Detecting Interpersonal Conflict in Issues and Code Review: Cross Pollinating Open- and Closed-Source Approaches

Huilian Sophie Qiu
Carnegie Mellon University
Pittsburgh, PA, USA
hsqq@cmu.edu

Bogdan Vasilescu
Carnegie Mellon University
Pittsburgh, PA, USA
vasilescu@cmu.edu

Christian Kästner
Carnegie Mellon University
Pittsburgh, PA, USA
kaestner@cs.cmu.edu

Carolyn Egelman
Google
Sunnyvale, CA, USA
cegelman@google.com

Ciera Jaspan
Google
Sunnyvale, CA, USA
ciera@google.com

Emerson Murphy-Hill
Google
Sunnyvale, CA, USA
emersonm@google.com

ABSTRACT

Interpersonal conflict in code review, such as toxic language or an unnecessary pushback, is associated with negative outcomes such as stress and turnover. Automatic detection is one approach to prevent and mitigate interpersonal conflict. Two recent automatic detection approaches were developed in different settings: a toxicity detector using text analytics for open source issue discussions and a pushback detector using logs-based metrics for corporate code reviews. This paper tests how the toxicity detector and the pushback detector can be generalized beyond their respective contexts and discussion types, and how the combination of the two can help improve interpersonal conflict detection. The results reveal connections between the two concepts.

LAY ABSTRACT

Software engineers often communicate with one another on platforms that support tasks like discussing bugs and inspecting each others' code. Such discussions sometimes contain interpersonal conflict, which can lead to stress and abandonment. In this paper, we investigate how to automatically detect interpersonal conflict, both by analyzing the text of the what the engineers are saying and by analyzing the properties of that text.

1 INTRODUCTION

In online communities and offline workplaces alike, interpersonal conflicts, understood broadly as including hostility, hate, aggression, toxic language, bullying, etc, has been a major concern and topic of research [3, 42, 47]. The consensus is not only that such forms of interaction are antisocial, but also that they are all associated with negative outcomes in the communities or groups where they take place, including decreased well-being, job satisfaction, stress, and turnover [35, 37, 50]. In addition, these outcomes tend to disproportionately affect members of underrepresented groups [4, 13, 62].

ICSE-SEIS'22, May 21–29, 2022, Pittsburgh, PA, USA

ACM ISBN 978-1-4503-9227-3/22/05.
https://doi.org/10.1145/3510458.3513019

In software engineering, the problem of interpersonal conflicts is also well recognized. For example, in software development, some communities and maintainers have a reputation for being toxic [20, 55, 58]. Although relatively milder, impolite language is seen as a barrier to newcomers [48, 59]. There are repeated anecdotes of sexist behavior, harassment, or contributors concealing their identity to avoid abuse [22, 40, 56, 60, 61]. More broadly, evidence is also starting to emerge about anger [21], negative emotions [19], impoliteness [43, 46], pushback [18], or directly toxicity in issue discussions [2, 12, 38, 50], code reviews [11], and Gitter developer communication [53]. The programming-related Q&A platform Stack Overflow is also notorious for being 'toxic' [9].

However, despite comparable agreement about the importance of the problem, there is relatively less progress in software engineering compared to other domains in terms of automatic detection for prevention or mitigation [30, 32]. Several factors contribute to this lag, including inherent difficulty of the problem, but also domain specificity of some toxic interactions and scarcity of labeled data.

Prior research on automatic detection of toxicity and related constructs in software engineering has room for improvement. In particular, we note that approaches published previously in the software engineering literature have generally all been based on textual analytics [10, 50]. For example, Raman et al. [50] experimented with different sets of features, all text-based, to train a classifier to detect open-source software (OSS) toxic issue discussions, which is defined as "rude, disrespectful, or unreasonable comment[s] that [are] likely to make someone leave a discussion" – a definition of toxicity used also in other public discussion forums such as Wikipedia or the New York Times, originating from Google's project Jigsaw [63]. However, follow-up work by Sarker et al. [53] showed that Raman et al. 's approach has limited generalizability.

Meanwhile, researchers have long been arguing that meta-information can be very useful to refine inconclusive classification [54]. For example, people with a history of hate speech are more likely to engage in such behavior again than people without any history [14]. In software engineering, Egelman et al. [18] showed that using only meta-information can detect pushback, defined as "the perception of unnecessary interpersonal conflict in code review while a reviewer is blocking a change request."

Notably, the two concepts – 'toxicity' as operationalized by Raman et al. [50] and 'pushback' as operationalized by Egelman et al. [18] – are similar, but distinct. For instance, while some types of

Egelman et al.'s pushback could be considered toxic (*e.g.*, personal attacks), others would not (*e.g.*, persistent nitpicking). Moreover, the types of software discussions analyzed and the study settings in the two studies are arguably very different — Egelman et al.'s classifier was applied only on code reviews internally at Google and Raman et al.'s classifier was applied only on public GitHub issues (not code reviews). Despite these difference, it seems possible that these two approaches could inform one another as a way to improve detection of interpersonal conflict.

In this paper, we contribute: (1) a comparison of how toxicity and pushback manifest in open source and in a company, and (2) a systematic evaluation of our ability to predict toxicity and pushback in different settings and using different approaches. To this end, we use existing and new labeled datasets that capture both concepts in open-source and corporate code reviews. We use 10-fold cross-validation to evaluate and compare the two previous classifiers and also develop a new combined classifier using features from both. Our results provide insights on how these classifiers work in different contexts. The comparisons and discussion also shed light on the relationship between the two concepts, toxicity and pushback, and the two settings, open source and corporate.

By improving the accuracy of automated approaches to detect toxicity, pushback, and possibly other forms of interpersonal conflict in software discussions, this research paves the way for designing tools to prevent, mitigate, and further study these phenomena, including designing interventions to offer just-in-time guidance to developers in such situations. A detector can also be a powerful tool for researchers studying the effectiveness of tool design and other interventions. More generally, this research offers an opportunity to apply a technique to both open and closed source software, possibly benefiting from synergies, a rarity in software engineering research, in our experience.

2 RELATED WORK

This paper builds directly on two recent approaches to detecting interpersonal conflict in software engineering artifacts, by Egelman et al. [18] and Raman et al. [50]. In Egelman et al.'s study at Google, the authors conducted interviews to develop the concept of *pushback* and designed logs-based metrics to detect pushback in code reviews. These metrics were rounds of a review, active reviewing time, and active shepherding time. Their logistic regression model obtained high recall (93%–100%) and low precision (6%–11%).

The other approach that this paper builds directly on is that of Raman et al. [50]. The authors manually annotated toxic issue threads from projects on the GitHub platform, and experimented with outputs from different sets of generic text-based classifiers to train a new classifier to detect toxic issue discussions specific for open source. They reported the highest 10-fold cross-validation accuracy when combining Stanford's Politeness Detector [15] with Google's Perspective API.[1] The present paper expands on Raman et al. 's text-based features, compares them with Egelman et al.'s classifier [18], and experiments with combining the two classifiers.

In addition to the pretrained general-purpose linguistic tools used by Raman et al., we also explore other linguistic techniques to detect interpersonal conflict. Vocabulary-based approaches have been used for text classification. Open-vocabulary analysis extracts features from the text being analyzed using statistical methods [45]. For example, Sood et al. [57] showed that an SVM classifier using binary presence and frequency of n-grams as features can be used to predict personal insults on social news sites. Monroe et al. [39] showed that the log odds-ratio of an n-gram (the frequency of being in one group of text divided by 1 minus the frequency) in two different groups can be used to identify n-grams that are over-represented in one group relative to the other. We build on Monroe et al.'s work in Section 5 by attempting to find out if there is a set of vocabulary that can distinguish between the positive labels (toxic or pushback) and the negative labels (non-toxic or non-pushback).

Closed-vocabulary analysis relies on predefined lists of words as features. Building on the classic linguistic theory of politeness by Brown and Levinson [6], Danescu-Niculescu-Mizil et al. [15] developed a computational parser for politeness strategies. Politeness theory divides politeness strategies into *positive politeness* and *negative politeness*. Positive politeness strategies encourage social connection and rapport, such as gratitude, optimistic sentiment, solidarity, etc. Negative politeness strategies try to minimize the imposition on the hearer, for example, by being indirect or apologizing for the imposition [6, 33, 34]. On the other hand, impolite behaviors can be direct questions (*e.g.*, "why?") or sentences that start with second-person pronouns, which may sound forceful. Prior studies showed that the politeness strategy parser [8] is able to predict if a conversation may turn awry [64] and can generalize well to various contexts. We build on this work by using politeness strategy features in our classifiers.

Finally, in the software engineering community, sentiment analysis [44] is a popular technique for analyzing issue discussions [21], pull request comments [25], and forum discussions [7]. Prior work has shown that sentiment analysis classifiers need to be trained using software engineering data because many traditionally negative phrases may have neutral sentiment in the context of software engineering [29], for example, "execute" (for a survey see Zhang et al. [65]). Popular software engineering sentiment analysis tools include Senti4SD [7] and SentiCR [1]. Senti4SD, developed by Calefato et al. [7], is trained on 4,000 posts extracted from Stack Overflow. This dataset is part of the Collab Emotion Mining Toolkit [41]. SentiCR [1] is trained on 1,600 manually labeled code review comments. In our study, we build on this work by using sentiment analysis developed for code reviews as a feature in our classifiers.

