Stress and Burnout in Open Source: Toward Finding, Understanding, and Mitigating Unhealthy Interactions

Naveen Raman,[♡] Minxuan Cao,[♣] Yulia Tsvetkov,[♣] Christian Kästner,[♣] Bogdan Vasilescu[♣]
[♡]University of Maryland, College Park, USA [♣]Carnegie Mellon University, USA

ABSTRACT

Developers from open-source communities have reported high stress levels from frequent demands for features and bug fixes and from the sometimes aggressive tone of these demands. Toxic conversations may demotivate and burn out developers, creating challenges for sustaining open source. We outline a path toward finding, understanding, and possibly mitigating such unhealthy interactions. We take a first step toward finding them, by developing and demonstrating a measurement instrument (an SVM classifier tailored for software engineering) to detect toxic discussions in GitHub issues. We used our classifier to analyze trends over time and in different GitHub communities, finding that toxicity varies by community and that toxicity decreased between 2012 and 2018.

1 INTRODUCTION

Sustaining open-source software is an important and difficult challenge. On the one hand, open source has a critical role in our software infrastructure, affecting directly or indirectly almost every software product and facet of modern life. Some argue that open source provides just as important infrastructure as roads and bridges do for the economy, yet its importance, and our dependence on it, are often not recognized [9]. On the other hand, open-source software, as all software, needs to be maintained. Continuous effort is needed to fix bugs and vulnerabilities and to evolve the software to accommodate new requirements to stay relevant. How to sustain such effort, be it from volunteers or through explicit support from corporations, is an open, controversially discussed problem.

Open-source practitioners have been raising awareness of *stress* and *burnout*. Community members are openly worried about mental and physical well-being of contributors and about exploitation of volunteers, including self-exploitation with the vague promise of building a profile that could help them find a better job, as evidenced by many recent blog posts, talks, podcasts, even entire conferences [e.g., 5, 17, 25, 27, 37]. A common theme is that open-source maintainers feel overwhelmed by the number of requests they receive (*e.g.*, bug reports, support requests). In addition, the transparency on social coding websites like GitHub raises stakes [6] in that mistakes are visible and can affect a contributor's reputation.

Even more important, more than just volume of requests and

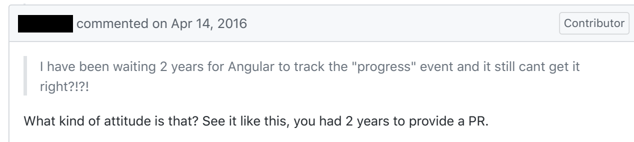

Figure 1: Excerpt from an issue discussion on GitHub.

high stakes, many maintainers complain about *the tone of these interactions*, sometimes formulated aggressively or from a position of entitlement, as in Figure 1. For developers, this can be draining, *e.g.*, "*GitHub notifications are a constant stream of negativity [...] Reading through these [...] can be mentally and emotionally exhausting*" [25]. Heightened levels of stress brought on by unhealthy interactions may make it harder for projects to attract, include, and retain a diverse talent pool. We argue that studying, understanding, and mitigating stress and burnout among open-source developers is an important and understudied research field. There are many important research questions, including: How prevalent is self-reported stress and burnout among open-source contributors? What are the causes of stress in open-source? How do they compare to other work environments? When is stress most damaging to open-source contributors and who is most at risk? What interventions are effective?

Our vision is to answer such questions empirically using mixed methods, relying heavily on public trace data. The advantages of such a computational approach are manifold. First, analyzing trace data avoids the recall errors and response biases typical of surveys and interviews. Second, it makes the ethical choice of avoiding to burden already potentially stressed individuals by asking them to recall or envision stressful interactions. Third, analyzing large, multidimensional samples offers statistical opportunities for modeling and hypothesis testing that are typically not present in smaller and simpler data sets from experiments, surveys, or interviews. Last but not least, operationalizing open-source stress factors using trace data paves the way to develop automated, non-invasive measures and models to help identify contributors that show signs of stress and projects at risk, as well as to design automated interventions.

