Springboard Capstone Project - Sentiment Predictor for Overwatch

Sine Gov

January 17, 2018

Introduction to Overwatch

Overwatch is a team based First Person Shooter video game that is played online with other people. As a team, your goal is to secure objectives while fighting off the other team. Each person selects from 26 characters and each character falls into one of four roles; Offense, Defense, Tank, or Healing. The composition of the team factors to your success in the matches. During a match, you have the ability to chat and emote certain actions you would like to take in the game.

Overwatch is an ever evolving game with new characters being added to the roster and changes to abilities or mechanics. These changes to a characters abilities, typically called balance changes, are done to help keep the characters from being too powerful, too weak, and to address explotable bugs within the character's ability kit. If a character is too powerful, matches in Overwatch will become predictable causing players to become frustrated and possibly leading them to stop playing. If you make a character too weak, you limit the options of characters also causing frustration since they may not able able to play their favorite characters. Additionally if a person chooses a weak character they may become victims to toxic behavior or language; players on their team may blame them for loosing or harrass them into playing a different character. This behavior becomes a drag on the community and these players may voice their frustrations on social media.

Another feature of Overwatch are cosmetic costumes, also known as skins, a player can obtain through obtaining loot boxes. Loot boxes can either be earned through regular play of Overwatch or bought with real money. With new patches, skins and special events are added to help keep the game exciting and fresh and consumers interested in investing in the product.

Social Media and Overwatch

Social media is a powerful tool that the developers of Overwatch use often. From social media, the developers communicate upcoming news and changes and also get a gauge of the community's feelings or thoughts about different aspects of the game. Additionally, social media provides reactions of the community after balance changes. Of course, these communities don't necessarily represent the totality of the player base, but it can still provide some useful information.

In order to review comments on social media it is often a manual process; someone needs to go through and read each comment and from that they can gain an understanding of the key issues the communitiy is discussing. This analysis attempts to automate this process to gather the major trends at a high level. Social media text data has been collected from Twitter, Reddit, and the Official Overwatch forums.

Goals of Project

Given the popularity of Overwatch and the amount of conversation available, it is important to be able to keep up with the sentiment of the community. With popularity you'll also receive lots of varying opinions and conversations relating to this game. With the volume of text, it becomes a challenge to keep abreast on trends and developing the actual game. To help prevent the developers from falling out of touch, a sentiment analyzer can be built to help them understand overarching trends. Additionally we can use the text data

gathered through this project to model an algorithm to identify negative sentiment which could be used to help customer service automate reports regarding toxic language.

Data Gathered

The data for this project was gathered from three sources: Reddit, the Official Overwatch Forums, and Twitter. Since this projects aims to analysis sentiment, text comments from each of these sites was collected. Each section below outlines how data was gathered and initially cleaned for each site. The method for gathering text data is described below. Code is included in appendix. All data was gathered on the same day: December 22, 2017.

Twitter

Using the twitteR and ROAuth libraries, the hashtag #Overwatch was used to collect tweets. After authenticating the session with Twitter, unique tweets in English were targeted for collection. Since the twitteR package outputs the data in a clean form, no additional data wrangling was needed.

Reddit

A custom script was written to gather data from reddit. The script first gathers all of the hyperlinks of all the topics on the first page of the subreddit /r/Overwatch. Once it has all the necessary links, it visits each link and copies each comment, the date of the comment, and the user name.

Overwatch Forums

Simliar to the method of collecting data from reddit, a custom script was written to gather data from the official Overwatch forums. The script first gathers all of the hyperlinks of all the topics on the first page of the subreddit /r/Overwatch. Once it has all the necessary links, it visits each link and copies each comment and and the user name. Note that there is code to scrape date data, however it works in rare instances. Date is not necessary for sentiment analysis however this code was not removed in the instance this data may become useful in the future.

Data Preparation

Each text comment was first tokenized and analyzed for sentiment using the bing and afinn lexicons from the tidytext library. The score from each of the words was used to calculate the overall sentiment of a given text comment using the formula:

$$Sentiment.score.of.single.comment = \frac{(score.of.words)}{total.words.in.comment}$$

Since each comment may have a varying amount of words, this formula was used to normalize the values between comments so direct comparisons could be done.

Exploration of the Data

With the data gathered, some initial exploration was performed to see if there were any trends or patterns.

Overwatch Forums Word Cloud

From the Overwatch Forums, we can see that many of the Heroes in Overwatch (Junkrat, Mercy, Genji, Mei, & Sombra) are showing up in the word cloud. Additionally, we see words that associate with the mechanics of the game: healing, ability, kill, team, nerf, and so forth. We know we are on the right track since a lot of these words are associated with the Overwatch game.

Reddit Word Cloud

From Reddit, we have similiar words appear in a word cloud, but now we are seeing different characters (Orisa, Winston, Doomfist, and so forth) show up. This could indicate that this community is interested in these characters somehow. This data could be used to make decisions (balance changes) around them depending on the conversations being had about them. Additionally we see the word skin which combined with the other characters being discussed, may mean that the players are speaking on the different skins they may be using or have.

