library(ggplot2)
library(ggthemes)
library(wordcloud)

#---wordcloud(n-p)---#

#the key.pol is a dataset in qdap keeping some polarity
words
#here we are going to extract this list but also add some new
words
pos.new<-scan("p-emojis.txt",what                    =
"character",comment.char = "")
pos.old<-subset(as.data.frame(key.pol),key.pol$y==1)
pos.words<-c(pos.new,pos.old[,1])
neg.new<-scan("n-emojis.txt",what                    =
"character",comment.char = "")
neg.old<-subset(as.data.frame(key.pol),key.pol$y==-1)
neg.words<-c(neg.new,neg.old[,1])
all.polarity<-sentiment_frame(pos.words,neg.words,1,-1)
```

```
#check the polarity using a sentence
#the function calculate using the following function
#sum(positive+negative+amplifier)/sqrt(number of words)
polarity('it is good',polarity.frame=all.polarity)
polarity('It is very good',polarity.frame=all.polarity)
polarity('it is bad',polarity.frame=all.polarity)
polarity('It is very bad',polarity.frame=all.polarity)

#read a file containing 1,000 airbnb reviews about stays in Boston
options(stringsAsFactors=F)

bos.pol<-polarity(ba.data12.o$text)

#plot the histogram of the polarity
ggplot(bos.pol$all,aes(x=polarity,y=..density..))+theme_gdocs()+
  geom_histogram(binwidth = .25,
                 fill="darkred",        colour="grey60", size=.2) +
  geom_density(size=.75)

#save the polarity back to the orginial dataset
ba.data12.o$screen_name<-scale(bos.pol$all$polarity)
#plotting wordclouds (one for positive and one for negative)

pos.comments<-
subset(ba.data12.o$text,ba.data12.o$screen_name>0)
neg.comments<-
subset(ba.data12.o$text,ba.data12.o$screen_name<0)

#compress it into a document with only two components
#(one for positive and one for negative)
pos.terms<-paste(pos.comments, collapse=" ")
neg.terms<-paste(neg.comments, collapse=" ")
all.terms<-c(pos.terms,neg.terms)
all.corpus<-VCorpus(VectorSource(all.terms))

#create the term by document matrix using tfidf
all.tdm<-TermDocumentMatrix(all.corpus,

control=list(weighting=weightTf,

removePunctuation=TRUE,

stopwords=stopwords(kind='en')))

#switch to matrix and add column names
all.tdm.m<-as.matrix(all.tdm)
colnames(all.tdm.m)<-c('positive','negative')

#build wordclouds
comparison.cloud(all.tdm.m,max.words=100)
colors=c('darkgreen','darkpurple')
```

III. Emotional Valence

```
#---Emotional Valence---#

#read files named tweet.ba.12 with rename tweet.b.12
ba.text <-c(ba.data12.o$text)

#obtain sentiment scores
ba<-get_nrc_sentiment(ba.text)
head(ba)

#Classify according to sentences
sentiment(ba.text)

#Seperate them
sentiment_by(ba.text)

#Find N/P words in every sentence
extract_sentiment_terms(ba.text) #找出每段话中的好词和坏词

#Plot the figure
plot(sentiment(ba.text))
```

IV. LSA

```
###------LSA------###

tfData <- read.csv("lala.csv")

install.packages("topicmodels")
install.packages("lsa")
install.packages("svs")

library("topicmodels")
library("lsa")
library("svs")

SEED <- 2100

calculateLDA <- function(tfData, topic_number,
```

```
estimateAlpha=TRUE){

   start.time <- Sys.time()

   topicmodel <- LDA(tfData$cleanedTrainMatrix, k=topic_number, control=list(seed=SEED, estimate.alpha=estimateAlpha))
   trainData <- posterior(topicmodel)[2]$topics
   testData <- posterior(topicmodel, tfData$cleanedTestMatrix)[2]$topics

   end.time <- Sys.time()
   time.taken <- end.time - start.time
   duration <- time.taken
   print(paste("Calculated LDA for ",topic_number," topics, duration=",duration," seconds"))
   list(topicmodel=topicmodel, ldaTrainData=trainData, ldaTestData=testData, duration=duration)
}

calculateLSA <- function(tfidfData, topic_number){

   start.time <- Sys.time()