In this paper, we take a first step towards realizing this vision, by developing and demonstrating a critical research instrument (a classifier) to detect toxic language in open-source issue discussions. Toxic language in open source can manifest in multiple ways, including hate speech and microaggressions found also elsewhere online (*e.g.*, Youtube), but also through open-source-specific displays of entitlement and urgency related to timing expectations as in Figure 1.

Our work builds on prior research on detecting toxic language—from hate speech to microaggressions—in the Natural Language Processing (NLP) community [3, 12, 32], making two main contributions. First, we show that toxic GitHub issue discussions (in English) can be identified using a combination of *pre-trained* detectors of negative sentiment, anger, impoliteness, and toxicity. Second,

ICSE-NIER'20, May 23–29, 2020, Seoul, Republic of Korea

ACM ISBN 978-1-4503-7126-1/20/05…$15.00
https://doi.org/10.1145/3377816.3381732

we show that classification accuracy can be further improved by domain adaptation, tailoring our detector to the context of software-engineering discussions. We demonstrate the potential of our classifier with three preliminary empirical studies. Our replication package is at https://github.com/CMUSTRUDEL/toxicity-detector.

2 RELATED WORK

We build on prior work that (a) has studied motivations of developers and users to see why conflict might arise and (b) has developed NLP tools to detect different forms of toxicity in different contexts.

Volunteering, motivations, and conflicts in open source. Researchers have extensively studied motivations of developers contributing to open source [e.g., 18, 24], revealing a multitude of intrinsic and extrinsic reasons, such as working on projects they enjoy or find useful. Despite increasing commercialization and professionalization, many contributors are volunteers [19, 38]. Yet, among the many reasons to contribute to open source, building one's professional reputation and signaling one's skills to potential employers are common ones [28].

At the same time, open source is broadly used in commercial projects, even for mission-critical components. Only a small number of users of an open-source project contribute to that project [19]. Given this asymmetry, high stakes, and the lack of a contractual relationship, users that demand changes from the project, be it additional features, specific changes (*e.g.*, perceived bugs or limitations), or better documentation, may be perceived as *entitled* [25, 26]. Within developer communities, there have been reports of insults and attacks [1, 21]. Beyond concerns for the maintainer's well being, toxic interactions are concerning for recuiting contributors [34].

Detecting Toxicity. The NLP community has achieved significant advances at detecting different forms of negativity and toxicity in text, *e.g.*, in movie reviews or social-media interactions, on which we build for our own toxicity detection instrument.

In the software-engineering community, *sentiment analysis* [30] is a popular such technique, used to analyze, among others, issue discussions, pull requests, email messages, and forum posts [e.g., 4, 16]. Similar approaches have been used to detect anger in issue reports [13]. Software engineering research has shown though that a sentiment-analysis classifier for software engineering tasks needs to be trained specifically on software engineering content [22], because traditional classifiers assign negative weights to many technical phrases such as "kill a process."

There is also related work on detecting toxicity in language, including hate speech, abuse, microagressions, and harassment [11, 36]. For example, *hate-speech detection* specifically looks for strong, toxic interactions [15], trained on comments in online forums [8, 31]. As for sentiment analysis, we expect that we will have to adjust existing classifiers for the software engineering context, where toxic interactions may be less direct, related to technical issues, or to timing.

3 DATA AND METHODS

At a high level, we manually labeled a sample of GitHub issue comments and trained a classifier to identify toxic comments, using features inspired by prior research on detecting toxic language in online communities. This section details the individual steps.

Table 1: Features used by our classifier.

Feature	Description
Length	Comment length, in characters
Frequency	TF-IDF weighted word frequencies
Politeness	As per Stanford's politeness detector [7]
Toxicity	As per Google's Perspective API (perspectiveapi.com)
Subjectivity, Polarity	As per the lexicon-based sentiment analysis Python library TextBlob (textblob.readthedocs.io)
Sentiment	As per the lexicon and rule-based sentiment analysis NLTK library [20] (VADER algorithm)
Anger	Number of anger words from the LIWC lexicon [35]

