CSCI 4152 - Project Report
Toxic Comment Classification

Alexandra Startsev (B00691787)
Brandon Poole (B00677266)
Jason Parsons (B00642710)

Submitted: Apr. 15, 2018

Abstract

The problem of filtering toxic online comments requires a strategy for dealing with unintended bias. This report presents two classification methods, Random Forest and Recurrent Neural Networks, as potential solutions for classifying toxicity. The Random Forest classification scheme showed a high degree of accuracy in classifying the split data set. The Recurrent Neural Network has yet to provide final results and is still a work in progress. The goal of trying multiple approaches is to compare and contrast the various machine learning techniques that can be applied to natural language processing tasks, and to increase our knowledge in these areas. This report details the methodology used to create both of our classifiers, and a detailed breakdown of the results. We also touch on the work currently being done in this field by other groups such as the Google Brain team.

1 Introduction

The problem of unintended bias in online comments is one of the main issues in filtering toxicity in online comments. Bias can occur when words associated with gender, race, or sexual orientation are labelled toxic regardless of the context in which they're used. This bias results in skewed accuracy results when classifiers are run on data sets made up of comments made online. Therefore, words that aren't toxic on their own get associated with toxicity by default with no regard for the context in which they are used.

In order to mitigate the issue of unintended bias, new techniques are being explored that place a greater emphasis on the context that words are within a sentence. In this sense, the context should drive the evaluation of toxicity rather than the presence of certain keywords. This task is difficult to automate due to the limitations of current machine learning approaches. As such, the job of filtering comments on sites like Wikipedia is left up to human moderators who must painstakingly go through each and every comment and filter them accordingly. This task is costly in time, money, and human resources.

This report explores several machine learning techniques for classification, with a goal towards improving the accuracy of classifying comments made on Wikipedia. This exploration begins with a look at work currently done by teams around the world using a myriad of different approaches. After which, attempts at classification using both the Random Forest and

Recurrent Neural Network classifiers is presented. The results of these attempts are then compared to current techniques to see if they improve upon current methodologies.

2 Related Work

The subject of this project is based upon the Toxic Comment Classification competition held on the Kaggle competition website. The competition is hosted by the Google Brain team, with an aim towards improving upon their own methods for classifying online comments. As part of the competition, an input file training set containing thousands of comments from Wikipedia contributors is supplied. The training set also contains six different class labels representing different levels of toxicity. These labels are: Toxic, Severe_Toxic, Obscene, Threat, Insult, and Identity_Hate. The competition also provides a test set input file that contains a list of unlabeled comments. The goal is to produce an output file that contains the comment id, followed by a value from 0-1 for each of the six class labels for each comment from the test set.

The main problem in classifying these comments identified by the Google Brain team is that of unintended bias. This occurs when sentences containing words that are not toxic in and of themselves are labelled as toxic due to their possible usage in toxic comments, a result known as false positive bias (Dixon, Li, Sorensen, Thain, Vasserman, 2017). The fallout from this is that groups associated with these words (the LGBTQ+ community, disadvantaged racial and gender groups, etc.) have positive comments unfairly removed leading to an imbalance in representation (Dixon, 2017).

Increasing fairness and inclusivity therefore becomes a series of tradeoffs where a balance between fairness and system performance must be struck (Viegas, Wattenberg, 2017). In attempt to strike this balance, the Google team has injected the training dataset with comments labelled non-toxic that include words associated with disadvantaged groups. The goal of this approach is to bring the number of toxic and non-toxic comments associated with these groups into balance before the classifier is even run (Dixon, et al., 2017).

Once this preprocessing task is done, classification can begin. The Google team used three evaluation models: a baseline, a bias-mitigated model, and a control group (Dixon, et al., 2017). All three models were then trained on an identical convolutional neural network, and the accuracy determined using different measurements of error (AUC (Area Under Curve), Pinned AUC). Their results show that the bias-mitigated model showed a reduction in overall unintended bias without sacrificing performance (Dixon, et al., 2017).

