Do Words Have Power? Understanding and Fostering Civility in Code Review Discussion

MD SHAMIMUR RAHMAN, ZADIA CODABUX, and CHANCHAL K. ROY, University of Saskatchewan, Canada

Modern Code Review (MCR) is an integral part of the software development process where developers improve product quality through collaborative discussions. Unfortunately, these discussions can sometimes become heated by the presence of inappropriate behaviors, often referred to as incivility. The negative behaviors encompass personal attacks, insults, disrespectful comments, and derogatory conduct. While researchers have extensively explored such incivility in various public domains, our understanding of its causes, consequences, and courses of action remains limited within the professional context of software development, specifically within code review discussions. To bridge this gap, our study draws upon the experience of 171 professional software developers representing diverse development practices across different geographical regions. Our findings reveal that more than half of these developers (56.72%) have encountered instances of workplace incivility, and a substantial portion of that group (83.70%) reported experiencing such incidents at least once a month. We also identified various causes, positive and negative consequences, and potential courses of action for uncivil communication. Moreover, to address the negative aspects of incivility, we propose a model for promoting civility that detects uncivil comments during communication and provides alternative civil suggestions while preserving the original comments' semantics, enabling developers to engage in respectful and constructive discussions. An in-depth analysis of 2K uncivil review comments using eight different evaluation metrics, followed by a manual evaluation, suggested that our proposed approach could generate civil alternatives significantly compared to the state-of-the-art politeness and detoxification models. Moreover, a survey involving 36 developers who used our civility model reported its effectiveness in enhancing online development interactions, fostering better relationships, increasing contributor involvement, and expediting development processes. Our research is a pioneer in generating civil alternatives for uncivil discussions in software development, opening new avenues for research in collaboration and communication within the software engineering context.

CCS Concepts: • **Human-centered computing → Empirical studies in collaborative and social computing**; • **Software and its engineering → Collaboration in software development**; • **Social and professional topics → Incivility**;

Additional Key Words and Phrases: Code Review, Civil and Uncivil Comments, Online Communities, Pretrained Translation Model

ACM Reference Format:
Md Shamimur Rahman, Zadia Codabux, and Chanchal K. Roy. 2024. Do Words Have Power? Understanding and Fostering Civility in Code Review Discussion. 1, 1 (April 2024), 25 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Authors' address: Md Shamimur Rahman, mdr614@usask.ca; Zadia Codabux, zadiacodabux@ieee.org; Chanchal K. Roy, chanchal.roy@usask.ca, University of Saskatchewan, Saskatoon, Saskatchewan, Canada, S7N1L5.

1 INTRODUCTION

Code review is a widely adopted software engineering process of inspecting developers' source code by other team members before merging code to the codebase. This process is also known as peer review because code changes submitted by a developer (also called author) are reviewed or verified by one or more peers. These practices ensure the software quality [8, 9, 82] through early detection of software bugs, technical debt, code smells, use of projects guidelines and coding style, and overall knowledge sharing among development teams [13, 14, 56, 59, 87]. Several well-known companies, including Microsoft [6], Google [72], and Oracle [22], and popular open-source projects (e.g., Android, QT, OpenStack [60] and ECLIPSE [50]) implement and get benefits from code review as a significant component of their development strategy.

Traditionally, code review required face-to-face interactive meetings and discussions among developers regarding checklist-based code inspection [29]. In contrast, todays' Modern Code Review (MCR) leverages convenient platforms such as Gerrit [60], ReviewBoard [12], Phabricator [85], GitHub pull request [11] with less formal asynchronous code changes discussion and also supports geographically distributed review [6]. Authors submit a set of cohesive modifications to the source code, known as a patch, which either introduces a new feature or resolves a software bug. Subsequently, a group of expert developers and maintainers, referred to as project reviewers, undertake the task of evaluating the patch and determining whether it should be incorporated into the project (patch acceptance) or rejected. During the review process, authors and reviewers engage in discussions encompassing various aspects of the submitted patches, such as their relevance to the project, the suitability of the proposed solution design, specific implementation details, and more. These discussions involve reviewers seeking clarifications, offering suggestions, and providing constructive feedback, while developers justify their approach and address reviewers' suggestions in an effort to persuade reviewers to accept the patch. Although the reviewers and contributing authors share a common objective of enhancing functionality and resolving issues within the software project, their interests during the code review process can diverge or even conflict, resulting in instances of disagreements, incivility, personal attacks, unnecessary disrespectful behavior, and interpersonal conflicts through their discussions or review comments [30, 44, 62, 65, 68, 74, 77, 78]. That way of communication could potentially decrease developers' productivity as they become frustrated by peers who eventually contemplate leaving the project permanently [13, 71]. Additionally, toxic interactions with existing team members could impede the successful onboarding of newcomers [46, 79].

