Xinyu Huang

December 19, 2019

Final Project Report

Overview

I want to find out tweets that are not negative from all the tweets sent by customers to airlines. I have access to two datasets. One contains all negative tweets, and another contains all non-negative tweets. I will use these two datasets to develop a classification model. This model can be used to classify whether a tweet sent by a customer to airline is negative (overall express a negative attitude such as anger, sadness and frustration) or not.

In the model building process, I first merge two datasets and build a document term matrix. Then I try different models and compare their accuracy to find the best model. I try Naive Bayesian, Support Vector Machine, and Random Forest. Finally, I find that Naive Bayesian model works best. After choosing the best model, I apply the model to the dataset with a specific tag. Each row of the dataset contains a tweet. The model could classify those tweets into negative or non-negative groups. I export all the non-negative tweets my model identified and evaluate those tweets manually on my own to get the precision of my classification.

Method

```
csvdata1 <- read.csv("/Users/xinyuhuang/Desktop/CIS434/Final Project/complaint1700.csv",
                     header=TRUE, sep=',')
csvdata2 <- read.csv("/Users/xinyuhuang/Desktop/CIS434/Final Project/noncomplaint1700.csv",
                     header=TRUE, sep=',')
csvdata1['complaint'] = 1
csvdata2['complaint'] = -1
csvdata = rbind(csvdata1,csvdata2)
y <- csvdata$complaint
```

First, I read in the data from two csv files to a data frame. I add one column ‘complaint’ to each data frame to represent their classification. If the tweet is negative, I assign the value to be 1,

else if the tweet is non-negative, I assign the value to be -1. Then I combine two data frames together.

```
docs <- Corpus(VectorSource(csvdata$tweet))
dtm.control = list(tolower=T, removePunctuation=T, removeNumbers=T,
                   stripWhitespace=T, stemming=T)
dtm.full <- DocumentTermMatrix(docs, control=dtm.control)
```

Second, I set up a corpus from the combined data frame to count the frequency of terms occur in each tweet. The rows represent the tweets sent to airlines and the columns represent terms contained in the tweets.

```
dtm <- removeSparseTerms(dtm.full,0.99)
X <- as.matrix(dtm)
Y <- as.factor(y)
```

I remove the sparse terms in the original document term matrix to build a new document term matrix. I transfer the document term matrix into a data frame and transfer the classification into factor for future use.

```
set.seed(1) # fixing the seed value for the random selection guarantees the same results in repeated runs
n=length(y)
n1=round(n*0.8)
n2=n-n1
train=sample(1:n,n1)
```

After that, I randomly select 80% of the data to be training data and the rest 20% to be validation data which will be used in the modeling process later.

```
Evaluation <- function(pred, true, class)
{

  tp <- sum( pred==class & true==class)
  fp <- sum( pred==class & true!=class)
  tn <- sum( pred!=class & true!=class)
  fn <- sum( pred!=class & true==class)
  precision <- tp/(tp+fp)
  recall <- tp/(tp+fn)
  F1 <- 2/(1/precision + 1/recall)
  F1
}
```

Before trying different models, I write a function to evaluate the performance of the model. I use F1 score to evaluate the accuracy of the model. F1 score has best value of 1 which means perfect precision and recall and has worst value of 0.

```
##########################################
#######   Support Vector Machine   ########
##########################################

svm.model1 <- svm(Y[train] ~ ., data = X[train,], kernel='linear', type = 'C', degree = 3)
pred1 <- predict(svm.model1, X[-train,])
table(pred1, Y[-train])
Evaluation(pred1, Y[-train], 1)
Evaluation(pred1, Y[-train], -1)

##########################################
#######   Support Vector Machine 2  ########
##########################################
svm.model2 <- svm(Y[train] ~ ., data = X[train,], kernel='linear', type = 'nu-classification', degree = 3)
pred2 <- predict(svm.model2, X[-train,])
table(pred2, Y[-train])
Evaluation(pred2, Y[-train], 1)
Evaluation(pred2, Y[-train], -1)

##########################################
#######   Support Vector Machine 3  ########
##########################################
svm.model3 <- svm(Y[train] ~ ., data = X[train,], kernel='polynomial', type = 'nu-classification', degree = 3)
pred3 <- predict(svm.model3, X[-train,])
table(pred3, Y[-train])
Evaluation(pred3, Y[-train], 1)
Evaluation(pred3, Y[-train], -1)
```

The next step is to try different model to find the model with highest F1 score.