3 RESEARCH QUESTIONS

Our overarching goal is to bridge the gap between the existing literature on toxicity [50] and pushback [18] in software development. Besides the two concepts themselves, there are three fundamental differences between the prior work studies in this area, which we systematically explore in this paper: (1) the context (open- *vs.* closed-source), (2) the type of discussion (issues *vs.* code review), and (3) the approach to classify (text-based *vs.* logs-based). Overall, we answer the following research questions and sub-questions:

First, we explore how well the two classifiers generalize beyond the respective settings in which they have been developed, while maintaining their specific target concepts (toxicity and pushback) and fundamental approaches to classification (text- and logs-based):

[1] https://perspectiveapi.com/

RQ$_1$. *How well do existing classifiers generalize across context and type of discussion?*

To answer this question, for each classifier we explore one additional setting beyond the one in which they have been developed. For the toxicity classifier [50], we experiment with open source code reviews in addition to the original issue discussions. Similarly, for the pushback classifier [18], we experiment with comments on open-source pull requests, the approximate equivalent of the original Google code reviews:

RQ$_{1.1}$. *How well does a text-based toxicity classifier designed for open-source issues perform when classifying toxicity in open-source pull requests?*

RQ$_{1.2}$. *How well does a logs-based pushback classifier designed for corporate code reviews perform when detecting pushback in open-source code reviews?*

Second, given the theoretical overlap between the concepts of pushback and toxicity, we explore how well the two fundamental approaches to classification, text-based (toxicity) and logs-based (pushback) generalize to detecting the other concept if appropriately trained on relevant data for that other concept:

RQ$_2$. *How well do existing classifiers generalize for both toxicity and pushback?*

RQ$_{2.1}$. *How well does a text-based (toxicity) classifier perform when classifying pushback, in both open and closed-source code reviews?*

RQ$_{2.2}$. *How well does a logs-based (pushback) classifier perform when classifying toxicity in open-source code reviews and issue discussions?*

Finally, we explore to what extent using design insights from one classification approach can be used to improve on the other:

RQ$_3$. *To what degree can combining existing approaches improve detection of toxicity and pushback?*

RQ$_{3.1}$. *How well can a combined text- and logs-based classifier classify toxicity?*

RQ$_{3.2}$. *How well can a combined text- and logs-based classifier classify pushback?*

For completeness, in addition to answering these questions, we also replicate the original experiments on toxicity in open source issues [50] and pushback in Google code reviews [18].

4 DATASETS

To answer our research questions, we used a mix of existing (whenever possible) and new datasets on toxicity and pushback. First, we used the two existing data sets from prior work on issue toxicity in open source [50] and code review pushback at Google [18]. Additionally, we created two new datasets on code review toxicity in open source and code review pushback in open source. Table 1 displays each of these four datasets as a row, labeled D1-D4, summarizes how each of our research questions and the prior work relates to each data set, and describes the size of the datasets.

4.1 Design Decisions and Tradeoffs

Before describing each dataset in detail, we note several important high-level design decisions, assumptions, and tradeoffs we had to make when creating the two new datasets, and in order to meaningfully compare results across all four datasets.

Unit of labeling. In the original toxic issue comments dataset by Raman et al. [50], ground truth labels are available for individual comments and the issue thread-level toxicity labels are an aggregation of comment-level labels, *i.e.*, if there is at least one comment labeled as toxic, the entire discussion is labeled as toxic. In contrast, the pushback code review dataset by Egelman et al. [18] contains only thread-level labels. Since we are reusing these datasets without relabeling, we maintain the same unit of labeling also in the two newly created datasets of the same concept.

Unit of classification. Our experiments focus on classifying toxic or pushback entities at the thread level, because the logs-based metrics, such as the rounds of review, used by Egelman et al. are not applicable for individual comments. However, because the text-based classifier works at the comment level, for pushback datasets where we only have thread-level labels, we had to assign each comment the same label as the thread-level label. We will discuss the limitation when we present the results.

The notion of code review. Our two new code review toxicity and pushback datasets are extracted from open-source projects on the GitHub platform whereas Egelman et al. 's dataset [18] was extracted from internal Google code reviews. In addition to the differences between the corporate and open-source contexts in terms of culture, process, and their observed consequences, the mechanics of code reviewing also differ. Google uses a proprietary dedicated code review management system [52] where all review comments are associated with specific code changes. On GitHub, projects typically manage code reviews as part of pull request threads. However, even though canonically code review comments on GitHub are expected to be attached to specific lines of code and can therefore be distinguished from more general discussion comments part of the same pull request thread, practices vary widely across projects [23]. For reasons of uniformity across projects when sampling candidates for manual labeling, and since we expect that indicators of pushback may occur across pull requests as a whole, not just review comments attached to specific changed lines, we consider the conceptual equivalent of a Google code review to be an entire GitHub pull request thread, including all its general and line-specific comments, *i.e.*, an "open-source code review thread" hereafter.

Representativeness. When sampling toxicity and pushback pull request candidates for manual labeling, we use several heuristics to narrow down the search space (details below) instead of random sampling. While this compromises the statistical representativeness of our datasets, it is necessary to do this since the two phenomena we study are relatively rare; random sampling is unlikely to discover many, if any, instances of these phenomena. We note that this is not only a limitation of the two prior work studies we build on, but also of all similar work on hate speech detection *etc.* [49]. Alternative approaches to building labeled datasets for hate speech detection are, as of 2021, still actively being researched [49].

Table 1: The relationship between our four datasets and their corresponding RQs.

	Classifiers			Number of Data Points
	Text-based	Logs-based	Combined	
D1 Toxic Open-Source Issue Comments	Raman et al. [50]	RQ2.2	RQ3.1	80 toxic, 160 non-toxic
D2 Toxic Open-Source Code Review Comments	RQ1.1	RQ2.2	RQ3.1	102 toxic, 204 non-toxic
D3 Pushback in Corporate Code Review	RQ2.1	Egelman et al. [18]	RQ3.2	493 pushback, 809 non-pushback
D4 Pushback in Open-Source Code Review	RQ2.1	RQ1.2	RQ3.2	201 pushback, 323 non-pushback

Open source vs corporate metrics. While we try to replicate Egelman et al. 's pushback detection method, some measures are unfortunately not observable on GitHub. For example, we cannot replicate "shepherding time," which in Egelman et al. 's study is the total amount of time an author spent actively viewing, responding to reviewer comments, or working on the selected code change, including looking up APIs or documentation. The public GitHub trace data about pull request threads captures only wall clock times, which is an overapproximation of the active shepherding time. We are particularly interested in evaluating how well such approximation metrics, that are less precise but more widely available outside of a corporate setting, can capture the same phenomena.

4.2 Toxic OSS Issues (D1; pre-existing)

This dataset, originally created by Raman et al. [50], consists of 80 GitHub issue discussions labeled as toxic by the authors. Starting from the GHTorrent database [24], Raman et al. [50] identified potentially toxic issue comments using the keyword "attitude" (the authors of the toxic comments are often criticized in the same thread by others, typically the project maintainers, about their attitude), and from issue threads "locked as too heated"—one of the mitigation strategies afforded by the GitHub platform. Raman et al. then manually reviewed a sample of candidate issue threads from this initial list and assigned ground truth toxicity labels.

We decided to replace the control group in Raman et al.'s dataset [50] because we noticed that those non-toxic comments' total number of characters is significantly shorter than for the toxic comments. Since a priori we have no reason to expect that toxic issues are generally longer than non-toxic issues, and we want to capture other aspects of toxic comments, we compiled a new set of non-toxic issues. Inspired by Egelman et al. [18], we constructed stratified samples by propensity score matching on the length of all comments within an issue thread (which is not used in any of our prediction models), after excluding code segments and comments from obvious bots [46], *e.g.*, a continuous integration tool. Our new set of non-toxic issues contains two non-toxic issues for every toxic issue.

4.3 Toxic OSS Code Review (D2; novel)

We compiled a dataset of 102 toxic open-source code review threads (*i.e.*, pull request threads with all their associated comments) and a separate corresponding control group of non-toxic open-source code review threads, using a similar approach to the one originally taken by Raman et al. [50] for issues. Specifically, we use three heuristics to narrow down the search space for candidates in the GHTorrent [24] database, followed by manual review and labeling. Egelman et al. [18] showed in their study of pushback that inter-rater agreement is very high when using multiple annotators, implying that a single annotator is sufficient. One author of the paper carried out the labeling independently, assigning "toxic" and "non-toxic" labels to the threads as a whole if at least one of the comments was considered to be toxic; when in doubt, we discussed the respective examples as a group and assigned labels collectively.

The three heuristics were:

- Locked as "too heated"—this built-in GitHub mitigation mechanism is available for both issues and pull requests; or
- Containing the keyword "attitude"; or
- Containing "code of conduct", a novel addition relative to Raman et al. 's heuristics [50]. We anecdotally observed that a project's code of conduct, when present, is invoked by maintainers when responding to a toxic comment.

Then, as in dataset D1, we performed propensity score matching on the total length of comments to assemble a control group containing two non-toxic open source code reviews for every toxic one.

4.4 Pushback in Corporate Code Review (D3; pre-existing)

We used the collection of code reviews gathered by Egelman et al. [18] from Google's internal corporate repository. The authors collected these using two methods:

First, Egelman et al. [18] pulled a stratified random sample of code reviews, then surveyed authors, reviewers, and other engineers about whether they perceived each code review as having elements of "pushback." The authors then labeled a code review as containing pushback if at least one respondent noted that the review contained at least one element of pushback. Code reviews are labeled as not containing pushback if (a) at least one person responded to a survey about it, and (b) all survey responses about that code review indicated that no elements of pushback were present.

Second, those same respondents could report a code review that they thought contained pushback. They labeled these reported code reviews as "containing pushback", except that we discarded those that participants indicated were problematic only because of excessive review delays, which are not part of Egelman et al. ' definition of pushback [18].