The Kaggle competition hosted by the Google team has a large number of teams contributing using a vast array of methodologies and approaches. As the competition has now ended, some of these teams have begun publishing their work as well as their results. A cursory scan of these approaches shows a wide array of techniques and new ideas that can be applied to the online comment classification problem domain. Whether these approaches improve on Google's current methods is unknown at this time, but the competition's results will hopefully prove to be beneficial to the NLP community at large in addition to Google themselves.

3 Problem Definition and Methodology

Classifying levels of toxicity in online comments is made difficult by the challenge of handling unintended bias in current classification schemes. Previous work on this problem has focused on unbiasing the data before classification is done. This work has already been done in the form of the competition's training data set. Therefore, the main task that follows is applying classification algorithms to this data set, and determining whether the results make sense (correct classification) with a high degree of accuracy.

Two approaches to classification were chosen in order to provide as wide a breadth of experience in natural language processing classification techniques. The classification methods chosen were the Random Forest classifier and a Recurrent Neural Network. These were chosen because both classifiers showed promise as possible solutions for this problem domain. The successes and failures of both approaches, as well as the methodologies used to set them up, are detailed in the following sections.

One thing that both techniques share in common is in the preprocessing of the original training set. While the Google Brain team has provided data that contains a balance of toxic and non-toxic comments, the format of the comments posed an additional concern. The original training and test sets contained all of the original punctuation, special characters, and blank spaces from their original postings on Wikipedia. In order to make the task of parsing and vectorizing the comments easier during classification, the choice was made to remove punctuation, special characters, and whitespace from both the training and test sets.

3.1 Random Forest Classifier

Random Forest is a supervised learning algorithm that learns on sub-samples of data to find points of differences between decisions. These points are float values that represent the breaking points in a decision between A and B, where A is one decision and B is another.

Pre-processing was especially important with Random Forest. The data was vectorized using GloVe, an unsupervised learning tool that vectorizes words given a set (Pennington et al., 2014), then assimilated into general comment vectors by repeated addition. These vectorized comments were then used instead of the initial csv file.

The total data was trained 6 different times for each classification of toxicity; thus resulting in 6 different Random Forest trees. Initial evaluation showed positive results with an average of ~90% accuracy. Of course accuracy was only one part of the equation. As the results were thoroughly reviewed, it was evident that the implementation hesitated when classifying comments as toxic (sensitivity).

Table 1: Sensitivity Evaluation with Depth 1

Toxic	Severely Toxic	Obscene	Threat	Insult	Identity Hate
1%	0%	0%	0%	0%	0%

Table 2: Sensitivity Evaluation with Depth 5

Toxic	Severely Toxic	Obscene	Threat	Insult	Identity Hate
25%	7.5%	20%	0%	17.5%	1%

Each tree was evaluated for sensitivity and corresponding precision with each increase in depth. It was evident that around depth 5, the loss of precision was exponentially larger than the sensitivity that was gained. Since the data was only tested on a subset of the training data, the expectation that the results would be higher when tested on 100% of the data is high.

Our Random Forest implementation was done completely in Python 3.5 with sklearn model representations and tools. The implementation definitely seems more like a script as it's a one-file repeatedly executed per iteration class.

3.2 Recurrent Neural Network Classifier

The Recurrent Neural Network classification method is a modified form of neural network in which outputs from neurons are fed back in as inputs. This allows the RNN to capture time series data in addition to their ability to work on sequences of words of arbitrary lengths as opposed to fixed-length sentences (Geron, 2017). This makes recurrent neural networks an ideal choice of classifier since the training and test data contains comments of differing lengths.

The current version of the attempted version of the RNN allows both the training and test sets to be read in. This is accomplished using the Pandas library. It then does a check for null values in the training set. This check did not show any null values, thus no further preprocessing of the data was needed. Once this is done, the comments and the class labels are split to help with running the classifier.

