Identification and Mitigation of Toxic Communications Among Open Source Software Developers

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

ABSTRACT

Toxic and unhealthy conversations during the developer's communication may reduce the professional harmony and productivity of Free and Open Source Software (FOSS) projects. For example, toxic code review comments may raise pushback from an author to complete suggested changes. A toxic communication with another person may hamper future communication and collaboration. Research also suggests that toxicity disproportionately impacts newcomers, women, and other participants from marginalized groups. Therefore, toxicity is a barrier to promote diversity, equity, and inclusion. Since the occurrence of toxic communications is not uncommon among FOSS communities and such communications may have serious repercussions, the primary objective of my proposed dissertation is *to automatically identify and mitigate toxicity during developers' textual interactions*. On this goal, I aim to: i) build an automated toxicity detector for Software Engineering (SE) domain, ii) identify the notion of toxicity across demographics, and iii) analyze the impacts of toxicity on the outcomes of Open Source Software (OSS) projects.

KEYWORDS

toxicity, developers' interactions, NLP, deep learning

ACM Reference Format:
Jaydeb Sarker. 2022. Identification and Mitigation of Toxic Communications Among Open Source Software Developers. In *37th IEEE/ACM International Conference on Automated Software Engineering (ASE '22), October 10–14, 2022, Rochester, MI, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3551349.3559570

1 INTRODUCTION

Prior research found evidence of toxic communications among various OSS communities [14, 19, 30, 40]. Although toxic communications are less frequent among OSS communities than in online platforms such as social media and opinion forums, it may have serious repercussions on the productivity or even survival of an OSS project. For example, being demotivated and frustrated with toxic communications, such as insults, threats, and sexual attacks from their peers, developers may leave an OSS project [3, 8]. Moreover, such communications often disproportionately impact newcomers, women, and other marginalized groups [17, 21, 41]. Therefore, toxicity is also a barrier to promote diversity, equity, and inclusion. Although proactive identification and mitigation of toxic communications is crucial, it is a challenge for large-scale OSS communities with thousands of members (e.g., Mozilla, Debian, and OpenStack) to manually check all communications for toxicity. Therefore, an automated identification and mitigation mechanism can significantly help these communities combating toxicity. In this context, my proposed dissertation aims to *identify and mitigate toxic communications among open source software developers*. I aim to achieve this goal based on three studies.

(Study 1) Develop a customized toxicity detector for the SE domain

Motivation: Despite the existence of many off-the-shelf toxicity detectors, those perform poorly in SE texts [37]. However, such performance degradation is hardly surprising, since prior research on SE domain-specific sentiment analysis tools [1, 23, 28] established needs for SE domain-specific natural language processing (NLP) tools. Although Raman et al. [35] developed the first SE domain-specific toxicity detector, it performed poorly on later studies [27, 34, 37]. To better understand toxicity and its impacts, a reliable toxicity detector for the SE domain, therefore is a need.

(Study 2) Develop a better understanding of the notion of toxicity among OSS developers representing various demographic groups

Motivation: According to Miller et al., toxicity among OSS communitis is a big umbrella of several antisocial behaviors, such as of trolling, flaming, hate speech, harassment, and cyberbullying [27]. In my prior study, we also introduced a definition of toxicity for the SE domain [37]. However, the notion of toxicity may differ based on on multitude of different factors, such as culture, ethnicity, country of origin, language, and relationship between the participants. A further study may help the developers: i) whether the phenomenon of toxicity differs different demographics, ii) to make conversation with another developer according to demographics. This study will help to analyze toxicity according to demographics and experience, and development of context aware toxicity detectors.

(Study 3) Analyze the impacts of toxicity on the outcomes of OSS projects

Motivation: I will analyze the impact of toxic comments on the open-source development community. This analysis may help the project management i) to take the required steps to mitigate the toxic interactions and ii) to improve the developers' relations. Moreover, this analysis may help the developers to get an overall idea of the negative impact of toxicity and inspire them not to use toxic

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

ACM ISBN 978-1-4503-9475-8/22/10…$15.00
https://doi.org/10.1145/3551349.3559570

conversations with their peers. This study aims to investigate: i) how toxicity impacts the diversity inclusion in SE practices, ii) how toxicity makes barriers for newcomers on-boarding, and iii) a better understanding of participants' characteristics and contextual factors that instigates toxic communications.