4.5 Pushback in OSS Code Review (D4; novel)

To construct an open source counterpart to the corporate code review pushback dataset, we replicated the survey instrument used by Egelman et al. [18], with only surface-level modifications to adapt to pull requests and their specifics on the GitHub platform instead of Google-specific terminology.

We then compiled a sample of GitHub code reviews that each:

- had at least 10 comments, to ensure that at least some amount of interpersonal interaction was present, and
- had no more than 50 comments, to limit the reading effort expected from survey respondents.

Additionally, to ensure some diversity in code review outcomes, half of the sampled code reviews were merged pull requests and half were closed without being merged. We emailed survey invitations to the authors and reviewers who display their emails publicly.

As with Egelman et al. 's survey [18], we also asked participants to report other code reviews that they thought contained pushback; 63 were reported this way. The reasons that these code reviews were reported as pushback are shown in Figure 7 in Appendix. We then labeled these discussions using the survey data in the same way as in Dataset D3. As a result, this dataset contains only conversation-level labels.

Since one can maximize the recall of a classifier by predicting all data points as positive, the minimum precision score is the percentage of positive data points. Therefore, to make D3 and D4 more fairly comparable, we downsampled D4's negative data points to match the positive-negative ration in D3. In the end, D4 contains 201 pushback threads and 323 non-pushback threads.

Table 2: N-grams that are over-represented in either class in *D2 Toxic OSS Code Review Comments*. N-grams with second-person pronouns are in bold. N-grams with software engineering terms are underlined.

	unigram	z-score	bigram	z-score	ngram	z-score
Toxic	**you**	12.172	it is	5.555	**if you want**	3.397
	people	7.292	**you want**	4.81	it is not	2.712
	even	7.097	that is	4.272	**do you think**	2.576
	do	6.71	going to	4.256	**you need to**	2.397
	what	6.644	**you are**	4.187		
	is	6.373	trying to	4.053		
	want	6.078	**if you**	3.682		
	your	5.796	to do	3.668		
	because	5.657	do not	3.556		
	why	5.547	**you think**	3.539		
Non-toxic	tests	-4.773	**could you**	-2.815		
	unit	-4.858	the pull	-2.889		
	vs	-4.982	as the	-3.137		
	file	-5.15	and the	-3.143		
	files	-5.165	of files	-3.296		
	for	-5.574	we can	-3.48		
	test	-5.76	pull request	-3.668		
	from	-5.872	code to	-3.856		
	at	-6.732	to the	-4.031		
	line	-6.782	instead of	-5.004	the pull request	-2.276

5 EXPLORATORY ANALYSIS

As a first step, before applying machine learning, we explored how well a more basic word-frequency approach could distinguish discussions with one label compared to the other (*e.g.*, toxic *vs.* non-toxic) in each of the four datasets. To this end, we used an open-vocabulary analysis [39] to automatically identify words and phrases that are used distinctly more often in one label than the other, and then manually reviewed these looking for themes. This analysis serves two purposes. First, it helps to triangulate that the manually assigned labels are meaningful, if "obvious" differences between the two classes are detectable using this independent approach. Second, it informs the design of more sophisticated automated classification, by identifying promising features.

Concretely, for the automated part we used log odds-ratio with Dirichlet prior [39] to identify n-grams that are significantly over-represented in positive labels, *i.e.*, those labeled as toxic or pushback, compared to the negative labels, *i.e.*, those labeled as non-toxic or non-pushback. Since our data sets do not have an equal volume of text, we measured frequency using the log of an n-gram's odds-ratio. Because some n-grams may appear only in one label and not the other, we added a smoothing Dirichlet to the vocabulary. We pre-processed the text by removing URLs and numbers. We did not remove stopwords before performing the analysis because removing them can interrupt sentences and potentially eliminate some meaningful n-grams. We then ranked n-grams by z-scores and kept those with absolute z-scores above 2.326, which corresponds to the statistical significance cutoff of $p < 0.01$. Finally, we kept the 10 unigrams, bigrams, and n-grams with the highest positive z-scores (from positive labels) and 10 with the lowest negative z-scores (from negative labels).

We then manually examined the usage of these n-grams in our data sets by sampling comments containing them. We looked for patterns in these comments that could help us distinguish toxic or pushback comments from non-toxic or non-pushback ones, respectively. That is, we applied this process to all four of our datasets

To illustrate the results of this exploratory analysis, consider the results for dataset D2 Toxic Open-Source Code Review Comments in Table 2. In the table, empty cells indicate that no more n-grams were above or below the z-score cutoff. Due to space constraints, the tables (Tables 3, 4, and 5) for the remaining three data sets are shown in Appendix. Below, we describe several patterns that we observed from this analysis.

Second Person Pronouns. One clear pattern we can observe from the word frequencies is the use of the second person pronoun "you" in toxic text, including phrases like "you are", "if you want". "You" is the unigram with the highest z-score in both D1 (Table 3) and D2. In Table 2, n-grams with second-person pronouns are in bold.

To investigate further, from D1 and D2 we randomly sample 10 toxic comments and 10 non-toxic comments that include "you." Some of these comments involve direct attacks on the second person recipient, such as "[y]ou don't care to be a part of the project," "[y]ou are expected to comply," "[y]ou decided to insult [...]." This echoes what Danescu-Niculescu-Mizil et al. [15] found: the use of second-person pronouns at the beginning of a sentence is more likely to be impolite. The same pattern is observed In D3 (Table 4).

In non-toxic comments in D2, the only n-gram that contains "you" is "could you", which is a negative politeness strategy that tries to minimize the imposition on the hearer by being indirect. The counterfactual form "could" is more polite than the future-oriented variant "can" [15]. This is also true in D3, where we again see some hedge words and other politeness strategies, such as "could you", "should be", and "seems" among non-pushback code reviews.

Gratitude. Gratitude is another common theme in non-pushback text, both in open and closed source code reviews (D3 and D4 (Table 5)). These n-grams included "thanks" and "thanks for" that appear among non-pushback code review comments.

Technical Discussion. In D1 and D2, we see many software engineering-related n-grams, *e.g.*, "tests" and "the pull", among non-toxic comments but almost none among toxic comments. In D3 and D4, we likewise see more technical terms among non-pushback comments. In Table 2, n-grams with software engineering terms are underlined.

Code of Conduct. We occasionally see "code of" and "the code of" appear in the top-10 lists. Typically, these two n-grams appear when referring to "the code of conduct", often as a reminder that someone violated the code of conduct. For example, one contributor wants the maintainer "to enforce the code of conduct [...]." Interestingly, we observe this pattern in D4 (pushback in open source code reviews), which was sampled without using this as a search term.

No Pattern and Overfitting. Finally, among all four datasets, we see some n-grams with no discernible rationale for why they might be indicators or contraindicators of toxicity or pushback. For instance, in Table 2, the bigrams consisting of only stop words, *e.g.*, "as the", "and the", and "to the", appear to just be noise, rather than true indicators of non-toxic open-source code review. As an example of overfitting, the top unigram in D3 ("<tech1>") indicates a widely-used, Google-specific piece of technology.

Overall, this exploration confirms that discussions in the positive labels, tend to shift focus away from the technical aspects themselves and onto interpersonal issues. The ground truth labels on all four datasets appear meaningful, since there are noticeable differences in the relative frequency of words and phrases between discussions with presence and absence of toxicity and pushback. Moreover, the analysis implies that there is substantial overlap between the two concepts of pushback and toxicity, suggesting that incorporating text-based features into classifiers for both concepts is worthwhile. However, the absence of a clear pattern for many n-grams suggests that a purely frequency-based approach would be insufficiently discriminatory for an accurate classifier. In what follows, we introduce more sophisticated classification approaches.

6 METHODS FOR CLASSIFICATION

6.1 Building classifiers for toxic comments and pushback in code reviews

Text-based features. In this paper, we reuse and improve the classifier developed by Raman et al. [50], which takes outputs from several text-based pretrained classifiers as features. We first preprocessed the text by removing URLs, quotes, numbers, *etc.* Then we feed the text into the following three pre-trained NLP classifiers, and use the outputs as features.

Following Raman et al. [50], we collect (1) the toxicity score and identity attack score from the Perspective API ([0, 1] range, with 1 being the most toxic/aggressive) and (2) count the occurrences of different politeness strategies using the politeness parser [8, 15] (normalized to [0, 1]). In addition, we used (3) a sentiment analysis tool developed for software engineering code review comments, SentiCR [1], with reportedly better performance on GitHub data than other sentiment analysis tools [65]. The output from SentiCR is either positive sentiment (1) or negative sentiment (-1).

Logs-based features. Because we are interested in answering whether the pushback classifier by Egelman et al. [18], which uses logs-based features, can be applied to open-source code review comments (RQ1), we calculated logs-based metrics for D2 and D4, the two novel datasets. Egelman et al.'s work on code review in the company used rounds of review, active reviewing time, and active shepherding time to build a classifier for pushback. They defined:

- *Rounds of review* as the number of batches of contiguously authored comments, as it "captures the extent to which there was back-and-forth between the author and reviewer."
- *Active reviewing time* is "the time invested by the reviewer in providing feedback," which includes actively viewing, commenting, or working on code review.
- *Active shepherding time* is the time "the author spent actively viewing, responding to reviewer comments, or working on the selected CR, between requesting the code review and merging the change into the code base."

The above "active" times may include time outside of code review, *e.g.*, editing files, but does not account for in-person conversations.

