Exploring the Impact of Toxic Comments in Code Quality

Jaime Sayago-Heredia[1][a], Gustavo Chango[1][b], Ricardo Pérez-Castillo[2][c] and Mario Piattini[3][d]
[1]*Escuela de Sistemas y Computación, Pontificia Universidad Católica del Ecuador, Sede Esmeraldas, Espejo y subida a Santa Cruz Casilla 08-01-0065, Ecuador*
[2]*Facultad de Ciencias Sociales de Talavera de la Reina, University of Castilla-La Mancha, Avenida Real Fábrica de Seda s/n 45600, Talavera de la Reina, Spain*
[3]*Information Technology & Systems Institute (ITSI), University of Castilla-La Mancha, Paseo de la Universidad 4, 13071, Ciudad Real, Spain*

Keywords: Sentiments Analysis, Toxic Comment Classification, GitHub, SonarQube, Commits, Software Quality, Software Engineering.

Abstract: Software development has an important human-side, which implies that developers' feelings have a significant impact to software development and could affect developers' quality, productivity, and performance. In this paper, we explore the process to find, understand and relate the effects of toxic emotions on code quality. We propose a tool and sentiments dataset, a clean set of commit messages, extracted from SonarQube code quality metrics and toxic comments obtained from GitHub. Moreover, we perform a preliminary statistical analysis of the dataset. We apply natural language processing techniques to identify toxic developer sentiments on commits that could impact code quality. Our study describes data retrieval process along with tools used for performing a preliminary analysis. The preliminary dataset is available in CSV format to facilitate queries on the data and to investigate in depth factors that impact developer emotions. Preliminary results imply that there is a relationship between toxic comments and code quality that may affect the quality of the software project. Future research will be the development of a complete dataset and an in-depth analysis for efficiency validation experiments along with a linear regression. Finally, we will estimate the code quality as a function of developers' toxic comments.

1 INTRODUCTION

Research in the field of software engineering has increasingly applied techniques and methods from other research areas (Novielli et al., 2018), such as sentiment analysis (Cheruvelil and Da-Silva, 2019; Ding et al., 2018; Guzman et al., 2014; Murgia et al., 2014). Software development has an important human-side, so it is evident that the developer's feelings have a significant impact on various issues related to software development, such as quality. Nowadays, software development projects depend on a large number of programmers who collaborate with each other in their efforts to develop a software system (Boehm, 1988). These efforts to build and maintain a software project are continuous and stressful for developers, which becomes a difficult problem to solve (Rezvani and Khosravi, 2019). Research community are concerned that these feelings could lead to buggy code and consequently poor quality code, as evidenced by some works (Asri et al., 2019; Cheruvelil and Da-Silva, 2019; Singh and Singh, 2018).

Developers are confronted with various problems every day and have to find a solution that require high levels of technical knowledge. For developers, these obstacles to successfully complete the development of a software project can be exhausting and stressful (Rezvani and Khosravi, 2019). This subsequently impacts on their ability to self-regulate their feelings and understanding (Hancock and Szalma, 2008), e.g. *commit messages* have a lot of negative feelings (Sinha et al., 2016). These messages contain an

[a] https://orcid.org/0000-0003-3657-5407
[b] https://orcid.org/0000-0003-3231-0153
[c] https://orcid.org/0000-0002-9271-3184
[d] https://orcid.org/0000-0002-7212-8279

Sayago-Heredia, J., Chango, G., Pérez-Castillo, R. and Piattini, M.
Exploring the Impact of Toxic Comments in Code Quality.
DOI: 10.5220/0011039700003176
In *Proceedings of the 17th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2022)*, pages 335-343
ISBN: 978-989-758-568-5; ISSN: 2184-4895

emotional expression (negative or toxic), which could either influence the quality of the code, or be a reflection of the fact that the quality of the code also influences the negative feelings of the developers. Therefore, there is a need to analyze, understand and relate developers' negative feelings and code quality, an important and understudied field of research. There are several important research questions that are raised in this research, for example: ¿Toxic comments are related to code quality during the software project lifecycle? ¿Developer’s toxic feelings related to the increase of bugs in software project? Our research was answer to some questions empirically using direct and indirect methods, and one of these indirect methods was development of a tool for analyzing large amounts of data and offering the possibility of statistical analysis and hypothesis testing. *Commit messages* can be used to extract developer sentiment (Ding et al., 2018), function as an important resource to analyze and understand a possible relationship between developer sentiment and code quality.

We have developed a research tool (classifier) to extract toxic comments from commit messages. Toxic language can be present in different places online (Facebook, YouTube and others), they are also present in commit messages software projects on GitHub. Toxic can manifest itself in multiple ways, in the case of software development, through messages corroborating the lack of help for a bug together with name calling, insults or threats. We carried out a preliminary study with data extracted with the constructed tool. This tool obtains toxic comments from commit messages of the selected projects from GitHub and jointly extracts the code quality metrics corresponding to commit messages of the software projects.

