Preemptive Toxic Language Detection in Wikipedia Comments Using Thread-Level Context

Vanja Mladen Karan and Jan Šnajder
Faculty of Electrical Engineering and Computing
Unska 3, Zagreb
{mladen.karan, jan.snajder}@fer.hr

Abstract

We address the task of automatically detecting toxic content in user generated texts. We focus on exploring the potential for preemptive moderation, i.e., predicting whether a particular conversation thread will, in the future, incite a toxic comment. Moreover, we perform preliminary investigation of whether a model that jointly considers all comments in a conversation thread outperforms a model that considers only individual comments. Using an existing dataset of conversations among Wikipedia contributors as a starting point, we compile a new large-scale dataset for this task consisting of labeled comments and comments from their conversation threads.

1 Introduction

Due to the ever-growing amount of user generated content online, manual moderation of such content is becoming increasingly difficult to scale up. On the other hand, the relative anonymity and lack of personal contact between participants of web conversations lowers inhibitions and increases the risk of toxic behavior, making adequate moderation increasingly important. Consequently, automated detection of toxic language in user generated content is an increasingly important area of research. While automated classification models are unlikely to ever fully replace human moderators, they can make their task easier by suggesting which content to prioritize for moderation.

The typical way to approach this problem is via supervised machine learning, where an input to a model is a user-generated text, and the output is a classification decision (toxic or non-toxic) or a numerical toxicity score. In this paper, we explore two possible extensions of this approach: *preemptive classification* and *thread-level models*.

While practically very useful, standard models are only applicable in a post-hoc scenario, i.e., to detect a toxic comment after if has already been posted. An alternate approach would be to have models detect situations that are likely to lead to toxic comments. If successful, such models would enable moderators to preemptively focus on potentially problematic discussion threads and then either intervene and guide the discussion away from conflict or respond in near real-time after the toxic comment is posted. Large-scale implementation of such near real-time moderation might be unnecessary and require too many moderators. However, for limited parts of discussions that are known to pertain to specially vulnerable social groups, this might be a feasible approach. Our first research question is whether such preemptive toxic comment detection is viable.

The second research question pertains to the benefits of including thread-level information when detecting toxic comments. Namely, most existing models consider every comment in isolation, therefore ignoring the context provided by the other comments in the discussion. For post-hoc models, while useful, this additional information may not be crucial, as the main indicators of toxicity are most present in the text of the comment being classified rather than in the rest of the thread. In the preemptive scenario, however, the model has access only to comments that appeared before a toxic one. We hypothesize that considering the entire thread of comments might be of greater importance in this case.

The contribution of this paper is threefold. First, using a large data set of conversations among Wikipedia contributors, we compile and make publicly available a new dataset with complete discussion threads and with semi-automatically generated toxicity labels. Secondly, we explore the viability of models for the preemptive toxic detection task. Third, we investigate the potential benefits of including thread-level information into models.

Proceedings of the Third Workshop on Abusive Language Online, pages 129–134
Florence, Italy, August 1, 2019.

2 Related Work

Many varieties of toxic language have been considered in NLP research, including sexism, racism (Waseem and Hovy, 2016a; Waseem, 2016), toxicity (Kolhatkar et al., 2018), hatefulness (Gao and Huang, 2017a), aggression (Kumar et al., 2018), attack (Wulczyn et al., 2017a), obscenity, threats, and insults. Waseem et al. (2017) proposed a systematic typology of toxic language.

Post-hoc detection of toxic text has been tackled by traditional machine learning approaches, such as logistic regression (Waseem and Hovy, 2016b; Davidson et al., 2017; Wulczyn et al., 2017b), and support vector machines (SVM) (Xu et al., 2012; Schofield and Davidson, 2017). However, the best performance is most often attained by deep learning approaches, such as convolutional neural networks (CNN) (Gambäck and Sikdar, 2017; Potapova and Gordeev, 2016; Pavlopoulos et al., 2017) and variants of recurrent neural networks (RNN) (Pavlopoulos et al., 2017; Gao and Huang, 2017b; Pitsilis et al., 2018; Zhang et al., 2018b). In our experiments we focus mainly on deep learning-based models.

