Toxic Overflow: Detecting Hateful Comments

Ahmet Burak Kahraman [1] **Furkan Çağlayan** [1] **Nur Bengisu Çam** [1]

Abstract

Understanding and processing natural languages, or NLP, is an important part of the machine learning field and a path for humankind to be able to communicate with computers. One of the uses of natural language processing is to prevent people from exposure to hateful speeches and bullying on the internet. Nowadays, having a toxic environment is a common problem in almost every social media platform. What we want is to overcome the toxicity on the web using machine learning. We used Wikipedia Talk Pages Comments dataset published by Jigsaw in December 2017. This dataset is highly unbalanced. We addressed this problem by using data augmentation techniques with word embeddings. We used well-known single algorithms but in this paper, we mostly focused on ensemble learning methods to outperform the results of each of these single algorithms. There are still some challenges in this topic, we left them as a future work for the next researchers.

1. Introduction

Building civilized and respectful interactions are important for communication in web platforms. Most people are more careful about communication in real life due to law or their social statues whereas on the internet they tend to act hateful, racist or sexist due to the anonymity. These disrespectful attitudes cause toxicity in social media platforms, online multiplayer games(Märtens et al., 2015) and chat groups. Many initiatives have taken place to improve the interaction quality and to reduce the toxicity, such as Perspective API[1], by Google and Jigsaw. Jigsaw also conducted a challenge on Kaggle[2] and provided a dataset that is made of Wikipedia comments which we will be using in this project. We wish users online could be polite and humane to each other so that kind of detectors would be absolute but sadly there is no sight of it so we need to build these toxic comment detectors.

In this paper, we will be experimenting with the Toxic Comment Classification Challenge dataset published by Jigsaw in December 2017. We applied some of the most renowned machine learning methods such as Naive Bayes, SVM and Decision trees. As we move forward in our project we will be using ensemble learning, by compositing classification algorithms. The first experimental results showed us that even though traditional methods give fine accuracy, the field is open to progress.

Toxic Comment Classification Challenge dataset consists of Wikipedia Talk Page comments and there are in total 6 class labels and these are toxic, severe-toxic, obscene, threat, insult, identity-hate. This is an unbalanced dataset and 22,468 samples have at least one class label whereas 201,081 samples have no label meaning that these samples are non-toxic. The language is mainly English but there are some outliers such as some comments written in different languages.

Since this dataset contains informal comments, there are many disorders such as abbreviations, irony, sarcasm and metaphors. We will talk about these problems throughout the paper.

Comment Text	toxic	insult
Are you f***er mother f***er have nothing to do but block University computers. Go and suck c***s	1	1
It's from one of the many books on various bands... I thought it was from Cult Rockers, but I just checked and Supertramp isn't in there. I will seek out a source for this quote.	0	0
Well you are ridiculous, in fact I suspect that you are Calton, please block me, I dont care	1	0

Table 1. Examples from dataset. Some words have been censored because of profanity.

[*]Equal contribution [1]Hacettepe University, Department of Computer Engineering. Correspondence to: <>.

BBM406 Fundamentals of Machine Learning, Fall 2018.

[1]https://www.perspectiveapi.com/

[2]https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge

The rest of the paper is as follows. Literature and related work review in section 2, description and details of the proposed methodology in 3, experimental setup and further analysis of the dataset in section 4, conclusion of the paper in section 5 and future work that is proposed for next researchers in section 6.

2. Related Work

To identify toxic or aggressive behavior on web platforms several approaches have been proposed. In one of the recent works(Risch & Krestel, 2018) recurrent neural network-based(GRU) ensembling and data augmentation were used. To augment the data, they translated samples to another language then translated back to the original language. With this method, they acquired 63% on English datasets and 60% on the Hindi dataset.

Another deep learning-based ensemble method (Zimmerman et al., 2018) was by Zimmerman et al. They used (Nakov et al., 2013) and (Waseem & Hovy, 2016)[3] for evaluation, reaching $\tilde{6}9\%$ and $\tilde{7}7\%$ accuracies respectively for each dataset. They trained 10 different CNN models and then took the average of the softmax outputs of all models to classify.

Hierarchical based methods have also been applied. In another work(Xiang et al., 2012), Xiang et al. treated twitter comments a composition of topical distributions. To extract topic models, they used Latent Dirichlet Allocation(Blei et al., 2003). After creating models for topics, they classified tweets as offensive or non-offensive, using feature vectors that are consisted of topical distributions and profane words. With these semi-supervised methods, they achieved 75.1% true positive rate over 4029 samples of test tweets.