Twitter Word Cloud

From the Twitter word cloud, we see a lot of html code (http and t.co) and terms related to streaming (twitch, stream, tonight, live, watch). This tells us that tweets are probably mostly comprised of links which tells us that the Twitter data may not be as useful for understanding what are communities are concerned with. Because of this, Twitter Data will not be analyzed in detail for the reminder of this report.

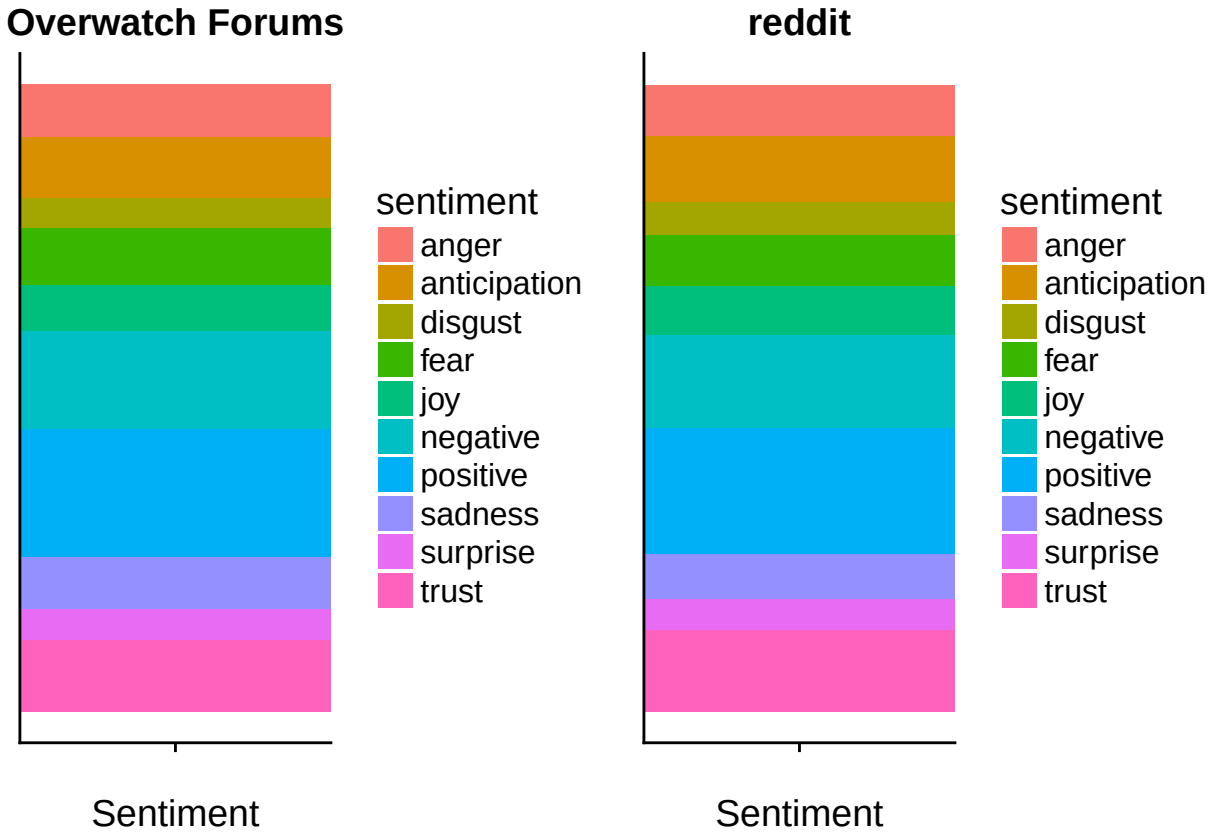
Sentiment

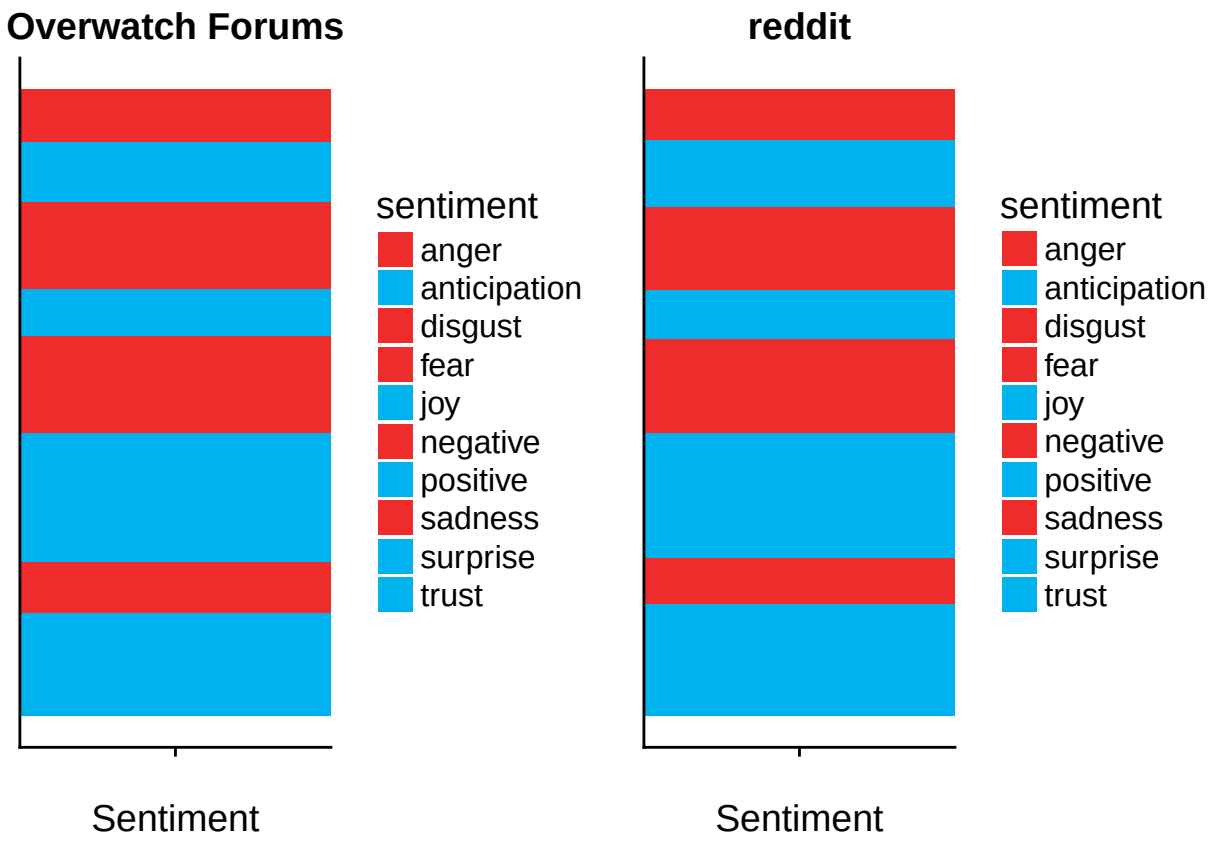