Our work makes a main contribution, verifying that toxic of messages can be identified through Natural Language Processing (NLP) techniques and obtaining a corpus of data with negative sentiments and toxic comments. Results preliminary study suggest a correlation between toxic comments and code quality. Moreover, future research we will explore how code quality can affect developers' emotions. This paper allows us to expand our research possibilities and areas involved in sentiment analysis that we will explore in next future.

2 BACKGROUND

This section provides background information about sentiment analysis, toxic comment classification and code quality.

2.1 Sentiment Analysis

Sentiment analysis (also known as opinion mining) was initially developed as an automated method to extract sentiment polarity from short texts published online, such as movie reviews, product reviews, *microblogs* and *tweets* (Sinha et al., 2016) that had a great application in the marketing world. Specifically, sentiment analysis is the study of the subjectivity and polarity of a manually written text (usually identified as neutral positive or negative) (Pang and Lee, 2008). There are some works analyzing sentiments in the software engineering domain where sentiments are analyzed (Cheruvelil and Da-Silva, 2019; Guzman et al., 2014; Kaur et al., 2018; Singh and Singh, 2018), using different artifacts such as GitHub (Pletea et al., 2014; Sinha et al., 2016), issue resolution with Jira (Ortu et al., 2016) and bug reports (Kritikos et al., 2020). Other research suggests uncertainties related to the unsuccessful application of sentiment analysis tools for software engineering (Asri et al., 2019; Sun, 2021). Existing tools require configurations for their specific use or adaptation for the specific context to be used and this can be crucial for the performance of the tool (Lin et al., 2018). Moreover, other works (Cheruvelil and Da-Silva, 2019; Ding et al., 2018; Guzman et al., 2014; Murgia et al., 2014), have focused on analyzing and classifying emotions in software artefacts and, at the same time, proposing emotion standards (sadness, happiness, anger, fear, etc.) in development teams. Guzmán et al. (Guzman et al., 2014) proposed a sentiment analysis approach for discussions in mailing lists and web-based software collaboration tools. Ding et al. (Ding et al., 2018) conducted an entity-level sentiment analysis by creating a dataset and a *SentiSW* tool. SentiSW is an entity-level sentiment analysis tool that consists of sentiment classification and entity recognition and classifies problematic comments with significantly higher accuracy than other tools. Murgia et al.(Murgia et al., 2014) analyzed development artefacts and problem reports to find out whether they contain any emotional information about software development through an automatic tool for emotion extraction in software development artefact. Authors (Cheruvelil and Da-Silva, 2019) applied a sentiment analysis tool to problem follow-up comments and observed how scores varied for problems with no reopening’s, with one reopening and with many reopening, suggesting that negative sentiment correlates with reopened problems, although the effect size appears to be quite small.

2.1.1 Sentiment Analysis Tools

Analyses described in the previous section focused on finding and locating emotions through tools that have not been created for the software engineering context. Sentiment analysis tools are available, both commercial (Lin et al., 2018) and free (Jongeling et al., 2017). Some sentiment analysis tools applied in software engineering are:

- *SentiStrength* is the most widely used in software engineering studies(Guzman et al., 2014; Novielli et al., 2014; Ortu et al., 2016).
- *NLTK* is a lexicon- and rule-based sentiment analysis tool, whose core is *VADER* (Valence Aware Dictionary and sEntiment Reasoner); *VADER* is specifically tailored to social network texts by incorporating a sentiment lexicon extracted from microblog contexts and manually validated by multiple independent human judges (Wilson and Hernández-Hall, 2014).
- *Stanford CoreNLP* is based on a recursive neural network, which differs from other tools in its ability to derive the sentiment of a sentence based on how the words compose the meaning of the sentence, rather than by summing the sentiment of individual words. It has been trained on movie reviews (Socher et al., 2013).

Tools described, encounter some discrepancies in terms of their effectiveness and coincidence of results, which may lead to different conclusions, making them not replicable when using different sentiment analysis tools (Jongeling et al., 2017). Specifically, several authors acknowledge (Howard et al., 2013; Jongeling et al., 2017; Lin et al., 2018) that it's necessary to build analysis tools targeted at specific datasets for software engineering so that they do not cast doubt on the validity of sentiment analysis results.

2.2 Code Quality

Code quality consists of characteristics such as capability, usability, performance, reliability and maintainability (Horch, 1996), security, portability, compatibility, performance efficiency, functional adequacy (ISO, 2011). Characteristics which affect the code in its efficiency, vulnerability and security(Liu and Woo, 2020). Several tools exist to detect software quality (Lewis et al., 2017). Open source projects such as Squale, SonarQube and CodeMetrics provide static analysis of target programs to check for coding errors (Ardito et al., 2020). There are programs to measure errors that affect code quality such as QScored (Thakur et al., 2020) and E-Quality (Erdemir et al., 2011). Are also paid tools such as Codacy, FortifyVeracode, Static Analysis, Reshift, NextGen Static Analysis (OWASP, 2021). Challenge for tools described is that exist a greater complexity than just reviewing code written by the developer, the literature review suggests that there is a relationship between code quality and emotions as demonstrated Gunsel (Gunsel, 2014) in which project complexity has an effect on the relationship between work emotions and code quality. Consequently, it is imperative to explore and understand whether there is a relationship between negative emotions such as developer feedback toxicity affecting and impacting the code quality of a software project.