There are hierarchical attention models(HAN) to classify the text documents. Attention networks are used to give more or less attention to some of the features that are more relevant or irrelevant since not all of the features(here in this case words) have the same level of importance. In (Yang et al., 2016), they proposed 2 levels of HAN. One is for words and the other is sentences in the documents. They used encoders to build the vectors from words and sentences, attention mechanisms to give more or less attention to each vector. They experimented with their HAN model with different datasets such as Yelp, Amazon, IMDB reviews and Yahoo answers. They got better results from the previous papers such as they correctly classified the 71.0% of the documents in Yelp'15 dataset, 49.4% of the documents in IMBD, 75.8% of the documents in Yahoo and 63.6% of the documents in Amazon dataset.

[3]https://github.com/zeerakw/hatespeech

There are common challenges in the toxic comment classification problem. The paper (van Aken et al., 2018)van Aken et al. "Challenges for toxic comment classification: An in-depth error analysis." states these common problems in Wikipedia Talk Pages and Twitter datasets. These problems are toxic comments without swear words or comments with swear words just to make irony and metaphors. They both make the classification harder since the models should consider the semantics as well. Another difficulty is, there are misspelled words in both of the datasets. To overcome this problem, they used word embeddings such as GloVe and FastText. They got a better result with FastText model embeddings. Both of these datasets are highly unbalanced meaning that models will predict the classes that have a high number of samples. They found that ensemble methods work better with such cases than other shallow methods or Neural Network architectural methods, so proposed ensemble learning methods to overcome these problems. Especially, with the classes that have less number of examples, ensemble learning techniques gave much better results than the shallow methods. They used Logistic Regression, RNN, CNN and word embeddings(GloVe and FastText). To build an ensemble learning method, they compared the results of each of these methods, chose the most powerful ones and combine them by using the gradient boosting decision trees.

There are some works that try to 'deceive' or 'fool' toxic detection algorithms. For example, in this paper(Hosseini et al., 2017), Hosseini et al. try to deceive Google Perspective API. Researchers alter the input comments to change the results in several ways. As their first approach, they tried to create spelling errors such as changing the "stupid" to "s.tupiiid". These spelling errors in the sentence cause perspective API to misclassify the results.

3. Methodology

We used the Wikipedia Talk Pages dataset published by Jigsaw in December 2017. As we mentioned earlier, this dataset is highly unbalanced. In this dataset there is a high number of "clean" comments, meaning that these comments do not belong to any of the other toxic labels and this is the most dominant class. 'Toxic' labeled class is the second most dominant class and it includes all other sub-toxic classes as its child class. These classes are *insult*, *identity hate*, *severe toxic*, *threat* and *obscene*. Respective class sample counts are given in table 2.

Label	#Samples
toxic	21,384
insult	11,304
identity hate	2,117
severe toxic	1,962
threat	689
obscene	12,140
clean	201,081

Table 2. Number of comments in each class

1.Pre-Processing Corpus. First of all, we pre-processed the dataset to clear up all the punctuations and redundant abbreviations. We used data augmentation techniques to increase the size of the classes that have fewer samples since not all the classes are suffering from the lack of data samples. We chose the threshold value for each label including "clean". But after we trained our model with this augmented dataset, we figured out that the accuracy metric for the "clean" class was much lower than the other classes. Since the toxic comment classification problem is actually multi-label rather than multi-class, most of the comments fit more than one of the classes. This increases the overall toxic comments so the "clear" comments become much smaller than the rest of the 6 classes. We figured out if we choose 20000 samples from "clear" class, we can increase the accuracy from 46% to 87% which is almost the same for the other classes.

Word	1st similar	2nd similar	3rd similar
book	books	paper	publisher
material	content	information	text
online	internet	web	available
stupid	dumb	pretentious	pathetic

Table 3. Top-3 most similar words from word2vec

2.Data Augmentation. We used Word2Vec word embeddings to augment the comments. Using word2vec, we can create dictionaries for each word in the training corpus and we can access the most similar k words. First, we used the pre-trained word2vec model from $nltk$ library(Loper & Bird, 2002). Then, we trained our own model with the dataset and compared the results. Unsurprisingly, our model gave slightly better results since not all the words in our dataset exist in the pre-trained model. You can see some example words from the dataset and their corresponding word embeddings in table 4 for both our model and pre-trained model. And in table 3, you can see some example words and their corresponding top-3 most similar words from our word2vec model.

Original	Our Model	Pre-trained
book	books	books
material	content	materials
stupid	dumb	dumb
online	internet	-Not in the dictionary-

Table 4. Word embedding examples extracted from the Wikipedia dataset.