Once we obtained an understanding of the topics discussed by the community, we took a look at the sentiment. Since we discovered Twitter data seems to contain a high amount of links and information about when a person was streaming, we exlcuded this data in further analysis.

Sentiment Words Used by Each Community

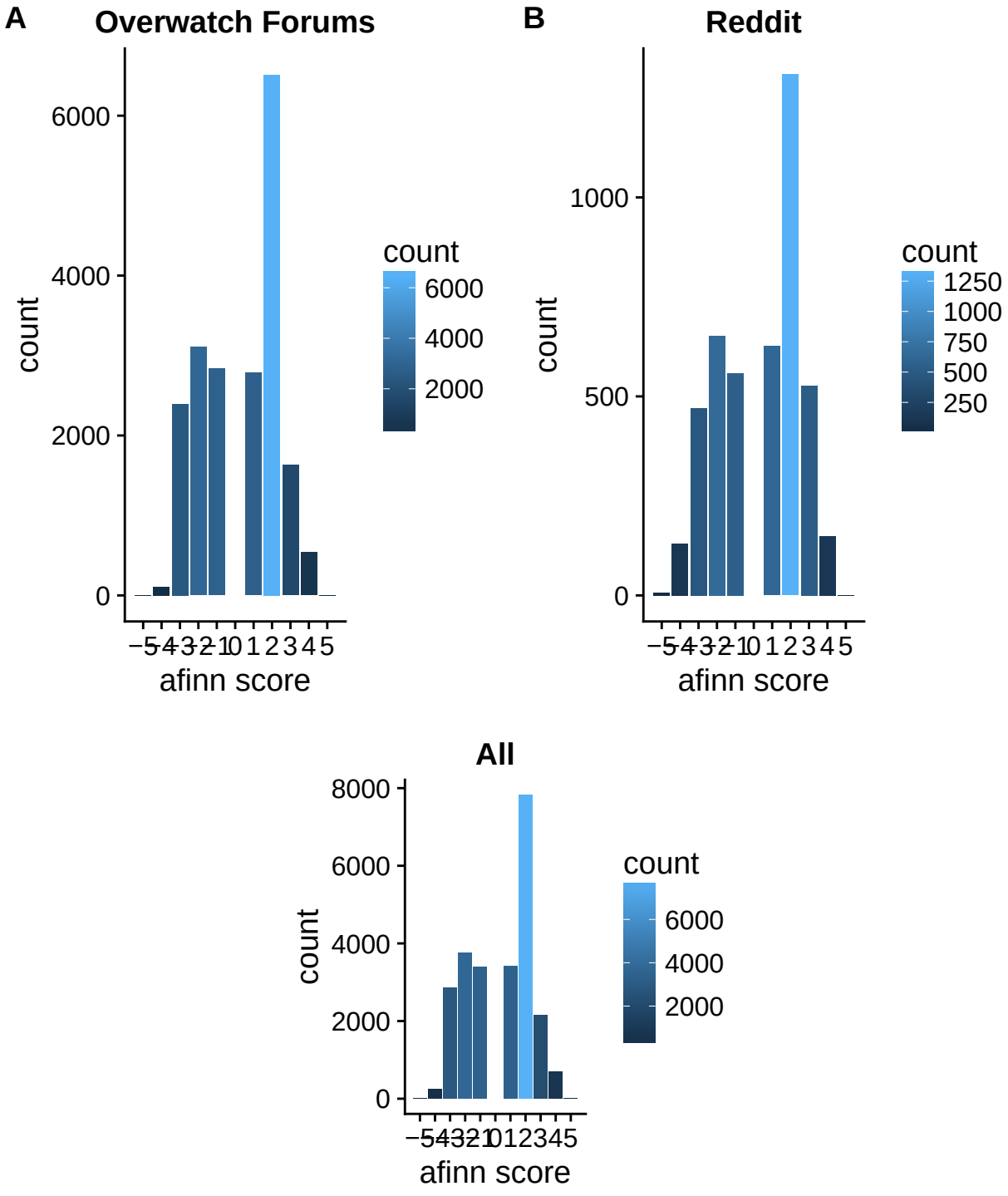
Two plots are shown below; the first with each sentiment colored differenty and the second colored by positve or negative sentiment. From these plots, we can see that the communities are using mostly positive words, however there is a significant amount of negative type of words that were used with all of these communities.