2.3 Toxic Comment Classification

In area of sentiment analysis, the classification of toxic comments is not clearly defined in scope. The research community has investigated the toxicity of comments on the web (Georgakopoulos et al., 2018). Risch (Risch et al., 2021) constructed a toxicity dataset describing its data origin along with a combination of the different targets found such as hate, attacks, aggressive, toxic, online harassment, abusive language, cyberbullying and offensive language, racism, sexism and insults. The definitions go according to each area to be classified; it is noted that the methods used for the analyses are similar because they are applied on social media platforms. Sarker (Sarker et al., 2020) in the area of software engineering validates the following broad definition of toxic content: "*An SE conversation will be considered as toxic, if it includes any of the following: offensive name calling, insults, threats, personal attacks, flirtations, reference to sexual activities, and swearing or cursing*". State-of-the-art review indicates that toxicity is part of sentiment analysis in the area of software engineering. This approach has been used to detect the psychological state of developers (Rousinopoulos et al., 2014). Authors (Guzman and Bruegge, 2013) used this technique to investigate the role of emotional awareness in development teams. Gachechiladze (Gachechiladze et al., 2017) used sentiment analysis to identify a classification for anger detection. Pletea (Pletea et al., 2014) suggests negativity increases when developers are concerned with the security of the software project. Other research has also explored the specific concept of happiness at work, connecting it to high quality software artefacts such as the work of Graziotin et al. (Graziotin et al., 2018) The sentiment expressed on websites like Stack Overflow is studied to analyze and classify toxic sentiments of users (Cheriyan et al., 2021). Carige (Carige and de Figueiredo Carneiro, 2020) indicates that positive and negative emotions have a tendency to influence developers' productivity, task quality and job satisfaction. We focus on understand and detecting toxic comments for the software engineering context from commit messages and choosing most effective method for extracting toxicity and relate the effects of toxic comments on the code quality of a software project during its lifecycle.

3 DATA COLLECTION AND METHODS

3.1 Project Selection

The software projects were obtained from the search performed on the software repository GitHub, which is the largest online platform and contain more than 3.4 million users (Li et al., 2017). This significant number of projects can contribute to our research, although it can be detrimental when selecting software projects that are irrelevant, so it is necessary to define both selection and exclusion criteria to filter those results. Inclusion and exclusion practices and strategies are valuable in several researches concerning software engineering (Petersen and Gencel, 2013). The following inclusion criteria were defined:

- The software project is in production.
- The metrics and reports are accessible from the SonarCloud platform.
- The team of developers of the software project must meet an average value of developers (5) which is an average value obtained from the pilot review of the software repositories. In addition, we applied other criteria based on the research of (Lenarduzzi et al., 2019):
- More than 10 releases
- More than 5000 commits,
- More than 1000 classes,
- More than 100000 lines of code.

Exclusion criteria are the english language must be used by the development team in commit messages, incomplete software and projects that are on the SonarQube platform but with low or no activity i.e., without any recent analysis or releases.

3.1.1 Classification Toxic Model

The context of the classification to be carried out to obtain the level of toxicity of developers' comments should be focused on software engineering as this is our area of study. The training data for the preliminary study is not large. The developed tool extracts the toxicity of commit messages with natural language processing (NLP) classification techniques suggested by state-of-the-art review (Geet et al., 2020; Saeed et al., 2019; Tare, 2017). Tool uses Microsoft's ML.NET library, that allows developers to build complex machine learning pipelines. Pipelines are often composed of multiple transformation steps that feature and transform the raw input data (Ahmed et al., 2019). The task used to train the model is binary classification. During the model training process, the model generator trains independent models with different options and binary classification algorithms to find the best performing model for the dataset (Sistema et al., 2019). Time required for model training is proportional to the amount of data. At the end of training the model the output will contain the algorithm that uses the model with the best performance on the input data. In our case it is the L-BFGS (Limited Broydon-Fletcher-Goldfarb-Shanno) algorithm which is a quasi-Newton optimization method of functions with a large number of parameters or of a high complexity (Bollapragada et al., 2018). It is a method that makes limited use of memory, using it optimally and in fewer algorithms for the same problem. L-BFGS allows obtaining the minimum of a function; it only needs the function and its gradient, but not the Hessian matrix, therefore, it is able to solve functions without restrictions in its parameters (Berahas and Takáč, 2020). The result of the model generated using the L-BFGS algorithm is satisfactory with a percentage rate of 78.03%.

