Analyzing Toxicity in Open Source Software Communications Using Psycholinguistics and Moral Foundations Theory

Ramtin Ehsani
Drexel University
Philadelphia, PA, USA
ramtin.ehsani@drexel.edu

Rezvaneh (Shadi) Rezapour
Drexel University
Philadelphia, PA, USA
shadi.rezapour@drexel.edu

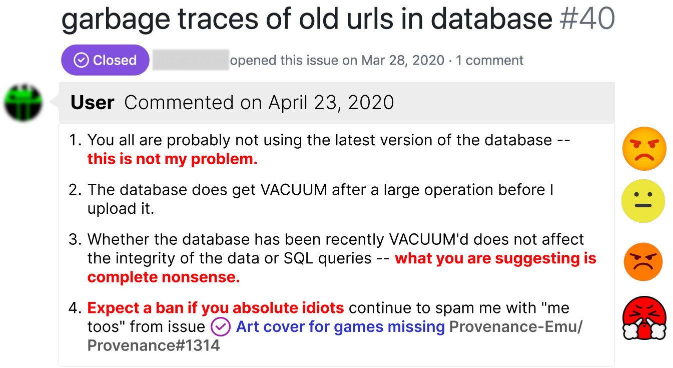
Preetha Chatterjee
Drexel University
Philadelphia, PA, USA
preetha.chatterjee@drexel.edu

***Abstract*—Studies have shown that toxic behavior can cause contributors to leave, and hinder newcomers' (especially from underrepresented communities) participation in Open Source Software (OSS) projects. Thus, detection of toxic language plays a crucial role in OSS collaboration and inclusivity. Off-the-shelf toxicity detectors are ineffective when applied to OSS communications, due to the distinct nature of toxicity observed in these channels (e.g., entitlement and arrogance are more frequently observed on GitHub than on Reddit or Twitter). In this paper, we investigate a machine learning-based approach for the automatic detection of toxic communications in OSS. We leverage psycholinguistic lexicons, and Moral Foundations Theory to analyze toxicity in two types of OSS communication channels; issue comments and code reviews. Our evaluation indicates that our approach can achieve a significant performance improvement (up to 7% increase in F1 score) over the existing domain-specific toxicity detector. We found that using moral values as features is more effective than linguistic cues, resulting in 67.50% F1-measure in identifying toxic instances in code review data and 64.83% in issue comments. While the detection accuracy is far from accurate, this improvement demonstrates the potential of integrating moral and psycholinguistic features in toxicity detection models. These findings highlight the importance of context-specific models that consider the unique communication styles within OSS, where interpersonal and value-driven language dynamics differ markedly from general social media platforms. Future work could focus on refining these models to further enhance detection accuracy, possibly by incorporating community-specific norms and conversational context to better capture the nuanced expressions of toxicity in OSS environments.**

***Index Terms*—moral principles, toxicity, open source, textual analysis**

I. Introduction

Open Source Software (OSS) projects are a societal good and they have increased the speed of digital advancement. However, most new OSS projects fail and many longstanding projects are abandoned by developers [1]. A common reason contributors abandon projects is because of social and emotional factors [2], e.g., a negative experience with other participants through a toxic conversation (as shown in Figure 1). Uncivil language can also deter newcomers from contributing to OSS [3]–[5]. While some OSS projects implement codes of conduct to define acceptable behavior, manually monitoring adherence is challenging for maintainers due to the high volume of daily communications [6]. If project maintainers and participants were able to proactively detect and prevent toxic communications through automated tools, it would lead to more inclusive and sustainable OSS.



garbage traces of old urls in database #40

[redacted] opened this issue on Mar 28, 2020 · 1 comment

User Commented on April 23, 2020

1. You all are probably not using the latest version of the database -- **this is not my problem.**
2. The database does get VACUUM after a large operation before I upload it.
3. Whether the database has been recently VACUUM'd does not affect the integrity of the data or SQL queries -- **what you are suggesting is complete nonsense.**
4. **Expect a ban if you absolute idiots** continue to spam me with "me toos" from issue Art cover for games missing Provenance-Emu/Provenance#1314

Fig. 1. A Toxic Conversation in OSS.

There are several challenges in automatically detecting toxic content in OSS. Directly applying toxicity detection tools trained on other domains (e.g., Google Perspective API) to software engineering (SE) corpora has proven ineffective due to the unique language and norms in SE [7], [8]. Toxicity in OSS is often nuanced, including insults resulting from technical disagreements and comments that are entitled, demanding, or arrogant [9]. Automated toxicity detectors often fail to capture this "covert toxicity", expressed through cynicism and entitlement in SE communications [10], [11]. Recent advances in Large Language Models (LLMs) have spurred interest in their application for toxicity detection on platforms like GitHub [12], [13]. However, LLMs also struggle with SE-specific toxicity (e.g., F-measure of 0.62 [12]), particularly when it lacks clear indicators such as offensive words or hate speech. For instance, statements like "*Such a plugin already exists...You have nobody to blame but yourself*" are misclassified, as LLMs may fail to recognize the subtle condescension and irony present in OSS communications.

Beyond Software Engineering, researchers in the Natural Language Processing (NLP) field have developed methods to automatically detect cyberbullying [14], offensive language [15], and hate speech [16], [17]. For example, using (psycho)linguistic markers derived from the Linguistic Inquiry and Word Count (LIWC) dictionary [18] has proven effective for analyzing toxic language across various communities [19]–[21]. Studies consistently demonstrate that linguistic styles and word choices offer valuable insights into individuals' thoughts, opinions, and emotions [22]–[24].