With these results, it's alarming how much negativity is used in the langauge, which is likely reflecting in the game as well. On a more positive note, there is a fair amount of trust which means there is time to improve the perception of the game with the community. We'll take another look at the sentiment using a different lexicon.




Overwatch Forums
reddit
sentiment
anger
anticipation
disgust
fear
joy
negative
positive
sadness
surprise
trust
Sentiment
Sentiment

Distribution of Sentiment Among the Three Communities We see that positivity is high for both communities, but the negative sentiments (negative scores) are still rather proportionally high. These communities are similar in terms of the sentiments they are sharing and we see this when the two communities are combined; the plots don't change much. This implies that there is similarity in thoughts and opinions.



Preliminary Findings in Data

To examine toxicity in another fashion, the average of the sentiment scores of comment was calculated for each community. Bing and Afinn scores were examined the scores are below:

Online Community/Website	Bing Score (Higher is more positive)
Reddit	0.13860
Official Overwatch Forums	0.11594

For the Bing lexicon, scores can fall between -1 or 1 with 1 being positive. When we look at the communities as a whole, it seems that the communities are slightly positive with Reddit having the highest score of 0.13860 of the pair.

Online Community/Website	Afinn Score (Higher is more positive)
Reddit	0.53664
Official Overwatch Forums	0.38761

For the afinn lexicon, scores can fall between -5 to 5 with 5 being the most positive. The communities seem slightly positive when examining them as a whole with Reddit having the highest score of 0.53664.

These scores confirms what we saw during the exploratory statistics of this data; the Overwatch community is slightly more positive, but only slightly.

Toxic Language

It seems that toxic language (which is defined as being unnecessarily aggressive, negative, or inflammatory) is impacting Overwatch on the community sites. Traditionally, competitive online multiplayer games are expected to have being toxic and while it is somewhat reassuring that the Overwatch communities are not completely overrun with toxicity, it is concerning that nearly half the comments are being classified by negative. This is an issue that should be addressed, however getting to the root cause of toxicity may be challenging. In the interim, we can use this toxic language found on the forums and reddit to develop an algorithm to detect negativity. As mentioned, this tool can become effective in helping customer service representatives address language complaints automatically saving them time and helping improve the reputation of Overwatch.

Automatic Sentiment Analyzer

To create the model with the best accuracy, several prototype models were created. 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 this was used for the basis of prediction.

Method

Using the supervised method of classification and regression trees (CART), several predictive models were created. The CART method was iterated on three times to create the most accurate model. 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 scores of the comments determined via the seniment

analyzers and the comments in the comment. Whether the comment was negative or positive was what the model tried to predict based on the words.

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

The accuracy for each model by lexicon is shown below.

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**

Implication of the Model

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.

Summary

From our analysis, we can see that the Overwatch community is experiencing some unrest. From the word clouds, it seemed the hero names were appearing prominently so there may be a correlation to them and the negativity. The silver lining in finding this is we can harness this data to help us create an algorithm to address negative language automatically.