The Pandas library is then used again to create a dictionary of words from the comments in the training set. Once this is complete, Google's Word2Vec model is used to vectorize the comments to put them in a form more useful to the RNN classifier. Once the data has been vectorized, it can be used to train the recurrent neural network using TensorFlow. After the RNN is trained, it can be run using the test set and a set of classification values from 0-1 for each of the six toxicity types will be produced with an accuracy value for each.

There is a trial and error phase that follows once the first batch of results is returned. The initial RNN model will be very simple, with a minimum number of hidden neuron layers in order to keep things simple. Inexperience with using neural networks means finding the right balance between the number of hidden layers to provide an optimal performance and fidelity mixture. There is also a concern with how to tune the hyperparameters of our RNN model which requires experimentation and experience to know what and how to tune them (Geron, 2017). For now, the model is set up in a manner where ease of use is the primary concern.

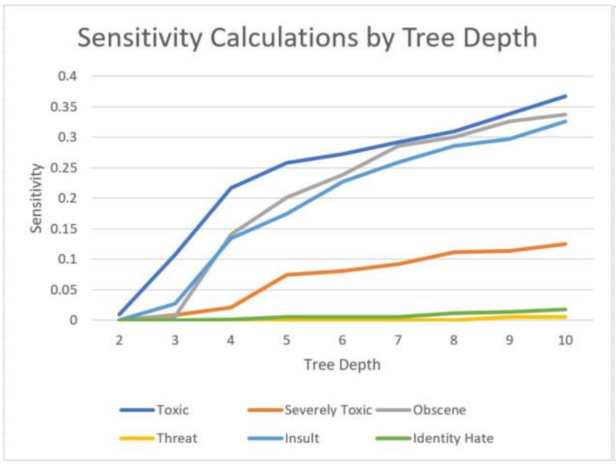
The code is written in the Python programming language due to the large number of user-friendly machine learning libraries available, and the ease with which Python can integrate them. The choice of Pandas, TensorFlow, and Word2Vec comes from research done to determine best practices for this problem domain. Attention was also paid to using as wide an array of tools as possible for education purposes so that the maximum amount of new knowledge of NLP tools and techniques could be gained. Such insight will help guide future work on this project.

4 Experimental Design

The setup for each classifier used the same pre-processed input training set, as well as the same test set. From there each method diverged on separate paths. Each classifier provided their own sets of challenges to be overcome, and require fine tuning in order to reach their full potential when applied to the problem of toxic comment classification. The following sections detail the results of each classifier when applied the training and test sets provided by the competition.

4.1 Results of Random Forest Classifier

The final results of the Random Forest classifier are more than fair. With an accuracy of all 6 categories above 90% (averaging 95%). As mentioned before, sensitivity was strictly compared to precision as shown below in Figures 1 and 2.



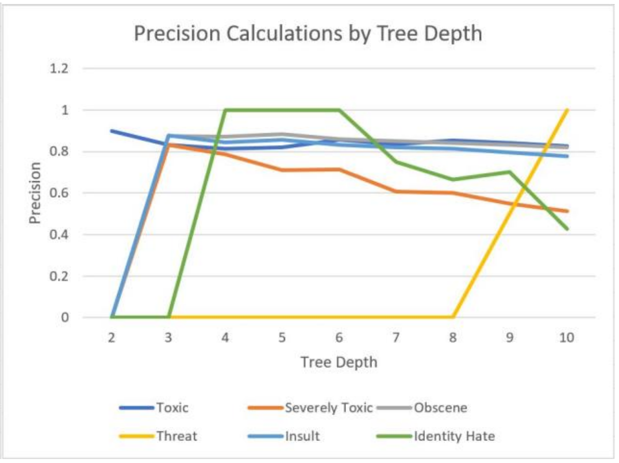


Figure 1 Figure 2

It’s easy to see around depth 5 that there are sharp dips in precision. The final depths per category range from 5 to 9, with Obscene laying on the lower end and Threats at the high.