Moral Foundations Theory (MFT) provides a complementary perspective by proposing that judgments are guided by intuitive appraisals based on personal values [25], [26]. People's values and personal norms affect their (spontaneous) attitude, decision-making process, and what they perceive as good or bad, and moral or immoral [27], [28]. MFT categorizes people's moral reactions and behavior into five foundations or principles which are each further characterized by two opposing values as virtues (good) and vices (bad); care/harm, fairness/cheating, authority/subversion, loyalty/betrayal, and purity/degradation [25], [26]. Previous research in psychology has addressed the connection between moral foundations and hate and showed that morality is a key feature of hatred; hate is connected to core moral beliefs and higher levels of moral emotions (e.g., contempt, anger, and disgust) [29], [30]. Ehsani et al. examined the role of morality in OSS toxicity and highlighted connections between moral principles and toxic interactions on GitHub issue threads [31]. Their qualitative study revealed that *purity/degradation* and *care/harm* were the most frequently observed moral principles, often linked to toxic threads that contain insults.

Building on Ehsani et al's work, we investigate if integrating moral dimensions with psycholinguistic markers can enhance automated toxicity detection in OSS communication. Specifically, we explore three machine learning-based classifiers (Support Vector Machine, Logictic Regression, Gradient Boosting) that leverage psycholinguistic lexicons (LIWC [23]) and MFT dimensions [25], [26] as features to detect toxicity in OSS communications. To ensure generalizability across different communication channels, we detect and quantify expressions of moral values as well as psychological markers in two types of software-related artifacts: (a) GitHub issue comments (dataset by [7]), and (b) Gerrit code review comments (extended dataset by [8]), to investigate the following research questions:

- **RQ1: How does the performance of toxicity models change when psycholinguistic cues are added as features?** We found that using psycholinguistic features, we can achieve a slight improvement (by $\sim 2\%$ in $F1_1$) in performance from baseline in identifying toxic instances across all classifiers.
- **RQ2: How does the performance of toxicity models change when moral values are added as features?** Adding morality on top of psycholinguistic features resulted in a significant jump in the performance of all classifiers ($\sim 2-7\%$ in $F1_1$).
- **RQ3: What types of SE texts are difficult to automatically detect as toxic using our techniques?** We conduct a qualitative error analysis to answer this question. Our observations provide several insights and potential areas of improvement to support future work in toxicity detection, e.g., using domain-specific dictionaries, understanding the context of the discussion, etc.

II. Background and Related Work

A. Moral Foundations Theory

Moral Foundations Theory classifies human behavior into five core principles [25]. Each principle represents a pair of opposing values. **Care/harm:** Rooted in our aversion to suffering—both for ourselves and others—this principle stems from our evolution as mammals, fostering virtues like kindness, compassion, and gentleness, while condemning cruelty and aggression. **Fairness/cheating:** This principle emphasizes justice and rights, associated with the evolutionary concept of reciprocal altruism. **Loyalty/betrayal:** Grounded in our tribal past, this principle encourages patriotism, heroism, trust, and self-sacrifice for the group, holding the ideal of "One for all, and all for one" as virtuous and viewing betrayal of social bonds as immoral. **Authority/subversion:** Informed by hierarchical social structures in primate history, this principle upholds virtues like leadership, deference, and respect for authority and traditions, while perceiving challenges to authority as immoral. **Sanctity/degradation:** Arising from the psychology of disgust and contamination, this principle views the body as a "temple" that can be defiled by immoral acts, promoting an elevated, noble approach to life.

B. Toxicity in Online Communities

Social media and online gaming communities are rife with online toxicity. Several factors contribute to online toxicity, including user anonymity, context collapse, and online disinhibition effect [32]. Different lexicons and annotated datasets have been used to study toxicity and abusive language on social media platforms. State-of-the-art approaches such as classic machine learning models using features such as TF-IDF, part-of-speech tags, and sentiment to detect hateful content [33], [34], deep learning [35], and transformer-based models [36] have been used in toxicity detection. In addition, using LLMs for toxicity has shown promises but it has limitations in generalizability and understanding nuanced forms of toxicity [37], [38]. Koh et al. [39] found that LLMs are unsuitable for blind toxicity evaluations within unverified factors. Recent research showed that the majority of works and models used for toxicity detection "encode" biases against marginalized groups [40]. For instance, Sap et al. [41] found strong associations between toxicity rating and the identities and beliefs of human coders. Consequently, these biases are embedded in off-the-shelf models such as Perspective API, which are vastly used in this domain.

C. Toxicity in Open Source

Similar to other online forums, OSS communication channels are not free of toxic content. Toxic behaviors have been responsible for causing stress and burnout [7], reduced developer motivation and productivity [42], leading to team

TABLE I
OVERVIEW OF OUR EXPERIMENT DATASETS

Dataset	Source	#Toxic	#Non-toxic	Total
Issue Comments	Github	101	303	404
Code Review	Gerrit	3,757	11,271	15028

attrition [43]. Systems to manage toxic comments have been used for several popular OSS platforms, e.g., GitHub. Miller et al. [9] studied different aspects of toxicity on GitHub, including how developers react to the current moderation mechanism, noting that the current system does not always resolve the problem and that a significant amount of burden continues to be placed on the project maintainers. Therefore automated interventions to detect and flag toxic conversations in OSS are necessary. Despite the presence of several state-of-the-art techniques for toxicity detection in blogs and tweets, applying these tools directly to the SE-related text is not effective due to several reasons such as longer texts containing references to code and other SE-specific factors [8], [11]. Towards domain-specific detection of toxic content, Raman et al. [7] proposed a machine-learning-based technique to detect toxic issue comments on GitHub. Their model performed best when using only two features, the Stanford Politeness score and Google Perspective API. This model was found to be not generalizable across other types of developer communications, such as code reviews and chats [8]. Another study detected offensive language on Stack Overflow, GitHub, and chats by using the Perspective API and regular expressions [44] as features. LLMs also have shown promise in toxicity detection in OSS; however, they struggle with identifying passive-aggressiveness and context-dependent toxic language [12].

Current approaches in toxicity detection are not generalizable in terms that they miss classifying toxic/biased words that are salient and cultural or domain-specific (e.g., in SE context) and do not show up in off-the-shelf lexicons and datasets. Preliminary results from analyzing toxicity in OSS through the lens of moral values have shown promise in enhancing the understanding of toxic behaviors [31], and our work builds upon the previous works in this domain and investigates the change in the model's effectiveness when using additional features based on psycholinguistic scores and moral values.