Next Steps

Further research regarding the negative sentiment should be performed. It may yield some interesting thoughts on changes that have occurred in the game. Additionally, tracking this data over time may also help with understanding the community better and may be worth investigating. In regards to the algorithm, it would be worth improving by using actual customer complaints. By training models that will help us detect negative language, we can automate this process freeing up our customer service representatives to do more valuable work such as restoring accounts and addressing billing issues. For now this algorithm can work as a proof of concept to show how this tool can be effective in addressing this complaints. Additionally, by addressing this negativity we can help improve the community for Overwatch players thus improving the reputation of the game overall.

Appendix

Scraping Code

Overwatch Forum Scrapper

```
library(rvest)
library(stringr)
library(tidyr)
library(RCurl)
library(dplyr)
library(curl)


#Read URL
ow.forums <- read_html("https://us.battle.net/forums/en/overwatch/22813879/")

#scrapes links from read URL
links <- ow.forums %>% html_nodes("a") %>% html_attr("href")

#Makes links scrape a data frame
links <- as.data.frame(links, row.names = NULL)

#Find the links for the forums (since links are /forums) and keep these.
keep<- grepl("^/forums", links$links)
keep <- as.data.frame(keep, row.names = NULL)

#bind the answers to the new dataframe
links <- cbind(links, keep)

#remove the rows of unneeded links
#links now contains all of the links for the forum post to be scrapped
links <- links[links$keep == "TRUE",]

#links only contains the tail end of the links. creating front_url to be concatenated later
front_url <- "https://us.battle.net"


#concatenate to create a full link and remove duplicates
full_links <- paste(front_url,links$links, sep = "")
#remove duplicate full_links
clean_links <- unique(full_links)


#new df ow.total
ow.total <- data.frame("link", "title", "time", "text", stringsAsFactors = FALSE)

#For loop to access URL from full_links
#loop for grabbing titles and topic
n <- 1
while(n <= length(clean_links)) {
  read_single_links <- read_html(clean_links[n])
```

```
  title_results <- read_single_links %>% html_nodes(".Topic-title")
  title <- xml_contents(title_results) %>% html_text(trim = TRUE)
  #line for testing to see new threads easier
  print("-----------------------------------------------------------------------------------------------
  print(clean_links[n])
  print(title)

  #pulls content of thread
  topic_results <- read_single_links %>% html_nodes(".TopicPost-bodyContent")
  topic <- trimws(topic_results)

  #cleanup of bodycontent text.
  clean_topic <- gsub("<.*?>", "", topic)
  clean_topic <- gsub("\n", "", clean_topic)


  #pulls date
  topic_time <- read_single_links %>% html_node(".TopicPost-timestamp")
  clean_time <- gsub('<.*content=\\"', "", topic_time)
  clean_time <- gsub('\">.*', "", clean_time)
  clean_time <- as.POSIXct(clean_time, format = "%m/%d/%Y %I:%M %p")
  clean_time <- paste(clean_time)
  print(clean_time)

  #adds topic to a vector with link, title using a for loop
  for(i in 1:length(clean_topic)) {
    ow.total <- rbind(ow.total, c(clean_links[n], title, clean_time[i], clean_topic[i]))
  }

  print(n)
  n<-n+1

}


####-----Cleaning up the Text column since it has dates merged into some of them
newdate <- c()
newtext <- c()

for(i in 1:length(ow.total$X.text.)) {
  if(grepl('\\d\\d/\\d\\d/.*?M', ow.total$X.text.[i]) == FALSE) {
    print("CLEAN")
  } else {
    print("UNCLEAN! CLEANING")
  m <- regexpr('\\d\\d/\\d\\d/.*?M',ow.total$X.text.[i])
  owfdates <- regmatches(ow.total$X.text.[i], m, invert = FALSE)
  owcleantext <- gsub('\\d\\d/\\d\\d/.*?M', '', ow.total$X.text.[i])
  newdate[i] <- owfdates
  newtext[i] <- owcleantext
  }
}
```

```
for(i in 1:length(ow.total$X.text.)) {
  if(is.na(newtext[i])) {
    print("Do not write over old value")
  } else if(newtext[i] != 'NA') {
    #print("Copy over new data")
    ow.total$X.text.[i] <- newtext[i]
    ow.total$X.time.[i] <- newdate[i]
  }
}

#remove duplicate values
str(ow.total)
newow.total_clean <- ow.total[!duplicated(ow.total[4]),]

#write the file
write.csv(newow.total_clean, 'OWFORUMS_manualcheck_12_30.csv')
```