The issue of incivility or inappropriate behavior in public discussions, particularly in software development and code review contexts, has received increasing attention in recent years. Researchers have extensively studied the characteristics, causes, and consequences of uncivil communication [14, 27, 30]. The state-of-the-art toxicity detector called ToxiCR [77] has been developed specifically for automatically identifying toxic code review comments. However, the existing research primarily focuses on categorizing the potential reasons and impacts of incivility by analyzing open-source code review comments without offering viable solutions to promote civil communication. Although these studies have suggested measures such as maintaining a code of conduct, providing encouragement, and implementing proper training, these approaches have proven less effective in real-world environments [84]. This is evident in cases like the Linux community and code review platforms like GitHub and Gerrit, where despite having established codes of conduct, instances of incivility persist in their code review discussions. Therefore, it is crucial to explore alternative methods to address the issue of uncivil communication within the realm of software development.

In this paper, we propose an approach designed to support developers in fostering civility in online discussions with their peers by offering constructive and respectful real-time suggestions when identifying uncivil or disrespectful comments. Specifically, our focus lies in transforming uncivil comments into positively-toned alternatives that preserve the same underlying semantic meanings. By adopting this approach, developers will be aware of the presence of toxic or disrespectful tones in their comments and receive alternative suggestions that convey the same message, thus, ensuring healthy communication and workplace dynamics. In order to accomplish our objectives, we gathered a total of 6.4K code review comments that were deemed uncivil in review discussions. These comments were sourced from prior research [30, 70, 76] on six projects, such as Android, Chromium OS, LibreOffice, AnkiDroid, and others. Subsequently, we manually converted these uncivil comments into positive-toned equivalents, and the dataset was then utilized to train various Neural Machine Translation (NMT) [7, 81] algorithms, ultimately creating a translator tool that provides alternative positive-toned and civil comments whenever uncivil comments are detected. Additionally, prior research [14, 28, 30, 31, 65] has primarily focused on open-source projects, relying only on the analysis of sentiment in comments exchanged among developers during code reviews. While this approach has provided valuable insights into the prevalence of incivility and even identified potential reasons and impacts, it has been limited by its inability to discern the underlying human motivations and the real-world consequences experienced by developers as a result of uncivil interactions. Therefore, our research seeks to broaden the perspective by incorporating individuals' experiences. It is worth noting that open-source communities extend the opportunity for voluntary contributions to software developers, contrasting with the more structured and often mandatory nature of software development within industrial contexts. This disparity in context can result in distinct dynamics when it comes to incivility and its impact. Thus, our study examines how incivility affects developers in industry settings, shedding light on the unique challenges in such environments. To achieve this, we conducted surveys among industry developers to gain insights into workplace communication dynamics, potential causes of incivility, and strategies employed to foster civil discussions. The contributions of our study are as follows:

- **A summarised knowledge on incivility:** We found potential reasons and impacts after reviewing six studies on incivility within the software development context. The literature, particularly centred on open-source projects, highlights inappropriate communication leading to reduced collaboration and productivity, increased project abandonment, and elevated stress levels. With limited prior studies, we surveyed 171 industry developers, gathering real-world insights into the practical effects of incivility.
- **A refinement of incivility detection model:** We significantly enhanced ToxiCR [77] by expanding the training dataset and conducting thorough testing across 40 GitHub projects, aligning with our research goals. The refined model now captures nuances like mocking, teasing, and flirtatious interactions, even without explicit profanity. Furthermore, we comprehensively compared the accuracy of ToxiCR [77], politeness detector [25, 52, 61], toxicity classifier [24, 51, 67], GPT4.0, and the refined model to detect uncivil comments using a manually labeled dataset.
- **A dataset for fostering civility:** We have compiled a novel dataset comprising 6.4K uncivil comments and their corresponding civil counterparts, meticulously curated from GitHub projects and prior research [30, 70, 76]. The quality of translation for these comments underwent rigorous evaluation by two independent evaluators, resulting in a robust inter-rater agreement (i.e., Cohens' Kappa [55] coefficient of +0.97). This high agreement solidifies the reliability and applicability of our dataset to promote civility within code review discussions and possesses the potential for fine-tuning pre-trained models in various domains [34].
- **A model for translating uncivil comments:** Our research introduces a novel translation tool aimed at fostering healthy and constructive communication patterns among developers during

code review discussions. This tool harnesses the capabilities of pre-trained language models[1], including T5, BART, NLLB, and MarianMT, all of which are specialized in text-to-text generation and fine-tuned using our prepared dataset. We also fine-tuned the popular GPT3.5[2] model with our dataset to boost the domain knowledge. When our detection model identifies an uncivil comment, the translation model seamlessly transforms it into a polite, respectful, and civil alternative. Users have the flexibility to choose from different translation models. Through rigorous manual analysis, we have demonstrated that the proposed tool generates civil alternatives for uncivil comments, achieving an impressive accuracy range of 79-95%.

- **An evaluation study using GitHub projects and through developers' perceptions:** We rigorously assessed our detection model across 40 GitHub projects, showcasing its effectiveness by outperforming all other detectors with precision, recall, and F1-score of 86%, 100%, and 92%, respectively. For the evaluation of our translation model, we conducted a comprehensive analysis using eight state-of-the-art politeness transfer and detoxification models on 2K uncivil comments. The results, measured across eight evaluation metrics, highlight the superior effectiveness of our translation model in fostering civility within code review discussions. Additionally, a survey involving 36 developers using our tool indicated unanimous agreement on the positive impact of our approach in fostering civility in online development discussions. All the implementations, analysis, and survey questionnaires are available in our replication package[3].

2 BACKGROUND AND RELATED WORK

This section presents an overview of the contextual details and explores relevant research across three main areas: i) the sociological concept of incivility, ii) investigations pertaining to incivility within Open Source System (OSS) development platforms, and iii) studies specifically centred on incivility within open-source code review discussions.

2.1 Sociological Concept of Incivility

Incivility in communication is a multifaceted phenomenon that is challenging to define due to its subjectivity, influenced by various factors like communication medium, location, culture, and the relationship between participants. Different fields have examined incivility and its opposite, civility, without reaching a consensus on its precise definition. While some equate civility with politeness, Bejan [10] argues that labeling someone as uncivil carries a more severe connotation than simply being impolite. According to Bejan, impoliteness can be tolerated to some extent, whereas incivility cannot. In the political context, Brooks and Geer [15] defined civility in terms of mutual respect. When investigating incivility on social media, Maity et al. [53] defined it as "*the act of sending or posting mean text messages with the intention of mentally hurting, embarrassing, or humiliating another person using electronic devices such as computers and cellphones.*" Coe et al. [21], focusing on online comments in news reports, proposed a broad definition of incivility as "*aspects of discussion that convey an unnecessarily disrespectful tone towards the discussion forum, its participants, or its topics.*" According to their definition, incivility is deemed unnecessary as it fails to contribute constructively to the discussion.

