Classifying Toxic Comments:

A report on using machine learning to find the bad ones

Dovid Burns

4/29/18

Springboard Capstone Project

1. Define the Problem

Toxic comments online are both damaging to people emotionally and prevent productive discussions about sensitive subjects. Many online platforms that allow for user comments such as Facebook, YouTube, Twitter, Wikipedia, Yelp and Instagram have difficulties ensuring that conversations are taking place in an appropriate way. This project will build a classifier using a dataset containing comments from Wikipedia's talk page edits to classify them as toxic or benign. The model can then be used in many platforms to automatically detect—and possibly remove—these comments before they offend participants or deter users from engaging in communications. Additionally, the model can be used as a tool to identify inappropriate users who would be issued serious warnings before being banned from the site.

2. Identify the Client

The client who created this dataset is the Conversation AI team. They are a research initiative founded by Jigsaw and Google who work on making tools to help improve online conversation. The area of interest for their analysis is the study of negative online behaviors, one of which is toxic comments. These comments are defined as being rude, disrespectful or other corrosive forms of language which shut down public online discussions. Conversion AI currently employs systems working to identify toxic language, but these models still make

many errors. This project will seek to correct these errors to create a stronger model for the Conversation AI team.

3. Describe the Data Set

This data set contains 159,571 comments taken from Wikipedia talk pages. Of these public comments, 15,294 are tagged as toxic. These have been flagged manually by human raters. Each comment provided is between 6 and 5,000 characters in length. The data set to be used is publicly hosted at https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge.

In order to create predictive variables besides for the words in the comments, the data was binned by length with each bin increasing in length by 100 characters. This was then analyzed for trends in the rate of toxic comments. Since there was such a wide range of comment length, we took log length as well to analyze and bin accordingly. We found seven distinct bins for analysis (see Table 1 below). The final variable created was the percent of uppercased letters in the comment. This was binned starting at 0-5%, increasing by 5% each time to see the trends in percent of toxic comments per group. We established four bins of percent uppercase as a useful variable (see Table 1 below).

Table 1: Binned Log-Length and Percent Uppercase Breakdowns

Log-Length Range	Num. of Comments	Percent Uppercase	Num. of Comments
0 <= X < 3	571	X < 0.1	148,493
3 <=X< 4	19,486	0.1 <= X < 0.45	8,771
4 <=X< 5	41,370	0.45 <=X < 0.55	257
5 <=X< 6	54,843	0.55 <= X	2050
6 <=X< 7	31,189		
7 <=X< 8	9,227		
8<=X	2,140		

The text in the comments required significant pre-processing before it was useful for analysis. The first steps were to remove all the new line characters, make the words lowercase, and remove apostrophes. After regex was used to filter out any “non-words” from the comments,

a clean string of words remained. We then vectorized these words using CountVectorizer to prepare them for the model. Additionally, the words were stemmed using a SnowballStemmer to simplify similar words for the model.

4. Other Potential Data

Because our client is interested in improving online communication in general—not exclusively comments on Wikipedia—there are other features and data that would strengthen our ability to predict toxic comments across multiple platforms. For example, if our data set was broader, including comments from YouTube, Facebook or Quora, the model built would most likely generalize better outside of Wikipedia. It might also be useful to analyze the amount of time that elapses between the initial view of a post and the submission of a response to this post. Perhaps users who spend more time considering their responses are less likely to post a toxic comment. Also, long-term tracking of users across multiple platforms would inform the model of toxic comments made by users over time and across the web. This information could very likely be a useful predictor of future toxic comments. Additionally, the length of time in which the user has been a member of the community in which they are posting toxic comments could be a significant prediction variable. Another interesting feature that could be created would be to examine the percent of words that were spelled incorrectly in a comment. Perhaps this feature would prove to have predictive value in our model

Lastly, it is very important to note that every one of these comments were labeled by humans as being toxic or benign. This introduces many possible issues into the model, the first of which is the subjective nature of humans identifying comments as toxic or offensive. This means that there will be some inherent margin of error in whatever model is created from this data. Another problem is the likelihood of human error. In order to possibly avoid these issues, a reasonable solution would be to have multiple people score each comment. Should disagreement arise between analysts, the comments would then be reviewed by yet another person or executive team for a final decision on the nature of the comments. Additionally, it is essential to note that a clear set of guidelines must be implemented by the people labeling the comments.