3.2 Tool Used to Collect Data

Tool is a web application built for data extracted. Figure 1, describes functionality of the tool that integrates and extracts commit messages from the GitHub Api which is the largest online platform and contains more than 3.4 million users (Li et al., 2017) and metrics from the SonarQube Api which is the most widely used tool on the market for code quality analysis (Lenarduzzi et al., 2020)). We use for its development .Net Visual Studio, C#, HTML5 and the ML.NET library that allows developers to build complex machine learning and LNP artifacts (Ahmed et al., 2019). First step is prepared dataset to be used for sentiment analysis. It is important to understand the dataset. The sentiment calculated from the commit message consists of a toxicity index, the numeric value of the sentiment provides a quantification. It is important determine and understand the words to understand the context of the sentence. When using the different words, we must differentiate the context from the software engineering to extract the correct sentiment values from the message. The tool uses different algorithms and trains separate models to find best performing model for dataset(Sistema et al., 2019). Tool extracts sentiment toxic from *commit messages* from project (GitHub) and metrics (SonarQube), result generates graphs with statistical analysis, dashboard by project and a dataset for each project.

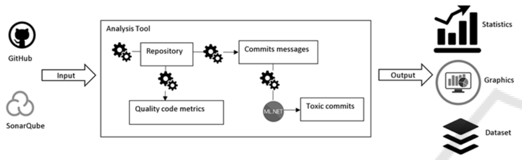

Figure 1: Overview of tool functionality.

3.3 Dataset Schema and Production

Figure 2 presents the schema of the dataset entities. It is made up of two linked databases, a relational database (SqlServer) and a NoSql database (MongoDB). By observing the tables of the first one, it is possible to identify their respective relationships and fields of each entity and normalisation. Repository table contains the information referring to the software projects with their respective repository header fields. Commit table contains the different data concerning the commit messages together with their toxicity level. Metrics table contains the quality values of SonarQube. UserRepository and User tables correspond to the security module of the tool. As for the second database (MongoDB), the collections are not normalised by the amount of data that can be repeated. CommitsByProject collection represents the repository and the respective documents such as commit, author, commiter, commitAuthor and toxicity. AnalysesByProject collection represents the information of each release extracted of repository. IssuesByProject collection contains the data of issues of extracted repository. MeasureHistoryByProject collection represents the information about the software quality metrics of each release of repository. The dataset obtained from the data extraction and cleaning can be consulted at the following link https://zenodo.org/record/6012674#.YgKvE-rMLt8

4 PRELIMINARY EMPIRICAL CASE STUDY

We conducted a preliminary study with a project software of the extracted toxic comments and metrics code quality dataset to validity check of tool. We study the value toxic per commit message of each release of the project, obtained the main quality metrics from SonarQube and executed a correlational analysis between the quality metrics and toxic comments.

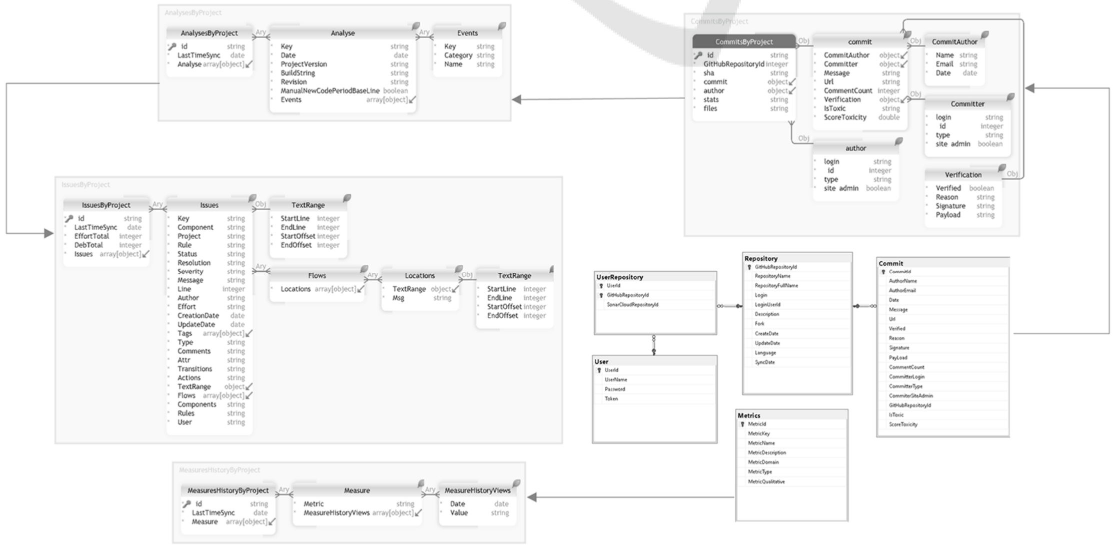

Figure 2: Diagram of database of tool.