Incivility is an overarching concept, encapsulating various related forms that collectively contribute to the erosion of respectful communication and healthy interactions. For instance, it comprises numerous toxic acts such as issuing threats, making offensive or sexually explicit remarks [26], engaging in insulting language or mockery [5], perpetrating harassment, bullying, griefing,

[1]https://bit.ly/TranslationModels-HuggingFace
[2]https://openai.com/blog/gpt-3-5-turbo-fine-tuning-and-api-updates
[3]https://zenodo.org/records/10775868

and trolling [1], as well as exhibiting various antisocial behaviors like hate speech, flaming, and cyberbullying [57]. Such encounters with offensive and toxic textual content within online platforms, exemplified by platforms like Facebook and Twitter, can exert detrimental consequences that may induce emotional distress [86, 89] and adverse psychological effects [38], discouraging active participation in online communities [4]. Research indicates that toxic and patronizing language [66] frequently targets members of minority groups [63, 83], potentially inciting real-life violence against them [20, 64]. Consequently, various studies have addressed the proliferation of harmful content on online platforms. Approaches such as text detoxification [24, 35, 42, 51, 67] transform textual content from being offensive, harmful, or uncivil into neutral, respectful, and constructive language. Politeness transfer [52, 61] involves enhancing the politeness level of texts, fostering courteous and considerate communication. Additionally, counter-responses to hateful speech [36, 73] provide a mechanism to address negativity, aiming to create an environment conducive to constructive discussion and fostering a more engaged and positive online discourse [58].

2.2 Incivility in OSS Platforms

Researchers in software engineering have investigated the effect of adverse communication in the OSS development process. The adverse communication styles have had a profound impact on developers across the spectrum, including newcomers who face communication barriers during their onboarding into OSS projects [46, 79, 80], as well as frequent contributors who endure stress and burnout due to toxic interactions [17, 71]. Recent studies have highlighted the presence of toxic communication within issue discussions on GitHub [57, 71]. Miller et al. [57] conducted a qualitative investigation to gain deeper insights into toxicity within OSS development and curated a sample of 100 Github issues that encompassed diverse forms of toxic interactions, such as insults, arrogance, trolling, entitlement, and unprofessional behavior. Their analysis also revealed notable differences in the manifestation of toxicity within OSS communities compared to other online platforms like Reddit or Twitter. Similar to incivility, numerous studies have investigated the notion of conflict within OSS development. Filippova et al. [32, 33] have investigated to understand the types, origins, and consequences of conflict in OSS projects from the perspective of contributors. Their findings indicate that conflicts often stem from divergent viewpoints on technical tasks, development procedures, and community norms. In assessing the efficacy of conflict management strategies, Huang et al. [43] determined that providing concrete, constructive suggestions at the technical level proved effective in mitigating the adverse effects of conflict.

2.3 Incivility in Code Review

Ferreira et al. [30] qualitatively investigated incivility in Linux Kernel Mailing Lists, revealing common forms of incivility in code review discussions: frustration, name-calling, and impatience, which raises awareness about the need for respectful and constructive communication in code review discussions. Egelman et al. [27] explored negative experiences during code review, termed *"pushback,"* where reviewers block change requests due to unnecessary conflict. They formulated three metrics that can be used to predict feelings of pushback due to negative interpersonal interactions during code review and highlight interactions that may reduce frustration and stress and benefit from a self-intervention [3]. Qiu et al. [68] explored the potential for extending the applicability of the toxicity detector and pushback detector beyond their original contexts and types of discussion. Additionally, it investigates how the integration of these two detectors [27, 71] can enhance the detection of interpersonal conflicts. Gunawardena et al. [41] surveyed 93 developers, highlighting how destructive criticism threatens gender diversity in software development, discouraging women from continuing.

However, the studies have no or limited discussions about specific strategies for mitigating uncivil communication that could lead to increased interpersonal conflicts among developers and other negative consequences. Although a few general approaches like implementing a code of conduct and providing training were mentioned, none delved into concrete solutions [84]. Furthermore, there was limited analysis on how developers address conflict issues, especially in closed-source projects. Therefore, our study sheds light on potential courses of action to resolve these issues and explores how an automatic civility model can promote a healthy and constructive communication environment.