III. Methodology

We developed a suite of machine learning-based techniques for automatically identifying toxic SE communications on GitHub. Our approach takes as input a text segment, either an issue comment or a code review comment, and classifies them as toxic or non-toxic.

A. Datasets

We leverage two publicly available labeled datasets of toxic communications in the SE domain: (1) a dataset of 1,597 GitHub *issue comments* (1,496 non-toxic, 101 toxic) [7], and (2) a dataset of 19,571 Gerrit *code review* comments (15,819 non-toxic, 3,757 toxic). The *code review* dataset is an extension of [8]. Both datasets are available in a public GitHub repository [45].

TABLE II
FEATURES TO IDENTIFY TOXICITY

Feature Set	Feature	Value
Baseline	Politeness	0 to +1
	Google's Perspective API	0 to +1
Psycholinguistic	LIWC (Analytic)	1 to 99
	LIWC (Clout)	1 to 99
	LIWC (Authentic)	1 to 99
	LIWC (Tone)	1 to 99
	LIWC (Swear)	0 to 100
	Sentiment	-1 to +1
Morality	Care (Virtue and Vice)	-1 to +1
	Fairness (Virtue and Vice)	-1 to +1
	Ingroup (Virtue and Vice)	-1 to +1
	Authority (Virtue and Vice)	-1 to +1
	Purity (Virtue and Vice)	-1 to +1

The existing datasets are imbalanced, with less than 6% and 20% toxic instances in the *issue comments* and *code review* datasets, respectively. For our experiments, we select a representative subset of the original data with a ratio of 1:3 toxic to non-toxic instances. More specifically, we use undersampling techniques to reduce the size of the non-toxic class but retain all the toxic instances in our datasets. Table I shows the details of our datasets.

B. ML Classification

We investigated several supervised machine learning-based approaches to automatically identify toxic texts. We describe the textual features followed by the suite of machine learning algorithms investigated for this classification task.

1) Features: We present three sets of features: Baseline (2 features), Psycholinguistic (6 features), and Morality (10 features). Table II lists the features in each set with their value range; descriptions of why and how we extract each feature follow.

Baseline Features: The state-of-the-art domain-specific toxicity detector for SE [7], leverages Google's Perspective API [46] and Stanford's Politeness Detector [47]. Of the additional feature combinations [7] experimented with (e.g., length of the text, subjectivity score, no. of anger words from LIWC), their model performed best when using only Politeness and toxicity score from the Perspective API. Hence, we use these two features in our baseline feature set.

Psycholinguistic Features: To understand how toxicity is represented in text, we use a subset of features from the Linguistic Inquiry and Word Count (LIWC) [23]. Miller et al. found that the most prevalent forms of toxicity in OSS are entitled, trolling, arrogant, and unprofessional comments from project users, and insults arising from technical disagreements [9]. We use the following LIWC features to identify the presence or the lack of these traits: (a) *Clout* for entitlement, (b) *Authentic* for trolling, (c) *Tone* for arrogance, (d) *Analytic* for unprofessionalism, and (e) *Swear words* for insults. These dimensions capture the linguistic and psychological cues relevant to OSS toxicity.

Sentiment is used for understanding people's emotions and affective states and is highly related to contentiousness [48]. In this work, we leverage Valence Aware Dictionary and sEntiment Reasoner (VADER) [49] to get the sentiment scores of texts in our dataset. VADER performs well on short, informal content like OSS communications and is widely used in different domains, specifically SE [50]. We used the compound score in VADER to get a single unidimensional measure of sentiment. The compound score sums the valence scores of each word in the lexicon and returns a "normalized, weighted composite score" for a given sentence.

Morality Features: As additional feature sets, we leverage Moral Foundations Theory [25], [51] to investigate the relationship between morality and toxicity. Our feature design is based on the premise that people's emotions, ideology, and culture can be reflected in their use of language [22]. Hatred and tension (online or offline) may be the result of differences in values (moral or personal). Therefore, finding representations of such information in user-generated texts can help in better understanding toxicity in online interactions. The Moral Foundations Dictionary (MFD) enables the measurement of MFT based on text data by associating 324 words with virtues and vices from the MFT [25], [51]. To extract moral values, we used MFDE, an enhanced version of MFD [24]. Compared to the original MFD, the enhanced lexicon consists of about 4,636 terms that were syntactically disambiguated and manually pruned and verified. To find morality in texts, we apply Distributed Dictionary Representations (DDR) [52]. DDR first computes the average of each dictionary and then computes the "loading" of a dictionary on a particular piece of text. We used MFDE as the seed words and created ten separate lists of words representing each moral value (vices and virtues of five moral categories). We used word2vec [53] to create vector representations of the words.

2) Classifiers: We trained multiple supervised machine learning-based classifiers in our study using Python scikit-learn package [54]. We explored other classifiers (e.g., Random Forests); however, we do not discuss them here, since they yielded significantly inferior results. Here, we provide an overview and explanation of our classifier choices.

Support Vector Machines (SVM) is a non-probabilistic classifier that maps input data into a feature space that maximizes the gap between the classification categories; i.e., *toxic*, and *non-toxic* in our study. SVM has been observed to achieve high accuracy in predicting toxic content in SE [7], [44].

Logistic Regression (LR) is a discriminative classification model that predicts the class by calculating the probability for each class and choosing the class with the highest probability. In our case, the class probability is the likelihood of a text being toxic. LR has been widely used for binary text classification in SE [55].

Gradient Boosting (GB) is an ensemble-based classification framework where a sequence of decision trees is constructed, and each tree minimizes the residual error of the preceding sequence of trees. Ensemble classifiers have been used in predicting offensive language in SE texts [44].