We applied these word embeddings into the comments in our dataset to create a sufficient number of data samples for the classes that have an insufficient number of data samples. You can see the examples of augmented comments from the dataset in table 5.

Original Comment	Augmented Comment
i ca not believe no one has already put up this page dilbert desktop games so i did	i ca not believe any only since yet put up here entry dilbert internet explorer snooker so myself ca not
a pair of jew hating weiner nazi schmucks	a pair of jew brainwashed weiner nazi schmucks
unsure at this stage its not very clear what wording is actually being discussed	unsure at this stage our but pretty obvious what wording represents what were explained

Table 5. Augmentation examples from the Wikipedia Talk Pages dataset.

As you can see from the table 5, there are miswritten words in the comments and we can not augment them since they do not exist in the word2vec dictionaries.

3.Boosting. After getting done with pre-processing and augmentation phases, we investigated well-known algorithms such as Naive-Bayes, SVM and Decision tree but mostly focused on ensemble learning by combining each of these algorithms using different parameters for each. To vectorize each word in the comments, we used **tf-idf vectorizer** rather than **tf** because this method is more useful to extract the more important words for classification. We selected the comments from each class randomly and changed the randomly selected words in those comments with their **top-n** words, again randomly.

4. Experimental Setup

Note#1: We bypassed *threat* class because the amount of data is insignificant.

Note#2: In following tables *identity hate* and *severe toxic* classes abbreviated as *i_hate* and *s_toxic*, respectively.

4.1. Preparing the Dataset

As mentioned in section 3, firstly we pre-processed the raw data to cleanup punctation. Some examples of the pre-processing operation is shown at tables 4.1 and 4.1.

	Sample
Original Comment	Online ambassador Hey there! I'd be happy to help out; have you read the Public Policy tutorials yet?
Pre-processed	online ambassador hey there i would be happy to help out have you read the public policy tutorials yet

Table 6. Pre-processing Effect-1

	Sample
Original Comment	Praise looked at this article about 6 months ago -much improved.]
Pre-processed	praise looked at this article about 6 months ago much improved

Table 7. Pre-processing Effect-2

The original dataset is very unbalanced. As shown in the figure 1, *clean* class takes up to %80.2 of the whole dataset. When we try to train this way, it gave around %99 accuracy, but this result is misleading because of the imbalance of the dataset.

Another problem with the dataset is there is no exact *clean* class. To overcome this we added a clean class with samples that don't belong to any other class.

To make it more balanced we sampled some data from each class and augmented them up to 3000. However this led to very poor results for *clean* class because it's more difficult to generalize for *clean* class. So we raised *clean* samples from 3000 to 20000. The result of the dataset after these operations and state of the data is shown at 2. Total sample size at this point is 35000.

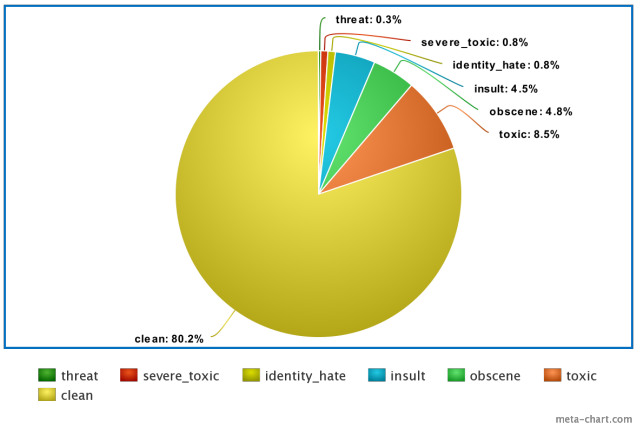

Figure 1. Original State of the Dataset

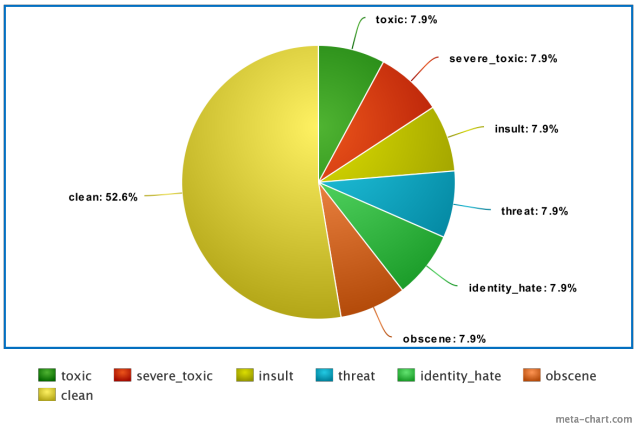

Figure 2. After-state of the Dataset

Naive Bayes. Naive-Bayes classifier is a probabilistic classifier based on Bayes Theorem. Idea is to use prior information to predict posterior, as shown in the below formula.

$$P(A/B) = \frac{P(B/A) * P(A)}{P(B)}$$

In other words, this formula can be interpreted as

$$posterior = \frac{prior * likelihood}{evidence}$$

We used multinomial Naive-Bayes from sklearn(Pedregosa et al., 2011) library to fit the data.