Toxic comments by release. We performed a percentage ranking of toxicity per release from 0 to 100, i.e. for each release there is a total of commit messages and in particular each comment has a toxicity index. We found, the highest amount of toxic comments are located in the range of 30 - 40 % per commit message. The amount of toxicity relevant for the following analysis is in the range of 90-100% toxic per commit message. We observe that amount of toxic comments increases with each project release.

Code quality metrics. Tool automatically extracts SonarQube's code quality metrics from the repository for each project release. We analyze, process, and clean the variables with information that significant to the study.

Correlational analysis. We statistically analyzed toxic comments and a possible relationship with quality metrics in our research. We study the relationship between variables, we performed a Pearson correlation (r) on our dataset. Pearson is a measure of linear association suitable for variables measured on an interval scale (Thirumalai and Member, 2017). Result obtained was thirty-two variables indicating from a low to moderate significant correlation ranging between values in from r=0.28 to r= 0.69 as see in table 1.

Table 1: Correlation toxic comments and code quality metrics.

Variable	R value
functions	0.39052521
duplicated_lines	0.64677504
coverage	0.32066621
complexity	0.40119754
comment_lines	0.40327453
comment_lines_density	0.57982721
duplicated_lines_density	0.69194964
file_complexity	0.63888768
violations	0.5330704
duplicated_blocks	-0.32833555
duplicated_files	0.56395605
statements	0.36686428
blocker_violations	0.52621986
major_violations	0.61798788
minor_violations	0.49356737
info_violations	-0.27603837
lines_to_cover	0.63816548
line_coverage	0.32066621
sqale_index	0.43315924
last_commit_date	0.62599543
open_issues	0.53678944
reopened_issues	0.23631395
confirmed_issues	-0.5486225
sqale_debt_ratio	0.31830114
new_sqale_debt_ratio	0.20616904
code_smells	0.51404113
bugs	0.28129013
reliability_remediation_effort	0.31833085
security_remediation_effort	0.52621986
security_rating	0.52621986
cognitive_complexity	0.40012998
new_development_cost	-0.41797483

To choosing the most significant variables for our study. We performed a processing of normalisation/rescaling of data set (Chango et al., 2021), achieve a CSV file. We then with Weka(Hutter et al., 2019) proceeded to apply attribute selection algorithms from the dataset to select the quality variables most strongly correlated with toxic comments. We obtained from data set two sets of 2 optimal variables (Table 2) for normalized data set and 6 optimal variables for discrete data set.

Table 2: List of selected variables.

# of selected	**selected variables**	**Selected Features**	**Type Data**
2	*duplicated_lines_density *lines_to_cover	1,12,25	Normalised
6	*blocker_violations *reopened_issues *confirmed_issues *new_sqale_debt_ratio *security_remediation_effort Security_rating	2,20,31,32, 34,39,40	Discretised

Our analyses suggest that exist a correlation between the code quality variables (Table 2) and the toxicity of *commit messages*, with the "duplicated lines density" variable r=0.69 having the highest correlation value and the "new sqale debt ratio" variable r=0.21 having the lowest correlation value.

These results imply that exist a relationship between toxic comments and code quality that can affect the quality of the software project in life cycle. Further analysis will be addressed in future research to study which dataset is more efficient in validation experiments together with linear regression and calculate the impact on code quality.

5 THREATS TO VALIDITY

The study has a scope on GitHub *commit messages* and toxic comments that could affect code quality. We do not consider other elements that are part of the software repository such as pull requests, branches, mailing lists or formal project documentation. Moreover, the tool could improve toxic calculation if we modify the training set. The next step of improvement is to use a larger training set, which allows us to increase the accuracy of obtaining toxicity along with a larger number of projects. Finally, the validation of our study could be improved by using other data sources.

6 CONCLUSIONS

Software quality is measured to find bugs and software problems in the development phase. Therefore, lack of quality can have serious consequences for the software product. In our research we propose a dataset to correlate the toxic comments from commit messages and software quality during the development phase of a software project. We analyze these possible correlations with a preliminary case study. In the case study, statistical analyses were applied to extract the significant variables for our study and to test the correlation with toxic comments. We found that there is a moderate correlation between the toxic comments and software quality metrics. Results obtained are in line with related work, what motivates us research to continued, how developer toxic sentiments might affect the code quality of a software project. Pilot study is small, so future work in this research will aim to replicate it with more software projects and an improved training set to obtain a broader and deeper account of the factors that affect developers' emotions and, in turn, code quality in the lifecycle of a software project. Moreover, future research we will explore how code quality can affect developers' emotions. This paper allows us to expand our research possibilities and areas involved in sentiment analysis that we will explore in next future.

