Toxic Comments Classification

Chris Danicic, Robert Deng, Chandan Gope

April 22, 2018

Abstract

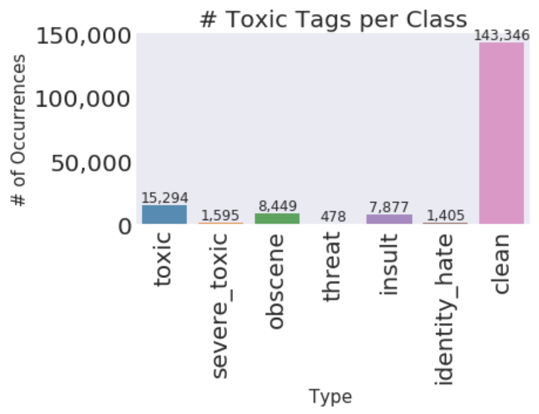
As social media platforms continue to expand reach, the human moderators of these networks will need automated tools for detecting and classifying inappropriate/toxic comments. Inspired by the “Deeper Attention to Abusive User Content Moderation” white paper, we start with a linear baseline model to establish benchmark accuracies and transition to more sophisticated non-linear models such as Convolutional Neural Networks (CNN) and Long Short Term Memory (LSTM). Although our dataset had 6 categories, we focused more on the Toxic vs Clean classification and our best model (CNN) achieved a total classification accuracy of ~95% with a False Positive of ~2% and False Negative of ~29%

Keywords: Toxic comments; classification; CNN; LSTM

Introduction

As social media platforms continue to expand reach, the human moderators of these networks will need automated tools for detecting and classifying inappropriate and abusive comments. Abusive participants not only discourage fruitful discussion, but can also eventually lead to the platform owner completely shutting down the ability to comment. Having a machine algorithm automatically flag abusive content/user would increase moderator efficiency and allow for smooth operation for general user. In this work we present machine learning classification models and test those across different operating parameters to explore what mechanisms work best to capture toxic language in online comments.

Exploratory Data Analysis



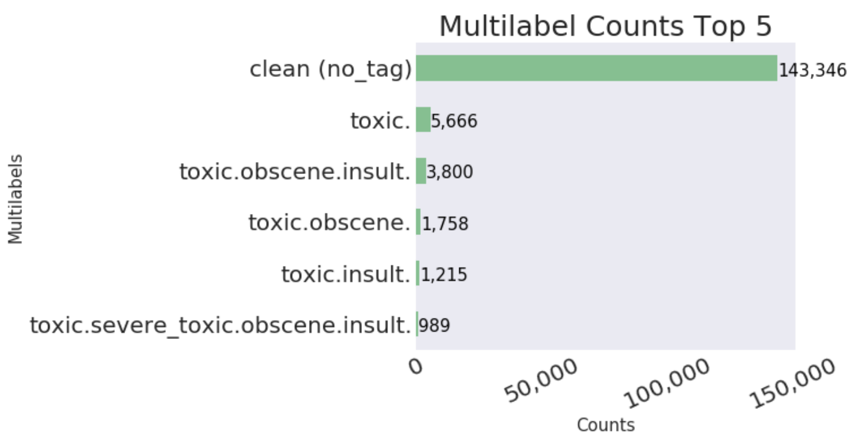


Figure 1: Distribution of comment classifications Figure 2: Distribution of multilabel classification showing co-occurances

We start by investigating our training set of 159,571 comments with 10% or 16,225 total unique comments that were flagged with 1 or more of the toxic tags in figure 1 (referred to as "unclean"). Within unclean comments, 60% were "multi-labels" with 2 or more tags with "toxic" was the most common overlap dominating 98% of total multi-labels in figure 2. Also, toxic, obscene, and insult were the most common triple tag combination, coinciding with their individual frequency counts. Furthermore, all severe_toxic comments were a subset of toxic - *and the permeation and sheer volume of "toxic" tags gives us the intuition in conclusion to prioritize categorizing "toxic" properly*. Next, looking at top unigrams, barring stop words, per tag by TF_IDF frequencies may give us a glimmer into the uniqueness of each category described by keywords.

Clean words speak to consequential response mechanisms ("block", "remove", "delete", and "user") as well as positive sentiments ("thank", "think", "add", and "edit"). The majority of unigrams for toxic tags are profane, invective, and demeaning words with lots of overlap. Threat and identity tags have a few distinct words in "kill", "death" and "n*****", "j**" respectively. Outlier words amongst toxic, threat, and insult tags include "stupid", "idiot", "hope" and "like", which seem like less offensive words individually, but could be offensive contextually. (hope -> hope you will die) *This gives us the intuition to use advanced models to evaluate keywords in sequence and context to unfurl their true meaning.*

Lastly, we explored a plethora of tertiary feature generation surfacing % count word unique and sentiment were the most relevant features. Using the VADER sentiment polarity scores, positive, neutral, and negative sentiment scores were generated for each comment. Comparing clean to unclean comments, all scores exhibited significant T-stat values: 102.4, -33.86, -86.32 respectively, likely due to the strong language used.

Model Selection

We begin our classification models drawing ideas from recent works on convolutional neural networks (CNN) and recurrent neural networks (RNN). Y. Kim [1] advocated simple CNNs with pre-trained vectors while J. Pavlopoulos [2] favored RNNs with additional tuning mechanisms (attention, da-CENT) over CNNs. Our CNN model was inspired by Y. Kim consisting of these layers: embedding, convolutional, 1-max pool, concatenation, dropout, and softmax layer for classification. We experimented with 100-dimension Glove feature vectors in the embedding layer and found slightly better results and improved training times. Experiments with other parameter did not yield fruitful results.