The remainder of the paper is organized as the following. Section 2 provides a brief overview of the research context and prior related works. Finally, Section 3 describes the three proposed studies and Section 4 concludes the paper respectively.

2 BACKGROUND

Toxicity: Due to influxes of toxic communications among various online platforms [24, 26], researchers have focused on automated toxicity identification techniques. The Jigsaw Conversational AI team defined toxicity as rude, disrespectful comments, which make a person leave the conversation [2]. Other researchers have also characterized various forms of toxicity, which include hate speech [9, 39], microaggressions [4], insults [29], and cyberbullying [36, 43]. However, toxic communications among OSS developers differ from those seen in other online platforms, as OSS communities are professional workplaces. Yet, most of the OSS communication channels such a mailing list [15], issue discussions [27, 35], and code reviews [17, 37] show occurrences of toxicity.

SE domain specific NLP tools: Most of the prior works on developing SE domain specific natural language tools focused on automated identification sentiments and opinions [1, 5, 20, 31]. Recently, several works have focused on characterizing SE domain specific toxicities and their identifications. To better understand the toxicity in FOSS development, Miller et al. have done a qualitative study on 100 GitHub toxic issue discussions [27] and they found that the most common types of toxicity are insults, entitled, and arrogant. Their study suggests that toxicity in GitHub differs from other domains like Wikipedia or Reddit [27]. After analyzing 1,545 Linux Kernel Mailing Lists, [15] found incivility (an antisocial behavior) where two-thirds of them contained frustration, name-calling, and impatience. To alleviate toxicity from the SE domain, Raman et al. [35] developed a toxicity detector for the SE domain with the combination of Perspective API [2] and Stanford Politeness detector [7]. Several studies found the significant low performance of this tool [35] in SE dataset as it was trained with only 654 labeled Github issue comments [27, 34, 37]. Further, Cheriyan et al. developed an offensive language detector that considers only swearing and profanity [6]. To prevent interpersonal conflict in issues and code reviews, Qiu et al. developed an automatic detection classifier [34] using 'toxicity classifier' [35]) and 'pushback classifier' [11]. Although two recent studies have attempted to develop toxicity detectors, performance of those tools are questionable [37].

Impacts of toxicity: In addition to a reliable toxicity detector, the developers should have a better understanding about the impacts of toxicity on their project outcomes. For example, the negative or controversial comments during code review took more time to complete the project than the neutral or positive reviews [12]. Another work found that destructive criticism during code reviews demotivated the women to continue their work on open-source projects [17]. After getting toxic comments on Github issue discussions, project maintainers sometimes took steps (i.e., locking issues, deleting issues, blocking users) [27] but this work is limited to only 100 samples.

3 RESEARCH APPROACH

The following subsections provide overviews of the three studies for my proposed dissertation.

3.1 Study 1: Develop a customized toxicity detector for the SE domain

Challenges: Since state-of-the-art toxicity detectors (i.e., Perspective API [2]) and one SE related toxicity detector [35] do not perform well in the SE text [37], we need to train the model with a large SE related labeled text. Manual labelling of toxic comments in the SE domain is a costly and time consuming process because toxic comments are rare [35, 37]. In addition, we need a SE domain specific rubric to label the text as some of the SE domain texts represent different meanings than general domain [35]. In my prior benchmark study [37], we have found off-the-shelf toxicity classifiers misclassified some common technical phrases (i.e., execute, kill, trash, junk, dirty, dead process) as toxic.

Research Methodology: To encounter the challenges of building the SE-specific toxicity detector, I have developed *ToxiCR*, a SE specific supervised-learning based toxicity detector for code review comments [38]. Before developing the ToxiCR, I have done a benchmark study where we empirically defined a rubric to label a text toxic or non-toxic in the SE domain [37]. During defining the rubric for detecting toxic interaction in the SE domain, we have selected Code Review (Android, Chromium OS, and LibreOffice projects) and Gitter chat messages (Instant Relay Chat) as previous studies suggested the presence of toxicity [35, 40] in those communities. Using a Python script with Gerrit's REST API, we mined all publicly available code reviews and excluded the bot comments like Paul et al. [30]. We have used a stratified sampling strategy with Google's Perspective API (PPA) [2] to select the probable toxic comments from code reviews [37]. To define the rubric, we went over 1000 SE texts and empirically developed a rubric to identify what is toxic or non-toxic with respect to the SE domain. Finally, we prepared a set of rules to label a text as toxic or not for our SE dataset. According to our definition, a text is considered to be toxic in SE domain if it contains any of the followings: offensive name calling, insults, curse or swearing, flirtations, sexual references, personal attacks, threats [37]. Further, I and another author from my prior benchmark study manually labeled 6,533 code review comments and 4,140 Gitter messages using the rubric [37]. We found that off-the-self toxicity detectors failed to detect toxicity in the SE domain with a low F1 score and accuracy. The SE domain texts contain technical words, code snippets, keywords which intensify the classifiers to misclassify the texts. These results and recent studies [27, 34] highly motivated us to develop a reliable toxicity detector in the SE domain.