3 STUDY DESIGN

To thoroughly examine the factors leading to incivility in OSS code review discussions and understand its impact by drawing insights from closed-source industries, we used a combination of qualitative and quantitative research methods. Our methodology entails summarising prior studies to outline the existing knowledge and surveying industry developers to gain their experience on incivility. Recognizing the importance of promoting civil communication, we developed a translator tool that can suggest constructive and positive alternatives when developers engage in inappropriate criticism of submitted code changes. Through our analysis and experiments, our objective is to address the following three Research Questions (RQs):

- **RQ1: How does incivility manifest within software development teams, and what are its discoursal consequences?** Here, we consolidate prior research and conduct a developer survey to uncover insights that can guide the necessity of fostering more constructive and harmonious discussion environments.
- **RQ2: What courses of action do developers commonly take when faced with incivility, and how can we implement strategies to promote a civil discussion environment?** This RQ identifies the usual courses of action developers generally take when confronted with incivility and determines effective strategies to promote civility in online discussions, including enhancing review comments with more respect and positive sentiments.
- **RQ3: How effective and feasible are the strategies for fostering civil discussions in open-source and industrial contexts?** Based on RQ2 findings, we aim to assess the effectiveness and practical usability of an automated approach that transforms uncivil comments into civil alternatives through a comprehensive comparative analysis, which includes state-of-the-art politeness transfer and detoxification models. The assessment involves utilizing eight evaluation metrics and conducting a survey with developers.

The methodology we have employed in our study can be comprehensively summarized as follows, outlining the approach and processes undertaken to investigate and address the research questions.

3.1 Outline Incivility on Existing Studies and Developers Survey

While the number of previous studies specifically addressing the concept of incivility in software development remains limited, we initiate our investigation by examining the first study [30] that explored the origins and impacts of uncivil communication during the code review process. To further explore relevant research, we examine the references and citations of the primary paper, continuing this process until no new studies are identified. Initially, we employ selection criteria that prioritize papers examining developer unhappiness, job satisfaction, emotions, burnout, toxic discussion, workplace bullying, project abandonment, negative feelings, review fairness, conflict, and related topics by thoroughly reviewing study titles. This approach leads us to identify 39 studies investigating non-coding concerns among developers, primarily within OSS platforms. However, since our study focuses specifically on incivility in code review discussions, we narrow our focus to retrieve only six relevant studies that specifically discuss the potential causes and consequences of uncivil behaviors among developers during code review.

Due to the limited available studies and their primary focus on code review discussions within OSS projects, as well as the lack of direct emphasis on code review in the conducted surveys [27, 30, 41], we intend to conduct an online survey targeting professional software developers available on LinkedIn and 17 companies. Our objective is to gain insights into the occurrence of incivility within their development teams, identify potential causes and impacts on team dynamics, and understand the strategies employed to address such incidents. The survey overview and participant selection process are discussed in the supplementary document.

3.2 Fostering Civility in Team Discussion

A civil way of communication in team discussions offers several benefits, including improved collaboration, enhanced communication effectiveness, increased trust and respect among team members, and ultimately, higher productivity and better overall team performance [27, 30, 40, 41, 71]. On the other hand, incivility can lead to strained relationships, reduced trust and cooperation, increased conflict, and decreased job satisfaction [30, 37, 39, 54]. Eventually, it can create a work environment that negatively affects the well-being and productivity of team members. Based on existing studies and our survey findings, incivility in software development and team dynamics has predominantly negative consequences. Given the widespread contribution of developers in OSS projects from diverse geographical locations, hybrid modes of work opportunities in industries, and the prevalence of communication channels for discussing development-related concerns, these channels serve as a significant source of incivility. To address this issue, we propose an automated approach to prevent the sending of uncivil comments and provide alternative civil sentences with the same meaning in real-time. This approach aims to promote civil communication while allowing commenters to express their thoughts respectfully. Notably, over 80% of the survey participants expressed agreement with our proposed approach, recognizing its potential to foster a civil communication environment. The development process of the proposed tool is outlined in the subsequent sections.

