Machine Learning

Sine Gov

January 14, 2018

Sentiment Predictor for Overwatch

To help alleviate the need to manually review every negative/toxic language customer service ticket, several prototype models that detects sentiment automatically has been created to determine which model will likely provide the best accuracy. The data set used to create the models was scrapped from reddit and the Overwatch forums. Each comment was then analyzed for negativity using two lexicons: afinn and bing and assigned a score accordingly. This score determined whether a comment was negative or positive and those results were recorded.

Type of Problem

Using the supervised method of classification and regression trees (CART), predictive models were created. Eighty percent of the data set was used for training the model and the remaining 20% of the data set was used to validate the model. The predictors used for these models are the comments themselves and the score for each of these comments determined via the seniment analyzers.

CART Models

The CART method was used to create this algorithm and the model was iterated on three times. The first model was created using the rpart's CART method with default values. The second model was created with cross validation of the CART method. The third model was created using the randomForest method of CART.

Evaluation

In the case of detecting toxic language, we want to make sure we are accurately capturing this behavior. The models were evaluated by computing the accuracy of the model using a confusion table. The higher our accuracy at identifing the toxic language, the better since we could get potentially remove these types of reports so our customer service reps can deal with higher priority/value tickets such as helping users gain access to hacked accounts or billing issues.

Results

Below the accuracy for each model by lexicon is shown.

First Model (Default values with no validation)

Below we see that the afinn method for detecting negative language is more accurate for model 1. Accuracy is not as high as we would like, but this provides us a start.

Model 1 Accuracy	bing	afinn
Forums	0.6877	**0.7084**
Reddit	0.6562	**0.6988**
Forums & Reddit	0.6688	**0.722**

Second Model (Cross Validation for each with best fit cp value)

With Cross Validation we are able to improve the accuracy of the models. For the most part the afinn method is better at obtaining accuracy except for reddit data which outperformed the afinn model by 0.004. This discrepency could be due to the random sampling that made the bing model better at detecting negative sentiment.

Model 2 Accuracy	bing	afinn
Forums	0.711	**0.7414**
Reddit	**0.7094**	0.705
Forums & Reddit	0.7058	**0.7525**

Third Model (random forest method)

The random forest method provided the best accuracy out of all three models. In this model, the afinn lexicon outperformed the bing lexicon.

Model 3 Accuracy	bing	afinn
Forums	0.804	**0.8072**
Reddit	0.7844	**0.8075**
Forums & Reddit	0.7926	**0.833**

Conclusion

From these models, it is likely that the afinn lexicon combined with the randomforest method for CART may be the best tools to help us detect negative sentiment. Keep in mind that these models only used data scrapped from community sites and they were scored based on a lexicon that may have incorrectly identified certain keywords (kill, mercy, etc) that may be neutral in the discussion of Overwatch. Using customer service reports validated by our customer service staff, we could get a model that would contain true accuracy and could be employed to save our staff time and ultimately money if the accuracy was high enough. As of now, this model provides a proof of concept for consideration of the creation of a similiar model.