IV. Evaluation Study

A. Evaluation Metrics

We use measures that are widely used for evaluation in information retrieval: precision, recall, F-measure, and ROC. To measure the fraction of automatically identified texts that are indeed toxic, we use precision, the ratio of true positives (TP) over the sum of true and false positives (FP). To see how often our approaches miss toxic data instances, we use recall, the ratio of true positives over the sum of true positives and false negatives (FN). F-measure combines these measures by harmonic mean. To measure robustness, we use ROC (Receiver Operating Characteristics), which represents the degree of separability between prediction classes. We compute precision, recall, and f-measure for non-toxic (class 0) and toxic (class 1) classes. Lastly, because our data is not completely balanced, we compute MCC (Matthews correlation coefficient), which is a correlation coefficient between observed and predicted binary classifications that is well suited for unbalanced data.

B. Procedures

We configured classifiers and ran them as follows.

1) Hyperparamter Tuning: We investigated several hyperparameters to adjust each classifier (using scikit-learn), and the following configurations produced the best classifications. For the rest of the parameters, we used the default values offered by scikit-learn.

- Support-Vector Machine (SVM): We used the *LinearSVC* class with the following parameters: C=10, max_iter=10000.
- Logistic Regression (LR): We used the *LogisticRegression* class with following parameters: C=1, max_iter=4000, multi_class='multinomial'.
- Gradient-Boosting (GB): We used the *GradientBoostingClassifier* class with the following parameters: learning_rate=1.0, n_estimators=1000, random_state = 0, max_depth=10, max_features='sqrt', min_samples_leaf=2.

2) Evaluation Process (RQ1, RQ2): For RQ1 and RQ2, results from the classifiers were obtained using stratified 5-fold cross validation i.e., the dataset was partitioned into five equal-sized sub-samples with stratification, ensuring that the original distribution of toxic and non-toxic instances is retained in each sub-sample.

3) Evaluation Process (RQ3): For RQ3, we developed two separate sets of test data from issue comments and code reviews to evaluate the effectiveness of our models, and investigate the properties of SE texts that were found to be challenging to automatically categorize. For the code review test set, we randomly selected 100 toxic and 100 non-toxic reviews from the original dataset, before creating the training described in the Methodology section. For the second dataset, since the issue comments data is small in size and only consists of 101 toxic instances, we additionally utilized the held out labeled issue test set created by [7], which consists of 194 issue threads labeled as toxic or non-toxic. Since the issue threads tend to be longer compared to our training data, we used the length of texts to filter out threads longer than 1,700 characters to have data similar to our training set. Our final issue comments test set consists of 58 issue comments (25 toxic, 33 non-toxic).

TABLE III

TOXICITY DETECTION RESULTS (ISSUE COMMENTS) (5-FOLD CROSS VALIDATION)

		P_0	R_0	$F1_0$	ROC_0	P_1	R_1	$F1_1$	ROC_1	MCC
Baseline	*LinearSVC*	84.13	97.00	90.09	71.29	84.92	45.57	58.87	71.29	54.05
	GradientBoosting	86.24	87.67	86.94	73.02	61.48	58.38	59.83	73.02	46.87
	LogisticRegression	83.34	98.00	90.05	69.83	88.89	41.67	56.08	69.83	53.26
Baseline+Psycholinguistic	*LinearSVC*	84.52	96.33	90.03	71.95	82.52	47.57	60.03	71.95	**54.14**
	GradientBoosting	86.27	92.00	89.01	74.24	71.97	56.48	**62.75**	**74.24**	52.98
	LogisticRegression	83.72	97.67	90.15	70.62	86.91	43.57	57.86	70.62	53.89
Baseline+Psycholinguistic+Morality	*LinearSVC*	87.50	90.33	88.79	75.90	70.59	61.48	64.74	75.90	**54.64**
	GradientBoosting	87.65	89.67	88.56	76.02	68.69	62.38	**64.83**	**76.02**	54.06
	LogisticRegression	84.85	95.00	89.62	72.26	77.36	49.52	60.13	72.26	52.55

TABLE IV

TOXICITY DETECTION RESULTS (CODE REVIEW) (5-FOLD CROSS VALIDATION)

		P_0	R_0	$F1_0$	ROC_0	P_1	R_1	$F1_1$	ROC_1	MCC
Baseline	*LinearSVC*	84.11	89.93	86.93	69.42	61.82	48.91	54.60	69.42	42.24
	Gradient Boosting	81.78	75.01	77.95	62.75	42.49	50.50	45.48	62.75	24.85
	Logistic Regression	84.15	89.90	86.93	69.49	61.81	49.08	54.71	69.49	42.32
Baseline+Psycholinguistic	*LinearSVC*	83.62	95.21	89.04	69.55	75.24	43.88	55.42	69.55	**47.96**
	Gradient Boosting	87.10	79.67	82.91	72.76	57.04	65.85	**60.09**	**72.76**	44.73
	Logistic Regression	84.39	92.73	88.36	70.56	68.91	48.40	56.85	70.56	46.82
Baseline+Psycholinguistic+Morality	*LinearSVC*	84.87	94.58	89.46	71.92	75.41	49.25	59.46	71.92	51.37
	Gradient Boosting	88.60	88.25	88.07	77.35	71.09	66.45	**67.50**	**77.35**	**57.03**
	Logistic Regression	85.82	93.00	89.25	73.35	72.01	53.70	61.43	73.35	51.94

C. Results and Discussion

Tables III and IV present the precision, recall, F-measure, ROC, and MCC, for each classification labels (0 for non-toxic, and 1 for toxic). To configure the feature sets, we first combine baseline features with psycholinguistic features (baseline + psycholinguistic), and then combine baseline features with both psycholinguistic and morality features (baseline + psycholinguistic + morality). We compare the baseline with both of these configurations.

*a) **RQ1.** How does the performance of toxicity models change when psycholinguistic cues are added as features?:* When using *baseline+psycholinguistic* features, we observe an improvement in performance from baseline in identifying toxic instances across all classifiers.