3.2.1 Automatic Translation of Uncivil Review Comments: Our proposed methodology for translating uncivil review comments into civil discourse consists of two main components: incivility detection and translation. In Step 1 of the process, comments are categorized as either civil or uncivil. Civil comments are made visible to peers directly, while toxic or uncivil ones proceed to Step 2, where they undergo translation using our translation tool. The translated comments are then reassessed in Step 1 to ensure civility. This iterative process continues until the comments meet civility standards or reach a predetermined number of iterations. We will provide a brief discussion of the models and their training dataset.

Detection Model: To identify whether a comment exhibits incivility, we initially employed the state-of-the-art toxicity detector ToxiCR[4] that can detect toxicity in code review interactions [75]. Although ToxiCR [77] has various choices of learning algorithms, we only replicated the Bidirectional Encoder Representations from Transformer (BERT) model for its superior performance, achieving 96% accuracy and 89% F1-score compared to other models [75, 77]. However, ToxiCR faces challenges in identifying uncivil comments that lack explicit profanity, such as mockery, provocation, flirtation, or references to gender, race, or identity. For instance, consider these comments: *"This code is a trainwreck. Do you even know how to code?"* and *"This is basic knowledge. How do you not get it?"*, ToxiCR struggled to flag these comments as toxic. We mitigated this limitation by creating a comprehensive dataset consisting of 4.7K comments, encompassing both uncivil and

[4]https://github.com/WSU-SEAL/ToxiCR

civil remarks. These comments were collected from GitHub projects and generated by ChatGPT[5], and underwent manual analysis. The main challenge encountered during comment generation was that ChatGPT inherently avoids producing negative or offensive sentences. Therefore, we crafted a base prompt: *"What is meant by Uncivil comments? Could you show some examples of uncivil comments and their civil alternatives for learning purposes in the context of code review in software development online discussion?"* We adapted the context to code quality, bug fixing, new features, and enhancements that usually come as pull requests. However, in certain cases, we prompted follow-up questions to extract examples. This resulted in 1.7k pairs of uncivil and corresponding civil comments. Initially, ToxiCR, when applied to the new dataset, achieved only a 25% F1-score. To enhance its performance, we combined the new dataset with ToxiCR, resulting in over 22K review comments for training and 2.4K for testing. This integration substantially improved the detection model, yielding an overall F1-score of 92%. A random manual evaluation of 1K review comments confirmed the superior performance of the updated model, as discussed in Section 4.

Translation Model: Our translation approach primarily utilizes pre-trained transformer models from the HuggingFace library[1], including T5 (Text-to-Text Transfer Transformer) [69], BART (Bidirectional and Auto-Regressive Transformers) [48], NLLB (No Language Left Behind) [23], and MarianMT (Multilingual and Efficient Neural Machine Translation) [47]. T5, developed by Google Research[6], is versatile and fine-tuned for various NLP tasks. Similarly, BART, from Facebook AI[7], excels in text generation while preserving original meanings. NLLB builds upon BART, extending machine translation to diverse languages, including low-resource ones. MarianMT specializes in multilingual translation and is trained on parallel corpora from various languages. We used multiple versions of each model with varying training parameters, resulting in seven fine-tuned translation models (e.g., T5-large, T5-base, T5-small, BART-large, BART-base, NLLB, and MarianMT). Additionally, we fine-tuned GP3.5[2] to translate uncivil comments into civil alternatives. Detailed analysis is presented in Section 4.

Dataset Collection: The process of preparing a dataset for translating incivility presents a significant challenge, as to the best of our knowledge, there is currently a lack of readily available data specifically designed for this purpose. As a result, creating a dataset that includes uncivil review comments and their corresponding civil counterparts requires considerable effort. First, we have collected the publicly available labeled toxic code review datasets [76]. The dataset includes 3,757 toxic comments with a lot of duplicated comments due to sampling for machine learning training. Additionally, we have collected uncivil review comments from Ferreira et al. [30] and Rahman et al. [70] and finally, the dataset comprising 6.4K code review comments ready for conversion into civil/non-toxic counterparts. The collected dataset and its translated version are available in our replication package[3] for further research.