Appendix

Full code for modeling

```
library(dplyr)
library(tidytext)
library(tm)
library(caTools)
library(rpart.plot)
library(e1071)
library(caret)
library(randomForest)

#-------------------------Reading Files Into R-------------------------#
#We are only interested in using reddit and forums data. Twitter was found to contain
#irrelevant data.

#read files into R. Ensure they are not factors
forums <- read.csv("OWFORUMS12_22_FINAL.csv", stringsAsFactors = FALSE)
reddit <- read.csv("REDDIT_12_22_FINAL.csv", stringsAsFactors = FALSE)
forandred <- read.csv("OWFORUMS&REDDIT_12_22_FINAL.csv", stringsAsFactors = FALSE)

#-------------------------Tokenize Data-------------------------#
#duplicating the text column so we have a copy of the text.
#This gets destroyed during tokenization, but we need it to group by.
forums$text_topic <- forums$text
reddit$text_topic <- reddit$X.text.
forandred$text_topic <- forandred$text

#tokenize the text column and group by text of each comment (text_topic)
tidy_forums <- forums %>%
  group_by(text_topic) %>%
  mutate(linenumber = row_number()) %>%
  unnest_tokens(word, text) %>%
  ungroup()

tidy_reddit <- reddit %>%
  group_by(text_topic) %>%
  mutate(linenumber = row_number()) %>%
  unnest_tokens(word, X.text.) %>%
  ungroup()

tidy_forandred <- forandred %>%
  group_by(text_topic) %>%
  mutate(linenumber = row_number()) %>%
  unnest_tokens(word, text) %>%
  ungroup()

#-------------------------Bing Sentiment Function-------------------------#

#bing maker is a function to get bing sentiment from tidy data
bing_maker <- function(tidy_data) {
  bing_data <- tidy_data %>%
```

```
    inner_join(get_sentiments("bing"))

  #create new columns to use in for loop calculations.
  #we will calculate the sentiment score by adding
  #all the sentiments and dividing by number of sentiments.
  bing_data$score <- 0
  bing_data$count <- 1

  #for loops codes the sentiment from bing sentiment into a -1 or 1.
  #1 is positive and -1 is negative.
  #Also counts the number of sentiment words to be used later.
  for(i in 1:length(bing_data$sentiment)) {
    if(bing_data$sentiment[i] == "negative") {
      bing_data$score[i] = -1
    } else {
      bing_data$score[i] = 1
    }
  }

  #calculates the sentiment score by adding up all sentiment values
  #and dividing it by the total for a single post.
  #text_topic is used since it identifies if a word belongs to a single comment.
  bing_data <- bing_data %>%
    group_by(text_topic) %>%
    mutate(sentimentscore = sum(score)/sum(count))

  #identify the negative values. used later to predict negative sentiment
  bing_data$negative <- as.factor(bing_data$sentimentscore < 0)

  #collapse the duplicated columns by text_topic
  bing_data <- bing_data[!duplicated(bing_data$text_topic),]

  bing_data <- data.frame(bing_data$text_topic, bing_data$sentimentscore, bing_data$negative)
  names(bing_data) <- c("forum_text","sentiment_score","negative")
  return(bing_data)
}

#--------------------Perform bing--------------------#
forum_bing <- bing_maker(tidy_forums)
reddit_bing <- bing_maker(tidy_reddit)
forandred_bing <- bing_maker(tidy_forandred)

#-------------------------Afinn Sentiment Function-------------------------#
#afinn maker is a function to get afinn sentiment from tidy data
afinn_maker <- function(tidy_data) {
  afinn_data <- tidy_data %>%
    inner_join(get_sentiments("afinn"))

  #create this column to help calculate sentiment score
  afinn_data$count <- 1

  #calculate sentiment score using afinn
  afinn_data <- afinn_data %>%
```

```
    group_by(text_topic) %>%