Reddit Scrapper

```
library(rvest)
library(tidyr)
library(RCurl)
library(dplyr)
library(curl)


#Read URL
ow.forums <- read_html("https://www.reddit.com/r/Overwatch/")

#scrapes links from read URL
links <- ow.forums %>% html_nodes("a") %>% html_attr("href")

#Makes scrapped links a data frame
links <- as.data.frame(links, row.names = NULL)

#Find the links for the forums (since links are /forums) and keep these.
keep<- grepl("^https://www.reddit.com/r/Overwatch/comments/", links$links)
#print(keep)
keep <- as.data.frame(keep, row.names = NULL)

#bind the answers to the new dataframe
links <- cbind(links, keep)

#remove the rows of unneeded links
#links now contains all of the links for the forum post to be scrapped
#paste to strip out unnecessary rows
links <- links[links$keep == "TRUE",]
links <- paste(links$links)

#new df ow.total
owreddit.total <- data.frame("link", "title", "time", "text", "user", stringsAsFactors = FALSE)
print(owreddit.total)
```

```r
n <- 1
while(n <= length(links)) {
  read_single_links <- read_html(links[n])

  #scrapes the link's title
  link_title <- gsub("https://www.reddit.com/r/Overwatch/comments/", "", links)
  link_title <- substr(link_title,8, nchar(link_title)-1)
  link_title <- gsub("_", " ", link_title)

  #scrapes comments
  comments <- read_single_links %>% html_nodes(".md")

  #convert from xml to usable format
  comments <- trimws(comments)
  comments <- gsub("<.*?>", "", comments)

  #script going too fast and wasn't cleaning
  comments <- gsub("\n", "", comments)


  #scrapes the .tagline node which has username data
  reddit_user <- read_single_links %>% html_nodes(".tagline")

  #Scrapes to the user name
  reddit_user <- gsub('.*user/', "", reddit_user)

  #Scrapes the stuff after class out
  reddit_user <- gsub('class.*', "", reddit_user)

  #clean up
  reddit_user <- gsub(" ","",reddit_user, fixed=TRUE)
  reddit_user <- gsub("\"","",reddit_user, fixed=TRUE)
  reddit_user <- gsub("<p","",reddit_user, fixed=TRUE)


  #scrapes comment timestamp by date
  comment_time <- read_single_links %>% html_nodes(".tagline, time")
  comment_time <- gsub('.*datetime=', "", comment_time)
  comment_time <- gsub('T.*', "", comment_time)
  comment_time <- gsub('\"', "", comment_time)
  comment_time <- gsub('<p.*', "", comment_time)


  #prints everything scrapped
  print("----------------------------------------------------------------------------------------------
  print("====link====")
  print(links[n])
  print(" ")
  print("====title====")
  print(link_title[n])
  print(" ")
  print("====comment====")
  print(comments[n])
```

```
  print(" ")
  print("====user====")
  print(reddit_user[n])
  print(" ")
  print("====date====")
  print(comment_time[n])
  print(" ")



  #adds topic to a vector with link, title using a for loop
  for(i in 1:length(comments)) {
    owreddit.total <- rbind(owreddit.total, c(links[n], link_title[n], comment_time[i], comments[i+2], 
  }

  print(n)
  n<-n+1
  #Sys.sleep(2)
}

write.csv(owreddit.total, file = "reddit12_22.csv")
```

Twitter Scrapper (Keys removed)

```
library(twitteR)
library(ROAuth)
# Download "cacert.pem" file
download.file(url="http://curl.haxx.se/ca/cacert.pem",destfile="cacert.pem")

cred <- OAuthFactory$new(consumerKey="xxxxx",
                         consumerSecret="xxxx",
                         requestURL='https://api.twitter.com/oauth/request_token',
                         accessURL='https://api.twitter.com/oauth/access_token',
                         authURL='https://api.twitter.com/oauth/authorize')

# Executing the next step generates an output --> To enable the connection, please direct your web brow
cred$handshake(cainfo="cacert.pem")

#save for later use for Windows
save(cred, file="twitter authentication.Rdata")

load("twitter authentication.Rdata")
#OLD- registerTwitterOAuth(cred)

consumer_key <- "xxxxxxxxxxxxxxxxxx"

consumer_secret <- "xxxxxxxxxxxxxxxxxx"

access_key <- "xxxxxxxxxxxxxxxxxx-xxxxxxxxxxxxxxxxxx"

access_secret <- "xxxxxxxxxxxxxxxxxx"
```

```
setup_twitter_oauth(consumer_key, consumer_secret, access_token=access_key, access_secret=access_secret)


# Grab latest tweets
print("Grabbing tweets!!!!! Will take a second.")
tweets_OW <- searchTwitter("#overwatch, -filter:retweets", n=14586, lang = "en")

df_tweets_OW <- twListToDF(tweets_OW)
write.csv(df_tweets_OW, file = "OWT_12-22.csv")
```

