Classifying Toxic Comments:

Using machine learning to find the bad ones

Dovid Burns

5/6/18

Springboard Capstone Project

1. Define the Problem

Toxic comments posted in public forums online are common, and they are so corrosive that they rapidly shut down otherwise engaging discussions. 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 and track inappropriate users, allowing platform administrators to issue serious warnings or altogether ban participants who chronically post toxic comments.

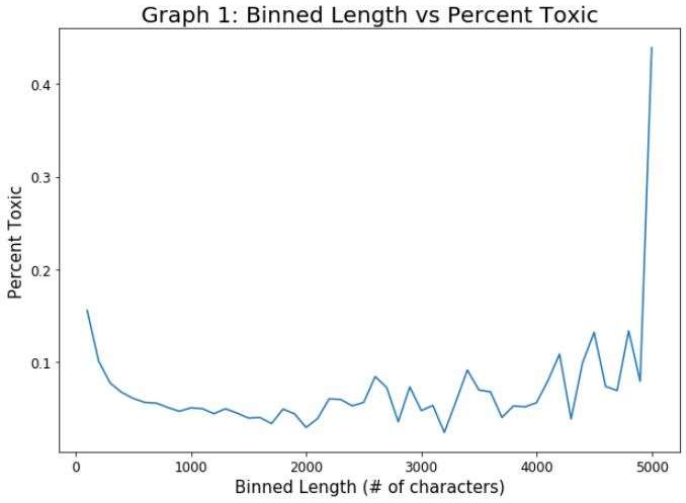
2. Identify the Client

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 remarks are defined as being rude, disrespectful or other corrosive forms of language that 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 in order to create a stronger model for the Conversation AI team.

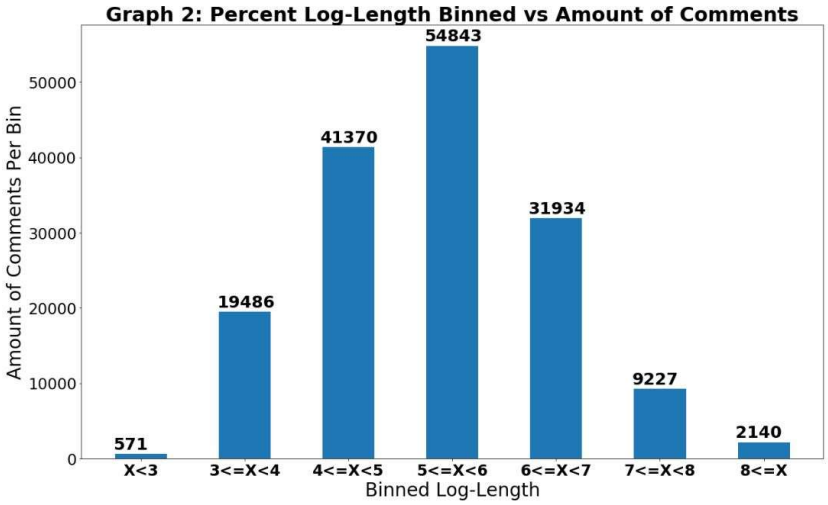
3. Describe the Data Set

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.

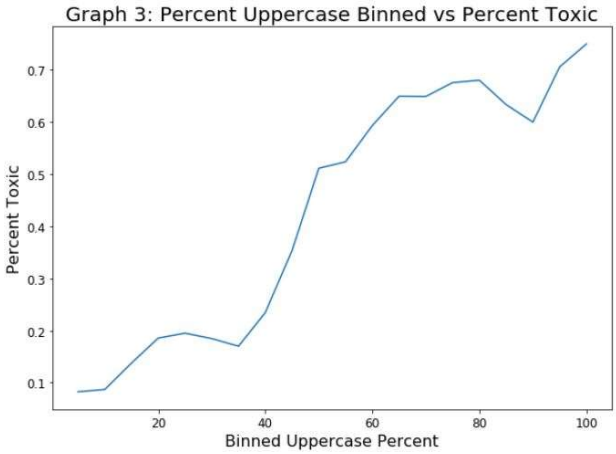
To create predictive variables, the data was binned by length with each bin increasing in length by 100 characters (see Graph 1).