    mutate(sentimentscore = sum(score)/sum(count))

  #collapse the duplicated columns by text_topic
  afinn_data <- afinn_data[!duplicated(afinn_data$text_topic),]

  afinn_data <- data.frame(afinn_data$text_topic, afinn_data$sentimentscore)
  names(afinn_data) <- c("forum_text","sentiment_score")

  afinn_data$negative <- 0

  #for loops codes the sentiment from bing sentiment into a -1 or 1.
  #1 is positive and -1 is negative.
  #Also counts the number of sentiment words to be used later.
  for(i in 1:length(afinn_data$sentiment_score)) {
    if(afinn_data$sentiment_score[i] < 0) {
      afinn_data$negative[i] = "TRUE"
    } else {
      afinn_data$negative[i] = "FALSE"
    }
  }

  return(afinn_data)
}

#--------------------Perform afinn--------------------#
forum_afinn <- afinn_maker(tidy_forums)
forum_afinn$negative <- as.factor(forum_afinn$negative)
reddit_afinn <- afinn_maker(tidy_reddit)
reddit_afinn$negative <- as.factor(reddit_afinn$negative)
forandred_afinn <- afinn_maker(tidy_forandred)
forandred_afinn$negative <- as.factor(forandred_afinn$negative)

#----------------PREP DATA FOR MACHINE LEARNING----------------#

#----------------Corpus Maker----------------#
#Function to reate a corpus and remove unnecessary words/text
#sentiment_data should be the dataset created from bing or afinn maker above
corpus_maker <- function(sentiment_data) {
  data_corpus <- Corpus(VectorSource(sentiment_data$forum_text))
  data_corpus <- tm_map(data_corpus, removePunctuation)
  data_corpus <- tm_map(data_corpus, tolower)
  data_corpus <- tm_map(data_corpus, removeWords, stopwords("english"))
  data_corpus <- tm_map(data_corpus, stemDocument)
  return(data_corpus)
}

#----------------Make Corpus----------------#
bforum_corpus <- corpus_maker(forum_bing)
breddit_corpus <- corpus_maker(reddit_bing)
bforandred_corpus <- corpus_maker(forandred_bing)

aforum_corpus <- corpus_maker(forum_afinn)
```

```r
areddit_corpus <- corpus_maker(reddit_afinn)
aforandred_corpus <- corpus_maker(forandred_afinn)


#----------------Process Corpus Function----------------#
#Function to process the corpus

process_corpus <- function(corpus_name, bing_name) {
  #Find frequencies of words
  freq <- DocumentTermMatrix(corpus_name)

  #removing the sparse terms
  sparse <- removeSparseTerms(freq, 0.995)

  #convert sparse data into a data frame
  sparse <- as.data.frame(as.matrix(sparse))

  #converts variable names to appropriate names (some may start with numbers)
  colnames(sparse) = make.names(colnames(sparse))

  #add dependent variable to the dataframe. We will be predicting this value.
  sparse$negative <- bing_name$negative
  return(sparse)
}

#----------------Processing Corpus----------------#
b_forum_predict <- process_corpus(bforum_corpus, forum_bing)
b_reddit_predict <- process_corpus(breddit_corpus, reddit_bing)
b_forandred_predict <- process_corpus(bforandred_corpus, forandred_bing)

a_forum_predict <- process_corpus(aforum_corpus, forum_afinn)
a_reddit_predict <- process_corpus(areddit_corpus, reddit_afinn)
a_forandred_predict <- process_corpus(aforandred_corpus, forandred_afinn)

#----------------Splitting into Test and Train Sets----------------#
#setting a seed to get consistent results when running multiple times
set.seed(1113)

#split the data into a training set and a test set.

#set the SplitRatio into variable so we can modify easily for all data sets.
#This ratio will go into the test set (20% goes into test set)
sr <- 0.2


#Bing Forums
bforum_split <- sample.split(b_forum_predict$negative, SplitRatio = sr)
bforum_test <- subset(b_forum_predict, bforum_split == TRUE)
