'Who built this crap?' Developing a Software Engineering Domain Specific Toxicity Detector

Jaydeb Sarker
Wayne State University
Detroit, USA
jaydebsarker@wayne.edu

ABSTRACT

Since toxicity during developers' interactions in open source software (OSS) projects show negative impacts on developers' relation, a toxicity detector for the Software Engineering (SE) domain is needed. However, prior studies found that contemporary toxicity detection tools performed poorly with the SE texts. To address this challenge, I have developed *ToxiCR*, a SE-specific toxicity detector that is evaluated with manually labeled 19,571 code review comments. I evaluate *ToxiCR* with different combinations of ten supervised learning models, five text vectorizers, and eight preprocessing techniques (two of them are SE domain-specific). After applying all possible combinations, I have found that *ToxiCR* significantly outperformed existing toxicity classifiers with accuracy of 95.8% and an $F1$ score of 88.9%.

KEYWORDS

developers interaction, toxicity, deep learning, NLP

ACM Reference Format:
Jaydeb Sarker. 2022. 'Who built this crap?' Developing a Software Engineering Domain Specific Toxicity Detector. In *37th IEEE/ACM International Conference on Automated Software Engineering (ASE '22), October 10–14, 2022, Rochester, MI, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3551349.3559508

1 PROBLEM AND MOTIVATION

Toxicity is a severe threat to online communities to maintain healthy relations. To curb the toxicity from online textual communication, Google's Conversational AI team developed Perspective API [1]. Though the OSS community is more professional than online communities, recent studies found the existence of toxicity during developers' communication [15, 20, 23]. Open source toxicity is a barrier for minority groups and newcomers. For example, due to getting toxic comments, a newcomer may leave the project immediately. Moreover, toxicity may have a negative correlation to promote diversity and inclusion in OSS communities. Studies also suggest that it takes a longer time to complete the project due to toxicity during developers' interactions [6]. Hence, identification of toxicity is crucial for OSS communities but it is highly time consuming for large scale OSS communities (i.e., Open Stack, Mozilla). Therefore, automatic identification of toxicity may help the OSS community to mitigate the toxicity from this domain. Though researchers proposed tools to detect toxicity using Natural Language Processing (NLP) and Machine Learning (ML) techniques, those failed to detect toxicity in the SE domain [20]. Therefore, Raman et al. encountered the challenge by developing a toxicity detector ('STRUDEL tool') that was trained with only 654 SE labeled texts [18]. Further studies found that the tool performed poorly on the SE texts [15, 17, 20].

Hence, it is necessary to develop a reliable toxicity detector for the SE domain to combat toxicity from OSS communities. To achieve this goal, I present *ToxiCR*, a supervised learning-based toxicity detector for the SE domain. *ToxiCR* is trained and tested using a large scale dataset of 19,571 code review comments. During our ten-fold cross validation based evaluations, the best performing model achieved 88.9% an $F1_1$ score of and 95.8% accuracy. We have made our dataset, tool, and evaluation results publicly available on Github at: https://github.com/WSU-SEAL/ToxiCR.

2 RELATED WORKS

Prior studies reported that toxicity in OSS projects caused burnout among developers [18] and newcomers may leave the project [20]. To better understand the toxicity in the OSS development, Miller et al. conducted a qualitative study on 100 Github issue discussions [15]. During their study, they mentioned the necessity of a reliable toxicity detector for the SE domain [15]. Similar to toxicity, recent studies also investigated incivility [8], pushback [5], interpersonal conflicts [17], and destructive criticism [11] during code reviews. Since researchers in the NLP domain developed toxicity detectors to combat toxicity in online discussions [1, 9], the SE specific toxicity detector is hardly available. Raman et al. [18] are the first ones who developed the SE specific toxicity detector ('STRUDEL tool' hereinafter) with the combination of Perspective API [1] and Stanford Politeness Detector [3]. Sarker et al. reported that the contemporary toxicity detectors including STRUDEL tool performed poorly on the SE dataset [20].