In the issue comments dataset (Table III), the $F1_1$ improves from 58.87% to 60.03% for *LinearSVC*, from 59.83% to 62.75% for *GradientBoosting*, and from 56.08% to 57.86% for *LogisticRegression*. When considering ROC_1 and ROC_2, we observe slight improvement ($\sim$ 1%) from the baseline, across all classifiers. Overall (except MCC), *GradientBoosting* performs better compared to the rest of the classifiers. However, when considering MCC, *LinearSVC* provides the best performance (54.14%).

In the code review dataset (Table IV), the $F1_1$ improves from 54.60% to 55.42% for *LinearSVC*, from 45.48% to 60.09% for *GradientBoosting*, and from 54.71% to 56.85% for *LogisticRegression*. Overall (except MCC), *GradientBoosting* performs better than the rest of the classifiers; we observe a significant improvement from the baseline with $\sim$15% (45.48% to 60.09%) in $F1_1$, $\sim$10% (62.75% to 72.76%) in ROC_1. However, for MCC that adjusts for class imbalance, *LinearSVC* provides better performance (47.96%). *LinearSVC* also provides a high precision (P_1 = 75.24%), but a lower recall (R_1 = 43.88%), indicating that it is more restrictive in labeling a code review comment as toxic.

We further looked into the values of psycholinguistic features in both toxic and non-toxic comments in our datasets. In both datasets, the average value of “Swear words” is significantly higher in toxic comments, 2.72 vs. 0.01 in toxic and non-toxic comments in issue comments, and 6.3 vs. 0.11 in the code review dataset. We also observe that, the average value of “Analytic” is lower in toxic comments, 30.96 vs 52.3 in toxic and non-toxic comments in issue comments, and 40.99 vs 44.8 in the code review dataset. Since, analytic scores represent the degree of formal, logical, and hierarchical thinking [56], a higher value shows that non-toxic comments were well-thought-out and thus exhibited more professionalism. Additionally, we found that the average value of “Authentic” scores is higher in toxic comments. Example of texts that have low authenticity scores include texts where a person is being socially cautious [57] and thus not involved in toxic behaviors such as trolling. These results also confirm [9]’s empirical observation of the existence of trolling, insults, and unprofessional comments as prevalent forms of OSS toxicity. We do not observe any significant patterns with the rest of the psycholinguistic features across both the datasets, which indicates the challenges of generalizing toxicity detection techniques for different types of data.

*b) **RQ2.** How does the performance of toxicity models change when moral values are added as features?:* Adding morality on top of *baseline+psycholinguistic* resulted in a significant jump in the performance of all classifiers.

In the issue comments dataset (Table III), while *LinearSVC* benefits the most from the addition of moral values, with around 4% increase in the $F1_1$, *GradientBoosting* achieves the highest performance of all models ($F1_1$ = 64.83%). Furthermore, using this model with *base-*

line+psycholinguistic+morality results in the highest ROC_1 as well as ROC_0. When considering MCC, we observe a slightly better result with *LinearSVC*. In the code review dataset (Table IV), the *GradientBoosting*'s performance jumps from 60.09% with *baseline+psycholinguistic* to 67.5%. MCC, ROC_1, and ROC_0 also significantly improved using this model.

We further looked into the morality features in both toxic and non-toxic comments and found that the average of all moral values are higher in toxic comments compared to the non-toxic ones in both datasets used in this paper. In both datasets, the average value of "degradation" (purity-vice) is significantly higher, 0.41 ± 0.05 vs. 0.35 ± 0.08 in toxic and non-toxic comments in issue comments, and 0.380 ± 0.07 vs. 0.34 ± 0.08 ones in the code review dataset.

Based on MFT, purity/degradation foundation is influenced by the "psychology of disgust and contamination". The result of our analysis is in line with previous work that found hate is conceptually closer to disgust and contempt, compared to anger and dislike [58].

c) ***RQ3.*** *What types of SE texts are difficult to automatically detect as toxic using our techniques?:* To answer RQ3, we performed classification error analysis. Specifically, we investigated the following questions:

RQ3.1. *What instances are misclassified using all feature sets?* We qualitatively analyzed the False Positives (FP) and False Negatives (FN) using all features, and GB as the classifier. We chose GB since it achieves the best classification performance overall. We found that 30 instances (4 issue comments, 26 code reviews) were marked as FP, and 45 instances (7 issue comments, 38 code reviews) were marked as FN, out of a total of 258 instances in our evaluation dataset.

The analysis procedure consisted of the following steps: (1) first we collected the data instances that were marked FP, and instances that were marked FN, using all features. (2) Following an open coding procedure [59], the authors of this paper independently studied the instances from step 1. We manually analyzed the conversations to identify the characteristics of conversations in each category (FP and FN), and recorded comments and reflections from the manual analysis in the form of short phrases, e.g., "contains entire sentence(s) written in uppercase". These insights helped us investigate additional characteristics that our features failed to capture. (3) The common observed characteristics in each category (FP and FN) were grouped. The analysis was performed in an iterative approach composed of multiple sessions, which helped in generalizing the hypotheses and revising the characteristics.

We manually analyzed the 30 instances marked FP when using all features, and observed that most of the instances contained domain-specific words that were incorrectly identified as toxic. For instance, *"If we add this board, will we have to start killing off the rest?..."*, where 'killing' refers to terminating an application. These errors could be handled using a domain specific dictionary. Other errors included misidentifying self-deprecating words as toxic. For instance, *"Oops, that was dumb. I actually had caught this while you were reviewing and fixed it."*. This is a limitation of using lexicon-based features, as words can have different meanings in different contexts.

We manually analyzed the 45 instances marked FN when using all features, and observed that toxicity in several of the instances could be understood only in the context of the discussion and with prior project-specific knowledge. For instance, *"extra stupid-people safe: do we want 'pwd -P' here, in case someone runs this in a symlinked directory? :)"*. The other toxic comments were insults without containing any negative terms such as swear words. For instance, in the comment *"...The questions are placed looking for fixes not closed stamps. Did they give you that stamp in Kindergarten?..."*, the words "stamps" and "Kindergarten" refer to a false sense of achievement. We also notice that some comments included entire sentences written in caps (or uppercase), which insinuates shouting or harsh tone, e.g., *"...@user already said why, stop asking. BUT CANT YOU JUST CHANGE BESTPLANET IN THE CODE AND SET IT TO PLANET 22..."*. These instances are misclassified by our models, since we are not leveraging the format of the text as a feature.