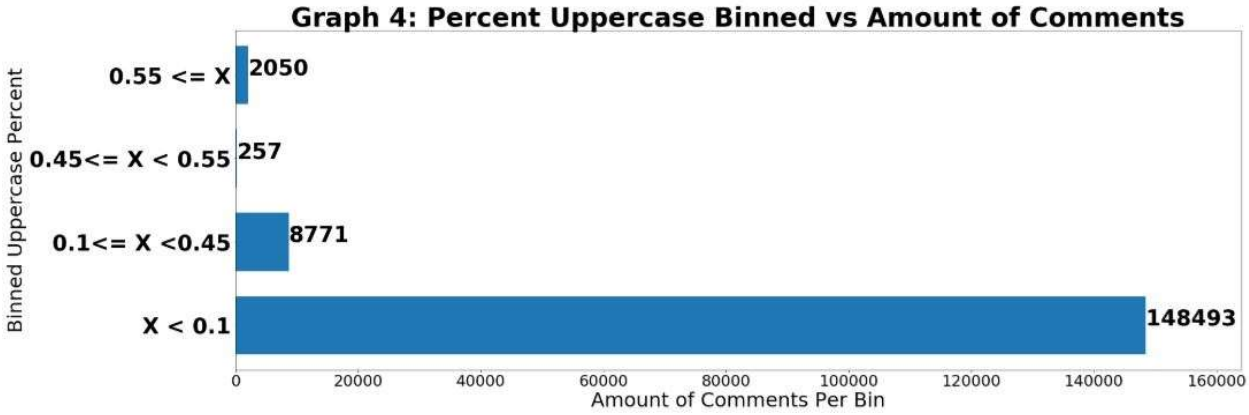
The resulting data was then analyzed for trends in the rate of toxic comments. Since there was such a wide range of comment length, we took log length as well to analyze and bin accordingly. We found seven distinct bins for analysis (see Graph 2 below).



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 (see Graph 3).



We established four bins of percent uppercase as a useful variable (see Graph 4 below).



The text in the comments required significant pre-processing before it was useful for analysis. The first steps were to remove all of 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

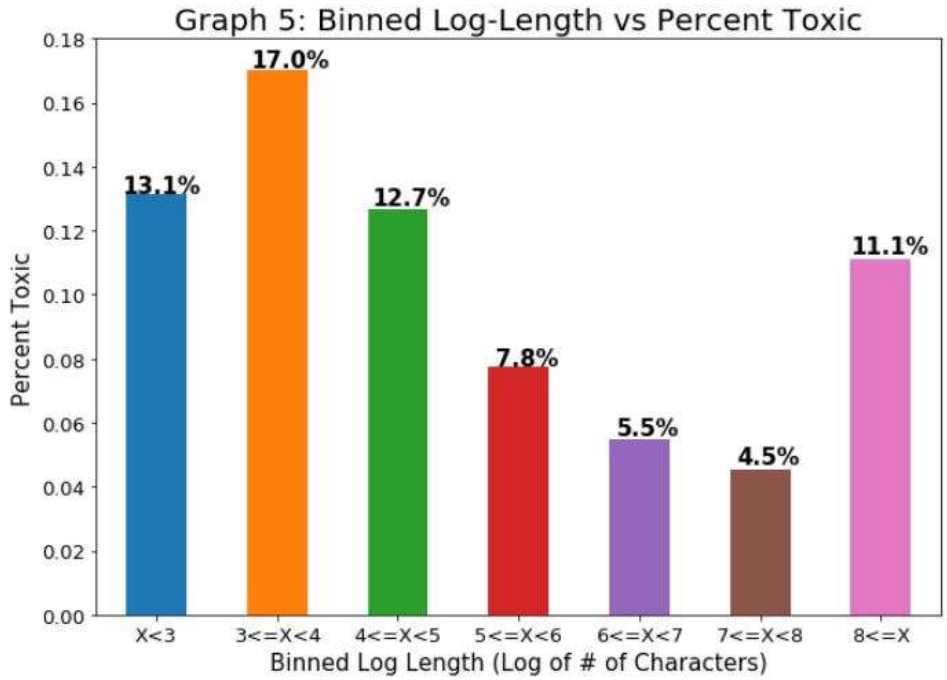
Because our client is interested in improving online communication in general—not exclusively comments on Wikipedia—there are other features and data that would strengthen our ability to predict toxic comments across multiple platforms. For example, if our data set was broader, including comments from 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. 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, meaning analysts will erroneously flag comments on occasion. To avoid these issues, a reasonable solution would be to have multiple people scoring 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 comments. Additionally, it is essential to note that a clear set of guidelines must be implemented by the people labeling the comments. Training of analysts would have to be effective and extremely consistent.