Data. With a few exceptions from blog posts, online discussions, and interviews [e.g., 5], no labeled data for toxic language in open source exists. We curated a dataset manually and incrementally. Toxic interactions seem to be rare but very stressful; given the low rate, random sampling seemed ineffective, so we identified two different strategies. First, we queried the GitHub API to identify issue threads that had been locked as "too heated". Among the 118,629 GitHub projects with any issues (according to our copy of GHTorrent [14]), we found 294 805 locked issues of which 654 where explicitly locked as "too heated" (providing a reason for locking is a very recent GitHub feature). Issue discussions locked as too heated often contain toxic behavior that was called out, *e.g.*, "*I'm locking the conversation. Inappropriate/unprofessional conduct will not be tolerated.*" We manually reviewed all the ones written in English, labeling their comments as either toxic or not (by extrapolating, we also labeled the issue as a whole as toxic, if at least one comment was toxic). Second, inspired by patterns found earlier, we searched through GHTorrent issue comments for *reactions*, by maintainers, containing the word 'attitude' (*e.g.*, Figure 1) and manually labeled them. In the end, using the two strategies we compiled a data set of 386 issue threads, 167 of which contain at least one toxic comment each, manually labelled.

After labelling, we split our data in two, half for training and half for testing. To increase the representativeness (our previous sampling was non-random) and the realism (toxic issues are relatively rare) of our training data, we further manually labelled 300 random issue threads, none of which were toxic, adding 225 of them (written in English, having at least one comment each) to the training set.

Classifier Features. The domain-specificity of toxicity in open-source suggests that a custom approach to classification is needed. Since we are limited by the relatively small amount of labelled data available for training, based on our review of the NLP literature we attempt to capture open-source toxicity using a combination of general pre-trained sentiment analysis, politeness, and abusive language detectors; for example, we use Google's pre-trained Perspective API for detecting "rude, disrespectful, or unreasonable comments" in non-software-specific online discussions (*e.g.*, Wikipedia). Our full set of features is described in Table 1.

Training. Our classification task is to assign a *toxic* or *non-toxic* label to a given issue comment (and by extension to the issue). To this end, we trained an SVM classifier. SVMs are often used to classify text [23], they tend to perform on par with other statistical classifiers and they outperform state-of-the-art neural network classifiers in low-resource training data scenarios like ours.

We used 10-fold nested cross validation to learn hyperparameters and evaluate the model [10]. Because of the imbalance in the training data, for each split, we adjusted the class weights, with a ratio r between non-toxic and toxic examples, where r is a hyperparameter. We grid searched over SVM hyperparameters $\gamma = \{1, 2, 2.5, 3\}$, $C = \{0.01, 0.05, 0.1, 0.5, 1, 10\}$; and $r = \{1, 1.5, 1.75, 2, 2.25\}$.

Tuning. A commonly recognized risk with NLP models is poor performance outside of the context where they have been trained [22]. For example, 'abort' and 'kill' have negative connotations in general English, but are mostly neutral in software engineering, *e.g.*, when referring to processes, leading to inaccurate predictions.

To alleviate this risk, we identified, using log odds with Dirichlet prior [29], words that are significantly overrepresented in software engineering language compared to general English, and replaced those words with a neutral filler word, so that the sentence structure would not be modified. Specifically, log odds with Dirichlet prior assumes words follow a Dirichlet distribution, and uses the distribution of software-engineering words along with the distribution of regular English words to estimate a confidence level for whether a word is software-engineering-specific; we use the typical $\alpha < 0.05$ cutoff. Our software engineering corpus comes from a random sample of 10K GitHub issues, and our generic English corpus comes from the Python library `wordfreq` [33], which uses seven corpora, including Wikipedia. For computational reasons, we apply this correction as a post-processing step, both at training and inference time, and only for comments initially predicted by our classifier as toxic, after which we re-compute all the features and re-classify the now-modified comment.

Evaluation. To quantify model accuracy during cross-validation, we use the $f_{0.5}$ score, because of the imbalance of our dataset and to value precision above recall. Of the different feature combinations we experimented with, our model performed best when using Politeness, Perspective, and after the tuning and post-processing steps described above. Our best classifier had a precision of 0.91 and a recall of 0.42. Feature ablation experiments show that removing features from our model decreases model performance, and adding features to our model does not improve performance.