RQ3.2. *What instances are misclassified by other feature sets but not morality?*
Overall, we observed the best performance when all features are used; specifically adding morality features significantly improves the performance (5% with SVM for issue comments, 7% with GB on code reviews). In this question, we investigate and gather insights on how adding morality features improves the classification of toxicity in OSS.

We selected the instances which were misclassified by GB classifier using only baseline and psycholinguistic features, but correctly classified when all feature sets are used (including morality). Using this selection procedure, we collected a set of 33 FPs (3 issue comments, 30 code reviews), and 22 FNs (2 issue comments, 20 code reviews). The analysis procedure was the same as RQ3.1.

We qualitatively analyzed the 33 FPs, and observed that the majority of comments contain domain-specific words. For instance, *"if you call response.body() .string() twice like this response.body().string() response.body().string() you will get this unsolvable error"* were misclassified using *baseline+psycholinguistic* features, but adding morality on top of that reduced model confusion. Another instance is *"Bad merge? You're shadowing oslo.config.cfg"* which was mistakenly classified as toxic using the baseline and *baseline+psycholinguistic* features. However, using moral words in addition to other features complemented each other, as also shown in other studies [52], which resulted in decreasing classification error in our models.

We qualitatively analyzed the 22 FNs, and observed that the majority of comments misclassified by other models include negative words such as "damn", "stupid", and "ugly" which are not included as swear words in LIWC. One of the shortcomings of LIWC is that the analysis is on the word level, which results in missing semantically related words in many cases. Using morality as a feature and the DDR method helped with capturing semantic similarity between words and concepts and going beyond word count. For instance, a comment *"this is pretty ugly. create it in one assignment"* is classified as non-toxic by the baseline and *baseline+psycholinguistic* features, but using moral words improved the performance of the model and resulted in a better accuracy.

Summary of Key Findings and Takeway. We found that psycholinguistic markers, such as "Swear words" and "Analytic" scores, proved valuable in distinguishing toxic from non-toxic content, while moral values (particularly "Degradation") captured nuanced forms of toxicity often linked to moral judgments. These features can be leveraged to build more accurate toxicity detectors for SE domains.

However, several challenges persist. SE-specific terms and indirect toxicity, including sarcasm, entitlement, and passive-aggressiveness, are often misclassified without domain-aware dictionaries and contextual understanding. Additionally, the complexity of OSS communication requires models that can account for conversation dynamics and project-specific knowledge. Future research should focus on refining these features and developing models capable of handling covert toxicity in diverse SE contexts, potentially by incorporating contextual markers and adaptive, SE domain-specific lexicons.

V. Conclusion and Future Work

This paper investigates the usefulness of psycholinguistic cues and moral values with toxicity in Open Source Software (OSS). Due to the use of domain-specific words and jargon in OSS, the majority of toxicity detection tools do not correctly identify toxic language in this domain. Therefore, there is a need to augment the current methods with features and information that are insightful for such task. A recent study [9] in OSS found features such as trolling, arrogant, and unprofessional comments mostly prevalent to toxicity in OSS. In this work, we leverage Linguistic Inquiry and Word Count (LIWC) to map these concepts to a subset of linguistic features in comments labeled as toxic and non-toxic. In addition, previous studies showed that people's instant emotions and ideologies can be reflected in their use of language. Moral Foundations Theory captures the reactions of people. We use an extended version of Moral Foundations Dictionary [24] to operationalize morality in text. Using these two sets of additional feature sets on top of a benchmark baseline model, we show that using moral values on top of psycholinguistic features improves the performance of our toxicity classifiers.

This study opens avenues for enhancing toxicity detection in OSS. Our scripts are available in our replication package [60]. Future work can focus on building larger, more balanced datasets to improve the generalizability of toxicity models across diverse OSS contexts. Additionally, moving beyond binary classification toward multi-dimensional toxicity categorization (e.g., personal harassment, implicit bias) will capture the complexity of toxic interactions, particularly those affecting underrepresented groups [61]. Additionally, developing models that understand the evolution of conversations—recognizing how tone, sarcasm, and passive-aggressiveness unfold over exchanges—could improve the detection of covert toxicity.

References

[1] J. Coelho and M. T. Valente, "Why modern open source projects fail," in *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, ser. ESEC/FSE 2017. New York, NY, USA: Association for Computing Machinery, 2017, p. 186–196.

[2] M. Gerosa, I. Wiese, B. Trinkenreich, G. Link, G. Robles, C. Treude, I. Steinmacher, and A. Sarma, "The shifting sands of motivation: Revisiting what drives contributors in open source," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, 2021, pp. 1046–1058.

[3] I. Steinmacher, T. Conte, M. A. Gerosa, and D. Redmiles, "Social barriers faced by newcomers placing their first contribution in open source software projects," in *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, ser. CSCW '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 1379–1392.

[4] H. S. Qiu, Y. L. Li, S. Padala, A. Sarma, and B. Vasilescu, "The signals that potential contributors look for when choosing open-source projects," *Proc. ACM Hum.-Comput. Interact.*, vol. 3, no. CSCW, nov 2019.

[5] R. Ehsani, M. M. Imran, R. Zita, K. Damevski, and P. Chatterjee, "Incivility in open source projects: A comprehensive annotated dataset of locked github issue threads," in *2024 IEEE/ACM 21st International Conference on Mining Software Repositories (MSR)*, 2024, pp. 515–519.

[6] P. Tourani, B. Adams, and A. Serebrenik, "Code of conduct in open source projects," in *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 2017, pp. 24–33.