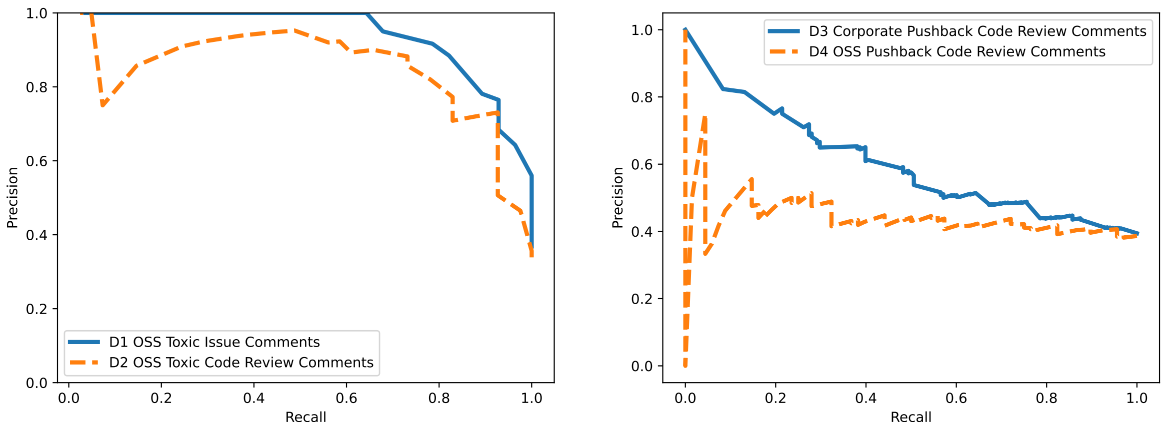
As discussed in Section 4, for GitHub data we could not exactly replicate all three logs-based metrics used by Google, because of differences between Google's code review tool and the GitHub pull request workflow. Therefore, by necessity we operationalized these metrics for open-source code review comments (D2 and D4) differently:

- We approximated *Rounds of review* as the number of comments on a pull request, since GitHub code review comments are not always grouped into batches the way Google's are.
- We approximated *Active shepherding time* as the time difference between the initial PR post and the last comment. Note that the difference between our shepherding time and the one by the company is that the company uses the actual amount of time an author spent on a code change, whereas ours is the wall-clock time of the entire review process, which may result in longer shepherding time overall.
- We did not attempt to approximate *Active reviewing time*, because we could not distinguish how much of the time between the submission of code and the last comment was taken by reviewers or by the author.

Training. We trained a random forest [5] classifier for each classification task because of its accuracy and robustness against overfitting [28, 31].

Following Raman et al. [50], we performed 10-fold nested cross validation to find the best model and reduce bias from random data splits. We first randomly split our labeled data into a training set (67%) and a test set (33%). We used stratified sampling to preserve the ratio between labels and ensure that each set contains both positive and negative labels.

We then fit and cross validate a random forest model using the training set for 10 trials. In each trial, the training set is further split



(a) Text-based classifier P-R curves on D1 and D2. **(b) Logs-based classifier P-R curves on D3 and D4.**

Figure 1: Text-based classifier P-R curves

into 10 folds randomly. Each fold is used once as a cross validation set, while the remaining 9 folds are used for training. The random forest model learns the best combination of hyperparameters, such as `n_estimators` and `max_depth`, optimizing for F1 score, the harmonic mean of precision and recall.

After each trial, we tested the random forest model with the combination of hyperparameters that produced the highest F1 score during training (67% of the entire labeled dataset) on the held-out test data (33% of the entire labeled dataset).

For the text-based classifier, the classification is performed at comment level. Then we aggregate the classifications to form thread-based labels. For pushback datasets (D3 and D4) where we only have thread-level labels, we assign all comments the same label as the thread-level label. For the logs-based classifier, the classification is performed at thread-level.

6.2 Classifier Performance Analysis

To evaluate the performance of our classifiers, we computed and compared the Areas Under the Precision-Recall (P-R) Curves, *i.e.*, the P-R AUC scores. Precision tells us how many comments labeled by our classifier as toxic/pushback are in fact toxic/pushback, and recall tells how many toxic/pushback comments in our test dataset are classified as toxic/pushback. P-R curves explore the classic precision/recall tradeoff in applications where the data is imbalanced [16], as is ours — toxicity and pushback are both relatively rare. P-R curves are also commonly used to evaluate classifiers when researchers care more about positive (toxic or pushback) than negative labels. This is also the case in our work — for downstream prevention, mitigation, and future research on toxicity and pushback, we believe that it is more important to identify true instances of toxicity and pushback than it is to identify that some comment or conversation is not toxic or pushback. A P-R AUC score summarizes the performance of a classifier into one value and can be interpreted as the average of precision scores calculated for different recall thresholds, with higher values (closer to 1) being preferable.

To compute the P-R curves, we uniformly vary the classifier's probability threshold for predicting the positive class, which corresponds to exploring the precision-recall tradeoff. To compare classifiers, we performed pairwise t-tests on their P-R AUC scores computed after the 10 cross-validation trials. At each trial, we applied the random forest classifier with the best hyperparameter combination on the held-out test data and computed an AUC score. As a result, from our 10-fold nested cross validation training process, we obtained 10 AUC scores (one per trial). For each t-test, we also report Cliff's delta measure of effect size.

In addition, we estimate the importance of each feature [31] in our random forest classifiers during the training phase, using a standard approach based on the mean overall improvement in a tree's impurity. The impurity, in classification tasks, is measured by the Gini index, interpreted as the probability of an item being incorrectly classified if it was randomly labeled according to the distribution of a specific feature [28].[2]

7 RESULTS

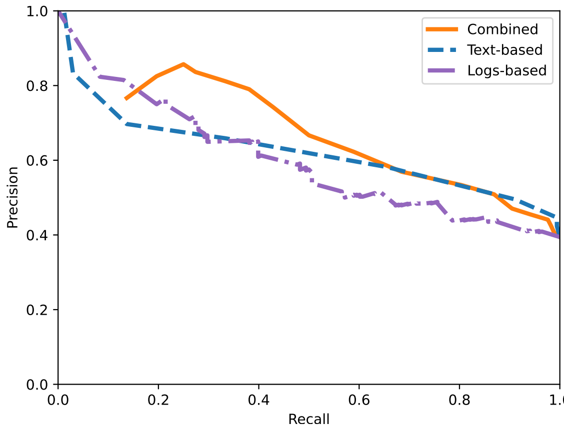
RQ 1: ***How well do existing classifiers generalize across context and type of discussion?***

To answer this question, we plot the P-R curves by the classifiers using the same features on different datasets and compare the average AUC scores. Figure 1 shows one of the curves from the 10 trials.

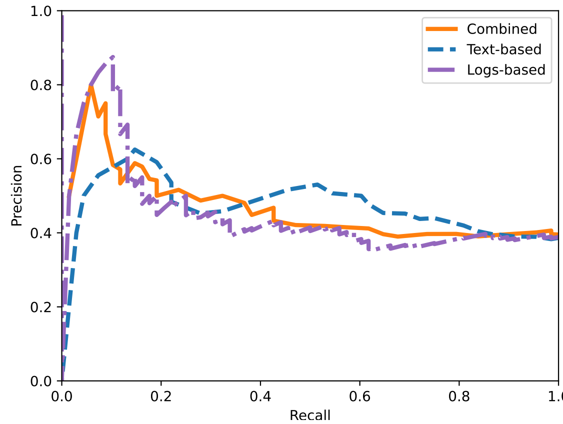
We start by comparing the P-R AUC scores for the text-based toxicity classifier on D2 (open-source code reviews) relative to the benchmark D1 (open-source toxic issues), answering $\mathbf{RQ}_{1.1}$. The P-R curves are shown in Figure 1a. We find that at the thread level, the text-based classifier performs better on D1 than on D2 (t = 5.640, p-value = 0.0001; Cliff's δ = 1 / large effect; the AUCs are 0.907 and 0.844 respectively).

We manually checked some randomly sampled toxic comments from D1 and D2 that our text-based classifier failed to identify. We found that some of them are responding to toxic behavior. For example, phrases like "you spent a long time insulting people" are

[2] Our code is available at https://doi.org/10.5281/zenodo.6051070



(a) P-R curves of all three classifiers on D3



(b) P-R curves of all three classifiers on D4

Figure 2: P-R curves on pushback classification

responses to someone else's insult and are clearly a signal of the presence of toxicity. Some other ones contain *covert toxicity*, such as sarcasm, entitlement, or the use of "?!" or emojis. Covert toxicity is difficult for language models to detect in general [36]. These comments also have a low predicted toxicity score by our classifier; some even use the word "please" as in "Please consider that this thread [...] is so problematic. [...] get this PR closed ASAP."

The impurity-based feature importance analysis (Figure 4a in Appendix) provides some explanations on what features are important in both datasets. The x-axis is the importance score of the features. The sum of importance scores of all features is 1. The two most important features during the training phase are from the Perspective API. They are followed by three politeness features: second person pronouns, the presence of negative words, (*e.g.*, "begging for complete code review" and "many bugs documented and unresolved"), and the use of first person pronouns. The use of second person pronouns echoes our findings of the word frequency analysis, where we see the use of "you" overrepresented in toxic text.

Reflecting on differences between the issue conversations and code review conversations that could cause the performance degradation when detecting toxicity in the latter case, we speculate two reasons based on exploring the two labeled datasets. One is that many code-specific comments are much shorter than discussion comments, yielding less linguistic information. The other possibility is that the code review conversations in our dataset more often include code chunks and removing inline codes may reduce information for the text-based classifier.

Next we compare the P-R AUC scores for the logs-based classifier on D3 (pushback in corporate code review) and D4 (pushback in open-source code review), answering **RQ**$_{1.2}$. The P-R curves are shown in Figure 1b. Our results show that the logs-based classifier has a lower performance when transferred to the open-source context, despite being retrained (t = 40.008, p-value $< 2.2e-16$; Cliff's $\delta = 1$; the average AUC scores are 0.693 and 0.445 for D3 and D4).