On a held-out test set (half the labelled data), our model achieves 75 % precision and 35 % recall. We additionally tested our classifier on 100,000 randomly sampled GitHub issues. We manually labeled 100 randomly selected issues that were predicted as toxic to estimate the precision of the classifier. We found that the classifier achieved 50 % precision on the random issues. This indicates that the classifier performs reasonably well outside of the training and validation sets. Some noise is acceptable for studying toxicity trends in the wild, assuming that wrong classifications are randomly distributed.

4 PRELIMINARY EMPIRICAL STUDIES

While our long-term agenda is much broader, we conducted three preliminary studies of toxicity in open-source projects to demonstrate possible uses of our measurement instrument. We study (1) whether toxic interactions in issue discussions have changed over time, (2) whether corporate and non-corporate projects are affected differently, and (3) whether communities around different programming languages are affected differently. We report initial observations, but leave a deeper exploration of these issues (*e.g.*, the influence of a community's culture) for future work.

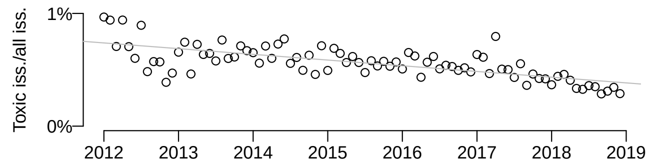

Figure 2: Rate of toxic issues over time decreases gradually

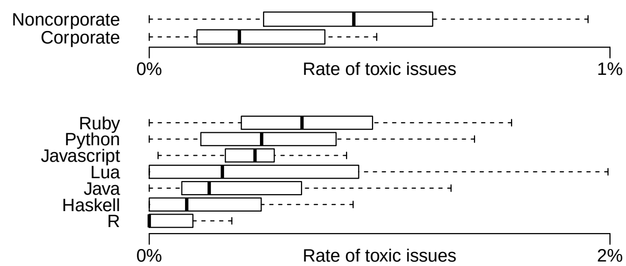

Figure 3: Corporate projects are less toxic than non-corporate projects; there are differences between languages.

Toxicity Over Time. We perceive the public conversations about toxicity, stress, and burnout in open source as a recent phenomena. We are interested to see whether this public attention corresponds with a measurable increase in toxic interactions. To that end, we use our instrument to automatically classify issue discussions in a longitudinal study. We classify all the 1 732 124 issues in GHTorrent from the second Monday of each month between 2012–2018 (this sampling strategy accounts for confounds such as the day of the week or time of the month). As expected, toxic issues are rare, with about 6 for every 1 000 issues. The rate of toxic issues decreases over time, as plotted in Figure 2. While we leave a deeper analysis of reasons for future work, we suspect that increased awareness of the issue may both cause a lower frequency of toxic interactions and more public discussions about the remaining cases.

Corporate vs. Non-Corporate. Suspecting that toxicity is targeted more at volunteers, we explore whether corporate-run projects are less exposed to toxic issue discussions than non-corporate projects. Specifically, we selected the 50 projects with the largest number of employees from corporations actively contributing (using email accounts to detect corporate affiliations) and selected the top 50 projects by number of stars not associated with a corporation. We then labeled 949 739 issues from these projects using our classifier. As shown in Figure 3, our results indicate that the rate of toxic issue discussions is substantially lower for corporate projects (statistically significant, Wilcoxon $p < .001$). We suspect that the increasing number of less toxic corporate projects on GitHub may lead to the overall reduced rate of toxic interactions, but again, we leave deeper explorations to future work.

Toxicity by Community. Cultural differences between communities [2] may also affect the degree of toxic interactions. We use programming languages as a proxy for communities and classified all 872 565 issues from the 30 most popular projects in each of 7 languages. Our findings (Figure 3) suggest differences in toxicity among communities, with R having the lowest rate of toxic discussions, Ruby the highest, and Lua the widest variance. Differences among communities and projects suggests that future research can study the role of community values, and the effectiveness of existing practices and interventions in natural experiments, where possible.