[7] N. Raman, M. Cao, Y. Tsvetkov, C. Kästner, and B. Vasilescu, "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*. New York, NY, USA: Association for Computing Machinery, 2020, p. 57–60.

[8] J. Sarker, A. K. Turzo, and A. Bosu, "A benchmark study of the contemporary toxicity detectors on software engineering interactions," *2020 27th Asia-Pacific Software Engineering Conference (APSEC)*, pp. 218–227, 2020.

[9] C. Miller, S. Cohen, D. Klug, B. Vasilescu, and C. Kästner, ""did you miss my comment or what?" understanding toxicity in open source discussions," in *2022 IEEE/ACM 44th International Conference on Software Engineering (ICSE)*, 2022, pp. 710–722.

[10] A. Lees, D. Borkan, I. D. Kivlichan, J. Nario, and T. Goyal, "Capturing covertly toxic speech via crowdsourcing," in *HCINLP*, 2021.

[11] H. S. Qiu, B. Vasilescu, C. Kästner, C. Egelman, C. Jaspan, and E. Murphy-Hill, "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)*, 2022, pp. 41–55.

[12] S. Mishra and P. Chatterjee, "Exploring chatgpt for toxicity detection in github," in *Proceedings of the 2024 ACM/IEEE 44th International Conference on Software Engineering: New Ideas and Emerging Results*, ser. ICSE-NIER'24. New York, NY, USA: Association for Computing Machinery, 2024, p. 6–10.

[13] L. Li, L. Fan, S. Atreja, and L. Hemphill, ""hot" chatgpt: The promise of chatgpt in detecting and discriminating hateful, offensive, and toxic comments on social media," *ACM Trans. Web*, vol. 18, no. 2, Mar. 2024.

[14] R. M. Kowalski, S. P. Limber, and P. W. Agatston, *Cyberbullying: Bullying in the Digital Age*, 2nd ed. Wiley Publishing, 2012.

[15] G. Xiang, B. Fan, L. Wang, J. Hong, and C. Rose, "Detecting offensive tweets via topical feature discovery over a large scale twitter corpus," in *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, ser. CIKM '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 1980–1984.

[16] F. D. Vigna, A. Cimino, F. Dell'Orletta, M. Petrocchi, and M. Tesconi, "Hate me, hate me not: Hate speech detection on facebook," in *ITASEC*, 2017.

[17] M. Chaudhary, C. Saxena, and H. M. Meng, “Countering online hate speech: An nlp perspective,” 2021.

[18] Y. R. Tausczik and J. W. Pennebaker, “The psychological meaning of words: Liwc and computerized text analysis methods,” *Journal of Language and Social Psychology*, vol. 29, no. 1, pp. 24–54, 2010.

[19] J. An, H. Kwak, C. S. Lee, B. Jun, and Y.-Y. Ahn, “Predicting anti-asian hateful users on twitter during covid-19,” in *Findings of the Association for Computational Linguistics: EMNLP 2021*, 2021, pp. 4655–4666.

[20] S. C. d. Silva, T. C. Ferreira, R. M. S. Ramos, and I. Paraboni, “Data-driven and psycholinguistics-motivated approaches to hate speech detection,” *Computación y Sistemas*, vol. 24, no. 3, pp. 1179–1188, 2020.

[21] J. Salminen, M. Hopf, S. A. Chowdhury, S.-g. Jung, H. Almerekhi, and B. J. Jansen, “Developing an online hate classifier for multiple social media platforms,” *Human-centric Computing and Information Sciences*, vol. 10, no. 1, pp. 1–34, 2020.

[22] H. C. Triandis, “The self and social behavior in differing cultural contexts.” *Psychological review*, vol. 96, no. 3, p. 506, 1989.

[23] R. L. Boyd, A. Ashokkumar, S. Seraj, and J. W. Pennebaker, “The development and psychometric properties of liwc-22,” *Austin, TX: University of Texas at Austin*, 2022.

[24] R. Rezapour and J. Diesner, “Expanded morality lexicon,” 2019, https://doi.org/10.13012/B2IDB-3805242_V1.1.

[25] J. Graham, J. Haidt, S. Koleva, M. Motyl, R. Iyer, S. P. Wojcik, and P. H. Ditto, “Moral foundations theory: The pragmatic validity of moral pluralism,” in *Advances in experimental social psychology*. Elsevier, 2013, vol. 47.

[26] J. Haidt and C. Joseph, “Intuitive ethics: How innately prepared intuitions generate culturally variable virtues,” *Daedalus*, vol. 133, no. 4, pp. 55–66, 2004.

[27] J. Haidt, “The emotional dog and its rational tail: a social intuitionist approach to moral judgment.” *Psychological review*, vol. 108, no. 4, p. 814, 2001.

[28] R. Rezapour, L. Dinh, and J. Diesner, “Incorporating the measurement of moral foundations theory into analyzing stances on controversial topics,” in *Proceedings of the 32nd ACM Conference on Hypertext and Social Media*, 2021.

[29] C. Pretus, J. L. Ray, Y. Granot, W. A. Cunningham, and J. J. Van Bavel, “The psychology of hate: Moral concerns differentiate hate from dislike,” Jun 2018.

[30] M. Atari, A. M. Davani, D. Kogon, B. Kennedy, N. A. Saxena, I. A. Anderson, and M. Dehghani, “Morally homogeneous networks and radicalism,” Jan 2021.

[31] R. Ehsani, R. Rezapour, and P. Chatterjee, “Exploring moral principles exhibited in oss: A case study on github heated issues,” in *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2023. New York, NY, USA: Association for Computing Machinery, 2023, p. 2092–2096. [Online]. Available: https://doi.org/10.1145/3611643.3613077

[32] J. Suler, “The online disinhibition effect,” *Cyberpsychology & Behavior: The Impact of the Internet, Multimedia and Virtual Reality on Behavior and Society*, vol. 7, no. 3, p. 321–326, Jun 2004.