Next steps would be to train on the full set of training data, then test on the csv file that links to the Kaggle competition. The main reason this wasn’t done was due to the fact that the competition expected probabilistic outputs instead of the true/false ones we had our model set up for.

4.2 Results of Recurrent Neural Network Classifier

At the time of this writing, the Recurrent Neural Network has not produced any results. It remains a work in progress and further research will be needed to bring it to fruition. Decisions on how many neurons, hidden layers, and hyperparameters to use, and how to tune them, are still an open consideration until this research is complete. One of the goals in this process is to find the balance between the number of hidden layers of neurons that provides the best model fidelity, while allowing for performance that allows the neural network to be run in an efficient manner on our current hardware.

In addition, the tuning of the hyperparameters is non-trivial, and may prove to be the most difficult part of setting up the RNN classifier. Decisions on hyperparameters require experience and expertise beyond the scope of this project.

The current RNN setup allows the training and test sets to be read in using Pandas. The preprocessing test to check for null values in the training shows that no null values in the data were found, therefore no further preprocessing of the data is done. The RNN model the uses Pandas to create a dictionary of words from the comments, at which point we attempt to use Google's Word2Vec technique to vectorize the comments and then run the RNN using TensorFlow. Unfortunately, the usage of Word2Vec has not lead to usable vectors for training our RNN. Unfamiliarity with the usage of Word2Vec has caused a large portion of time being used towards understanding how to apply this technique to this project's data sets.

In order to solve this difficulty, the next step for future work on this classifier is to solve the Word2Vec issue, and train our vectorized data on a basic RNN. The number of neurons and hidden layers to use, as well as how to tune the hyperparameters, remains an open issue and will require experimentation using trial and error mixed with further research on best practices. Despite being a work in progress, attempting to implement a recurrent neural network has been invaluable in providing insight on how neural networks work, and what tools and libraries are currently available to help facilitate their implementation.

5 Conclusion

The accuracy of the Random Forest classification scheme suggests that this method may provide a way forward for classifying toxic comments. The Recurrent Neural Network also shows promise, but has a higher degree of complexity. This is due to the inherent nature of neural networks which often contain multiple hidden layers and require fine tuning of the hyperparameters used. These classification methods are only two among many different machine learning techniques that can be applied to natural language processing tasks.

The difficulty in classifying toxic comments online is due to unintended bias present in current classification methods. Bias presents itself when words that are gendered, racial, or sexualy oriented in nature are present. If a positive comment using one of these words is classified as toxic simply due to its presence in the sentence, then this is a failure of our classification system. This report presents two possible methods of mitigating this unintended bias and producing a classification system that handles context in a responsible way.

Toxicity continues to be a major issue in online communities. In order to tackle this issue, we need more robust tools for detecting toxic comments. Due to the costs in time and money involved in having human beings manually policing comment sections, an automated system for handling toxicity is needed. The Random Forest and RNN solutions are a first step towards tackling this complex issue.

References

Dixon, L., Li, J., Sorensen, J., Thain, N., Vasserman, L. (2017). Measuring and mitigating unintended bias in text classification. Association for the Advancement of Artificial Intelligence.

Dixon, L. (2017). Conversation corpora, emotional robots, and battles with bias. https://github.com/conversationai/unintended-ml-bias-analysis/blob/master/presentations/Conversation%20Corpora%2C%20Emotional%20Robots%2C%20and%20Battles%20with%20Bias%20-%20Wikimedia%20Research%20Talk%20-%20Nov%202017.pdf

Geron, A. (2017). Hands-On Machine Learning with Scikit-Learn and TensorFlow. Sebastopol, California: O'Reilly.

Pennington, J. Socher, R. Manning, C. (2014). GloVe: Global Vectors for Word Representation. https://nlp.stanford.edu/projects/glove/

Viegas, F., Wattenberg, M. (2017). Fairness in machine learning. https://github.com/conversationai/unintended-ml-bias-analysis/blob/master/presentations /AI-with-the-best%20fairness%20presentation.pdf