Class	Accuracy
toxic	%47.4
s_toxic	%67.0
insult	%65.8
obscene	%65.8
i_hate	%71.6
clean	%60.0

Table 8. Naive Bayes Results

Results from the table 8 shows that Naive Bayes alone won't offer the best accuracy.

Decision Tree: Decision Tree is a tree-based structure that can be used in both regression and classification problems. A decision tree creates a decision structure by constantly splitting nodes. Node split can be based on information gain and entropy. The information gain of the decision tree is shown below.

$$IG(X) = H(Y) - H(Y/X)$$

Class	Accuracy
toxic	%62.5
s_toxic	%76.3
insult	%63
obscene	%60
i_hate	%79.5
clean	%49.0

Table 9. Decision Tree Results

Table 9 shows that some class accuracies are increased but classes like *clean* and *obscene* accuracies are decreased.

SVM: SVM(Support Vector Machine) or SVC(Support Vector Classifier) is a supervised learning algorithm(Cortes & Vapnik, 1995) that takes advantage of the kernel trick to transform non-linear calculations to a higher dimension. Normally SVCs are binary classifiers, but using one-vs-rest algorithm it can be turned into a multi-class classifier. One of the hyper-parameters of the SVM is the kernel. We used the RBF kernel for our experiments.

Class	Accuracy
toxic	%58.3
s_toxic	%74.6
insult	%59.8
obscene	%60.0
i_hate	%78.8
clean	%56.3

Table 10. SVM Results

Looking at the table 10 overall accuracy is decreased, comparing to Naive Bayes and Decision Tree algorithms.

4.2. Ensemble learning with AdaBoost

AdaBoost(short for Adaptive Boosting) is a machine learning algorithm that was introduced by Yoav Freund and Robert Schapire(Freund et al., 1999). The main idea behind boosting is to stack a bunch of weak classifiers to create a strong classifier. The adaptive part of the AdaBoost algorithm is the change in sample weights in each iteration. According to that rule, misclassified data will have more weight and other data will have less weight after each classification. We tried two versions of AdaBoost, the first one with Naive Bayes as a base estimator of the ensemble and the second one with Decision Tree as a base estimator of the ensemble.

Class	Accuracy
toxic	%81.8
s_toxic	%89.1
insult	%76.6
obscene	%84.1
i_hate	%82.3
clean	%85.8

Table 11. AdaBoost with Naive Bayes Results

Class	Accuracy
toxic	%86.3
s_toxic	%86.6
insult	%83.3
obscene	%86.5
i_hate	%87.3
clean	%87.7

Table 12. AdaBoost with Decision Tree Results

Comparing the Naive Bayes results in table 8 and Naive Bayes with AdaBoost results in table 11, we can see *obscene* and *insult* accuracies are decreased but all other class accuracies are increased. And comparing the simple Decision Tree from table 9 and Decision Tree with AdaBoost results from table 12, we can clearly see accuracies for all classes have increased, even generalizing *clean* class at acceptable rates with %77.5 accuracy.

4.3. Further Analysis of Results

Results from the above sections show that simple classifiers as Naive Bayes and Decision Tree can be very powerful with Boosting. Although accuracies are raised by a lot, some drawbacks limit the accuracy from further increase.

First one of them is the lack of ability of these classifiers to identify irony, metaphors or toxicity without the use of profane words. Another misclassified category would be abbreviations. One of such example can be seen at table 4.3.

Sample	Prediction
f yourself with your family	clean

Table 13. Misclassification example

Below examples show that model has also difficulties classifying mocking and irony.

Sample	Prediction
last warning oooooh i am really scared now ooooooooooh	clean
jog on hitler nobody wants to hear from you	clean
too bad the soviets did not exterminate your entire family album	clean

Table 14. Misclassification examples

5. Conclusion

Toxicity	**True**	**False**
Positive	2610	403
Negative	3388	599

Table 15. Test set results

Looking at the table 15, our augmented and AdaBoosted Decision Tree model has a *sensitivity* of %81.3 and %89.3 *specificity*. This means our model can detect toxic clean data better than toxic data. But as we previously mentioned in 4.3, this topic has some challenges and open to future improvements.