We speculate that there are two main reasons for the lower performance. First, limited by the information publicly available on GitHub, we could only compute measures for two of the three logs-based features used originally inside Google. Therefore, we have less information. Indeed, in D3, reviewing time, the feature missing in D4, ranks as the most important (Figure 5 in Appendix). Second, our measure of shepherding time computed for open-source code reviews is only an approximation, using wall-clock time rather than the amount of time spent actively working on code in review. Therefore, the logs-based features we computed for open-source data are not as accurate as those on corporate data.

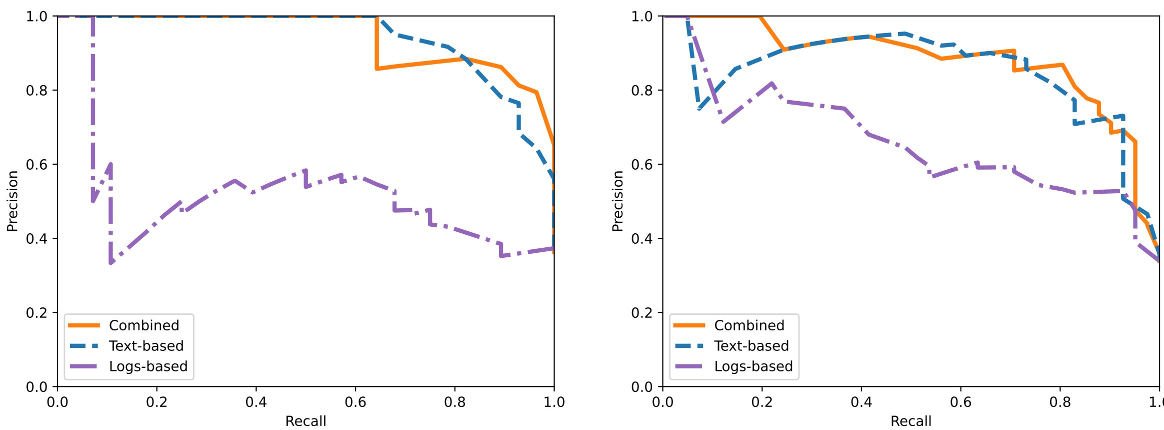
> **Summary**: *Both the text-based classifier and the logs-based classifier have performance degradation when generalizing to other contexts.*

RQ$_2$: ***How well do existing classifiers generalize for both toxicity and pushback?***

To answer this research question, we compare the performance of the classification approach originally designed for one construct (toxicity or pushback) to the classification approach originally designed for the other construct.

We start by evaluating the performance of the text-based classifier on datasets D3 and D4, compared to the performance of the logs-based classifier as a benchmark, answering **RQ**$_{2.1}$. Figure 2 shows one of the P-R curves from the 10 trails.

On *D3 Pushback in Corporate Code Review*, the text-based classifier outperforms the logs-based classifier on average (t = 9.766, p-value = $1.304e-08$; Cliff's $\delta = 1$ / large effect; the mean AUC scores are 0.757 and 0.693 for the text-based and logs-based classifiers respectively); note, this logs-based classifier is the one using all three measures of pushback, available inside the company Google. This suggests that pushback as a construct shares many linguistic similarities with toxicity. In addition, the better performance of



(a) P-R curves of all three classifiers on D1

(b) P-R curves of all three classifiers on D2

Figure 3: P-R curves on toxicity classification

the text-based classifier suggests that, in a corporate setting, interpersonal conflicts can be more subtle than delay of reviews or excessive comments.

For *D4 Pushback in Open-Source Code Review*, comparing the P-R AUC scores shows that, unlike previously on D3, the text-based classifier and the logs-based have a similar performance (t = 0.246, p-value = 0.810; the average P-R AUC scores are 0.447 for the text-based classifier and 0.445 for the logs-based one); note, the logs-based classifier in this case contains only the measures available publicly on the GitHub platform.

One possible explanation is that in contrast to the previous corporate dataset, there are relatively fewer examples of open-source pushback code reviews in our sample that could be traced back to reasons with linguistic markers. Therefore, there is less for the text-based classifier to discern. To test this hypothesis, we compiled a subset of D4, D4-1, containing as positive examples all the reported pushback open-source code review threads that are linked to linguistic markers (Figure 7 in Appendix), such as "harsh comments" (39 out of 63 self-reported pull requests), and as negative examples the remaining self-reported pushback open-source code review threads with only likely non-linguistic markers, such as "excessive review delays." Comparing the performance of the logs-based and text-based classifiers on D4-1 does not support our hypothesis: the text-based classifier underperforms the logs-based one ($t = -5.072$, p-value = 0.0002; Cliff's $\delta = -0.86$ / large effect; the average AUCs are 0.566 and 0.679 respectively). The reason could be that pushback threads labeled with linguistic-related reasons are often labeled with non-linguistic ones too, *e.g.*, "requesting a change without justification."

Another possible explanation is that since pushback classification is done at the thread level, within a thread the actual comments indicative of pushback are too rare for the whole text of the threads to be significantly different on average between the pushback and non-pushback classes. To test this, we created dataset D4-2, in which we assigned pushback labels at the comment level. Specifically, we used the responses to our survey asking participants to copy-paste the text fragments indicating pushback, in addition to offering pushback pull requests as a whole, to identify which comments in the thread contained those exact fragments. We then labeled those comments as pushback and all other comments in the same threads as non-pushback. Then we performed the classification and thread-level aggregation as usual. Comparing the performance of thread-level text- vs logs-based classification on D4-2, we observe that the text-based classifier now outperforms the logs-based one (t = 2.591, p-value = 0.026; Cliff's δ = 0.54 / large effect; the average AUCs are 0.534 and 0.471 respectively), supporting our hypothesis.

The feature importance analysis (Figure 4b in Appendix) for the text-based classifier on both pushback datasets D3 and D4 present some insights into what linguistic features are associated with pushback comments. On both datasets, the toxicity score and identity attack score from the Perspective API have the highest importance. They are followed by several politeness strategies. The third most important feature in D3 is the presence of positive lexicons whereas in D3 is the number of hedge words, such as "likely", "maybe", "seems". Having second person pronouns is also an important feature to classifying D3 Pushback in Corporate Code Review but less so to D4 Pushback in Open-Source Code Review.

Summary: *When detecting pushback, the text-based classifier performs better than the logs-based classifier for corporate code review comments, but they have similar performance for open-source code review comments.*

Next we compare the performance of the logs-based classifier against the performance of the text-based classifier on detecting toxicity, answering $\mathbf{RQ}_{2.2}$. We plotted the P-R curves for the text-based and the logs-based classifiers on D1 and D2, shown in Figure 3. We find that the text-based classifier performs better than the logs-based one on both D1 (t = 45.515, p-value $< 2.2e-16$; Cliff's $\delta = 1$; P-R AUC scores are 0.907 and 0.516 respectively) and D2 (t = 13.591, p-value $= 2.22e-10$; Cliff's $\delta = 1$; P-R AUC scores are 0.844

and 0.665, respectively). The good performance of the text-based classifier implies that toxicity is more of a linguistic phenomenon. Meta-data, such as the logs-based features we computed, could not capture enough information to distinguish toxic language.

Summary: *The logs-based classifier does not perform as well as the text-based one when detecting toxic open-source issues and code review comments.*

RQ$_3$: *To what degree can combining existing approaches improve detection of toxicity and pushback?*

We start by comparing P-R AUC scores of the text-based and the logs-based classifiers against that of the combined classifier when detecting toxicity, on both D1 and D2, which answers **RQ$_{3.1}$**. The P-R curves are shown in Figure 3. Overall, we find that the combined classifier has better performance than the logs-based classifiers but is similar to the text-based classifier. On D1, the combined classifier outperforms the logs-based one (a t-test between the logs-based classifier and the combined classifier: $t = -51.975$, p-value $< 2.2e-16$; Cliff's $\delta = -1$; the AUC scores are 0.516 and 0.895 respectively) but is indistinguishable from the text-based classifier (a t-test between the text-based classifier and the combined classifier: $t = 0.376$, p-value = 0.712, the text-based classifier's AUC is 0.907).

Similarly, on D2, the combined classifier outperforms the logs-based classifier (a t-test between the logs-based classifier and the combined classifier: $t = -24.226$, p-value = $9.001e-12$; Cliff's $\delta : -1$; AUCs are 0.665 and 0.871 respectively). However, the combined classifier outperforms the text-based classifier (a t-test between the text-based classifier and the combined classifier: $t = -2.3884$, p-value = 0.0363; Cliff's $\delta : -0.58$ / large effect; the AUC of the text-based classifier is 0.844).

The feature importance analysis (Figure 6a in Appendix) shows that text-based features are more important in detecting toxicity than logs-based features. This suggests that, again, toxicity is more about the language than the logs-based metrics. The toxicity score and identity attack by the Perspective API have the highest importance. They are followed by the two logs-based features, which are followed by several politeness strategies. The use of second-person pronouns is also among the top 5 most important features, which echoes our findings in the word frequency analysis.

Summary: *For toxicity, the combined classifier has a similar performance to the text-based one on toxic issue comments but a better performance on toxic code review comments. The combined classifier performs better than the logs-based one in detecting toxicity.*

Finally, we compare the AUC scores between the text-based and the combined classifier and between the logs-based and the combined classifier when detecting pushback (D3 and D4), which answers **RQ$_{3.2}$**. The P-R curves are shown in Figure 2.

On *D3 Corporate Pushback Code Review Comments*, the combined classifier performs better than the logs-based (a t-test between the logs-based and the combined classifier: $t = -12.511$, p-value = $2.108e-08$; Cliff's $\delta = -1$ / large effect; AUC are 0.693 and 0.755) but about the same as the text-based one (a t-test between the text-based and the combined classifier: $t = 0.363$, p-value = 0.723; the text-based classifier's AUC is 0.757).