All the above-mentioned approaches deal with the post-hoc scenario. Other work we are aware of that explores the preemptive scenario is that of Zhang et al. (2018a). There, the task is to predict – given an initial courteous exchange of two user comments – whether the third comment will be toxic. The authors create a manually labeled data set and perform an extensive study on which pragmatic and rhetorical devices are indicative of conversation toxicity. Moreover, there is the work of (Liu et al., 2018), where a logistic regression classifier with a rich feature set (including thread level features) is evaluated on a data set of manually labeled 30000 Instagram comments. In contrast, the data set produced in our work is much bigger, but has only silver labels.

In our work we consider the use of thread-level information for toxic comment detection. Within the scope of this work we limit ourselves to simple mechanisms for including this information into deep learning models. Recently, deep learning models have been proposed that leverage graph structures, such as TreeLSTM (Tai et al., 2015) and GraphSAGE (Hamilton et al., 2017), which might be useful for modeling thread-level structure in our task. We leave the investigation of this possibility for future work.

3 Dataset

At the time of writing we were not aware of the data set from (Liu et al., 2018). At first we considered using the dataset from (Zhang et al., 2018a), but found it rather small (∼1200 examples) for deep learning approaches. Furthermore, this dataset was constructed using a very carefully designed methodology for a specific experiment – detecting whether a toxic comment will appear given a courteous initial exchange of two comments. We are interested in a more general case, where conversation threads might be longer and not necessarily start in a courteous manner. Moreover, we aimed at a setting which would better reflect the realistic working conditions in which our models would be used and allow us to measure their practical impact. Consequently, we decided to create a new dataset from the data collected by Hua et al. (2018). It contains the entire conversational history of comments on Wikipedia modeled as a graph of actions. The possible actions are *Creation*, *Addition*, *Modification*, *Deletion*, and *Restoration*. Automatically derived toxicity scores are also provided for each example.

We apply the following steps to this dataset:

Step 1. Filter the data to remove all threads with less than 2 different participants. This leaves ∼8.7M threads.

Step 2. Apply all *Modification* actions, to update the comments to their most recent version.

Step 3. Flag comments that were deleted. A comment is considered deleted if there is a *Deletion* action on it, without a subsequent *Restoration* action that would undo the effect.

Step 4. Split the threads into the train (70%), dev (15%), and test (15%) sets. The split is done across time: the test set contains the most recent threads, while the train set contains the least recent.

Step 5. Semi-automatically label the examples for toxicity. An example is considered toxic if its toxicity or severe toxicity scores are above 0.64 or 0.92, respectively. [1] and it was deleted by a person who is not the comment author. This heuristic takes into account two signals: (1) the fact that a toxic classifier has high confidence for a comment and (2) the fact the comment was deleted.

[1] The same thresholds as those used by Hua et al. (2018) tuned to give equal error rate on a dataset of Wikipedia comments manually labeled for toxicity. The thresholds yield 86% precision and 84% recall on the tuning data. The thresholds differ as the number of severely toxic comments in the tuning data was much smaller than the number of toxic comments.

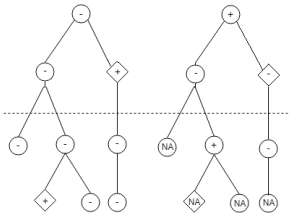

Figure 1: An illustration of examples generated for the post-hoc (left tree) and preemptive (right tree) case. Diamond shaped nodes represent toxic comments while round nodes represent non-toxic comments. Positive examples for the classifier are denoted by + and negative ones by –. Examples denoted by NA are not considered by the classifier: these are the leaf nodes in the preemptive scenario. The filtering from Step 7 has the effect of ignoring all examples with an insufficient number of preceding comments, e.g., for $L_{min} = 2$ all comments above the dotted line would be ignored in the *rich-context* setting.

Considering only deleted examples as toxic would yield noisy labels, as comments are often deleted for reasons other than being toxic. Manual inspection of the silver-labeled dataset reveals that the combination of the toxicity classifier and observed deletion is effective in identifying some of the toxic comments. However, this approach fails to identify those toxic comments which were not deleted or those for which the toxicity classifier failed. The former issue is not problematic, as it was shown by Hua et al. (2018) that toxic comments on Wikipedia get deleted by the community very quickly. Thus toxic comments that are not deleted are quite rare. The latter issue, however, represents a limitation of our work. Our results apply only to those types of toxic language that are detectable by current post-hoc models. Extending this data set to account for more complex types of toxic language would require considerable annotation effort and we leave it as a possibility for future work.