I try tune different parameters and try many SVM models. I got the best SVM result from the second model:

```
pred2  -1   1
   -1 257  94
   1   94 235
> Evaluation(pred2, Y[-train], 1)
[1] 0.7142857
> Evaluation(pred2, Y[-train], -1)
[1] 0.7321937
```

```
##########################################
##########   Naive Bayesion   #############
##########################################

nb.model <- naiveBayes(X[train,], Y[train], nfold = 10, laplace = 0)
pred <- predict(nb.model, X[-train,])
table(pred, Y[-train])
Evaluation(pred, Y[-train], 1)
Evaluation(pred, Y[-train], -1)
```

The next model I try is naïve bayesion. I get a result of:

```
pred  -1   1
  -1 256  85
  1   95 244
> Evaluation(pred, Y[-train], 1)
[1] 0.7305389
> Evaluation(pred, Y[-train], -1)
[1] 0.7398844
```

```
##########################################
#######   Random Forest  ########
##########################################
rf.model <- randomForest(X[train,], Y[train], ntree=800, sampsize=200, threshold=0.7)
predrf <- predict(rf.model, X[-train,])
table(predrf, Y[-train])
Evaluation(predrf, Y[-train], 1)
Evaluation(predrf, Y[-train], -1)
```

Finally I try random forest. I also tune many different parameters, the best result I get is:

```
predrf  -1   1
    -1 251  95
    1  100 234
> Evaluation(predrf, Y[-train], 1)
[1] 0.7058824
> Evaluation(predrf, Y[-train], -1)
[1] 0.7202296
```

Overall, the best model with highest F1 score is naïve bayesion. So next I am going to use naïve bayesion to classify the tweets from dataset with my specific tag.

```
library(DBI)
library(RMySQL)

driver <- dbDriver("MySQL")
myhost <- "localhost"
mydb   <- "studb"
myacct <- "cis434"
mypwd  <- "LLhtFPbdwiJans8F@S207"

conn <- dbConnect(driver, host=myhost, dbname=mydb, myacct, mypwd)

# hE'p0o41<Kur
temp <- dbGetQuery(conn, "SELECT * FROM proj4final WHERE tag=\"hE'p0o41<Kur\"")

dbDisconnect(conn)
```

I first connenct to the databease to retrive the tweets with my own tag. There are 4555 tweets in the data frame.

```
temp_docs <- Corpus(VectorSource(temp$tweet))

dtm.control = list(tolower=T, removePunctuation=T, removeNumbers=T,
                   stripWhitespace=T, stemming=T)
temp_dtm.full <- DocumentTermMatrix(temp_docs, control=dtm.control)
temp_dtm <- removeSparseTerms(temp_dtm.full,0.99)
temp_X <- as.matrix(temp_dtm)
temp_pred = predict(nb.model, temp_X)
output = temp[temp_pred==-1,c(1,5)]
write.csv(output,"output_final.csv", row.names = FALSE)
```

I set up a corpus to count the frequency of terms occur in each tweet and remove sparse terms. Then I transfer the document term matrix into a data frame and transfer the classification into factor. Finally I apply the model I choose to classify the tweets. I need a csv file of non-negative tweets consist of two columns. The first column contains the id from original table and the second column conyains the tweet content. I select them and export them as a CSV file. I get a file with 1787 rows which means there are 1787 tweets out of 4555 tweets are classified as non-negative.

As the last step to evaluate the precision of my model's classification, I insert a column into the CSV file and evaluate those tweets manually to see if they are really non-negative. I give a value 1 if the classification is correct and 0 if the classification is wrong.

```
#calculate precision
evaluation_result <- read.csv("/Users/xinyuhuang/Desktop/CIS434/Final Project/Xinyu_Huang.csv",
                   header=TRUE, sep=',')
sum(evaluation_result$evaluation)/nrow(evaluation_result)
```

I read in the csv file with manual classification and calculate the precision of my classification by calculating the sum of the second column devided by the total number of rows in my csv file.

```
[1] 0.5599104
```

I got a precision of nearly 0.56.

Conclusion

I build a model to classify whether a tweet sent by a customer to airline is negative (overall express a negative attitude such as anger, sadness and frustration) or not.

In the model building process, I first merge two datasets and build a document term matrix. Then after trying different models and compare their accuracy, I find that Naive Bayesian model works best. I apply this model to the dataset with a specific tag. The dataset is classified into negative or non-negative groups. I export all the non-negative tweets my model identified and evaluate those tweets manually on my own to get the precision of my classification. I finally got a precision of nearly 0.56.

Reflection

This result is not really good. I think the reasons are: 1) training data set are not big enough and not accurate enough; 2) I could have tried more models and better tuning them if there are more time; 3) I could have found better way to optimize the document term matrix with different weight among different terms to help the model to perform better.