For developing ToxiCR, we have manually labeled a large scale dataset which is shown in table 1. In my benchmark study, we have 6,533 labeled code review texts and additionally two of us labeled 13,038 code review comments from open stack projects. The inter-rater agreement (Cohen's Kappa) score is 0.92. Overall,

Table 1: An overview of manual labeled dataset

Dataset	# total texts	# toxic	# non-toxic
Code review-I [37]	6,533	1,310	5,223
Code review-II [38]	13,038	2,447	10,591
Code review (combined)	19,571	3,757	15,819

Table 2: Mean performances of the top three models

Model + Preprocessing	P_1	R_1	$F1_1$	Acc
GRU + (profane_count, id_split)	0.897	0.856	0.876	0.954
RF + profane_count	**0.917**	0.845	0.879	0.955
BERT + keyword-removal	0.907	**0.874**	**0.889**	**0.958**

we have used 19,571 labeled code review comments to build our model where 19.2% samples are toxic (3,757).

We have followed the state-of-the art supervised-learning and NLP based techniques to build the toxicity detector. ToxiCR is a combination of machine learning models, preprocessing techniques, and vectorizers. We empirically evaluated and added ten supervised learning models with ToxiCR which includes five conventional machine learning (CLE) algorithms (Decision Tree, Logistic Regression, Support Vector Machine, Random Forest (RF), Gradient-Boosted Decision Tree), four deep neural networks (DNN) (Long Short Term Memory (LSTM) [18] , Bidirectional LSTM [16], Gated Recurrent Unit [13], Deep Pyramid CNN [22]), and one Transformers (Bidirectional Encoder Representations from Transformers (BERT) [10]). We have done five mandatory preprocessing of the code review texts (URL removal, Contraction expansion, Symbol removal, Repetition elimination, Adversarial pattern identification) and three optional preprocessings (Identifier splitting (id_split), Programming Keywords Removal, profane-count). For vectorizing the text to fit into machine learning models, we use five different word vectorizers. We set and tuned the model parameters according to the binary classification strategies.

During the training of CLE models, we randomly split the dataset and have done 10-fold cross validation where 9 groups are used for training and another one is used for testing in each splitting [38]. DNN and BERT models have a probability to overfit. To find the best fit model, we split the dataset into train: test: validation as 8:1:1 in each fold of 10-fold cross validation. During each fold of training with 80% of the data, the trained system are tested with 10% of validation data. This validation set helps each model to find its best hyperparameters at the end of the training. To do that, we also use the EarlyStopping function that monitors minimum val loss from Keras library.

Research Progress: During the evaluation of ToxiCR [38], we have done 10-fold cross validations 5 times and computed the mean of all metrics (precision, recall, F1 for both class, and accuracy). Without applying optional preprocessing, we have found that RF provides the best $F1_1$ score (0.859) among all CLE models which was also higher than DPCNN and LSTM models. GRU with GloVe [32] embedding provided an $F1_1$ score as 0.875 which is also statistically significant. Overall, the best model is BERT which achieved $F1_1$ score (0.887) and accuracy (0.957). Further, we also investigate whether three of the optional preprocessings or their combinations improve the performance of any model. To understand the results more clear after applying optional preprocessings, I put the results metrics for toxic class (Precision (P_1), Recall (R_1), and $F1_1$) and overall accuracy of the top three models on table 2. Moreover, I have added which types of preprocessing techniques provided the top performances of those models. We have found that CLE models boost their performance with optional preprocessings than DNN and transformers. Finally, we got a combination with BERT and keyword-removal preprocessing which performed the best with 0.889 $F1_1$ score and 0.958 accuracy.