Comparing the results from tables 8, 9 and 10 with the results from tables 11 and 12 shows that AdaBoosting and ensemble learning method greatly increases the prediction accuracy over toxic comment detection tasks.

Examples from table 4.1 show that pre-processing makes comments more human-readable and easier to interpret.

Examples from table 5 shows that word2vec based text augmentation can help to make distinct classes more evenly distributed. And since tf-idf vectorizer takes advantage of unique words, this augmentation method can create even better unique samples, hence improves the generalization of the model. Using ensemble learning based methods allowed us to outperform the results of the shallow models. With using text augmentation, first we made each of these 7 classes to have equal number of comments. Unfortunately, this approach was not successful since the overall number of toxic comments(combination of 6 labels except "clear") became much more larger than the number of "clear" labelled comments. Since our model reach high test accuracy, high sensitivity and high specificity rates, we can confidently say our proposed augmentation model was successful.

6. Future Works

We mentioned that there are miswritten words in the dataset and you can see the examples of comments in table 5. In the work of Van Aken et al. "Challenges for Toxic Comment Classification: An in-depth error Analysis." paper they fixed this problem with sub-word embeddings that are trained on the Twitter dataset using GloVe and FastText models. They got better results with FastText. These methods can be applied to fix the miswritten words so that we can also vectorize them and use them for augmenting the comments. There are also abridgments in the comments, we can try to vectorize them as well.

We can try to use hierarchical models such as BERT, ALBERT, and NNLM. These models are stronger since they consider the positions of the words to create the word embeddings. BERT stands for "Bidirectional Encoder Representations from Transformers" and also referred to as "Mask Language Model". ALBERT stands for "A Lite BERT" and both BERT and ALBERT belong to Google AI. They both are neural-network-based models with a complex structure. ALBERT is proposed to increase the training speed of the BERT by using parameter reduction techniques. NNLM stands for Natural Language Modelling. This model has a different structure from BERT and ALBERT. Hierarchical models provide us power representations than word-based representation models such as FastText, GloVe and Word2Vec since they consider not just the words itself but also positions of the words using the masking strategy on some of the words. Researches found that ALBERT outperforms BERT, so in the future, these models can be integrated into this topic to get stronger and more representative embeddings. Since we have only experimented with one word-based model which is Word2Vec, in the future, FastText and GloVe models can be used in order to compare the results with our paper.

In the future, we believe that toxic detection models will improve. At the same time, toxic users will also try to deceive detectors which will also push researchers including us to create better detection algorithms to reduce the toxicity on the internet.

References

Blei, D. M., Ng, A. Y., and Jordan, M. I. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan): 993–1022, 2003.

Cortes, C. and Vapnik, V. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

Freund, Y., Schapire, R., and Abe, N. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999.

Hosseini, H., Kannan, S., Zhang, B., and Poovendran, R. Deceiving google's perspective API built for detecting toxic comments. *CoRR*, abs/1702.08138, 2017. URL http://arxiv.org/abs/1702.08138.

Loper, E. and Bird, S. Nltk: the natural language toolkit. *arXiv preprint cs/0205028*, 2002.

Märtens, M., Shen, S., Iosup, A., and Kuipers, F. Toxicity detection in multiplayer online games. 12 2015. doi: 10.1109/NetGames.2015.7382991.

Nakov, P., Rosenthal, S., Kozareva, Z., Stoyanov, V., Ritter, A., and Wilson, T. SemEval-2013 task 2: Sentiment analysis in twitter. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pp. 312–320, Atlanta, Georgia, USA, June 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/S13-2052.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Risch, J. and Krestel, R. Aggression identification using deep learning and data augmentation. 08 2018.

van Aken, B., Risch, J., Krestel, R., and Löser, A. Challenges for toxic comment classification: An in-depth error analysis. *CoRR*, abs/1809.07572, 2018. URL http://arxiv.org/abs/1809.07572.

Waseem, Z. and Hovy, D. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL Student Research Workshop*, pp. 88–93, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-2013. URL https://www.aclweb.org/anthology/N16-2013.

Xiang, G., Fan, B., Wang, L., Hong, J., and Rose, C. Detecting offensive tweets via topical feature discovery over a large scale twitter corpus. pp. 1980–1984, 10 2012. doi: 10.1145/2396761.2398556.

Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., and Hovy, E. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1480–1489, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1174. URL https://www.aclweb.org/anthology/N16-1174.

Zimmerman, S., Fox, C., and Kruschwitz, U. Improving hate speech detection with deep learning ensembles. 05 2018.