bforum_train <- subset(b_forum_predict, bforum_split == FALSE)

#Bing Reddit
breddit_split <- sample.split(b_reddit_predict$negative, SplitRatio = sr)
breddit_test <- subset(b_reddit_predict, breddit_split == TRUE)
```

```
breddit_train <- subset(b_reddit_predict, breddit_split == FALSE)

#Bing both
bforandred_split <- sample.split(b_forandred_predict$negative, SplitRatio = sr)
bforandred_test <- subset(b_forandred_predict, bforandred_split == TRUE)
bforandred_train <- subset(b_forandred_predict, bforandred_split == FALSE)

#Afinn Forums
aforum_split <- sample.split(a_forum_predict$negative, SplitRatio = sr)
aforum_test <- subset(a_forum_predict, aforum_split == TRUE)
aforum_train <- subset(a_forum_predict, aforum_split == FALSE)

#Afinn Reddit
areddit_split <- sample.split(a_reddit_predict$negative, SplitRatio = sr)
areddit_test <- subset(a_reddit_predict, areddit_split == TRUE)
areddit_train <- subset(a_reddit_predict, areddit_split == FALSE)

#Afinn Both
aforandred_split <- sample.split(a_forandred_predict$negative, SplitRatio = sr)
aforandred_test <- subset(a_forandred_predict, aforandred_split == TRUE)
aforandred_train <- subset(a_forandred_predict, aforandred_split == FALSE)

#----------------Model Functions----------------#
#First model created. Automatic. No validation used. Default values used.
CARTMODELMAKER <- function(traindata) {
  cartmodel1 <- rpart(negative~., data = traindata, method = "class")
  return(cartmodel1)
}

#Second Model using cross validation.

#First set folds and cross validation method. Then determine increments.
fitOWControl = trainControl(method="cv",number=10)
cartGrid <- expand.grid(.cp=(1:50)*0.00001)

#function to find the best cp
cpmaker <- function(traindata,fold,increment) {
  fitOWControl = trainControl(method="cv",number=fold)
  cartGrid <- expand.grid(.cp=(1:50)*increment)
  cpvalue <- train(negative ~., data=traindata, method="rpart", trControl=fitOWControl, tuneGrid=cartGr
  return(cpvalue)
}

#build the model using results from CARTMODELTWO
CARTMODEL2 <- function(traindata, cp) {
  cartmodel2train <-  rpart(negative ~., data=traindata, method="class",control=rpart.control(cp=cp))
}

#Third Model (Random Forest)
RFMODEL <- function(trainingset) {
  randomforestmodel <- randomForest(negative ~., data = trainingset)
}
```

```
#Function to test the model with test data
CARTMODELTEST <- function(model,testdata) {
  predictmodel <- predict(model, newdata=testdata, type="class")
  return(predictmodel)
}

#Function to show the results
tablechecker <- function(testdata, modelresults) {
  tablecheck <- table(testdata$negative, modelresults)
  confusionMatrix(tablecheck)
}

#----------------1st Model (Automatic), Testing----------------#

#----------Bing Test----------#
#Model using the training set
bingforummodel1 <- CARTMODELMAKER(bforum_train)
bingredditmodel1 <- CARTMODELMAKER(breddit_train)
bingforandredmode1l <- CARTMODELMAKER(bforandred_train)
prp(bingforummodel1)
prp(bingredditmodel1)
prp(bingforummodel1)

#Test the model using test set
bingforumtest1 <- CARTMODELTEST(bingforummodel1, bforum_test)
bingreddittest1 <- CARTMODELTEST(bingredditmodel1, breddit_test)
bingforandredtest1 <- CARTMODELTEST(bingforandredmode1l, bforandred_test)


#----------Afinn Test----------#
afinnforummodel1 <- CARTMODELMAKER(aforum_train)
afinnredditmodel1 <- CARTMODELMAKER(areddit_train)
afinnforandredmode1l <- CARTMODELMAKER(aforandred_train)
prp(afinnforummodel1)
prp(afinnredditmodel1)
prp(afinnforandredmode1l)

#Test the model using test set
afinnforumtest1 <- CARTMODELTEST(afinnforummodel1, aforum_test)
afinnreddittest1 <- CARTMODELTEST(afinnredditmodel1, areddit_test)