5. Initial Findings

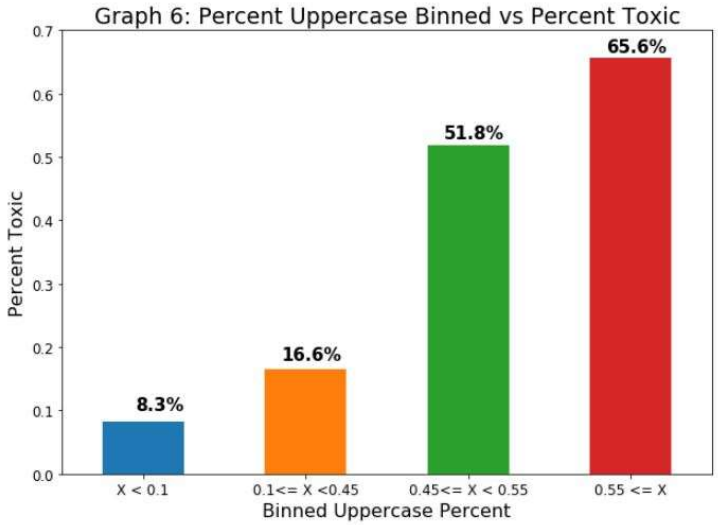
Disclaimer: the dataset contains text that may be considered profane, vulgar, or offensive.

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 less than 0.001 for both variables. The rates of toxic comments per Binned Log-Length and Binned Percent Uppercase are demonstrated in Graphs 5 and 6 below.



As indicated above in Graph 5, 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.



While a similar analysis cannot be performed for the vectorized words, the Random Forest Classifier gives the most important features in the model. When we ran a model with only the vectorized words as features, we found that several words strongly predicted toxic comments (see Table 1). We then created a subset of the dataset to find groups of comments that contain each of these words. Each word-specific subset was then analyzed to find the percent of that subset that were tagged as toxic comments (PERCENT_TOXIC Table 1). Lastly, we did a chi-squared test to find the statistical significance of comments containing each of these words verses percent toxic (P-VALUE Table1). All the p-values from these chi-squared tests were less than 0.001, confirming that these words are statistically significant indicators in determining toxicity. The words have been censored in Table 1 to reduce abrasiveness.

Table 1: Random Forest Important Features

WORD	IMPORTANCE	PERCENT_TOXIC	P-VALUE
f*ck	0.1	94.1	< 0.001
f*cking	0.091	95	< 0.001
sh*t	0.053	78.6	< 0.001
b*tch	0.048	90.1	< 0.001
stupid	0.033	61.2	< 0.001
suck	0.028	85.3	< 0.001
a*s	0.027	14.5	< 0.001
f*ggot	0.024	93.7	< 0.001
idiot	0.021	67.5	< 0.001
d*ck	0.019	73.7	< 0.001
as*hole	0.016	90.3	< 0.001
gay	0.016	54.7	< 0.001
c*ck	0.012	68.6	< 0.001
c*nt	0.012	87.5	< 0.001
bastard	0.012	81.8	< 0.001
hell	0.012	14	< 0.001
p*nis	0.01	68.9	< 0.001
n*gger	0.01	81.7	< 0.001
loser	0.008	43.6	< 0.001
f*g	0.008	88.6	< 0.001

6. Machine Learning

a. Process Overview

The machine learning process has many stages. We first pre-processed the data to be usable in the models and then created three different models using a Multinomial Naïve Bayes, Random Forest and AdaBoost classifiers. We created ROC curves for each model to demonstrate effectiveness and picked the best model based on the highest AUC score on the test data.

b. Data Pre-Processing

After vectorizing the words per comment and the binned variables, the final step was to combine these in a way that a machine learning model could learn from them. We first ran a get_dummies function on both 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. 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.