   lsa_train_tdidf <- createLSATrain(tfidfData$cleanedTrainMatrix, dims=topic_number)
   lsa.training.set.tfidf = lsa_train_tdidf$matrix

   lsa_test_tdidf <- createLSATest(tfidfData$cleanedTestMatrix, lsa_train_tdidf$lsa)
   lsa.testing.set.tfidf = lsa_test_tdidf$matrix

   end.time <- Sys.time()
   time.taken <- end.time - start.time
   duration <- time.taken

   list(lsaTrainData=lsa.training.set.tfidf, lsaTestData=lsa.testing.set.tfidf, duration=duration)
}

createLSATrain <- function(rawTrainMatrix, dims=dimcalc_share(share=0.8)){
   start.time <- Sys.time()
   m=as.matrix(rawTrainMatrix)
   dim(m)
   lsa_m=lsa(t(m),dims)
   dim(lsa_m$tk)
   dim(lsa_m$dk)
   length(lsa_m$sk)

   lsa.training.set = t(diag(lsa_m$sk) %*% t(lsa_m$dk))
   dim(lsa.training.set)

   end.time <- Sys.time()
   time.taken <- end.time - start.time
   duration <- time.taken

   list(matrix=lsa.training.set, lsa=lsa_m, duration=duration)
}

createLSATest <- function(rawTestMatrix, lsa_m){

   start.time <- Sys.time()
   m_test=as.matrix(rawTestMatrix)
   dim(t(m_test))

   lsa.testing.set <- t(diag(1,dim(lsa_m$tk)[2]) %*% t(lsa_m$tk) %*% t(m_test))

   end.time <- Sys.time()
   time.taken <- end.time - start.time
   duration <- time.taken

   list(matrix=lsa.testing.set, duration=duration)
}

createPLSA <- function(tfData, topic_number){
   fast_plsa(tfData, topic_number)
}
term.freq1 <- term.freq[1:60,]
ba.lsa <- createPLSA(as.matrix(term.freq1),20)
```

V. SVM

```
setwd("D:/R/R-profile/Text-mining Ass")
pdata <- read.csv("la.csv", header = T)

library(caret)
library(e1071)
library(Matrix)
library(SparseM)
library(tm)
library(SnowballC)
library(naivebayes)
```

```
###---Seperate&Train
#split into testing and training
set.seed(123)
split<-createDataPartition(pdata$V2, p=0.7, list=FALSE)
traindata<-pdata[split,]
testdata<-pdata[-split,]

#function to deal with unmatched terms
match.matrix<-
function(text.col,original.matrix=NULL,weighting=weightTf)
{
  control<-list(weighting=weighting)
  training.col<-
    sapply(as.vector(text.col,mode="character"),iconv,
           to="UTF8",sub="byte")
  corpus<-VCorpus(VectorSource(training.col))
  matrix<-DocumentTermMatrix(corpus,control=control);
  if( !is.null(original.matrix)){
    terms<-
      colnames(original.matrix[,

which(!colnames(original.matrix) %in%
colnames(matrix))])
    weight<-0
    if(attr(original.matrix,"weighting")[2]=="tfidf")
      weight <-0.000000001
    amat<-matrix(weight,nrow=nrow(matrix),
                 ncol=length(terms))
    colnames(amat)<-terms
    rownames(amat)<-rownames(matrix)
    fixed<-as.DocumentTermMatrix(
      cbind(matrix[,which(colnames(matrix) %in%

colnames(original.matrix))],amat),
      weighting=weighting)
    matrix<-fixed
  }
  matrix<-matrix[,sort(colnames(matrix))]
  gc()
  return(matrix)
}

#function to clean the data
#create a pre-processing function using tm functions and the
above two
clean.d<-function(x){
  x<-tolower(x)
  x<-removeWords(x,stopwords('en'))
  x<-removePunctuation(x)
  x<-stripWhitespace(x)
  return(x)
}

#clean the training data and build a term by document matrix
traindata$V1<-clean.d(traindata$V1)
testdata$V1<-clean.d(testdata$V1)

###SVM training & predicting
#clean the training data and build a term by document matrix
clean.train<-clean.d(traindata$V1)
train.dtm<-
match.matrix(clean.train,weighting=tm::weightTfIdf)

train.matrix<-as.matrix(train.dtm)
train.matrix<-Matrix(train.matrix,sparse=T)