ACKNOWLEDGEMENTS

This study has been partially funded by the G3SOFT (SBPLY/17/ 180501/ 000150) project funded by the 'Dirección General de Universidades, Investigación e Innovación – Consejería de Educación, Cultura y Deportes; Gobierno de Castilla-La Mancha'. This work is part of the SMOQUIN project (PID2019-104791RB-I00) funded by Spanish MICINN. This work is also part of the BIZDEVOPS-Global (RTI2018-098309-B-C31) project, Ministerio de Economía, Industria y Competitividad (MINECO) & Fondo Europeo de Desarrollo Regional (FEDER).

REFERENCES

Ahmed, Z., Amizadeh, S., Bilenko, M., Carr, R., Chin, W. S., Dekel, Y., … Zhu, Y. (2019). Machine learning at microsoft with ML.NET. *ArXiv*, 2448–2458.

Ardito, L., Coppola, R., Barbato, L., & Verga, D. (2020). A Tool-Based Perspective on Software Code Maintainability Metrics: A Systematic Literature Review. *Scientific Programming*, *2020*. https://doi.org/10.1155/2020/8840389

Asri, I. El, Kerzazi, N., Uddin, G., Khomh, F., & Janati Idrissi, M. A. (2019). An empirical study of sentiments in code reviews. *Information and Software Technology*, *114*(June), 37–54. https://doi.org/10.1016/j.infsof.2019.06.005

Berahas, A. S., & Takáč, M. (2020). A robust multi-batch L-BFGS method for machine learning*. *Optimization Methods and Software*, *35*(1), 191–219. https://doi.org/10.1080/10556788.2019.1658107

Boehm, B. W. (1988). A Spiral Model of Software Development and Enhancement. *Computer*, *21*(5), 61–72. https://doi.org/10.1109/2.59

Bollapragada, R., Mudigere, D., Nocedal, J., Shi, H. J. M., & Tang, P. T. P. (2018). A Progressive Batching L-BFGS Method for Machine Learning. *35th International Conference on Machine Learning, ICML 2018*, *2*, 989–1013.

Carige, R. S., & de Figueiredo Carneiro, G. (2020). Impact of developers sentiments on practices and artifacts in open source software projects: A systematic literature review. *ICEIS 2020 - Proceedings of the 22nd International Conference on Enterprise Information Systems*, *2*(ICEIS), 31–42. https://doi.org/10.5220/0009313200310042

Chango, W., Cerezo, R., & Romero, C. (2021). Multi-source and multimodal data fusion for predicting academic performance in blended learning university courses. *Computers and Electrical Engineering*, *89*(November 2020). https://doi.org/10.1016/j.compeleceng.2020.106908

Cheriyan, J., Savarimuthu, B. T. R., & Cranefield, S. (2021). Norm Violation in Online Communities – A Study of Stack Overflow Comments. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *12298 LNAI*, 20–34. https://doi.org/10.1007/978-3-030-72376-7_2

Cheruvelil, J., and Da-Silva, B. C. (2019). Developers' sentiment and issue reopening. *Proceedings - 2019 IEEE/ACM 4th International Workshop on Emotion Awareness in Software Engineering, SEmotion 2019*, 29–33. https://doi.org/10.1109/SEmotion.2019.00013

Ding, J., Sun, H., Wang, X., & Liu, X. (2018). Entity-level sentiment analysis of issue comments. *Proceedings - International Conference on Software Engineering*, 7–13. https://doi.org/10.1145/3194932.3194935

Erdemir, U., Tekin, U., & Buzluca, F. (2011). E-quality: A graph based object oriented software quality visualization tool. *Proceedings of VISSOFT 2011 - 6th IEEE International Workshop on Visualizing Software for Understanding and Analysis*. https://doi.org/10.1109/VISSOF.2011.6069454

Gachechiladze, D., Lanubile, F., Novielli, N., & Serebrenik, A. (2017). Anger and its direction in collaborative software development. *Proceedings - 2017 IEEE/ACM 39th International Conference on Software Engineering: New Ideas and Emerging Results Track, ICSE-NIER 2017*, 11–14. https://doi.org/10.1109/ICSE-NIER.2017.18

Geet, A., Illina, I., Fohr, D., Landscapes, T. N., Toxic, A., Sa, A. G. D., … Lorraine, U. De. (2020). Towards Non-Toxic Landscapes : Automatic Toxic Comment Detection Using DNN. In *Second Workshop on Trolling, Aggression and Cyber- bullying (LREC, 2020)*.