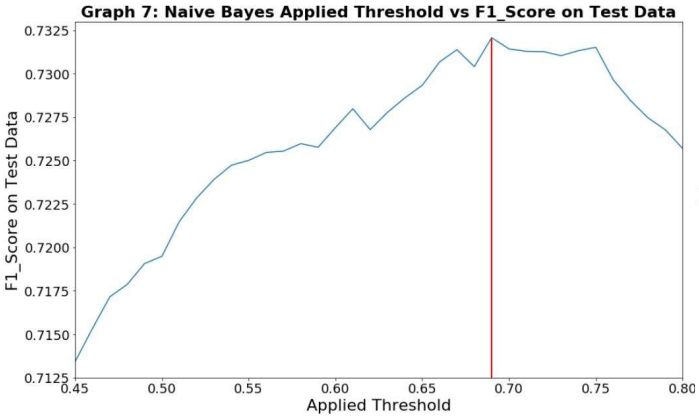
c. Naïve Bayes Classifier

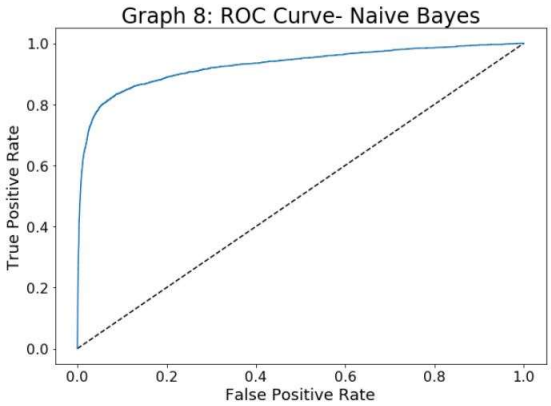
The optimal hyper-parameters for count-vectorizer as determined by grid search with Multinomial Naïve Bayes classifier optimizing for roc_auc was a min_df of 15, a max_df of 0.2 and no max features. Using these parameters, we also found that a value of Alpha =1 was optimal for the Multinomial Naïve Bayes classifier. The following scores were obtained when we ran this algorithm on the stemmed data (see Table 2):

Table 2: Multinomial Naïve Bayes Classifier Scores

Metric	**Score**
Accuracy on Training Data	94.6%
Accuracy on Test Data	94.3%
F1- Score on Test Data	0.72
AUC on Test Data	**0.928**

When we examined various thresholds besides the default 50%, we found that a threshold greater than 0.69 gave the highest F1-Score of 0.732 (see Graph 7). The ROC Curve is illustrated in Graph 8 below.





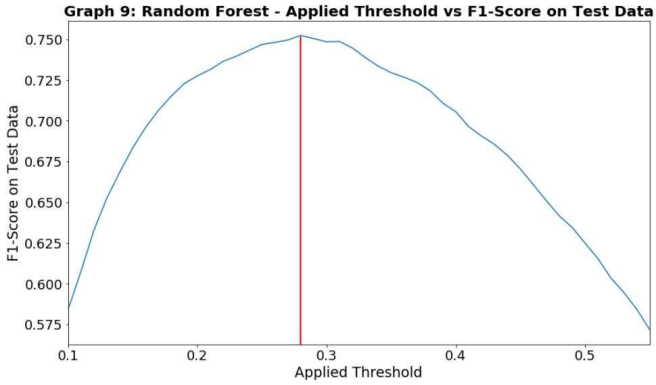
d. Random Forest Classifier

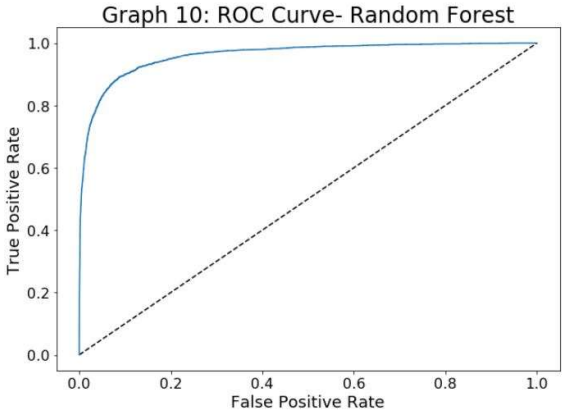
The optimal hyper-parameters for count-vectorizer as determined by a grid search with a Random Forest Classifier optimizing for roc_auc was a min_df of 10, a max_df of 0.2 and no max features. The grid search optimizing roc_auc found that a Random Forest with bootstrap= False, criterion= 'gini', max_depth= 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 3):