Dataset Preparation: Table 1 in the supplementary document presents the rubric for translating 6.4K uncivil review comments into civil language. This rubric is comprised of eight translation rules (TRs) that guide the translation process. The first seven rules, derived from Sarker et al. [77], outline criteria for labelling review comments as toxic. Acknowledging that incivility encompasses broader aspects beyond explicit toxicity, we introduced the eighth rule specifically to identify comments lacking explicit profanities, and ToxiCR [77] faced challenges in this regard. The task of converting uncivil comments was initially a collaboration between the first author and ChatGPT[5]. The given prompt was *"Give a civil, modest, polite, decent, gentle version of this following comment maintaining original meaning and normal online conversational patterns. The 'normal*

[5]https://chat.openai.com/
[6]https://github.com/google-research/text-to-text-transfer-transformer
[7]https://bit.ly/BART-MetaResearch

online conversational pattern' refers to the typical way people engage in conversation in the context of online communication. (comment)". However, due to ChatGPTs' limitations in maintaining civil conversation, including inconsistent translations, use of alternative uncivil words, overly formal language, preference for complex sentences, and reduced accuracy with lengthy comments, the author manually reviewed ChatGPTs' output with follow-up instructions to change prior responses to ensure that the comments adhered to conventional online communication norms. This thorough process ensured that the resulting civil comments conveyed the intended message while promoting respectful communication. After the initial conversion, two independent reviewers assessed each comment and reached a consensus on the translations. In cases of disagreement, they discussed and revised as needed. The inter-rater reliability, measured by Cohens' Kappa coefficient [55], was +0.97, signifying nearly perfect agreement between the reviewers. We employed the generated civil versions of the uncivil ones as non-toxic labeled data for training the detection model, as elaborated in the *"Detection Model"* section.

3.2.2 Model Evaluation: In our research, we conducted extensive evaluations of the models we developed. Firstly, we employed two evaluation approaches to assess the detection model. We used a test dataset from previous research and gathered a substantial dataset of over 200K open-source code review comments from 40 GitHub projects, selecting based on specific criteria, including over 100 contributors, 10,000 commits, and active pull requests. From this dataset, we randomly sampled 1K comments (at a 95% confidence level with a 5% error margin) for manual labeling. Two independent evaluators assigned labels, demonstrating a high level of agreement (Cohens' Kappa coefficient of +0.91). We then compared these labels to our detection models' predictions. Moreover, we incorporated the Standford Politeness Detector (PD) [25], a non-polite text detector used by politeness transfer models [52, 61], alongside a toxic text classifier[8] employed in detoxification models [24, 51, 67], and leveraged insights from the recent GPT4.0[9] model for a comprehensive comparative analysis on incivility detection.

Secondly, to evaluate the translation model, we employed a set of eight automatic evaluation metrics (details in Section 5 of the Supplementary document), including Incivility Decrease, Semantic Similarity [88], Sentence Similarity [92], Length Dissimilarity of Translated Texts, Perplexity [18], and three sentiment analyzers (SentiCR [2], SentiStrength-SE [45], and RoBERTa [16, 49]). Despite our translation model being specifically tailored for the code review context, we conducted a comprehensive comparison with domain-specific politeness transfer models (Polite ChatBot [61] and politeness transfer by Madan et al. [52]), detoxification models (Marco [42], ParaDetox [51], CondaBert and ParaGedi [24], COUNT [67], and DiffuDetox [35]), as well as the GPT4.0 model. Subsequently, we randomly selected 100 toxic comments from the collected review comments on GitHub and manually assessed the translation quality. This evaluation considered three criteria: i) the extent of minimal change required to transform the original comment into a civil one, ii) the preservation of the original meaning, and iii) the selection of appropriate words and sentence structures, with a deliberate avoidance of overly formal language and complex sentence constructions. These meticulous evaluations allowed us to rank the performance of our models in enhancing the civility of code review online interactions.