On the contrary, on *D4 Open-Source Pushback Code Review Comments*, the performance of the logs-based classifier is similar to the the combined classifier ($t = -2.1171$, p-value = 0.052; the average AUC scores are 0.445 and 0.455 respectively). Similarly, the combined classifier's performance is indistinguishable from that of the text-based classifier (a t-test between the text-based and the combined classifier: $t = -0.929$, p-value = 0.373, the text-based classifier's AUC is 0.447).

From the feature importance analysis on the combined classifier on our two pushback datasets D3 and D4 (Figure 6b in Appendix) shows that the logs-based features have higher importance than the text-based ones. Among the text-based ones, toxicity score and identity attack have the highest importance, followed by several politeness strategies.

Summary: *For classifying pushback in code reviews, the combined classifier performs better than the logs-based classifier but about equivalently to the text-based classifier in a corporate setting; and performs about equivalently to the text-based classifier and the logs-based classifier in an open-source setting.*

8 DISCUSSION

Classifiers' cross-domain application. For **RQ$_1$**, we found that prior classifiers' performance [18, 50] degrades when applied to new datasets. For open-source code review comments, one reason may be that, compared to issues, discussions in PRs are generally more technical, and hence, less personal. One reason the logs-based classifier performed relatively poorly in open-source code review may be that we were not able to accurately reproduce one of the corporate pushback features, active shepherding time.

Relationship between toxicity and pushback. By answering **RQ$_2$**, how well can the classifiers generalize across domains and datasets, we can conclude some relationship exists between the two concepts. Pushback is initially centered around delays in code review, which is associated with lower productivity [18], whereas toxicity is centered more around the negative interactions among contributors during code review [50]. However, Egelman et al. [18] reported that, in addition to lengthy reviews, pushback is also characterized by interpersonal conflict. This is supported by our finding that the text-based classifier has a better performance than the logs-based one on pushback detection in a corporate setting, suggesting that pushback in a corporate setting is more subtle than lengthy discussions or delayed reviews. Similarly, in open-source, toxic language is also a significant part of pushback. Among the pushback code review comments users reported, more than half of them have reasons related to communication (Figure 7 in Appendix). However, we found that the logs-based features did not improve toxicity detection. This suggests that toxicity is mostly about language, and meta-data cannot capture the nuance.

Corporate vs. open-source settings. When answering **RQ$_2$**, we were also able to compare the two contexts, corporate and open source. We found that the text-based classifier works better than the logs-based one when classifying corporate pushback. However, it was surprising that the logs-based classifier and the text-based one have similar performance when classifying open-source pushback. This differs from the impression we had from the survey responses. From the survey responses, we observed many complaints about maintainers delaying the review process. When looking at some of the PRs, we saw that many of the maintainers mentioned having a holiday or being busy with day jobs as reasons for the delay. One comment from the open-source pushback survey reflected that "It's not PR and not about code review, but it's about open source world."

Moreover, both the text-based and the logs-based classifiers have better performance on corporate pushback code review comments than on open-source ones. This suggests some differences between the two datasets. Perhaps these differences arise from uniformity in Google's code review practices [52] compared to the multitude of practices used on GitHub [23].

This also raises the issue of transferring our results to other settings. When answering $\mathbf{RQ}_1$, we found that using the same set of features on data from a different context resulted in lower performance. However, the multiple levels of comparisons we conducted in this study can act as a guideline while developing a system for toxicity and pushback detection in other contexts.

Prediction vs. classification. In this paper, we performed classification on conversations after they had concluded, largely because logs-based features are not applicable to individual comments. As a result, our current models cannot yet be applied to all scenarios where automated detection of toxicity or pushback are of interest, *e.g.*, comment-level classification for just-in-time intervention. Instead, we target primarily scenarios where thread-level classification is needed, *e.g.*, to reflect on when discussions have gone awry (of interest to practitioners) or to detect and study when, how, and why toxicity and pushback occur (of interest to researchers).

Future work can explore how to use text-based features to do real-time detection and offer editing suggestions. Cheriyan et al. [10] proposed a Conflict Reduction System that can rephrase offensive sentences. However, their datasets are heavily focused on swearing and profanity. Our findings can greatly enrich the set of text features that can be used to detect and prevent potential toxic comments.

Text analytics improvements. Our text classifier combined three different NLP techniques, but other NLP techniques on larger datasets is a future research direction. Some paths that can be explored include using text embedding [17] or conversational structure [64]. One could also use Snorkel [51], a weak supervision model, to help augment our labeled dataset.

Prior studies have shown that general NLP models may not be directly applicable to software engineering corpora [27, 29]. For example, "error" and "test" are mostly neutral in the software engineering context but have negative connotations in general English. Han et al. [26] report that Perspective API can misclassify toxic inputs due to a domain mismatch or novel lexicon of toxicity. Therefore, some fine-tuning is needed on top of the Perspective API to attain better performance. Raman et al. [50] suggested fine-tuning a classifier using a domain-specific lexicon. However, this is a difficult task that needs careful design and evaluation. Thresholds and datasets are all variables that can be explored. Moreover, when evaluating the effectiveness of the domain-specific lexicon tuning, how do we decide what words should be in the list and what should not? These questions are worth exploring in the future.

9 THREATS TO VALIDITY

Internal validity. The data we used for training and testing our classifiers is small in two respects. The first is from a machine learning perspective, where more data often yields more reliable conclusions. The second is from an ecosystem perspective; the data we studied represents a small subset of all the discussions going on within GitHub and Google, likely limiting the generalizablity of our results.

Another limitation is that our data, both existing and newly collected, rely on human raters to judge interpersonal conflict. While Egelman and colleagues' showed some degree of reliability across different raters, nonetheless perceptions of interpersonal conflict invariably differ from person to person. Such differences threaten the true accuracy of our ground truth data.

External validity. A major threat to generalizability is the context in which we collected our data. For corporate code reviews, we used data from Google; classifying code reviews in other companies would likely yield different results. Likewise, our other datasets are from GitHub; data obtained from other platforms may also yield different results.

Construct validity. The lack of comment-level labels in pushback datasets D3 and D4 likely confused the classifiers using text-based features. Because all comments within a pushback conversation share the same label, some neutral or positive comments are also labeled as pushback. Since our text-based classifier works on the comment level, it can get confused when seeing comments associated with polite strategies (*e.g.*, indirect start) and impolite strategies (direct questions) that are both labeled as pushback.

In our analysis, we bridged concepts and contexts in prior work [18, 50], between open and closed source; and issues and code reviews. However, we did not exhaustively explore this space. For instance, we did not collect data for toxic corporate code reviews or issues. Given the results that the text-based classifier works well on Google's pull requests, using it to detect or understand toxic comments may be worthwhile future work.

10 CONCLUSION

In this paper, we cross-pollinated with two techniques designed to detect interpersonal conflict. In applying these text- and logs-based techniques to broader contexts than those for which they were originally designed, we uncovered several novel insights. For instance, we found that prior work that detected code review pushback using logs data [18] can be improved substantially by analyzing the text contained in those code reviews. While the opposite was not true – logs data did not improve issue toxicity detection – we nonetheless found that logs can be a useful feature in toxicity classifiers. Building on these techniques, we envision a future where tools can help software developers learn from or avoid interpersonal conflict, enabling projects to be more inclusive of a wider variety of contributors.

ACKNOWLEDGEMENTS

This work supported in part by a Google Award for Inclusion Research. Thanks to Danny Berlin, Adam Brown, Mark Canning, and the Engineering Productivity Research team for their help during this research. Also thanks to Nithum Thain, Lucas Dixon, and Jeffrey Sorensen for their help and feedback on this study.

REFERENCES

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

[2] Adam Alami, Marisa Leavitt Cohn, and Andrzej Wąsowski. 2019. Why does code review work for open source software communities?. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 1073–1083.

[3] Lora Aroyo, Lucas Dixon, Nithum Thain, Olivia Redfield, and Rachel Rosen. 2019. Crowdsourcing subjective tasks: the case study of understanding toxicity in online discussions. In *Companion proceedings of the 2019 world wide web conference*. 1100–1105.

[4] Nicole A Beres, Julian Frommel, Elizabeth Reid, Regan L Mandryk, and Madison Klarkowski. 2021. Don't You Know That You're Toxic: Normalization of Toxicity in Online Gaming. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–15.

[5] Leo Breiman. 2001. Random forests. *Machine learning* 45, 1 (2001), 5–32.

[6] Penelope Brown and Stephen C Levinson. 1987. *Politeness: Some universals in language usage*. Vol. 4. Cambridge university press.

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

[8] Jonathan P. Chang, Caleb Chiam, Liye Fu, Andrew Wang, Justine Zhang, and Cristian Danescu-Niculescu-Mizil. 2020. ConvoKit: A Toolkit for the Analysis of Conversations. In *Proceedings of Special Interest Group on Discourse and Dialogue*.

[9] Jithin Cheriyan, Bastin Tony Roy Savarimuthu, and Stephen Cranefield. 2017. Norm violation in online communities–A study of Stack Overflow comments. In *Coordination, Organizations, Institutions, Norms, and Ethics for Governance of Multi-Agent Systems XIII*. Springer, 20–34.

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

[11] Moataz Chouchen, Ali Ouni, Raula Gaikovina Kula, Dong Wang, Patanamon Thongtanunam, Mohamed Wiem Mkaouer, and Kenichi Matsumoto. 2021. Anti-patterns in modern code review: Symptoms and prevalence. In *2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 531–535.

[12] Sophie Cohen. 2021. Contextualizing toxicity in open source: a qualitative study. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1669–1671.