Exploratory Statitistics

```
library(dplyr)
library(tidytext)
library(ggplot2)
library(wordcloud)
library(reshape2)
library(RCurl)

#read owforums data into R. Ensure they are not factors
reddit <- read.csv(text=getURL("https://raw.githubusercontent.com/ujumqin/SB_CapstoneProject/master/Fina
twitter <- read.csv(text=getURL("https://raw.githubusercontent.com/ujumqin/SB_CapstoneProject/master/Fi
forums <- read.csv(text=getURL("https://raw.githubusercontent.com/ujumqin/SB_CapstoneProject/master/Fina
withouttwitter <- read.csv(text=getURL("https://raw.githubusercontent.com/ujumqin/SB_CapstoneProject/mas


#reddit <- read.csv("REDDIT_12_22_FINAL.CSV", stringsAsFactors = FALSE)
#twitter <- read.csv("OWTWITTER_12-22_FINAL.csv", stringsAsFactors = FALSE)
#forums <- read.csv("OWFORUMS12_22_FINAL.CSV", stringsAsFactors = FALSE)


#duplicating the text column so we have a copy of the text. this text is used to group
reddit$text_topic <- reddit$X.text.
twitter$text_topic <- twitter$text
forums$text_topic <- forums$text
withouttwitter$text_topic <- withouttwitter$text

#-------------------------Tokenize Text-------------------------
#tokenize the text column for reddit
tidy_reddit <- reddit %>%
  group_by(text_topic) %>%
  dplyr::mutate(linenumber = row_number()) %>%
  unnest_tokens(word, X.text.) %>%
  ungroup()

#tokenize the text column for twitter
tidy_twitter <- twitter %>%
  group_by(text_topic) %>%
  dplyr::mutate(linenumber = row_number()) %>%
  unnest_tokens(word, text) %>%
  ungroup()
```

```r
#tokenize the text column for forums
tidy_forums <- forums %>%
  group_by(X.) %>%
  dplyr::mutate(linenumber = row_number()) %>%
  unnest_tokens(word, text) %>%
  ungroup()

tidy_notwitter <- withouttwitter %>%
  group_by(X.) %>%
  dplyr::mutate(linenumber = row_number()) %>%
  unnest_tokens(word, text) %>%
  ungroup()

#-------------------------Bing Sentiment-------------------------
reddit_bing <- tidy_reddit %>%
  inner_join(get_sentiments("bing")) %>%
  count(word, sentiment, sort = TRUE) %>%
  ungroup()

twitter_bing <- tidy_twitter %>%
  inner_join(get_sentiments("bing")) %>%
  count(word, sentiment, sort = TRUE) %>%
  ungroup()

forum_bing <- tidy_forums %>%
  inner_join(get_sentiments("bing")) %>%
  count(word, sentiment, sort = TRUE) %>%
  ungroup()

withouttwitter_bing <- tidy_notwitter %>%
  inner_join(get_sentiments("bing")) %>%
  count(word, sentiment, sort = TRUE) %>%
  ungroup()

#-------------------------Plotting Data (Bing)-------------------------
reddit_bing %>%
  group_by(sentiment) %>%
  top_n(10) %>%
  ungroup() %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(word, n, fill = sentiment)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~sentiment, scales = "free_y") +
  labs(y = "Reddit Most Used Sentiment Words",
       x = NULL) +
  coord_flip()

twitter_bing %>%
  group_by(sentiment) %>%
  top_n(10) %>%
  ungroup() %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(word, n, fill = sentiment)) +
```

```
  geom_col(show.legend = FALSE) +
  facet_wrap(~sentiment, scales = "free_y") +
  labs(y = "Twitter Most Used Sentiment Words",
       x = NULL) +
  coord_flip()

forum_bing %>%
  group_by(sentiment) %>%
  top_n(10) %>%
  ungroup() %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(word, n, fill = sentiment)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~sentiment, scales = "free_y") +
  labs(y = "Overwatch Forums Most Used Sentiment Words",
       x = NULL) +
  coord_flip()

withouttwitter_bing %>%
  group_by(sentiment) %>%
  top_n(10) %>%
  ungroup() %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(word, n, fill = sentiment)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~sentiment, scales = "free_y") +
  labs(y = "Most Used Sentiment Words By Overwatch Community",
       x = NULL) +
  coord_flip()
#-------------------------nrc Sentiment-------------------------
reddit_nrc <- tidy_reddit %>%
  inner_join(get_sentiments("nrc"))

twitter_nrc <- tidy_twitter %>%
  inner_join(get_sentiments("nrc"))

forum_nrc <- tidy_forums %>%
  inner_join(get_sentiments("nrc"))

withouttwitter_nrc <- tidy_all %>%
  inner_join(get_sentiments("nrc"))

#-------------------------Plotting Data (nrc)-------------------------
ggplot(reddit_nrc, aes(x="", fill=sentiment)) +
  geom_bar(width = 1) +
  ggtitle("reddit")

reddit_bar

reddit_bar + coord_polar("y")

ggplot(forum_nrc, aes(x="", fill=sentiment)) +
  geom_bar(width = 1) +
```

```
  ggtitle("Overwatch Forums")

forum_bar

forum_bar + coord_polar("y")


ggplot(twitter_nrc, aes(x=sentiment)) +
  geom_bar(aes(y=..count.., fill =..count..)) +