3 APPROACH AND UNIQUENESS

Context and Rubric: The meaning of toxicity is a complex phenomena and prior works have different views to label a text as toxic. Jigsaw Conversational AI [1] team provided a definition of toxicity as "a text that makes a person to leave the discussion due to disrespectful, rude, or unreasonable comments." However, the OSS community is highly professional, it is needed to adopt the SE specific rubric to mark a text as toxic. Since prior studies lack a reliable definition of toxicity in the SE domain, I and my colleague went through 1000 code review texts and empirically developed

ASE '22, October 10–14, 2022, Rochester, MI, USA

ACM ISBN 978-1-4503-9475-8/22/10…$15.00
https://doi.org/10.1145/3551349.3559508

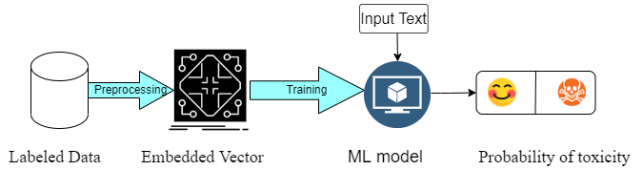

Figure 1: A simplified overview of ToxiCR

a rubric to identify a text in the SE domain as toxic or not [20]. According to our definition, when a comment in the SE domain contains any of them (i.e., swearing, cursing, insults, identity attack, threat, sexual attacking words), it would be labeled as toxic.

Dataset: Since code review is a process of direct interaction between developers, there is a high chance of toxic texts during code reviews. Therefore, following the rubric [20], I and my colleague manually labeled 6,533 code review comments from Android, Chromium, and Libreoffice projects. Since deep learning models need a large amount of data during training to achieve a high performance, I follow the exact process of our prior study and label 13,025 code reviews from open stack projects [21]. The inter-rater agreement (Cohen's Kappa) is 0.92 and 19.1% data (3,757 comments) are labeled as toxic among 19,571 labeled comments.

Text Preprocessing and Vectorizers: Figure 1 represents a simplified overview of *ToxiCR*. I have written the tool using Python language with the TensorFlow framework. Since code review comments contain URLs, technical jargons, code snippets, we have applied five mandatory preprocessing and three optional preprocessing steps. Five mandatory preprocessing includes URL removal, Contraction expansion, Symbol removal, Repetition elimination, and Adversarial pattern identification. These five steps help to remove the unnecessary features. I empirically evaluated the impact of remaining three optional preprocessing which includes Identifier splitting, Programming Keywords Removal, and Count profane words. I have used five different vectorizers with *ToxiCR*. The vectorizers include TfIdf, three context-free (i.e., Word2Vec [14], GloVe [16], fastText [2]), and context-based Bidirectional Encoder Representations from Transformers BERT [4] pretrained vectorizer. However, I evaluated TfIdf with five classical and ensemble (CLE) models, three context-free with Deep Neural Networks (DNN) models, and BERT uses its pretrained context-based vectorizer.

Machine Learning Models: I have used five CLE models including Decision Tree, Logistic Regression, Support-Vector Machine, Random Forest (RF), and Gradient-Boosted Trees from scikit-learn in python. Since I have evaluated *ToxiCR* with a large-scale dataset, I choose four DNN models. The DNN includes Deep Pyramid CNN (DPCNN) [13], Long Short Term Memory (LSTM) [12], BiLSTM [10], and Gated Recurrent Unit (GRU) [7]. Recent years, BERT [4] outperformed other state-of-the-art models in text classification. Each of the DNN models has three basic layers. After preprocessing the input text, the embedding layer maps the input text to the embedded matrix. The second layer is the Hidden State Layer that takes the input as an embedded matrix and captures the high level semantics of the words. This layer has four different blocks of DNNs (i.g., DPCNN, LSTM, GRU). A user can choose any model during training time. Finally, we have a classification layer to project output. Similar to the DNN model, I used the BERT-base pretrained vectorizer in the BERT model. I use a dropout layer to prevent overfitting with the BERT model. Further, I choose 'binary cross entropy' as loss function for DNNs and BERT models. I use the 'sigmoid' activation function with DNNs and 'linear' activation with BERT .