My paper of these findings: 'benchmark study' has been accepted in 27th Asia-Pacific Software Engineering Conference (APSEC) [37] and 'ToxiCR' [38] is currently under review (major revision submitted) to ACM Transactions on Software Engineering and Methodology (TOSEM).

Contributions The contributions of this study include: i) a rubric to identify toxic texts from software developer communications; ii) a large scale labeled dataset of 19, 571 texts; and iii) a reliable toxicity detector for SE domain, which is publicly available at: https://github.com/WSU-SEAL/ToxiCR.

3.2 Study 2: Develop a better understanding of the notion of toxicity among OSS developers representing various demographic groups

Problem Definition: Unlike other text classification problems (i.e., spam, online abuse), toxicity is a highly subjective [25]. For example, in the OSS community females got more negative comments than men during code reviews [31]. Moreover, newcomers may get more demotivated than experienced ones due to getting toxic comments from their peers. To understand how people from different demographics perceive toxicity online, Kumar et al. conducted a survey from 17,280 participants across the United States [25]. Surprisingly, they found a diverse labeling from the raters and no single demographic factors can define the construction of toxicity. However, there are two limitations of that study: i) they conducted their survey only inside the United States which does not cover the whole region of the world, and ii) they provided the survey questions that were not related to the SE conversations. In the SE domain, no such study to understand toxicity across demographic factors and experience has been done yet. To fill these research gaps in understanding the toxicity across the demographics in the SE domain, I set my first research question as:

RQ1: Does the phenomenon of toxicity differ across the demographics in the OSS community?

According to the previous research [25], state-of-the-art classifiers failed to detect toxicity when the data is labeled according to demographics and experiences. Hence, I set my next goal as preventing toxicity from different demographic perspectives and personal experiences. So, I set my second research question:

RQ2: How can we mitigate toxicity in demographic perspectives?

Challenges: First, it is crucial to find out a diverse developers community to conduct a survey. Second, we need to compile a set of questionnaires that can be used to understand the perspectives

of toxicity across demographics. Finally, to identify toxicity according to demographics, developers need a fine-grained generalized classifier that can not not only detect toxicity but also detect the sub-versions (i.e., insult, threat) of toxicity. Therefore, to build a fine-grained toxicity classifier, we need a large-scale labeled dataset that can provide diverse versions of toxicity.

Research Methodology: To get a better understanding of the notion of toxicity among OSS developers from different demographic (i.g., Nationality, Culture, Gender, LGBTQ+, Religion and Race) groups, I will conceptually replicate the study of Kumar et al. [25]. Conceptual replication stands for using the different research methodologies with the same set of data/ questions [42]. In this work, I will not use the exact same data and demographics from [25], but I will do partial conceptual replication. First, I will prepare a survey for OSS developers across the whole world to understand how they perceive the OSS interactions as toxic. The survey includes 15 comments from the SE interactions (i.e., code review) to label and the information of demographic characteristics from the participants. Second, a participant can label each comment from a range of 5 toxic (non, slightly, moderately, very, and extreme) categories. Finally, participants will label each comment into subcategories of toxicity (insult, threat, identity attack, sexually explicit, flirtation, and profanity). During the subcategory labeling, participants can label each comment into multiple subcategories.

From my first part of this study, I will consider the comments that will be rated moderately toxic or higher as toxic text. Further, I will do a statistical analysis to find out how demographics (gender, age, religion, LGBTQ+, race) and experiences influence to perceive a text as toxic. Second part of this study will help to answer the RQ2. I aim to build a fine-grained toxicity detector that will detect the different sub-versions of toxicity for example insulting, identity attack, insulting, threat etc. After getting the response from the developers, I may gather a large datasets from different perspectives of toxicity that will be labeled by the participants. If I will need more dataset, I will label it from the perception of the participants labeling. Finally, I will apply state-of-the art NLP and machine learning techniques to build a fine grained multi-class tool to detect toxicity in the SE domain.

Research Progress: I am working on setting the questionnaires for the survey. After getting permission from the IRB of my university, I will send this survey among 6000 developers with different demographics across the world.

Expected contributions: Expected contributions from this study include: i) a better understanding how a developers' demographics influence toxicity; ii) a better understanding of subcategories of toxicity; and iii) a fine grained tool to detect different perspective of toxicity.