  theme(axis.text.x = element_text(size = 10, angle = 45, hjust = 1, vjust = 1)) +
  ggtitle("Twitter") +
  coord_cartesian(ylim = c(0,11000), expand = TRUE)

ggplot(forum_nrc, aes(x=sentiment)) +
  geom_bar(aes(y=..count.., fill =..count..)) +
  theme(axis.text.x = element_text(size = 10, angle = 45, hjust = 1, vjust = 1)) +
  ggtitle("Overwatch Forums") +
  coord_cartesian(ylim = c(0,11000), expand = TRUE)

ggplot(withouttwitter_nrc, aes(x=sentiment)) +
  geom_bar(aes(y=..count.., fill =..count..)) +
  theme(axis.text.x = element_text(size = 10, angle = 45, hjust = 1, vjust = 1)) +
  ggtitle("All Communities") +
  coord_cartesian(ylim = c(0,24000), expand = TRUE)

#-------------------------afinn Sentiment-------------------------
reddit_afinn <- tidy_reddit %>%
  inner_join(get_sentiments("afinn"))

twitter_afinn <- tidy_twitter %>%
  inner_join(get_sentiments("afinn"))

forum_afinn <- tidy_forums %>%
  inner_join(get_sentiments("afinn"))

withouttwitter_afinn <- tidy_notwitter %>%
  inner_join(get_sentiments("afinn"))

#-------------------------Plotting data (Afinn)-------------------------

ggplot(twitter_afinn, aes(x=score)) +
  geom_bar(aes(y=..count.., fill= ..count..)) +
  labs(x="afinn score") +
  xlab("afinn score")+
  coord_cartesian(ylim = c(0,7000), expand = TRUE) +
  ggtitle("twitter") +
  scale_x_continuous(breaks = -5:5)

ggplot(forum_afinn, aes(x=score)) +
  geom_bar(aes(y=..count.., fill=..count..)) +
  labs(x="afinn score") +
  xlab("afinn score")+
  coord_cartesian(ylim = c(0,7000), expand = TRUE)+
```

```
  ggtitle("Overwatch Forums") +
  scale_x_continuous(breaks = -5:5)

ggplot(reddit_afinn, aes(x=score)) +
  geom_bar(aes(y=..count.., fill=..count..)) +
  labs(x="afinn score") +
  xlab("afinn score")  +
  coord_cartesian(ylim = c(0,7000), expand = TRUE)+
  ggtitle("Reddit") +
  scale_x_continuous(breaks = -5:5)

ggplot(withouttwitter_afinn, aes(x=score)) +
  geom_bar(aes(y=..count.., fill=..count..)) +
  labs(x="afinn score") +
  xlab("afinn score")  +
  coord_cartesian(ylim = c(0,13000), expand = TRUE)+
  ggtitle("All") +
  scale_x_continuous(breaks = -5:5)

#-------------------------Word Clouds-------------------------

tidy_forums %>%
  anti_join(stop_words) %>%
  count(word) %>%
  with(wordcloud(word, n, max.words = 50))

tidy_reddit %>%
  anti_join(stop_words) %>%
  count(word) %>%
  with(wordcloud(word, n, max.words = 50))

tidy_twitter %>%
  anti_join(stop_words) %>%
  count(word) %>%
  with(wordcloud(word, n, max.words = 50))

tidy_all %>%
  anti_join(stop_words) %>%
  count(word) %>%
  with(wordcloud(word, n, max.words = 50))

tidy_notwitter %>%
  anti_join(stop_words) %>%
  count(word) %>%
  with(wordcloud(word, n, max.words = 50))

#-------------------------Word Clouds (negative vs positive)-------------------------
par(mfrow=c(2,2))

tidy_forums %>%
  inner_join(get_sentiments("bing")) %>%
  count(word, sentiment, sort = TRUE) %>%
  acast(word ~ sentiment, value.var = "n", fill = 0) %>%
```

```
  comparison.cloud(colors = c("gray20", "gray80"),
                   max.words = 80)

tidy_reddit %>%
  inner_join(get_sentiments("bing")) %>%
  count(word, sentiment, sort = TRUE) %>%
  acast(word ~ sentiment, value.var = "n", fill = 0) %>%
  comparison.cloud(colors = c("gray20", "gray80"),
                   max.words = 80)

tidy_twitter %>%
  inner_join(get_sentiments("bing")) %>%
  count(word, sentiment, sort = TRUE) %>%
  acast(word ~ sentiment, value.var = "n", fill = 0) %>%
  comparison.cloud(colors = c("gray20", "gray80"),
                   max.words = 80)

tidy_notwitter %>%
  inner_join(get_sentiments("bing")) %>%
  count(word, sentiment, sort = TRUE) %>%
  acast(word ~ sentiment, value.var = "n", fill = 0) %>%
  comparison.cloud(colors = c("gray20", "gray80"),
                   max.words = 80)
```

Sentiment Analyzer Algorithm

```
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)
```

```r
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
  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.
  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
  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)
}
```

```r
#--------------------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.
  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)
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.
```

```r
#set the SplitRatio into variable so we can modify easily for all data sets. This ratio will go into th
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(afinnforandredmode11, 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)
```