Georgakopoulos, S. V., Vrahatis, A. G., Tasoulis, S. K., & Plagianakos, V. P. (2018). Convolutional neural networks for toxic comment classification. *ACM International Conference Proceeding Series*. https://doi.org/10.1145/3200947.3208069

Graziotin, D., Fagerholm, F., Wang, X., & Abrahamsson, P. (2018). What happens when software developers are (un)happy. *Journal of Systems and Software*, *140*, 32–47. https://doi.org/10.1016/j.jss.2018.02.041

Gunsel, A. (2014). The Effects of Emotional Labor on Software Quality: the Moderating Role of Project Complexity. *Journal of Global Strategic Management*, *2*(8), 96–96. https://doi.org/10.20460/jgsm.20148 15645

Guzman, E., Azócar, D., & Li, Y. (2014). Sentiment analysis of commit comments in GitHub: An empirical study. *11th Working Conference on Mining Software Repositories, MSR 2014 - Proceedings*, 352–355. https://doi.org/10.1145/2597073.2597118

Guzman, E., & Bruegge, B. (2013). Towards emotional awareness in software development teams. *2013 9th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, ESEC/FSE 2013 - Proceedings*, 671–674. https://doi.org/10.1145/2491411.2494578

Hancock, P. A., & Szalma, J. L. (2008). Performance under stress. *Performance Under Stress*, (January 2008), 1–389. https://doi.org/10.21139/wej.2017.013

Horch, J. W. (1996). Metrics and models in software quality engineering. *Control Engineering Practice*. https://doi.org/10.1016/0967-0661(96)81493-6

Howard, M. J., Gupta, S., Pollock, L., & Vijay-Shanker, K. (2013). Automatically mining software-based, semantically-similar words from comment-code mappings. *IEEE International Working Conference on Mining Software Repositories*, 377–386. https://doi.org/10.1109/MSR.2013.6624052

Hutter, F., Kotthoff, L., & Vanschoren, J. (2019). *Automated Machine Learning. The Springer Series on Challenges in Machine Learning. Automated Machine Learning. The Springer Series on Challenges in Machine Learning.* Springer. https://doi.org/10.1007/978-3-319-00960-5_6

ISO. (2011). ISO - ISO/IEC 25010:2011 - Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models. Retrieved November 24, 2021, from https://www.iso.org/standard/35733.html

Jongeling, R., Sarkar, P., Datta, S., & Serebrenik, A. (2017). On negative results when using sentiment analysis tools for software engineering research. *Empirical Software Engineering*, *22*(5), 2543–2584. https://doi.org/10.1007/s10664-016-9493-x

Kaur, A., Singh, A. P., Dhillon, G. S., & Bisht, D. (2018). Emotion Mining and Sentiment Analysis in Software Engineering Domain. *Proceedings of the 2nd International Conference on Electronics, Communication and Aerospace Technology, ICECA 2018*, (Iceca), 1170–1173. https://doi.org/10.1109/ICECA.2018.8474619

Kritikos, A., Venetis, T., & Stamelos, I. (2020). *An Empirical Investigation of Sentiment Analysis of the Bug Tracking Process in Libre Office Open Source Software. IFIP Advances in Information and Communication Technology* (Vol. 582 IFIP). Springer International Publishing. https://doi.org/10.1007/978-3-030-47240-5_4

Lenarduzzi, V., Lomio, F., Huttunen, H., & Taibi, D. (2020). Are SonarQube Rules Inducing Bugs? *SANER 2020 - Proceedings of the 2020 IEEE 27th International Conference on Software Analysis, Evolution, and Reengineering*, 501–511. https://doi.org/10.1109/SANER48275.2020.9054821

Lenarduzzi, V., Saarimäki, N., & Taibi, D. (2019). The technical debt dataset. *ACM International Conference Proceeding Series*, (May), 2–11. https://doi.org/10.1145/3345629.3345630

Lewis, W. E., Dobbs, D., & Veerapillai, G. (2017). *Software Testing and Continuous Quality Improvement (3rd ed.)*. Auerbach Publications. https://doi.org/https://doi.org/10.1201/9781439834367

Li, L., Goethals, F., Baesens, B., & Snoeck, M. (2017). Predicting software revision outcomes on GitHub using structural holes theory. *Computer Networks*, *114*, 114–124. https://doi.org/10.1016/j.comnet.2016.08.024

Lin, B., Zampetti, F., Bavota, G., Penta, M. Di, Lanza, M., Oliveto, R., & Di Penta, M. (2018). Sentiment Analysis for Software Engineer-ing: How Far Can We Go. *Dl.Acm.Org*, 94–104. Retrieved from https://doi.org/10.1145/3180155.3180195

Liu, X., & Woo, G. (2020). Applying Code Quality Detection in Online Programming, 56–60.