3.3 Study 3: Analyze the impacts of toxicity on the outcomes of OSS projects

Problem Definition and Challenges: To understand the impact of toxicity in the open-source community, we will need a large scale study among the popular projects. Recently, Miller et al. found some after effects of using toxic comments in OSS discussions, for example closing, locking issues, deleting or hiding the comments and blocking users [27] but their study is limited to only 100 Github issue discussions. [17] found that negative code review demotivates the female developers and [33] studied that newcomers got frustrated due to getting impolite comments in open source projects. To get a broader idea of the impact of toxicity on the outcomes of OSS projects, I set my first research question as:

RQ1: How does toxicity impacts measurable outcomes of an OSS project?

Further, I will analyze what characteristics instigate beyond the toxic comments in open-source projects. Hence, I set my second research question as:

RQ2: Which factors commonly work as triggers of toxicity among OSS communities?

Research Methodology: To do a large-scale empirical study on understanding the impact of toxicity on the outcomes of OSS projects, I have curated a large-scale dataset from GhTorrent and Gerrit code reviews. I have selected ten popular projects (i.g., go, libreoffice, ovirt) and picked the inline and review comments from Gerrit code reviews. Moreover, I have selected 1000 popular GhTorrent projects for my empirical analysis. Since I aim to analyze only the details of toxic comments, I have started my work to collect the details of the projects where toxic comments occurred. To complete this task, I plan to use the ToxiCR classifier [38] to identify the toxic comments from the Gerrit and GhTorrent dataset.

For answering the RQ1, we have resolved the gender for all contributors on those 10 Gerrit projects. Moreover, I will find out who are new contributors to those projects. Further, I and my colleague will manually investigate those selected toxic comments (classified by ToxiCR) and label them into different sub-categories of anti-social behaviors (i.g., insults, sexual attack, threat). I will summarize the after impacts of those projects outcomes where toxicity occurs. After analyzing the entire conversation where toxicity occurs, I will apply mixed methods to find out which characteristics trigger toxicity in open-source projects to answer the RQ2. Finally, I will propose some necessary steps from the results of mixed method analysis on how to mitigate the toxic interactions in FOSS projects.

Research Progress: I have generated toxicity scores on ten projects of gerrit for inline and review comments using ToxiCR tool.

Expected contributions: Expected contributions include: i) empirical evidence regarding the impact of toxicity on the outcomes of OSS projects; ii) a better understanding of how toxicity impacts various demographic groups; and iii) a set of recommendations to prevent as well as mitigate toxicity from the OSS communities.

4 CONCLUSION

In my doctoral dissertation, I focus on providing a way to the OSS community to make healthy communication by *identification and mitigation of toxicity during their interactions*. To achieve this goal, I have planned to complete three studies. These three studies will help the open-source software developers i) to use an automatic detection of toxicity tool in the SE domain, ii) understand the notion of toxicity according to demographic factors, and iii) provide a better understanding of the impact of toxicity in OSS projects.

REFERENCES

[1] Toufique Ahmed, Amiangshu Bosu, Anindya Iqbal, and Shahram Rahimi. 2017. SentiCR: a customized sentiment analysis tool for code review interactions. In *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 106–111.

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

[3] Anonymous. 2014. Leaving Toxic Open Source Communities. https://modelviewculture.com/pieces/leaving-toxic-open-source-communities

[4] Luke Breitfeller, Emily Ahn, David Jurgens, and Yulia Tsvetkov. 2019. Finding microaggressions in the wild: A case for locating elusive phenomena in social media posts. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*. 1664–1674.

[5] Fabio Calefato, Filippo Lanubile, Federico Maiorano, and Nicole Novielli. 2018. Sentiment polarity detection for software development. *Empirical Software Engineering* 23, 3 (2018), 1352–1382.

[6] Jithin Cheriyan, Bastin Tony Roy Savarimuthu, and Stephen Cranefield. 2021. Towards offensive language detection and reduction in four Software Engineering communities. In *Evaluation and Assessment in Software Engineering*. 254–259.

[7] 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).

[8] R Van Wendel De Joode. 2004. Managing conflicts in open source communities. *Electronic Markets* 14, 2 (2004), 104–113.

[9] Fabio Del Vigna12, Andrea Cimino23, Felice Dell'Orletta, Marinella Petrocchi, and Maurizio Tesconi. 2017. Hate me, hate me not: Hate speech detection on facebook. In *Proceedings of the First Italian Conference on Cybersecurity (ITASEC17)*. 86–95.