Step 6. Generate examples from each thread in the train/dev/test set for the (1) preemptive scenario and (2) post-hoc scenario as shown in Figure 1. For the post-hoc scenario, examples are generated from all comments in the tree. Positive examples are those comments that are labeled as toxic, while all other comments are negative examples. In the preemptive scenario, examples are generated from all comments that are not leaves of the tree. Positive examples are those comments that have a toxic child and no toxic ancestors, while all other comments are negative examples. The number of preceding comments available for each comment in this data can vary from 0 to over 100 (median is 2). Consequently, to differentiate from the setting in the next step, we will refer to this setting as the *real-context setting*.

Setting	Scenario	train	dev	test
real-context	post-hoc	357K	47K	35K
real-context	preemptive	226K	32K	25K
rich-context	post-hoc	119K	21K	15K
rich-context	preemptive	53K	11K	8K

Table 1: Statistics of the generated data set variants.

Step 7. While the previous setting is more realistic, in order to better assess their full potential, we wished to evaluate the thread level models in a setting where the context provided by preceding comments is always available. To this end, we filter the examples from the previous step such that only those are left that have at least $L_{min} = 2$ comments on the path to the root.[2] We will refer to the datasets obtained in this step as being in the *rich-context setting*.

As the label distribution of the examples obtained in this way is extremely skewed (positive examples account for 0.5 to 1 percent of the data, depending on the setting and scenario), we undersample the negative class. For completeness, we also retain the non-undersampled versions of the dev and test sets for some of the experiments.

Lastly, to additionally evaluate the quality of the silver labels we manually labeled 100 examples from the balanced version of the data set. We found that on these examples the silver labels have 0.51 precision and 1.00 recall. This yields 0.67 F1 measure and is somewhat lower than the expected 0.85 obtained for this classifier in (Hua et al., 2018). The difference indicates that the thresholds from (Hua et al., 2018) obtained on non-deleted comments from Wikipedia may not perform equally well on deleted comments. To address this and increase the quality of the labels, more deleted comments should be manually labeled and thresholds retuned using, e.g., the same error rate method of (Wulczyn et al., 2017a).

Some statistics of the finally generated data set are given in Table 1, and some examples in Table 2. We make the dataset and the code for generating it available.[3]

[2] Following the choice of (Zhang et al., 2018a).

[3] http://takelab.fer.hr/data/pretox

Text	Toxic
Go fuck yourself. Anways.	+
Lazy Bast*rd	+
Superman is a total loser batman would win	+
Sick of this article and the shitty writing	+
When did it become an inherently evil thing to be concerned for human welfare?	–
Please elaborate; this is fascinating.	–

Table 2: Examples of comments from the dataset.

Setting	Scenario	train	dev	test
real-context	post-hoc	357K	47K	35K
real-context	preemptive	226K	32K	25K
rich-context	post-hoc	119K	21K	15K
rich-context	preemptive	53K	11K	8K

Table 3: Statistics of the generated data set variants.

4 Models

We implement several baseline models to get some preliminary results on this data.

Our simplest model is a linear support vector machine (SVM) on TF-IDF weighted unigrams and bigrams. We include the most frequent 10k n-grams into the model, and tune the C hyperparameter on the dev data. This model ignores thread context, even when it is available.

For the deep learning models we use an neural network based encoder to derive a vector representation for each comment in our data. We denote this encoder as enc_{com}. For models that ignore preceding comments, the output of this encoder is fed directly to linear and softmax layers and produces a classification decision for each comment. Thus, the output of our model for a comment $\mathbf{c}$, which is a sequence of word embeddings, could be written as:

$$y_{\mathbf{c}} = softmax(\mathbf{W}^T enc_{com}(\mathbf{c}))$$

For models that take preceding comments into account, the input is not a single comment but a sequence of comments, $\mathbf{t} = (\mathbf{c_1}, \ldots, \mathbf{c_n})$, which includes the comment to classify, $\mathbf{c_n}$, and all of its ancestors, $(\mathbf{c_1}, \ldots, \mathbf{c_{n-1}})$. We first apply enc_{com} to each individual $\mathbf{c_i}$, obtaining comment representations $\mathbf{r_i} = enc_{com}(\mathbf{c_i})$. We then feed the sequence $\mathbf{s} = (\mathbf{r_1}, \ldots, \mathbf{r_n})$ as features into another encoder, which we denote by enc_{thr}. The output of the model for the given input is similar as before:

$$y_{\mathbf{t}} = softmax(\mathbf{W}^T enc_{thr}(\mathbf{s}))$$

For implementing the encoder, we performed preliminary experiments with convolutional neural networks (CNN) (Krizhevsky et al., 2012), long short-term memory networks (LSTM) (Hochreiter and Schmidhuber, 1997), and gated recurrent units (GRU) (Cho et al., 2014), tuning hyperparameters on the development data. We found that, on the development data for this task, GRU and LSTM perform similarly and slightly better than CNN. We also found that bidirectional recurrent models perform slightly better than standard ones. To represent the words we use the freely available 50-dimensional GloVE embeddings (Pennington et al., 2014) trained on 6 billion tokens. Preliminary experiments also reveal models perform better when the embeddings are also updated during training. For our final experiments we use two BiLSTM models with a cell/hidden-state size of 50 to implement enc_{com} and enc_{thr}. We use Adam (Kingma and Ba, 2015) to train the models with a learning rate of 0.001, minibatch size of 128, and early stopping using the dev set. We denote the variants of the model that are thread-agnostic and thread-aware as BiLSTM and BiLSTM-context, respectively. All models are implemented in PyTorch (Paszke et al., 2017) and the code is available online.[4]

5 Results

The results are given in Table 4. Each column represents one dataset variant. All differences within the same variant are statistically significant at $p<0.05$ (tested used bootstrap resampling).

While differences across different dataset variants are not directly comparable, there is a tendency for models to perform much better in the post-hoc scenario than in the preemptive scenario, which is expected. Preemptive models are, however, able manually labeled to beat the random baseline and achieve scores that are numerically similar to those of Zhang et al. (2018a) on their data.

The BiLSTM-context model performs similarly or worse than the BiLSTM model in all cases except the preemptive real case where context does help, but both LSTM-based models are outperformed by a simple SVM. This indicates that the additional information provided by the thread context is, in this case, not very useful for determining the correct class. Manual inspection of the data set confirms that humans could determine the toxicity of most comments without referring to the thread for additional context.This intuition is invalid in

[4] http://takelab.fer.hr/data/pretox

	Preemptive		Post-hoc	
Model	Real	Rich	Real	Rich
Random	.500	.500	.500	.500
SVM	**.601**	.578	.883	.893
BiLSTM	.558	**.620**	.901	**.902**
BiLSTM-context	.586	.602	**.904**	.900

Table 4: Results in various settings and scenarios. Random is the expected performance of choosing a random class with uniform probability across the classes. The numbers are F1-scores on perfectly balanced test data.

cases when the thread context for preemptive detection already contains a toxic comment. A presence of a toxic comment in a thread is a good indicator of a situation where more toxicity will follow. Thus considering the entire thread leads to better performance in such cases. This, however, is not true *preemptive* detection, as toxicity already occurred earlier in thread. Consequently, we filtered out such cases from the data by requiring comments that are positive examples for the preemptive case to have no toxic ancestors (as described in Chapter 3). It is however worth mentioning that, for this reason, our preliminary experiments which omit this filtering step indeed showed more noticeable benefits of having the thread available in the preemptive case.

We also note that the unbalanced nature of this data has a very negative effect on performance in a practical setting. For example, even after tuning the classification threshold to maximize F1 on unbalanced dev data, the F1-score for the best post-hoc model on the unbalanced test is still below 0.5. Thus, more work is required to make models for this task that are applicable in a real-world setting.

6 Conclusion

We compiled a large semi-automatically labeled dataset for studying preemptive toxic language detection in Wikipedia conversations. We explored two types of deep learning models for this task: those that only consider a single comment and those that take into account the context by considering preceding comments in a conversation. In our experiments, the context-sensitive models did not significantly outperform context-agnostic ones. While all preemptive models would beat a random baseline, their performance is still too low for practical applications.

There are numerous possibilities for future work. One is to employ more sophisticated graph-based deep learning methods such as GraphSAGE (Hamilton et al., 2017). Another direction would be exploring ways to better address the class unbalance typical for this task. Yet another possibility would be to enrich the input features with information available about the user who is commenting, e.g., whether they had toxic comments in the past, or their personality profile derived from text using models such as that of Gjurković and Šnajder (2018). Finally, combining deep learning with discourse and pragmatic features, such as those of Zhang et al. (2018a), might be a good next step to improve the models in the preemptive setting.