Table 1: Performance of the models for toxic (1) class

Models	P_1	R_1	$F1_1$	Acc
STRUDEL tool (retrain) [22]	0.85	0.86	0.85	0.94
RF + profanity	**0.917**	0.845	0.879	0.955
GRU +(profane, id)	0.897	0.856	0.876	0.954
BERT + keyword	0.907	**0.874**	**0.889**	**0.958**

4 RESULTS AND CONTRIBUTIONS

Model Training: During the experiment of each model, I have done an extensive evaluation using five times 10-fold cross validation and taken the mean of each evaluation metrics. I measure precision, recall, and F1 score for both toxic (1) and non-toxic (0) classes where we prioritize the F1-score of toxic class ($F1_1$) most. Moreover, I use stratified sampling to split the data and make sure the similar ratios between two classes. To customize the hyperparameters in DNN and BERT models, I split the data into 8:1:1 ratios for training, validation, and testing. In each epoch of the training, the EarlyStopping function with minimum value loss prevents the model from overfitting by using the validation set. The threshold value to being toxic is set to >= 0.5

Results: I have done three steps of evaluation. To get the baseline performances, I evaluated four state-of-the-art baseline models (i.e., Perspective API [20], pretrained STRUDEL tool [20], DPCNN [20], and retrained STRUDEL tool [22]) with our dataset. Among all state of the art models, the retrained STRUDEL tool achieved the best performance with 0.85 $F1_1$ score. During evaluation of our ten models, I have not applied the three optional preprocessing initially and RF outperformed all CLE models with 0.859 $F1_1$ score. On the other hand, the GRU achieved 0.875 $F1_1$ score with GloVe embedding and secured top position among all DNN models. Without the optional preprocessing, BERT achieved the best $F1_1$ score (0.887) and accuracy 95.7%.

To understand the impact of optional preprocessing, I have applied different combinations of the three optional preprocessings. Table 1 represents the results of the top three models performance for toxic class after applying optional preprocessing where they outperformed the baseline STRUDEL model. RF boosts performance with profane count preprocessing ($F1_1$ score is 0.879) that outperforms the best GRU model. However, DNN and BERT models do not improve a lot after applying the optional preprocessings. The best model for ToxiCR is BERT with keyword removal preprocessing that achieved 0.889 $F1_1$ score and 95.8% accuracy. I believe that *ToxiCR* will be helpful to prevent toxicity for the OSS communities.

Contributions: I have contributed the following in this study: i) developed a rubric for the SE domain to label toxicity and labeled 19,571 code review comments, ii) developed publicly available SE domain specific toxicity detector [19], iii) empirical evaluation of ten ML models and three optional preprocessings, and iv) finding the best possible combination.

REFERENCES

[1] Conversation AI. [n.d.]. What if technology could help improve conversations online? https://www.perspectiveapi.com/

[2] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching Word Vectors with Subword Information. *arXiv preprint arXiv:1607.04606* (2016).

[3] Cristian Danescu-Niculescu-Mizil, Moritz Sudhof, Dan Jurafsky, Jure Leskovec, and Christopher Potts. 2013. A computational approach to politeness with application to social factors. *arXiv preprint arXiv:1306.6078* (2013).

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. (June 2019), 4171–4186. https://doi.org/10.18653/v1/N19-1423

[5] Carolyn D Egelman, Emerson Murphy-Hill, Elizabeth Kammer, Margaret Morrow Hodges, Collin Green, Ciera Jaspan, and James Lin. 2020. Predicting developers' negative feelings about code review. In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*. IEEE, 174–185.