[13] Mia Consalvo. 2012. Confronting toxic gamer culture: A challenge for feminist game studies scholars. (2012).

[14] Maral Dadvar, Dolf Trieschnigg, Roeland Ordelman, and Franciska de Jong. 2013. Improving cyberbullying detection with user context. In *European Conference on Information Retrieval*. Springer, 693–696.

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

[16] Jesse Davis and Mark Goadrich. 2006. The relationship between Precision-Recall and ROC curves. In *Proceedings of the 23rd international conference on Machine learning*. 233–240.

[17] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

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

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

[20] Isabella Ferreira, Kate Stewart, Daniel German, and Bram Adams. 2019. A longitudinal study on the maintainers' sentiment of a large scale open source ecosystem. In *2019 IEEE/ACM 4th International Workshop on Emotion Awareness in Software Engineering (SEmotion)*. IEEE, 17–22.

[21] Daviti Gachechiladze, Filippo Lanubile, Nicole Novielli, and Alexander Serebrenik. 2017. Anger and its direction in collaborative software development. In *2017 IEEE/ACM 39th International Conference on Software Engineering: New Ideas and Emerging Technologies Results Track (ICSE-NIER)*. IEEE, 11–14.

[22] R Stuart Geiger. 2017. Summary analysis of the 2017 github open source survey. *arXiv preprint arXiv:1706.02777* (2017).

[23] Georgios Gousios, Martin Pinzger, and Arie van Deursen. 2014. An exploratory study of the pull-based software development model. In *Proceedings of the 36th International Conference on Software Engineering*. 345–355.

[24] Georgios Gousios and Diomidis Spinellis. 2012. GHTorrent: GitHub's data from a firehose. In *2012 9th IEEE Working Conference on Mining Software Repositories (MSR)*. IEEE, 12–21.

[25] Emitza Guzman, David Azócar, and Yang Li. 2014. Sentiment analysis of commit comments in GitHub: an empirical study. In *Proceedings of the 11th working conference on mining software repositories*. 352–355.

[26] Xiaochuang Han and Yulia Tsvetkov. 2020. Fortifying toxic speech detectors against veiled toxicity. *arXiv preprint arXiv:2010.03154* (2020).

[27] Matthew J Howard, Samir Gupta, Lori Pollock, and K Vijay-Shanker. 2013. Automatically mining software-based, semantically-similar words from comment-code mappings. In *2013 10th working conference on mining software repositories (MSR)*. IEEE, 377–386.

[28] Hemant Ishwaran. 2007. Variable importance in binary regression trees and forests. *Electronic Journal of Statistics* 1 (2007), 519–537.

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

[30] David Jurgens, Eshwar Chandrasekharan, and Libby Hemphill. 2020. A just and comprehensive strategy for using NlP to address online abuse. In *57th Annual Meeting of the Association for Computational Linguistics, ACL 2019*. Association for Computational Linguistics (ACL), 3658–3666.

[31] S Jalil Kazemitabar, Arash A Amini, Adam Bloniarz, and Ameet Talwalkar. 2017. Variable importance using decision trees. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 425–434.

[32] Svetlana Kiritchenko, Isar Nejadgholi, and Kathleen C Fraser. 2021. Confronting abusive language online: A survey from the ethical and human rights perspective. *Journal of Artificial Intelligence Research* 71 (2021), 431–478.

[33] Robin Lakoff. 1973. The logic of politeness: Or, minding your p's and q's. In *Proceedings from the Annual Meeting of the Chicago Linguistic Society*, Vol. 9. Chicago Linguistic Society, 292–305.

[34] Robin Lakoff. 1977. What you can do with words: Politeness, pragmatics and performatives. In *Proceedings of the Texas conference on performatives, presuppositions and implicatures*. ERIC, 79–106.

[35] Nolan Lawson. 2017. What it feels like to be an open-source maintainer. *Read the Tea Leaves. https://nolanlawson. com/2017/03/05/what-it-feels-like-to-be-an-open-source-maintainer* (2017).

[36] Alyssa Lees, Daniel Borkan, Ian Kivlichan, Jorge Nario, and Tesh Goyal. 2021. Capturing Covertly Toxic Speech via Crowdsourcing. In *Proceedings of the First Workshop on Bridging Human–Computer Interaction and Natural Language Processing*. 14–20.

[37] Jan Lehnardt. 2017. Sustainable Open Source: The Maintainers Perspective or: How I Learned to Stop Caring and Love Open Source. https://writing.jan.io/2017/03/06/sustainable-open-source-the-maintainers-perspective-or-how-i-learned-to-stop-caring-and-love-open-source.html. [Online; accessed 19-July-2021].

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

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

[40] Dawn Nafus. 2012. 'Patches don't have gender': What is not open in open source software. *New Media & Society* 14, 4 (2012), 669–683.

[41] Nicole Novielli, Fabio Calefato, and Filippo Lanubile. 2018. A Gold Standard for Emotions Annotation in Stack Overflow. In *Proc. of 15th Int'l Conf. on Mining Software Repositories* (Gothenburg, Sweden) *(MSR 2018)*. 14–17. https://doi.org/10.1145/3196398.3196453

[42] Adewale Obadimu, Esther Mead, Muhammad Nihal Hussain, and Nitin Agarwal. 2019. Identifying toxicity within youtube video comment. In *International Conference on Social Computing, Behavioral-Cultural Modeling and Prediction and Behavior Representation in Modeling and Simulation*. Springer, 214–223.

[43] Marco Ortu, Bram Adams, Giuseppe Destefanis, Parastou Tourani, Michele Marchesi, and Roberto Tonelli. 2015. Are bullies more productive? Empirical study of affectiveness vs. issue fixing time. In *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*. IEEE, 303–313.

[44] Bo Pang and Lillian Lee. 2009. Opinion mining and sentiment analysis. *Comput. Linguist* 35, 2 (2009), 311–312.

[45] Gregory Park, H Andrew Schwartz, Johannes C Eichstaedt, Margaret L Kern, Michal Kosinski, David J Stillwell, Lyle H Ungar, and Martin EP Seligman. 2015. Automatic personality assessment through social media language. *Journal of personality and social psychology* 108, 6 (2015), 934.

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

[47] John Pavlopoulos, Jeffrey Sorensen, Lucas Dixon, Nithum Thain, and Ion Androutsopoulos. 2020. Toxicity Detection: Does Context Really Matter? *arXiv preprint arXiv:2006.00998* (2020).

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

[49] Md Mustafizur Rahman, Dinesh Balakrishnan, Dhiraj Murthy, Mucahid Kutlu, and Matthew Lease. 2021. An Information Retrieval Approach to Building Datasets for Hate Speech Detection. *arXiv preprint arXiv:2106.09775* (2021).

[50] Naveen Raman, Minxuan Cao, Yulia Tsvetkov, Christian Kästner, and Bogdan Vasilescu. 2020. Stress and Burnout in Open Source: Toward Finding, Understanding, and Mitigating Unhealthy Interactions. In *International Conference on Software Engineering, New Ideas and Emerging Results (ICSE)*. ACM, 57–60.

[51] Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid training data creation with weak supervision. In *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, Vol. 11. NIH Public Access, 269.

[52] Caitlin Sadowski, Emma Söderberg, Luke Church, Michal Sipko, and Alberto Bacchelli. 2018. Modern code review: a case study at google. In *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice*. 181–190.

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

[54] Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In *Proceedings of the fifth international workshop on natural language processing for social media*. 1–10.

[55] Daniel Schneider, Scott Spurlock, and Megan Squire. 2016. Differentiating Communication Styles of Leaders on the Linux Kernel Mailing List. In *Proceedings of the 12th International Symposium on Open Collaboration*. 1–10.

[56] Vandana Singh and William Brandon. 2019. Open source software community inclusion initiatives to support women participation. In *IFIP International Conference on Open Source Systems*. Springer, 68–79.

[57] Sara Owsley Sood, Elizabeth F Churchill, and Judd Antin. 2012. Automatic identification of personal insults on social news sites. *Journal of the American Society for Information Science and Technology* 63, 2 (2012), 270–285.

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

[59] Igor Steinmacher, Tayana Conte, Marco Aurélio Gerosa, and David Redmiles. 2015. 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*. 1379–1392.

[60] Parastou Tourani, Bram Adams, and Alexander Serebrenik. 2017. Code of conduct in open source projects. In *2017 IEEE 24th international conference on software analysis, evolution and reengineering (SANER)*. IEEE, 24–33.

[61] Bogdan Vasilescu, Daryl Posnett, Baishakhi Ray, Mark GJ van den Brand, Alexander Serebrenik, Premkumar Devanbu, and Vladimir Filkov. 2015. Gender and tenure diversity in GitHub teams. In *Proceedings of the 33rd annual ACM conference on human factors in computing systems*. 3789–3798.

[62] Sebastian Wachs, Michelle F Wright, and Alexander T Vazsonyi. 2019. Understanding the overlap between cyberbullying and cyberhate perpetration: Moderating effects of toxic online disinhibition. *Criminal Behaviour and Mental Health* 29, 3 (2019), 179–188.

[63] Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. Ex machina: Personal attacks seen at scale. In *Proceedings of the 26th international conference on world wide web*. 1391–1399.

[64] Justine Zhang, Jonathan P Chang, Cristian Danescu-Niculescu-Mizil, Lucas Dixon, Yiqing Hua, Nithum Thain, and Dario Taraborelli. 2018. Conversations gone awry: Detecting early signs of conversational failure. *arXiv preprint arXiv:1805.05345* (2018).

[65] Ting Zhang, Bowen Xu, Ferdian Thung, Stefanus Agus Haryono, David Lo, and Lingxiao Jiang. 2020. Sentiment analysis for software engineering: How far can pre-trained transformer models go?. In *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 70–80.