SVM.model <- svm(x=train.matrix,
y=as.factor(traindata$V2),
                 kernel="linear")

#clean the testing data and build a term by document matrix
clean.test<-clean.d(testdata$V1)
test.dtm<-
match.matrix(clean.test,weighting=tm::weightTfIdf,
                          original.matrix=train.dtm )

test.matrix<-as.matrix(test.dtm)
test.matrix<-Matrix(test.matrix,sparse=T)

#apply the model into the testing set and see the results
preds<-predict(SVM.model,as.matrix(test.matrix))
table("SVM-Prediction" = preds, "Actual" = testdata$V2)

### ROC CURVE for SVM
install.packages("pROC")
library(pROC)
modelroc <- roc(as.numeric(testdata$V2),as.numeric(preds))
plot(modelroc, print.auc=TRUE, auc.polygon=TRUE,
grid=c(0.1, 0.2),
     grid.col=c("green", "red"), max.auc.polygon=TRUE,
     auc.polygon.col="skyblue", print.thres=TRUE)

### SVM-predicting ba
```

```
set.seed(123)
clean.ba <- clean.d(ba.data12.o$text)
ba.dtm<-match.matrix(clean.ba,weighting=tm::weightTfIdf,
                                original.matrix=train.dtm )

ba.matrix<-as.matrix(ba.dtm)
ba.matrix<-Matrix(ba.matrix,sparse=T)

preds.ba<-predict(SVM.model,as.matrix(ba.matrix))
table(preds.ba)
```

VI. Improved Naive Bayes

```
library(e1071)
library(dplyr)
library(tm)
library(naivebayes)
library(caret)
library(MASS)
library(caTools)

#create the vector format
trainvector <- as.vector(traindata$V1)
testvector <- as.vector(testdata$V1)

#create corpus for data
traincorpus <- VCorpus(VectorSource(trainvector))
testcorpus <- VCorpus(VectorSource(testvector))

#create the term-by-document matrix
trainmatrix <- DocumentTermMatrix(traincorpus)
testmatrix <- DocumentTermMatrix(testcorpus)

# feature selection
dim(trainmatrix)

# restrict the DTM to use only the frequent words using the
'dictionary' option
fivefreq <- findFreqTerms(trainmatrix, 5)
length((fivefreq))

# Use only 5 most frequent words (fivefreq) to build the
DTM
trainmatrix.nb <- DocumentTermMatrix(traincorpus,
control=list(dictionary = fivefreq))
dim(trainmatrix.nb)

testmatrix.nb <- DocumentTermMatrix(testcorpus,
control=list(dictionary = fivefreq))
dim(testmatrix.nb)


# Function to convert the word frequencies to yes (presence)
and no (absence) labels
convert_count <- function(x) {
  y <- ifelse(x > 0, 1,0)
  y <- factor(y, levels=c(0,1), labels=c("No", "Yes"))
  y
}

# Apply the convert_count function to get final training and
testing DTMs
trainNB <- apply(trainmatrix.nb, 2, convert_count)
testNB <- apply(testmatrix.nb, 2, convert_count)

# Train the classifier
system.time( classifier <- naiveBayes(as.matrix(trainNB),
as.factor(traindata$V2)) )

# Use the NB classifier we built to make predictions on the
test set.
system.time( pred <- predict(classifier, testNB) )

# Create a truth table by tabulating the predicted class labels
with the actual class labels
table("Naive Bayes-Predictions"= pred, "Actual" =
testdata$V2 )

### ROC CURVE for Bayes

install.packages("pROC")
library(pROC)
modelroc <- roc(as.numeric(testdata$V2),as.numeric(pred))
plot(modelroc, print.auc=TRUE, auc.polygon=TRUE,
grid=c(0.1, 0.2),
     grid.col=c("green", "red"), max.auc.polygon=TRUE,
     auc.polygon.col="skyblue", print.thres=TRUE)

### test in ba
dtm.ba.nb <- DocumentTermMatrix(ba.bayes.corpus,
control=list(dictionary = fivefreq))
ba.NB <- apply(dtm.ba.nb, 2, convert_count)
system.time( ba.pred <- predict(classifier, ba.NB) )
table("Predictions"= ba.pred)
```