Threats to Validity. Our study is limited to issue discussions on GitHub tracked in GHTorrent. It does not include other forms of communication, such as forums, mailing lists, or face-to-face interactions at conferences. While we evaluated our classifier in Section 3, due to the large number of issues analyzed in our study, we did not verify all classification results. Although we have little reason to expect systematic bias, there is a risk that our classifier may perform differently in different subpopulations.

Additionally, our classifier has relatively low precision on random issues, and low recall on the held-out test set. This might be a result of overfitting to the training set. A larger training and validation set should be used to reduce these issues. Larger validation sets allow for more fine tuning of parameters, which could make the classifier more accurate. Data with more varied sources could also improve the classifier.

5 CONCLUSION

We argue that developer stress and burnout are important threats to open-source sustainability, and suggest a larger research program to find, understand, and mitigate unhealthy interactions. As a key component of such a research program, we report on initial steps to detect toxic interactions in GitHub issue discussions, which seem particularly stressful to maintainers. We design a classifier and demonstrate its utility with three preliminary studies. Our results show promise, and could be used to inform the design of automated, non-invasive measures and models to both help identify contributors and projects exposed to higher levels of toxicity (and likely also stress), as well as to intervene to avoid such toxic comments in the first place (*e.g.*, by flagging them for moderation before being posted). We are excited to see such systems developed, evaluated, and deployed in the near future.

Acknowledgements. Raman was supported through CMU's Research Experiences for Undergraduates in Software Engineering. The researchers were supported in part by awards from the National Science Foundation and the Alfred P. Sloan Foundation. Many thanks to collaborators for useful suggestions: Carolyn Rose and Jim Herbsleb at CMU; Julia Ferraioli, Erin McKean, and Emerson Murphy-Hill at Google.

REFERENCES

[1] Anonymous. Leaving toxic open source communities, 2014. modelviewculture.com/pieces/leaving-toxic-open-source-communities.

[2] Christopher Bogart, Christian Kästner, James Herbsleb, and Ferdian Thung. How to break an api: Cost negotiation and community values in three software ecosystems. In *Proc. FSE*, pages 109–120, 2016.

[3] Luke Breitfeller, Emily Ahn, David Jurgens, and Yulia Tsvetkov. Finding microaggressions in the wild: A case for locating elusive phenomena in social media posts. In *Proc. Conf. Empirical Methods in Natural Language Processing*, 2019.

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

[5] Brett Cannon, Adam Stacoviak, and Jerod Santo. The changelog, episode 318: A call for kindness in open source, October 2018. Podcast.

[6] Laura Dabbish, Colleen Stuart, Jason Tsay, and Jim Herbsleb. Social coding in GitHub: transparency and collaboration in an open software repository. In *Proc. CSCW*, pages 1277–1286, 2012.

[7] Cristian Danescu-Niculescu-Mizil, Moritz Sudhof, Dan Jurafsky, Jure Leskovec, and Christopher Potts. A computational approach to politeness with application to social factors. In *Proc. ACL*, pages 250–259, 2013.

[8] Nemanja Djuric, Jing Zhou, Robin Morris, Mihajlo Grbovic, Vladan Radosavljevic, and Narayan Bhamidipati. Hate speech detection with comment embeddings. In *Proc. WWW*, pages 29–30, 2015.

[9] Nadia Eghbal. Roads and bridges: The unseen labor behind our digital infrastructure. Technical report, Ford Foundation, 2016.

[10] Tom Fearn. Double cross-validation. *NIR news*, 21(5):14–15, 2010.

[11] Darja Fišer, Ruihong Huang, Vinodkumar Prabhakaran, Rob Voigt, Zeerak Waseem, and Jacqueline Wernimont, editors. *Proc. Works. Abusive Language Online*, 2018.

[12] Paula Fortuna and Sérgio Nunes. A survey on automatic detection of hate speech in text. *ACM Computing Surveys (CSUR)*, 51(4):85, 2018.

[13] Daviti Gachechiladze, Filippo Lanubile, Nicole Novielli, and Alexander Serebrenik. Anger and its direction in collaborative software development. In *Proc. ICSE-NIER*, pages 11–14, 2017.