5. Initial Findings

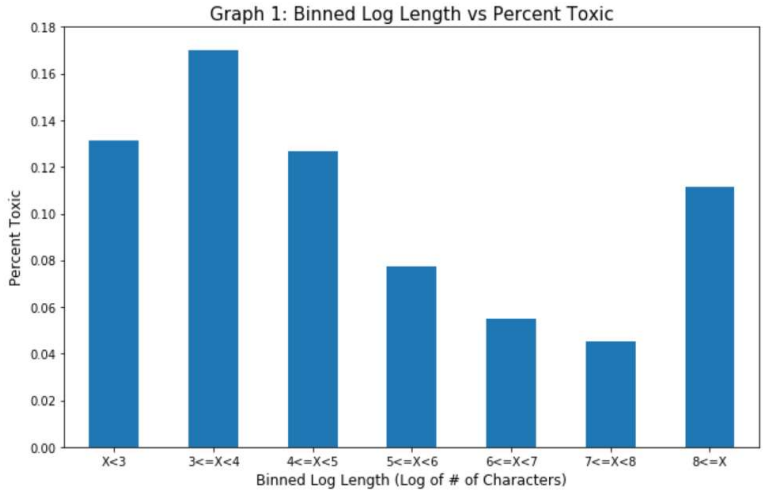
Disclaimer: the dataset contains text that may be considered profane, vulgar, or offensive.

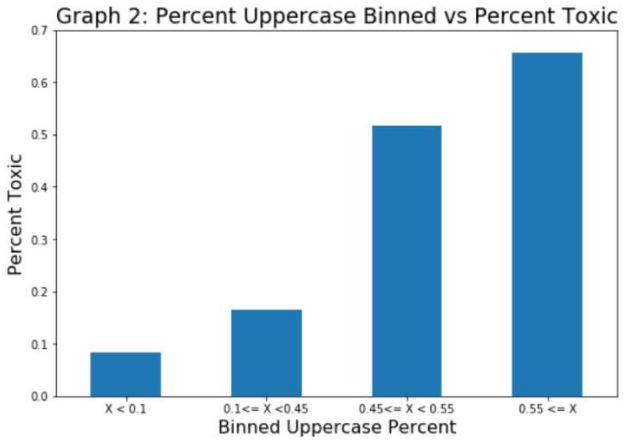
The created variables percent of uppercase characters and log-length of each comment were both strong indicators of comment toxicity. In order to verify that these were statistically valid variables to use, a chi-squared contingency test from the scipy.stats packages was used to find a p-value of zero for both variables. The rates of toxic comments per Log-Length and Percent Uppercase are demonstrated in Table 2.

Table 2: Binned Log-Length and Percent Uppercase Toxic Percent's

Log-Length Range	Percent Toxic	Percent Uppercase	Num. of Comments
0 <= X < 3	13.1%	X < 0.1	8.3%
3 <=X< 4	17.0%	0.1 <= X < 0.45	16.6%
4 <=X< 5	12.7%	0.45 <=X < 0.55	51.8%
5 <=X< 6	7.8%	0.55 <= X	65.6%
6 <=X< 7	5.5%		
7 <=X< 8	4.5%		
8<=X	11.1%		

As indicated above in Table 2, a few clear patterns arise. The overall trend for length is that comment length and likelihood of toxicity are negatively correlated. This makes sense: people generally write offensive comments in brief, not typically taking the time or effort to compose toxic essays. This trend fails at the beginning and end of our data, meaning very short comments do not have the highest frequency of toxic posts, while extremely long comments show an increased tendency to be toxic. The trend in percent uppercase is very clear as well: the percent of characters that are capitalized is positively correlated with likelihood of comment toxicity. This is both a strong statistical indicator as well as a logical communication pattern. Capitalization typically expresses increased vocal volume, which is consistent with communicating anger verbally. People seeking to express toxic ideas would likewise wish to communicate their anger, even in the context of online discourse. Both of these significant trends are illustrated in Graph 1 and Graph 2 below.





While a similar analysis cannot be performed for the vectorized words, the Random Forest Classifier give the most important features in the model. When we ran a model with only the vectorized stemmed words as features, we were able to find the following stemmed words that most strongly predicted toxic comments. The words have been censored in Table 3 to reduce abrasiveness.

Table 3: Random Forest Important Features

Stemmed Words	Feature Importance
F*ck	0.215
Sh*t	0.048
Idiot	0.038
B*tch	0.036
Suck	0.034
*SS	0.03
F*ggot	0.025
C*nt	0.024
Stupid	0.023
*sshol	0.023
D*ck	0.02
Gay	0.018
Bastard	0.015

6. Machine Learning

a. Data Pre-Processing

After vectorizing the words per comment and the binned variables, the final step was to combine these in a way that that a machine learning model could learn from them. We first ran a get_dummies function on both of the binned variables, the Log-length and percent capitalized. We dropped the first dummy column to prevent variable correlation. The features that are created by count vectorizer are stored in a sparse matrix with elements in Compressed Sparse Column format. We used the hstack method from the scipy.sparse library to combine our dummy variables with the vectorized words. This completed the data pre-processing stage. We then split the data into training and testing groups that allowed for simulated model scoring on new data. In order to make sure we had the optimal hyper-parameters, we ran a cross-validated grid search for the count-vectorizer method on both the Naïve Bayes and Random Forest Classifiers. Once we had the best

parameters for the count-vectorizer, we did a similar cross-validated grid-search on the Naïve Bayes and Random Forest Classifiers.