[33] T. Davidson, D. Warmsley, M. Macy, and I. Weber, “Automated hate speech detection and the problem of offensive language,” in *Proceedings of the international AAAI conference on web and social media*, vol. 11, no. 1, 2017, pp. 512–515.

[34] A. M. Founta, C. Djouvas, D. Chatzakou, I. Leontiadis, J. Blackburn, G. Stringhini, A. Vakali, M. Sirivianos, and N. Kourtellis, “Large scale crowdsourcing and characterization of twitter abusive behavior,” in *Twelfth International AAAI Conference on Web and Social Media*, 2018.

[35] S. Agrawal and A. Awekar, “Deep learning for detecting cyberbullying across multiple social media platforms,” in *European conference on information retrieval*. Springer, 2018, pp. 141–153.

[36] V. Isaksen and B. Gambäck, “Using transfer-based language models to detect hateful and offensive language online,” in *Proceedings of the Fourth Workshop on Online Abuse and Harms*, 2020, pp. 16–27.

[37] D.-H. Lee, H. Cho, W. Jin, J. Moon, S. Park, P. Röttger, J. Pujara, and R. K.-w. Lee, “Improving covert toxicity detection by retrieving and generating references,” in *Proceedings of the 8th Workshop on Online Abuse and Harms (WOAH 2024)*. Mexico City, Mexico: Association for Computational Linguistics, Jun. 2024, pp. 266–274.

[38] Y.-S. Wang and Y. Chang, “Toxicity detection with generative prompt-based inference,” 2022.

[39] H. Koh, D. Kim, M. Lee, and K. Jung, “Can llms recognize toxicity? a structured investigation framework and toxicity metric,” 2024.

[40] X. Zhou, M. Sap, S. Swayamdipta, Y. Choi, and N. A. Smith, “Challenges in automated debiasing for toxic language detection,” in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, 2021, pp. 3143–3155.

[41] M. Sap, S. Swayamdipta, L. Vianna, X. Zhou, Y. Choi, and N. A. Smith, “Annotators with attitudes: How annotator beliefs and identities bias toxic language detection.” Seattle, United States: Association for Computational Linguistics, Jul. 2022, pp. 5884–5906.

[42] A. Bosu and J. C. Carver, “Impact of peer code review on peer impression formation: A survey,” *2013 ACM / IEEE International Symposium on Empirical Software Engineering and Measurement*, pp. 133–142, 2013.

[43] I. Steinmacher, I. Wiese, A. P. Chaves, and M. A. Gerosa, “Why do newcomers abandon open source software projects?” in *2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, 2013, pp. 25–32.

[44] J. Cheriyan, B. T. R. Savarimuthu, and S. Cranefield, “Towards offensive language detection and reduction in four software engineering communities,” in *Evaluation and Assessment in Software Engineering*, ser. EASE 2021. New York, NY, USA: Association for Computing Machinery, 2021, p. 254–259.

[45] “Cmustrudel.” [Online]. Available: https://github.com/CMUSTRUDEL/toxicity-detector

[46] G. P. API, “https://www.perspectiveapi.com/,” accessed Aug, 2022.

[47] C. Danescu-Niculescu-Mizil, M. Sudhof, D. Jurafsky, J. Leskovec, and C. Potts, “A computational approach to politeness with application to social factors,” in *ACL*, 2013.

[48] M. Pennacchiotti and A.-M. Popescu, “Detecting controversies in twitter: a first study,” in *Proceedings of the NAACL HLT 2010 workshop on computational linguistics in a world of social media*, 2010, pp. 31–32.

[49] C. J. Hutto and E. Gilbert, “Vader: A parsimonious rule-based model for sentiment analysis of social media text,” in *ICWSM*, 2014.

[50] B. Lin, F. Zampetti, G. Bavota, M. Di Penta, M. Lanza, and R. Oliveto, “Sentiment analysis for software engineering: How far can we go?” in *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*, 2018, pp. 94–104.

[51] J. Graham, J. Haidt, and B. A. Nosek, “Liberals and conservatives rely on different sets of moral foundations.” *Journal of personality and social psychology*, vol. 96, no. 5, p. 1029, 2009.

[52] J. Garten, J. Hoover, K. M. Johnson, R. Boghrati, C. Iskiwitch, and M. Dehghani, “Dictionaries and distributions: Combining expert knowledge and large scale textual data content analysis,” *Behavior research methods*, vol. 50, no. 1, pp. 344–361, 2018.

[53] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” 2013.

[54] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, “Scikit-learn: Machine learning in python,” *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[55] P. Chatterjee, K. Damevski, N. Kraft, and L. Pollock, “Automatically Identifying the Quality of Developer Chats for Post Hoc Use,” in *Transactions on Software Engineering and Methodology (TOSEM)*, ser. TOSEM ’20, 2020.

[56] J. W. Pennebaker, C. K. Chung, J. Frazee, G. M. Lavergne, and D. I. Beaver, “When small words foretell academic success: The case of college admissions essays,” *PLOS ONE*, vol. 9, p. 1–10, Dec 2015.

[57] M. L. Newman, J. W. Pennebaker, D. S. Berry, and J. M. Richards, “Lying words: Predicting deception from linguistic styles,” *Personality and Social Psychology Bulletin*, vol. 29, no. 5, pp. 665–675, 2003.

[58] C. A. Martínez, J.-W. van Prooijen, and P. A. Van Lange, “Hate: Toward understanding its distinctive features across interpersonal and intergroup targets.” *Emotion*, vol. 22, no. 1, p. 46, 2022.

[59] P. Runeson, M. Host, A. Rainer, and B. Regnell, *Case Study Research in Software Engineering: Guidelines and Examples*, 1st ed. Wiley Publishing, 2012.

[60] “Anonymized repository.” [Online]. Available: https://anonymous.4open.science/r/toxicity-morality-analysis-2A1F/README.md

[61] S. Zacchiroli, “Gender differences in public code contributions: A 50-year perspective,” *IEEE Software*, vol. 38, no. 2, pp. 45–50, 2021.