Murgia, A., Tourani, P., Adams, B., & Ortu, M. (2014). Do developers feel emotions? An exploratory analysis of emotions in software artifacts. *11th Working Conference on Mining Software Repositories, MSR 2014 - Proceedings*, 262–271. https://doi.org/10.1145/2597073.2597086

Novielli, N., Calefato, F., & Lanubile, F. (2014). Towards discovering the role of emotions in stack overflow. *6th International Workshop on Social Software Engineering, SSE 2014 - Proceedings*, (November), 33–36. https://doi.org/10.1145/2661685.2661689

Novielli, N., Calefato, F., & Lanubile, F. (2018). A gold standard for emotion annotation in stack overflow. *Proceedings - International Conference on Software Engineering*, 14–17. https://doi.org/10.1145/3196398.3196453

Ortu, M., Murgia, A., Destefanis, G., Tourani, P., Tonelli, R., Marchesi, M., & Adams, B. (2016). The emotional side of software developers in JIRA. *Proceedings - 13th Working Conference on Mining Software Repositories, MSR 2016*, 480–483. https://doi.org/10.1145/2901739.2903505

OWASP. (2021). Source Code Analysis Tools | OWASP. Retrieved December 6, 2021, from https://owasp.org/www-community/Source_Code_Analysis_Tools

Pang, B., & Lee, L. (2008). Opinion Mining and Sentiment Analysis. *Found. Trends Inf. Retr.*, *2*(1–2), 1–135. https://doi.org/10.1561/1500000011

Petersen, K., & Gencel, C. (2013). Worldviews, research methods, and their relationship to validity in empirical software engineering research. *Proceedings - Joint Conference of the 23rd International Workshop on Software Measurement and the 8th International Conference on Software Process and Product Measurement, IWSM-MENSURA 2013*, 81–89. https://doi.org/10.1109/IWSM-Mensura.2013.22

Pletea, D., Vasilescu, B., & Serebrenik, A. (2014). Security and emotion: Sentiment analysis of security discussions on GitHub. *11th Working Conference on Mining Software Repositories, MSR 2014 - Proceedings*, 348–351. https://doi.org/10.1145/2597073.2597117

Rezvani, A., & Khosravi, P. (2019). Emotional intelligence: The key to mitigating stress and fostering trust among software developers working on information system projects. *International Journal of Information Management*, *48*(January), 139–150. https://doi.org/10.1016/j.ijinfomgt.2019.02.007

Risch, J., Schmidt, P., & Krestel, R. (2021). Data Integration for Toxic Comment Classification : Making More Than 40 Datasets Easily Accessible in One Unified Format.

Rousinopoulos, A.-I., Robles, G., & González-Barahona, J. M. (2014). Sentiment Analysis of Free/Open Source Developers: Preliminary Findings From a Case Study. *Revista Eletrônica de Sistemas de Informação*, *13*(2). https://doi.org/10.5329/resi.2014.1302006

Saeed, H. H., Shahzad, K., & Kamiran, F. (2019). Overlapping toxic sentiment classification using deep neural architectures. *IEEE International Conference on Data Mining Workshops, ICDMW*, *2018–Novem*, 1361–1366. https://doi.org/10.1109/ICDMW.2018.00193

Sarker, J., Turzo, A. K., & Bosu, A. (2020). A benchmark study of the contemporary toxicity detectors on software engineering interactions. *Proceedings - Asia-Pacific Software Engineering Conference, APSEC*, *2020–Decem*, 218–227. https://doi.org/10.1109/APSEC51365.2020.00030

Singh, N., & Singh, P. (2018). How Do Code Refactoring Activities Impact Software Developers' Sentiments? - An Empirical Investigation into GitHub Commits. *Proceedings - Asia-Pacific Software Engineering Conference, APSEC*, *2017–Decem*, 648–653. https://doi.org/10.1109/APSEC.2017.79

Sinha, V., Lazar, A., & Sharif, B. (2016). Analyzing developer sentiment in commit logs. *Proceedings - 13th Working Conference on Mining Software Repositories, MSR 2016*, 520–523. https://doi.org/10.1145/2901739.2903501

Sistema, I., Pomoću, P., & Net, M. L. (2019). DEVELOPMENT OF RECOMMENDER SYSTEMS USING ML . NET, (September).

Socher, R., Perelygin, A., Wu, J. Y., Chuang, J., Manning, C. D., Ng, A. Y., & Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. *EMNLP 2013 - 2013 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 1631–1642.

Sun, K. (2021). Exploiting the Unique Expression for Improved Sentiment Analysis in Software Engineering Text.

Tare, P. (2017). Toxic Comment Detection and Classification. In *31st Conf. on Neural Information Processing Systems (NIPS 2017),* (pp. 1–6).

Thakur, V., Kessentini, M., & Sharma, T. (2020). QScored: An Open Platform for Code Quality Ranking and Visualization. *Proceedings - 2020 IEEE International Conference on Software Maintenance and Evolution, ICSME 2020*, 818–821. https://doi.org/10.1109/ICSME46990.2020.00101

Thirumalai, C., & Member, I. (2017). Analysing the Concrete Compressive Strength using Pearson and Spearman. *International Conference on Electronics, Communication and Aerospace Technology*, 215–218.

Wilson, J., & Hernández-Hall, C. (2014). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. *Eighth International AAAI Conference on Weblogs and Social Media*, 18. Retrieved from https://www.aaai.org/ocs/index. php/ICWSM/ICWSM14/paper/viewPaper/8109