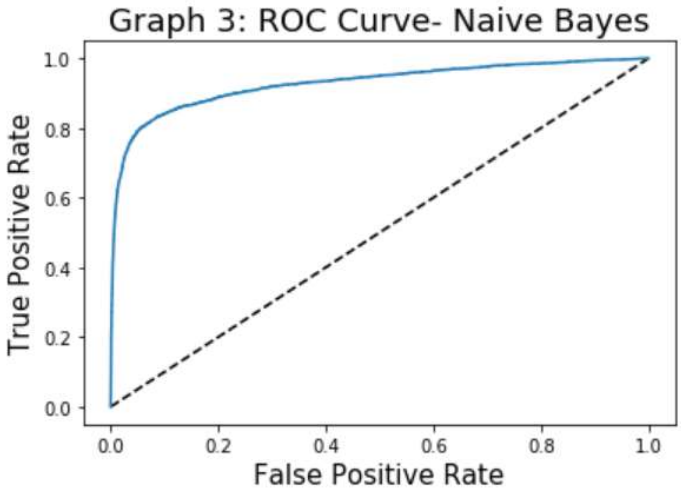
b. Naïve Bayes Classifier

The optimal hyper-parameters for count-vectorizer with Multinomial Naïve Bayes classifier optimizing for roc_auc was a min_df of 15, a max_df of 0.2 and no max features. Using these parameters, we also found that a value of Alpha =1 was optimal for the Multinomial Naïve Bayes classifier. The following scores were obtained when we ran this algorithm on the stemmed data (see Table 4):

Table 4: Multinomial Naïve Bayes Classifier Scores

Metric	Score
Accuracy on Training Data	94.6%
Accuracy on Test Data	94.3%
AUC on Test Data	0.93
F1- Score on Test Data	0.72

When we examined different thresholds besides the default 50%, we found that a threshold greater than 0.55 gave the highest F1-Score of 0.725. The ROC Curve is illustrated in Graph 3 below.



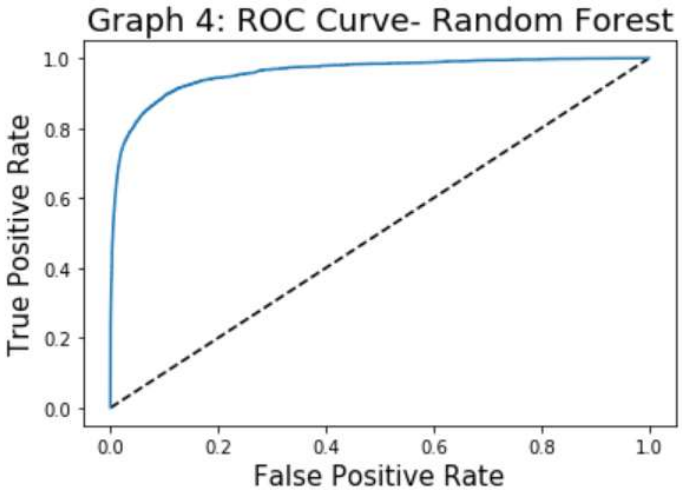
c. Random Forest Classifier

The optimal hyper-parameters for count-vectorizer with a Random Forest Classifier optimizing for roc_auc was a min_df of 10, a max_df of 0.2 and no max features. The grid search optimizing roc_auc found that a Random Forest with bootstrap= False, criterion= 'gini', max_depth= none, max_features= 'auto', min_samples_leaf=10, and min_samples_split= 2 optimized the model. Using the above stated parameters on stemmed words, the model achieved the follow results (see Table 5):

Table 5: Random Forest Classifier Scores

Metric	**Score**
Accuracy on Training Data	94.3%
Accuracy on Test Data	94.3%
AUC on Test Data	0.96
F1- Score on Test Data	0.59

When we analyzed different thresholds besides the default 50%, we determined that a threshold of greater than 0.29 gave the highest F1-Score of 0.733. The ROC Curve is shown in Graph 4 below.



7. Conclusion

Both algorithms Naïve Bayes and Random Forest accurately predicted which comments were toxic. Random Forest had an AUC of 0.96 and a F1-Score of 0.733, which was slightly higher than the AUC of 0.93 and F1-Score of 0.725 indicated for Naïve Bayes. Our client can definitely improve their current system of tagging comments using either of these two models. Because our client is seeking to improve online dialog, we recommend that public platforms use our Random Forest model to implement a system of blocked words based on the model's top predictors for toxic comments. When the application blocks predictably toxic comments, public dialog will be able to maintain the level of decency necessary to sustain meaningful online discourse. This model could be implemented real-time in its entirety to ensure most toxic comments are flagged as they are being made. Unfortunately, the program would likely erroneously flag some comments as toxic. These errors would need to be addressed in a timely manner by staff trained to manage the application and its errors. Lastly, a summary of all of a user's comments can be run through this algorithm to see how many of them are likely to be toxic. A score could then be generated from the comment history for each user, which could be used to warn those with a history of making toxic comments. Further, those users with poor scores could be flagged for extra monitoring, or they could be outright banned from posting future comments. Overall, this report shows that there is real room for machine learning to contribute in a meaningful way to maintaining the safety and appropriateness of online discourse.