afinnforandredtest1 <- CARTMODELTEST(afinnforandredmode1l, aforandred_test)

#----------------2nd Model (Cross Validated), Testing----------------#

#----------Bing Test----------#
#Figure out the cp values. May take a minute to find values
cpbforums <- cpmaker(bforum_train, 10, 0.002)
cpbreddit <- cpmaker(breddit_train, 10, 0.002)
cpbforandred <- cpmaker(bforandred_train, 10, 0.001)
cpbforums
cpbreddit
cpbforandred
```

```
#Create models with new cp values
bingforummodel2 <- CARTMODEL2(bforum_train, 0.002)
bingredditmodel2 <- CARTMODEL2(breddit_train, 0.002)
bingforandredmodel2 <- CARTMODEL2(bforandred_train, 0.003)
prp(bingforummodel2)
prp(bingredditmodel2)
prp(bingforandredmodel2)

#Test the models
bingforumtest2 <- CARTMODELTEST(bingforummodel2, bforum_test)
bingreddittest2 <- CARTMODELTEST(bingredditmodel2, breddit_test)
bingforandredtest2 <- CARTMODELTEST(bingforandredmodel2, bforandred_test)


#----------Afinn Test----------#
#Figure out the cp values. May take a minute to find values
cpaforums <- cpmaker(aforum_train, 10, 0.0001)
cpareddit <- cpmaker(areddit_train, 10, 0.0001)
cpaforandred <- cpmaker(aforandred_train, 10, 0.0001)
cpaforums
cpareddit
cpaforandred

#Create models with new cp values
afinnforummodel2 <- CARTMODEL2(aforum_train, 0.0028)
afinnredditmodel2 <- CARTMODEL2(areddit_train, 0.0041)
afinnforandredmodel2 <- CARTMODEL2(aforandred_train, 0.0012)
prp(afinnforummodel2)
prp(afinnredditmodel2)
prp(afinnforandredmodel2)

#Test the models
afinnforumtest2 <- CARTMODELTEST(afinnforummodel2, aforum_test)
afinnreddittest2 <- CARTMODELTEST(afinnredditmodel2, areddit_test)
afinnforandredtest2 <- CARTMODELTEST(afinnforandredmodel2, aforandred_test)

#----------------3rd Model (Random Forest), Testing----------------#
#!!!!!!!!!!!WARNING! TAKES A LONG TIME TO RUN!!!!!!

bingforummodel3 <- RFMODEL(bforum_train)
bingredditmodel3 <- RFMODEL(breddit_train)
bingforandredmodel3 <- RFMODEL(bforandred_train)

afinnforummodel3 <- RFMODEL(aforum_train)
afinnredditmodel3 <- RFMODEL(areddit_train)
afinnforandredmodel3 <- RFMODEL(aforandred_train)

#Test the models
bingforumtest3 <- CARTMODELTEST(bingforummodel3, bforum_test)
bingreddittest3 <- CARTMODELTEST(bingredditmodel3, breddit_test)
bingforandredtest3 <- CARTMODELTEST(bingforandredmodel3, bforandred_test)
afinnforumtest3 <- CARTMODELTEST(afinnforummodel3, aforum_test)
afinnreddittest3 <- CARTMODELTEST(afinnredditmodel3, areddit_test)
```

```
afinnforandredtest3 <- CARTMODELTEST(afinnforandredmodel3, aforandred_test)


#-------------------------Compare Models-------------------------#

#Forum Models (bing)
tablechecker(bforum_test, bingforumtest1)
tablechecker(bforum_test, bingforumtest2)
tablechecker(bforum_test, bingforumtest3)

#Reddit Models (bing)
tablechecker(breddit_test, bingreddittest1)
tablechecker(breddit_test, bingreddittest2)
tablechecker(breddit_test, bingreddittest3)

#Combined Forum & Reddit Models (bing)
tablechecker(bforandred_test, bingforandredtest1)
tablechecker(bforandred_test, bingforandredtest2)
tablechecker(bforandred_test, bingforandredtest3)

#Forum Models (afinn)
tablechecker(aforum_test, afinnforumtest1)
tablechecker(aforum_test, afinnforumtest2)
tablechecker(aforum_test, afinnforumtest3)

#Reddit Models (afinn)
tablechecker(areddit_test, afinnreddittest1)
tablechecker(areddit_test, afinnreddittest2)
tablechecker(areddit_test, afinnreddittest3)

#Combined Forum & Reddit Models (afinn)
tablechecker(aforandred_test, afinnforandredtest1)
tablechecker(aforandred_test, afinnforandredtest2)
tablechecker(aforandred_test, afinnforandredtest3)
```