Table 3: Random Forest Classifier Scores

Metric	**Score**
Accuracy on Training Data	94.3%
Accuracy on Test Data	94.3%
F1- Score on Test Data	0.625
AUC on Test Data	0.961

When we analyzed different thresholds besides the default 50%, we determined that a threshold of greater than 0.28 gave the highest F1-Score of 0.752. The ROC Curve is shown in Graph 4 below.





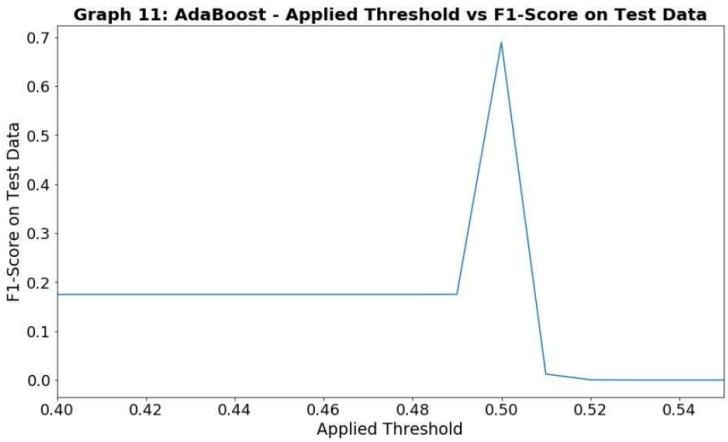
e. AdaBoost Classifier

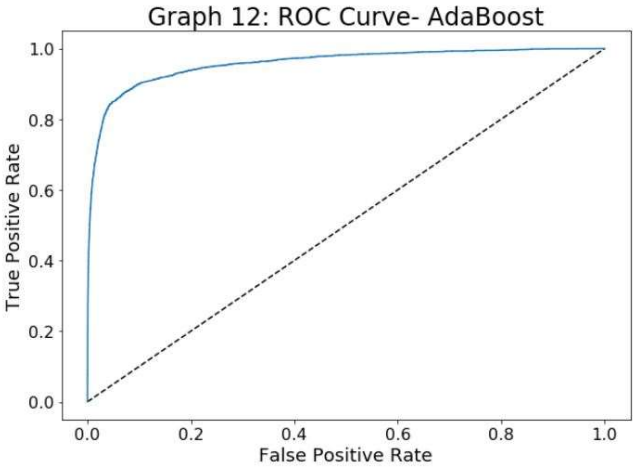
The optimal hyper-parameters for count-vectorizer as determined by a grid search with a AdaBoost Classifier optimizing for roc_auc was a min_df of 0, a max_df of 0.4 and 10,000 for max features. The grid search optimizing roc_auc found that a AdaBoost with a learning rate of 0.1 and 5,000 estimators optimized the model. Using the above stated parameters on stemmed words, the model achieved the follow results (see Table 4):

Table 4: AdaBoost Classifier Scores

Metric	**Score**
Accuracy on Training Data	95.4%
Accuracy on Test Data	95.2%
F1- Score on Test Data	0.69
AUC on Test Data	0.957

When we analyzed different thresholds besides the default 50%, we determined that applying any threshold other than the default of 0.5 dramatically decreased the F1-Score (see Graph 11). The ROC Curve is shown in Graph 12 below.





7. Conclusion

All three models accurately predicted which comments were toxic. Random Forest had the best scores with an AUC of 0.961 and a F1-Score of 0.752 after applying a threshold. Our client can improve their current system of tagging comments using any of these models. Because our client is seeking to improve online dialog, we recommend that public platforms implement a system of blocked or flagged words based on the Random Forest model's top predictors. 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, it is important to note that a summary of all a user's comments can be run through this algorithm to see how many of them are toxic. A score could then be generated from the comment history for each user, which could be used to warn users and the platforms they frequent of their 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.