[10] 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

[11] 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.

[12] 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.

[13] 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).

[14] Samir Faci. 2020. The Toxicity Of Open Source. https://www.esamir.com/20/12/23/the-toxicity-of-open-source/

[15] 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.

[16] 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.

[17] 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).

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

[19] Nasif Imtiaz, Justin Middleton, Joymallya Chakraborty, Neill Robson, Gina Bai, and Emerson Murphy-Hill. 2019. Investigating the effects of gender bias on GitHub. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 700–711.

[20] Md Rakibul Islam and Minhaz F Zibran. 2018. SentiStrength-SE: Exploiting domain specificity for improved sentiment analysis in software engineering text. *Journal of Systems and Software* 145 (2018), 125–146.

[21] Carlos Jensen, Scott King, and Victor Kuechler. 2011. Joining free/open source software communities: An analysis of newbies' first interactions on project mailing lists. In *2011 44th Hawaii international conference on system sciences*. IEEE, 1–10.

[22] 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.

[23] Robbert Jongeling, Proshanta Sarkar, Subhajit Datta, and Alexander Serebrenik. 2017. On negative results when using sentiment analysis tools for software engineering research. *Empirical Software Engineering* 22, 5 (2017), 2543–2584.

[24] Robin M Kowalski, Susan P Limber, and Patricia W Agatston. 2012. *Cyberbullying: Bullying in the digital age*. John Wiley & Sons.

[25] Deepak Kumar, Patrick Gage Kelley, Sunny Consolvo, Joshua Mason, Elie Bursztein, Zakir Durumeric, Kurt Thomas, and Michael Bailey. 2021. Designing Toxic Content Classification for a Diversity of Perspectives. In *Seventeenth Symposium on Usable Privacy and Security (SOUPS 2021)*. 299–318.

[26] Megan Lindsay, Jaime M Booth, Jill T Messing, and Jonel Thaller. 2016. Experiences of online harassment among emerging adults: Emotional reactions and the mediating role of fear. *Journal of interpersonal violence* 31, 19 (2016), 3174–3195.

[27] 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)*.

[28] Nicole Novielli, Daniela Girardi, and Filippo Lanubile. 2018. A benchmark study on sentiment analysis for software engineering research. In *2018 IEEE/ACM 15th International Conference on Mining Software Repositories (MSR)*. IEEE, 364–375.

[29] Sünje Paasch-Colberg, Christian Strippel, Joachim Trebbe, and Martin Emmer. 2021. From insult to hate speech: Mapping offensive language in German user comments on immigration. *Media and Communication* 9, 1 (2021), 171–180.

[30] Rajshakhar Paul, Amiangshu Bosu, and Kazi Zakia Sultana. 2019. Expressions of sentiments during code reviews: Male vs. female. In *2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 26–37.

[31] Rajshakhar Paul, Amiangshu Bosu, and Kazi Zakia Sultana. 2019. Expressions of Sentiments during Code Reviews: Male vs. Female. In *Proceedings of the 26th IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER '19)*. IEEE.

[32] 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.

[33] Huilian Sophie Qiu, Yucen Lily Li, Susmita Padala, Anita Sarma, and Bogdan Vasilescu. 2019. The signals that potential contributors look for when choosing open-source projects. *Proceedings of the ACM on Human-Computer Interaction* 3, CSCW (2019), 1–29.

[34] 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.

[35] 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.

[36] Kelly Reynolds, April Kontostathis, and Lynne Edwards. 2011. Using machine learning to detect cyberbullying. In *2011 10th International Conference on Machine learning and applications and workshops*, Vol. 2. IEEE, 241–244.

[37] 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.

[38] 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).

[39] Anna Schmidt and Michael Wiegand. 2019. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media, April 3, 2017, Valencia, Spain*. Association for Computational Linguistics, 1–10.

[40] 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.

[41] Igor Steinmacher and Marco Aurélio Gerosa. 2014. How to support newcomers onboarding to open source software projects. In *IFIP International Conference on Open Source Systems*. Springer, 199–201.

[42] Sayma Sultana and Amiangshu Bosu. 2021. Are Code Review Processes Influenced by the Genders of the Participants? *arXiv preprint arXiv:2108.07774* (2021).

[43] 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*. 656–666.