[6] Ikram El Asri, Noureddine Kerzazi, Gias Uddin, Foutse Khomh, and MA Janati Idrissi. 2019. An empirical study of sentiments in code reviews. *Information and Software Technology* 114 (2019), 37–54.

[7] Nelly Elsayed, Anthony S Maida, and Magdy Bayoumi. 2019. Deep Gated Recurrent and Convolutional Network Hybrid Model for Univariate Time Series Classification. *International Journal of Advanced Computer Science and Applications* 10, 5 (2019).

[8] Isabella Ferreira, Jinghui Cheng, and Bram Adams. 2021. The" Shut the f** k up" Phenomenon: Characterizing Incivility in Open Source Code Review Discussions. *Proceedings of the ACM on Human-Computer Interaction* 5, CSCW2 (2021), 1–35.

[9] Spiros V Georgakopoulos, Sotiris K Tasoulis, Aristidis G Vrahatis, and Vassilis P Plagianakos. 2018. Convolutional neural networks for toxic comment classification. In *Proceedings of the 10th Hellenic Conference on Artificial Intelligence*. 1–6.

[10] Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural networks* 18, 5-6 (2005), 602–610.

[11] Sanuri Dananja Gunawardena, Peter Devine, Isabelle Beaumont, Lola Garden, Emerson Rex Murphy-Hill, and Kelly Blincoe. 2022. Destructive Criticism in Software Code Review Impacts Inclusion. (2022).

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

[13] Rie Johnson and Tong Zhang. 2017. Deep pyramid convolutional neural networks for text categorization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 562–570.

[14] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[15] Courtney Miller, Sophie Cohen, Daniel Klug, Bogdan Vasilescu, and Christian Kästner. 2022. "Did You Miss My Comment or What?" Understanding Toxicity in Open Source Discussions. In *In 44th International Conference on Software Engineering (ICSE'22)*.

[16] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1532–1543.

[17] Huilian Sophie Qiu, Bogdan Vasilescu, Christian Kästner, Carolyn Egelman, Ciera Jaspan, and Emerson Murphy-Hill. 2022. Detecting Interpersonal Conflict in Issues and Code Review: Cross Pollinating Open-and Closed-Source Approaches. In *2022 IEEE/ACM 44th International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*. IEEE, 41–55.

[18] Naveen Raman, Minxuan Cao, Yulia Tsvetkov, Christian Kästner, and Bogdan Vasilescu. 2020. Stress and burnout in open source: Toward finding, understanding, and mitigating unhealthy interactions. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: New Ideas and Emerging Results*. 57–60.

[19] Jaydeb Sarkar, Asif Turzo, Ming Dong, and Amiangshu Bosu. 2022. ToxiCR: Replication package. Github. https://github.com/WSU-SEAL/ToxiCR.

[20] Jaydeb Sarker, Asif Kamal Turzo, and Amiangshu Bosu. 2020. A Benchmark Study of the Contemporary Toxicity Detectors on Software Engineering Interactions. In *2020 27th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 218–227.

[21] Jaydeb Sarker, Asif Kamal Turzo, Ming Dong, and Amiangshu Bosu. 2022. Automated Identification of Toxic Code Reviews Using ToxiCR. *arXiv preprint arXiv:2202.13056* (2022).

[22] Jaydeb Sarker, Asif Kamal Turzo, Ming Dong, and Amiangshu Bosu. 2022. WSU SEAL implementation of the STRUDEL Toxicity detector. https://github.com/WSU-SEAL/toxicity-detector/tree/master/WSU_SEAL.

[23] Megan Squire and Rebecca Gazda. 2015. FLOSS as a Source for Profanity and Insults: Collecting the Data. In *2015 48th Hawaii International Conference on System Sciences*. IEEE, 5290–5298.