11 APPENDIX

Table 3: Over and underrepresented words in *D1 Toxicity in Open-Source Issues Comments*. N-grams with second-person pronouns are in bold. N-grams with software engineering terms are underlined.

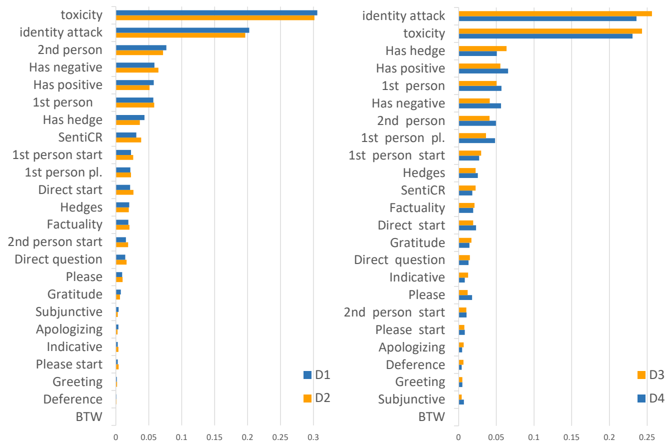
	unigram	z-score	bigram	z-score	ngram	z-score
Toxic	**you**	30.77	this is	12.756	this is not	6.013
	it	23.724	in the	11.822	**you want to**	5.217
	that	22.437	**you are**	11.651	**you need to**	4.869
	of	22.051	it is	10.608	there is no	4.303
	and	21.318	**you have**	9.389	**if you want**	4.272
	is	18.917	to be	9.371	**you have to**	4.036
	this	18.524	**that you**	9.145	to do with	4.036
	your	18.121	**if you**	8.727	**if you want to**	3.971
	have	16.647	to do	7.535	part of the	3.94
	what	15.62	have to	7.514	the problem is	3.799
Non-toxic	via	-3.526	team and	-2.825		
	unit	-3.82	plenty of	-2.838		
	team	-3.871	of experience	-2.954		
	assigned	-3.979	with our	-2.972		
	returns	-4.32	and provide	-2.972		
	function	-4.452	to remove	-3.037		
	item	-5.104	with an	-3.042		
	ticket	-5.121	issue was	-3.263		
	duplicate	-5.528	assigned to	-3.44		
	click	-5.62	looking for	-3.573		

Table 4: Over and underrepresented words in *D3 Pushback in Corporate Code Review*. N-grams with second-person pronouns and gratitude are in bold. N-grams with software engineering terms are underlined.

label	unigram	z-score	bigram	z-score	ngram	z-score
Push back	<tech1>	5.352	**you want**	3.04	**you want to**	2.792
	tests	4.452	want to	2.849	on nov at pm	2.637
	<tech2>	3.683	of these	2.849	nov at pm	2.577
	our	3.599	of our	2.626		
	build	3.564	is to	2.575		
	libraries	3.362	if we	2.525		
	break	3.245	depend on	2.464		
	thing	3.197	we use	2.464		
	see	3.177	the cl	2.441		
	rollback	3.152	this case	2.311		
Non-push-back	submit	-5.338	to represent	-3.831	make sure the	-2.566
	groups	-5.485	to me	-3.834	to do the	-2.64
	feature	-5.514	how about	-3.882	not sure if	-2.69
	<tech3>	-5.64	to submit	-3.96	seems to be	-2.805
	map	-5.664	this function	-4.106	in this cl	-2.813
	rate	-6.042	the new	-4.286	which is not	-2.919
	thanks	-6.303	**could you**	-4.363	**do you have**	-3.189
	section	-6.336	for the	-4.432	how do we	-3.604
	the	-6.492	change the	-4.439	to change the	-4.009
	for	-6.9	**thanks for**	-5.291	**thanks for the**	-4.891

Table 5: Over and underrepresented words in *D4 Pushback in Open-Source Code Review*. N-grams with second-person pronouns, gratitude, and "code of conduct" are in bold. N-grams with software engineering terms are underlined.

label	unigram	z-score	bigram	z-score	ngram	z-score
Push back	runtime	17.511	is of	6.622	**the code of**	3.957
	suggestion	9.676	the project	6.452	the new format	3.721
	argument	9.32	**code of**	6.171	for the new	3.457
	us	8.762	of type	6.006	the commit message	3.185
	people	8.35	the linter	4.638	to the project	3.096
	timer	8.218	read the	4.583	as long as	3.003
	non	7.254	it is	4.313	the number of	3.003
	high	7.068	social media	4.186	to read the	2.957
	requirements	6.29	the old	4.13	just wanted to	2.874
	de	6.276	commit message	4.026	we dont want	2.874
Non-push-back	access	-5.923	the following	-3.402		
	struct	-5.992	an error	-3.412		
	config	-6.197	it seems	-3.536	is going to	-2.311
	tests	-6.282	the same	-3.715	it would be	-2.5
	server	-6.351	**thank you**	-3.786	all of the	-2.5
	line	-6.431	the server	-4.021	this should be	-2.802
	field	-6.632	did not	-4.047	it seems that	-2.872
	build	-7.262	the tests	-4.12	**thank you for**	-2.972
	info	-7.309	file line	-4.287	let me know	-3.111
	error	-7.319	line in	-5.301	file line in	-4.287



(a) D1 vs D2 **(b) D3 vs D4**

Figure 4: Text-based classifier feature importance scores.

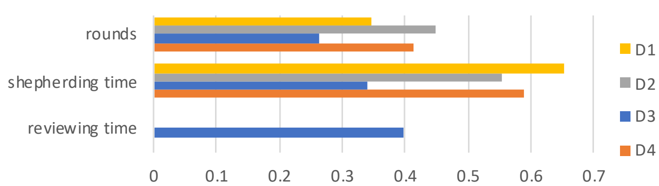
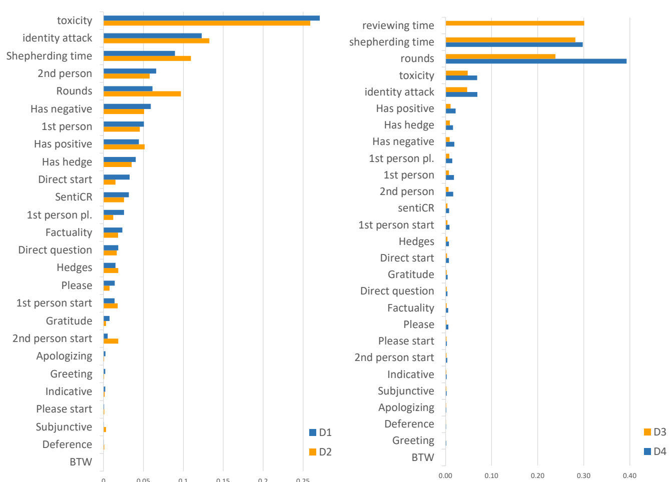


Figure 5: Logs-based classifiers' feature importance



(a) Toxicity (D1 and D2) **(b) Pushback (D3 and D4)**

Figure 6: Combined classifiers' feature importance

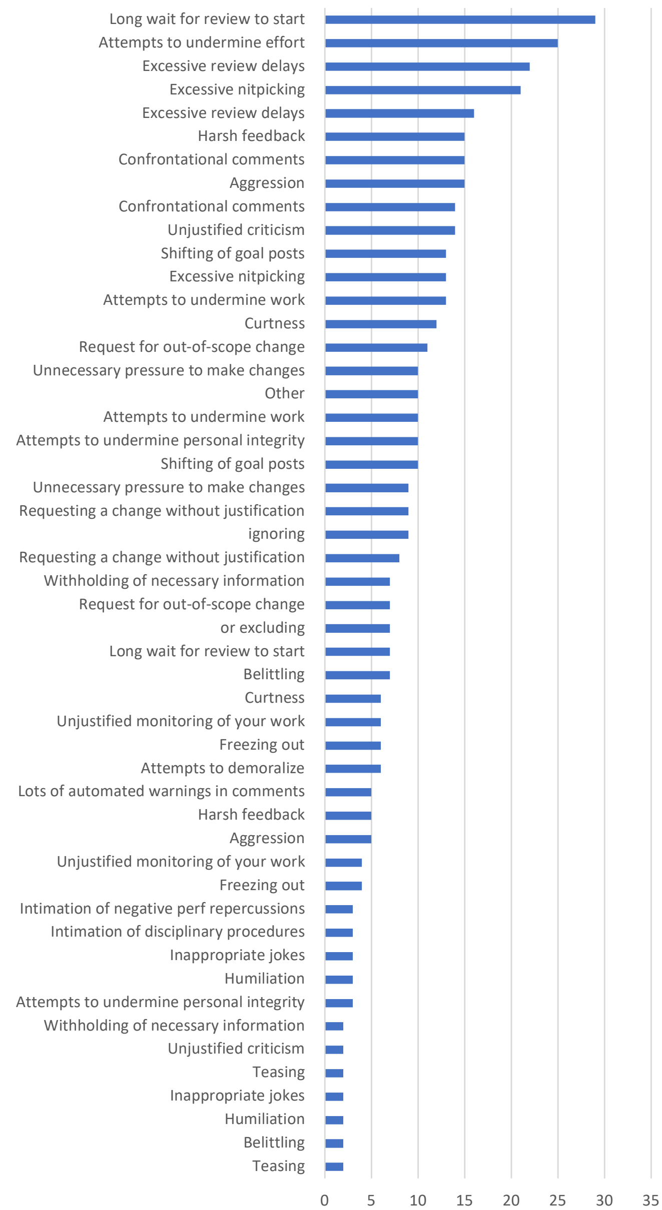


Figure 7: Reasons for pushback in OSS