Our first LSTM model was stacked with the following layers: embedding, bidirectional LSTM, global max pool, dropout, dense with Relu activation, dropout, dense with sigmoid activation and 6 dimensional output for categorical properties. The sigmoid function was chosen in lieu of softmax due to the nature of the multi-label task. Our second LSTM model was similar to our first but without the global max pooling and first dropout layer. While max pooling layers have been shown to improve accuracy by Lee et al [4] in some comparisons, max pooling was eliminated in order to sample more of the input sequence. Lastly, the parameters of the top performing bidirectional LSTM (no max pooling) with 10% dropout were used to construct a final LSTM trained to detect only the "toxic" category in order to compare it to the performance of the CNN models.

Methods

Text preprocessing was essential to clean and standardize our data for modelling. We started by using regex to delete any non alpha-numeric character and convert any numbers with NUM, streamlining our input. Next, we note the variation in word length for our comments. 90% of comments had 150 words or less and 96% had 250 or less which inspired our max sequence length parameters. As a result, comments get clipped and to retain any possible toxic words after the max sequence length, our second preprocessing method prioritizes toxic words learned by TF-IDF frequencies.

The EDA noted the imbalance of unclean vs clean comments (10% vs 90%), which inspired us to use weighted_cross_entropy_with_logits and class aware batching. We were able to compute the weighted cross entropy and sample equally from both clean/unclean classes and noticed both methods were similar; batching had a slight edge with a decrease in false negative rate.

Results and Discussion

The first round of bidirectional LSTM model testing consisted of varying the maximum features, the maximum sequence length, and dropout. Like other models, an 80/20 split of train and test was used, with 10% of the (80%) train set reserved for training validation. Each model was trained in 4 epochs, which was sufficient to reach stability in categorical cross-entropy loss using Adam optimizer. The results are as shown in Table 1. It is important to note that categorical accuracy reflects the accuracy in predicting each of the 6 categories for every sample, thus is higher than a simple raw accuracy of predicting all six categories correctly.

Bidirectional LSTM with global max pooling layer				
	10,000 maximum features		20,000 maximum features	
	0.1 dropout	0.2 dropout	0.1 dropout	0.2 dropout
150 sequence length	99.27%	99.35%	99.11%	99.26%
250 sequence length	99.35 %	99.15%	97.69%	99.26%

Table 1, Categorical accuracies of bidirectional LSTM with global max pooling

The second round of bidirectional LSTM model were trained for 4 epochs on the same data sets as the first, with the dropout rate kept constant at 10%. The global max pooling layer and first 10% dropout layer were eliminated. From Table 2 below, the categorical accuracies for the second bidirectional LSTM model set were overall marginally higher than the first set, potentially due to the extra information that is passed to the dense layers with the elimination of the global max pooling layer. For this set, we collected additional false positive and false negative scores to determine the model parameters to train against the single category data used in the CNN development, shown in Table 3 below. Bidirectional LSTM with 20,000 maximum features and a max sequence length of 250 words performed that best of the four models. The 20,000 max feature, 250 max sequence model samples the greatest range of vocabulary and comment content between the 4 models, which might explain its higher accuracies.

Bidirectional LSTM (no max pooling layer)						
	10,000 maximum features			20,000 maximum features		
	0.1 dropout			0.1 dropout		
	Categorical accuracy	"Toxic" False Positive	"Toxic" False Negative	Categorical accuracy	"Toxic" False Positive	"Toxic" False Negative
150 sequence length	99.39%	25.37%	25.84%	99.36 %	7.57%	22.84%
250 sequence length	98.50 %	21.00%	22.81%	99.40%	14.73%	11.11%

Table 2, Categorical accuracy, false positive and false negative percentages of each model (bidirectional LSTM without the global max pooling layer and first 10% dropout layer) with respect to the primary category "toxic".

Baseline, CNN and Bidirectional LSTM Comparison				
	Baseline (Logistic Regression)	CNN	CNN (Class aware batching)	Bi-LSTM 20000 maxfeature, 250 max len
Overall accuracy	93.48%	95.65%	95.26%	96.00%
False positive "toxic"	6%	1%	2%	15.44 %
False negative "toxic"	13%	36%	29%	26.26 %

Table 3, Accuracy comparison summary

From Table 3, it seems the CNN model (with class-aware batching) achieves the best balance of False Positive, False Negative and Overall accuracy.

Conclusion

After our exploratory data analysis and model comparison, it is clear that the majority of toxic comments can be captured by focusing on key words (bag-of-words) such as obscenities and racial slurs. Although the neural network models performed better than the baseline logistic regression, simple linear models work well and may be sufficient as cheap gateway filter. CNN/RNN techniques do improve the accuracy with respect to the simple bag-of-words models, but we believe we could achieve even greater improvements with a larger and more balanced training data.

Reference

[1] Yoon Kim. 2014. Convolutional neural networks for sentence classification. In EMNLP, pages 1746– 1751.
[2] Joh Pavlopoulos et al, "Deeper Attention to Abusive User Content Moderation", Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pg 1136–1146 Copenhagen, Denmark, September 7–11, 2017.
[3] https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html
[4] Lee, Ji Young, and Franck Dernoncourt. "Sequential Short-Text Classification with Recurrent and Convolutional Neural Networks." Proceedings of NAACL-HLT. 2016.

Github link to source code

https://github.com/chandangope/w266-finalproject.git