Datastory Milestone

Sine Gov

January 10, 2017

Detecting Toxic Language in Online Multiplayer Computer Games

Introduction

Toxic language, which is defined as language that is unnecessarily aggressive or inflammatory, is a problem that plagues online communities, especially in online multiplayer computer games. Aside from causing a bad experience while in game, toxicity has a financial impact on companies with products that suffer the problem. Players who experience toxic behavior often report these incidents to customer service representatives, which must then be reviewed. There are methods that are currently employed to combat false reports: thresholds are put into place so customer service representatives are not inundated with reports, however if an online multiplayer game is plagued by toxic behavior, there will still be several reports that must be reviewed. This becomes not only costly from a customer service perspective, but now PR must be involved to help address the perceived toxicity issue in the game. This negative image could also mean loss of current players and prevention of new players from playing the game.

In an ideal situation, we could identify and address the root causes of negativity in these games, however it is a complicated problem with no obvious trend; people will become toxic in game for various reasons and it may well be due to external factors outside of the game itself with the game only serving as a catalyst to trigger the behavior. Until the root cause can be identified, an intermediate solution is necessary to help alleviate the strain that toxicity puts on customer service departments and the online game community.

An automated solution could be designed to identify toxic behavior and action players before a customer service representative needs to address the issue. Using machine learning and data collected from various sources, an algorithm that will identify whether language used could be considered as negative or positive could be created. This algorithm could detect toxicity before a player reports the behavior thus saving the individuals effort for needing to fill out a report. In addition, by not requiring a report to detect toxic language, customer service representatives are free to address other issues. If this algorithm were to have a high accuracy, it could virtually eliminate the need for human intervention for toxic behavior, saving money and time for the game developers and improving the online game experience and possibly the online community.

For this project, the game Overwatch, an competitive online first person shooter, will be examined. For more information about Overwatch, visit the official Overwatch product page: https://playoverwatch.com/en-us

Data Set

The data was gathered from three online communities relating to Overwatch: Twitter, the official Overwatch forums, and the Overwatch subreddit on Reddit. A script scraping text comments for each site was written and employed. Additional data such as user name, time of post, link to post was gathered for each text comment, however these were not used in the development of the algorithm. All data was gathered on the same day: December 22, 2017.

Limitations of Data Set

This data set does not contain any actual in game chat text, which is where toxicity would be reported. Additionally, this data does not contain human ratings for the sentiment of the text; the bing and afinn lexicons were used to determine the sentiment scores. Although these lexicons are useful for detecting sentiment in normal speech, the language of gamers has specific nuances that may not be caught or misclassified by this library. For example, words like ‘kill’ or ‘Mercy’ are commonly used in Overwatch to describe actions in the game, but may carry a neutral sentiment.

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.

Code for Data Preparation

Below is the code used to determine the sentiment for each comment. Only the Overwatch forums is shown below, but this method is also done for reddit and Twitter.

```
library(dplyr)
library(tidytext)
library(tm)
library(caTools)


#----------------PULL DATA FROM FILE & TOKENIZE----------------#
#read owforums data into R. Ensure they are not factors
owforums <- read.csv("OWFORUMS12_22_FINAL.csv", stringsAsFactors = FALSE)

#duplicating the text column so we have a copy of the text.
#this text is destroyed during tokenization
owforums$text_topic <- owforums$text

#tokenize the text column
tidy_owforums <- owforums %>%
  group_by(X.) %>%
  mutate(linenumber = row_number()) %>%
  unnest_tokens(word, text) %>%
  ungroup()

#----------------BING SENTIMENT----------------#

#get the sentiment for the tokenized words
owforums_sentimentbing <- tidy_owforums %>%
  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.
owforums_sentimentbing$score <- 0
```

```
owforums_sentimentbing$count <- 1

#this 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(owforums_sentimentbing$sentiment)) {
  if(owforums_sentimentbing$sentiment[i] == "negative") {
    print("negative")
    owforums_sentimentbing$score[i] = -1
  } else {
    print("positive")
    owforums_sentimentbing$score[i] = 1
  }
}

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

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

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

#----------------AFINN SENTIMENT----------------#
owforums_sentimentafinn <- tidy_owforums %>%
  inner_join(get_sentiments("afinn"))

owforums_sentimentafinn$count <- 1

owforums_sentimentafinn <- owforums_sentimentafinn %>%
  group_by(X.) %>%
  mutate(sentimentscore = sum(score)/sum(count))

owforums_sentimentafinn <- owforums_sentimentafinn[!duplicated(owforums_sentimentafinn$text_topic),]
```

Preliminary Findings in Data

To determine how toxic a community was in comparison to others, the average of the sentiment scores of comment was calculated for each community. Bing and Afinn scores were examined. Below are the tables of the scores:

Online Community/Website	Bing Score (Higher is more positive)
Twitter	0.27714
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 twitter having the highest score of 0.277.

Online Community/Website	Afinn Score (Higher is more positive)
Twitter	1.00597
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 Twitter having the highest score of 1.006.

These scores confirms what we saw during the exploratory statistics of this data; the Overwatch community is slightly more positive, but only slightly. Knowing this, we could use this data to help us find negativity in comments or statements by using machine learning and teaching the an algorithm to detect negative sentiment/language.

Approach

The approach that will be taken for this project is different from the original proposal. I originally wanted to identify a way to see what items/topics were possibly causing negativity. I wanted to use this information to inform the designers on how they should proceed in designing their games. I decided against this for a few reasons. First, game design is often full of nuance and intuition and while people may complain on internet forums, the vast majority may actually agree with or like the changes made to a game; the vocal minority does not represent the vast majority. Second, the tools required to perform this kind of analysis were not well understood by myself. Lastly, this algorithm to detect negative language will provide a a more immediate solution to a complicated problem that requires more investigation.