[14] Georgios Gousios. The GHTorrent dataset and tool suite. In *Proc. MSR*, pages 233–236, 2013.

[15] Joshua Guberman, Carol Schmitz, and Libby Hemphill. Quantifying toxicity and verbal violence on Twitter. In *Proc. CSCW Companion*, pages 277–280, 2016.

[16] Emitza Guzman, David Azócar, and Yang Li. Sentiment analysis of commit comments in GitHub: an empirical study. In *Proc. MSR*, pages 352–355. ACM, 2014.

[17] Eran Hammer. A new social contract for open source, February 2018. Blog post, hueniverse.com/86d1fcf3e353.

[18] Guido Hertel, Sven Niedner, and Stefanie Herrmann. Motivation of software developers in open source projects: an internet-based survey of contributors to the Linux kernel. *Research Policy*, 32(7):1159–1177, 2003.

[19] Eric von Hippel and Georg von Krogh. Open source software and the "private-collective" innovation model. *Organization Science*, 14(2):209–223, 2003.

[20] Clayton J Hutto and Eric Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Proc. ICWSM*, 2014.

[21] Carlos Jensen, Scott King, and Victor Kuechler. Joining free/open source software communities: An analysis of newbies' first interactions on project mailing lists. In *Proc. HICSS*, pages 1–10. IEEE, 2011.

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

[23] Dan Jurafsky and James Martin. *Speech and language processing*. Pearson, 2014.

[24] Sandeep Krishnamurthy. On the intrinsic and extrinsic motivation of floss developers. *Knowledge, Technology & Policy*, 18(4):17–39, 2006.

[25] Nolan Lawson. *What it feels like to be an open-source maintainer*, March 2017. Blog post, nolanlawson.com/2017/03/05/.

[26] Jan Lehnardt. Sustainable open source: The maintainers perspective or: How i learned to stop caring and love open source, March 2017. Blog post, writing.jan.io.

[27] Pia Mancini et al. Sustain: A one day conversation for open source software sustainers – the report. Technical report, Sustain Conference Organization, 2017.

[28] Molly K. Maskrey. *Career Direction*, pages 25–70. Apress, Berkeley, CA, 2016.

[29] Burt L Monroe, Michael P Colaresi, and Kevin M Quinn. Fightin'words: Lexical feature selection and evaluation for identifying the content of political conflict. *Political Analysis*, 16(4):372–403, 2008.

[30] Bo Pang, Lillian Lee, et al. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1–2):1–135, 2008.

[31] Haji Mohammad Saleem, Kelly P Dillon, Susan Benesch, and Derek Ruths. A web of hate: Tackling hateful speech in online social spaces. *First Workshop on Text Analytics for Cybersecurity and Online Safety at LREC 2016*, 2016.

[32] Anna Schmidt and Michael Wiegand. A survey on hate speech detection using natural language processing. In *Proc. Int'l Works. Natural Language Processing for Social Media*, pages 1–10, 2017.

[33] Robert Speer, Joshua Chin, Andrew Lin, Lance Nathan, and Sara Jewett. wordfreq: v1. 5.1, 2016.

[34] Igor Steinmacher, Marco Aurelio Graciotto Silva, Marco Aurelio Gerosa, and David F Redmiles. A systematic literature review on the barriers faced by newcomers to open source software projects. *Information and Software Technology*, 59:67–85, 2015.

[35] Yla R Tausczik and James W Pennebaker. The psychological meaning of words: Liwc and computerized text analysis methods. *Journal of Language and Social Psychology*, 29(1):24–54, 2010.

[36] Zeerak Waseem, Wendy Hui Kyong Chung, Dirk Hovy, and Joel Tetreault, editors. *Proc. Works. Abusive Language Online*. ACL, 2017.

[37] Tim Wood. moment().endof('term'), July 2016. Blog post, https://medium.com/timrwood/moment-endof-term-522d8965689.

[38] Frances Zlotnick. Github open source survey 2017, June 2017. doi: 10.5281/zenodo.806811.