References

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language. In *Proceedings of ICWSM*.

Björn Gambäck and Utpal Kumar Sikdar. 2017. Using Convolutional Neural Networks to Classify Hate-speech. In *Proceedings of the First Workshop on Abusive Language Online*, pages 85–90.

Lei Gao and Ruihong Huang. 2017a. Detecting online hate speech using context aware models. In *Proceedings of Recent Advances in Natural Language Processing (RANLP)*, pages 260–266, Varna, Bulgaria.

Lei Gao and Ruihong Huang. 2017b. Detecting online hate speech using context aware models. *arXiv preprint arXiv:1710.07395*.

Matej Gjurković and Jan Šnajder. 2018. Reddit: A gold mine for personality prediction. In *Proceedings of the Second Workshop on Computational Modeling of Peoples Opinions, Personality, and Emotions in Social Media*, pages 87–97.

Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Yiqing Hua, Cristian Danescu-Niculescu-Mizil, Dario Taraborelli, Nithum Thain, Jeffery Sorensen, and Lucas Dixon. 2018. Wikiconv: A corpus of the complete conversational history of a large online collaborative community. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2818–2823. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. pages 1–13.

Varada Kolhatkar, Hanhan Wu, Luca Cavasso, Emilie Francis, Kavan Shukla, and Maite Taboada. 2018. The sfu opinion and comments corpus: A corpus for the analysis of online news comments.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Ritesh Kumar, Aishwarya N. Reganti, Akshit Bhatia, and Tushar Maheshwari. 2018. Aggression-annotated Corpus of Hindi-English Code-mixed Data. In *Proceedings of the 11th Language Resources and Evaluation Conference (LREC)*, pages 1425–1431, Miyazaki, Japan.

Ping Liu, Joshua Guberman, Libby Hemphill, and Aron Culotta. 2018. Forecasting the presence and intensity of hostility on instagram using linguistic and social features. In *Twelfth International AAAI Conference on Web and Social Media*.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *NIPS-W*.

John Pavlopoulos, Prodromos Malakasiotis, and Ion Androutsopoulos. 2017. Deep learning for user comment moderation. In *Proceedings of the First Workshop on Abusive Language Online*, pages 25–35.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Georgios K Pitsilis, Heri Ramampiaro, and Helge Langseth. 2018. Detecting offensive language in tweets using deep learning. *arXiv preprint arXiv:1801.04433*.

Rodmonga Potapova and Denis Gordeev. 2016. Detecting state of aggression in sentences using cnn. *arXiv preprint arXiv:1604.06650*.

Alexandra Schofield and Thomas Davidson. 2017. Identifying Hate Speech in Social Media. *XRDS: Crossroads, The ACM Magazine for Students*, 24(2):56–59.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1556–1566.

Zeerak Waseem. 2016. Are you a racist or am i seeing things? annotator influence on hate speech detection on twitter. In *Proceedings of the first workshop on NLP and computational social science*, pages 138–142, Austin, Texas.

Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding Abuse: A Typology of Abusive Language Detection Subtasks. In *Proceedings of the First Workshop on Abusive Langauge Online*.

Zeerak Waseem and Dirk Hovy. 2016a. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL student research workshop*, pages 88–93, San Diego, California.

Zeerak Waseem and Dirk Hovy. 2016b. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL student research workshop*, pages 88–93.

Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017a. Ex machina: Personal attacks seen at scale. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1391–1399. International World Wide Web Conferences Steering Committee.

Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017b. Ex machina: Personal attacks seen at scale. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1391–1399. International World Wide Web Conferences Steering Committee.

Jun-Ming Xu, Kwang-Sung Jun, Xiaojin Zhu, and Amy Bellmore. 2012. Learning from bullying traces in social media. In *Proceedings of the 2012 conference of the North American chapter of the association for computational linguistics: Human language technologies*, pages 656–666. Association for Computational Linguistics.

Justine Zhang, Jonathan Chang, Cristian Danescu-Niculescu-Mizil, Lucas Dixon, Yiqing Hua, Dario Taraborelli, and Nithum Thain. 2018a. Conversations gone awry: Detecting early signs of conversational failure. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1350–1361.

Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018b. Detecting Hate Speech on Twitter Using a Convolution-GRU Based Deep Neural Network. In *Lecture Notes in Computer Science*. Springer Verlag.