Additionally, we conducted a survey where 36 participants interacted with our detection and translation models, providing their comments and assessing the model-generated outputs. This survey provided invaluable user feedback and real-world insights into the practical usability and effectiveness of our models.

[8]https://huggingface.co/s-nlp/roberta_toxicity_classifier
[9]https://openai.com/research/gpt-4

4 RESULTS

In this section, we present the outcomes and findings that respond to our research questions, shedding light on various aspects related to incivility within code review discussions.

4.1 RQ1: Manifestations of Incivility and Their Discoursal Impact

We conducted a two-fold analysis to address RQ1 concerning existing knowledge and developer perspectives on incivility. First, we synthesized prior research discussing incivility, destructive criticism, interpersonal conflict, and their effects on development. Second, we conducted an online survey with 171 developers from different roles and industries worldwide. The following sections provide detailed insights into these analyses.

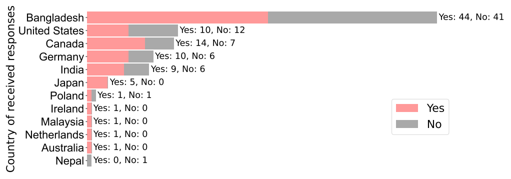
4.1.1 Prior Research Knowledge on Incivility. Table 2 in the supplementary materials provides a comprehensive overview of key findings extracted from prior studies that investigated the origins and impacts of incivility within software development. An informal survey by Ferreira et al. [30] unveiled that code review discussions held significant potential for contentious interactions, including personal attacks, mockery, name-calling, threats, and vulgarity. To delve deeper into the sources and widespread impact of uncivil communication, their findings revealed that a substantial portion (66.66%) of non-technical Linux kernel emails exhibited uncivil characteristics. The primary triggers for uncivil discussions included inappropriate solutions proposed by developers, poor code quality, disregard for instructions, and violations of community conventions. Interestingly, uncivil comments also originated from developers when they received suboptimal suggestions or solutions, faced rejections lacking proper explanations, or perceived excessive effort demanded by reviewers. Similarly, Miller et al. [57] identified prevalent forms of incivility and toxicity, notably featuring entitled, arrogant, and insulting comments, often stemming from technical and ideological disagreements. Additionally, users' inability to utilize applications, reviewers receiving inadequate code with proper problem descriptions and incorrect language versions, and newcomers or authors encountering a lack of reviews and prior negative experiences were identified as potential triggers for toxicity in discussions. Furthermore, four other studies [19, 27, 39, 41] delved into gender and power-related discrimination within development teams. They highlighted various sources of incivility, including destructive criticism of work, an excessive focus on identifying faults, and unnecessary requests for code changes. A lack of appreciation or support from peers often accompanied these sources. Conversely, developers faced harsh criticism due to factors such as low code quality, violations of community standards, and non-compliance with reviewers' instructions.

Incivility within software development carries significant repercussions, profoundly affecting both individuals and collaborative teams. The rejection of patches, especially after developers have invested considerable effort, often results in frustration and can lead to retaliatory actions, including verbal attacks on reviewers [30]. This toxic environment may compel reviewers to discontinue communication with the offending developers or reject patches without providing adequate explanations [30, 57]. In more severe cases, maintainers may opt to replace reviewers who engage in destructive criticism and even resort to banning developers from further contributions, further intensifying the impact on team dynamics [30, 41, 57]. On a broader scale, incivility can result in the disengagement of newcomers to the development process, hindering their integration into the team. The stress and burnout experienced by team members subjected to incivility can have a detrimental effect on their performance and motivation. Turnover intentions may increase as individuals seek more supportive and less hostile work environments [41]. Furthermore, incivility can erode team unity, teamwork, and effectiveness. It can create an unsafe space where team members are hesitant to seek help or express their ideas for improvement, leading to a stagnation of innovation [41]. Incivility can also raise questions about the value of individual contributions and skills within the team. In some instances, it may even manifest as discrimination, particularly against minorities and women, further perpetuating a hostile atmosphere [19, 39, 41]. Regarding individual experiences, incivility can impact moods, resulting in discomfort and reduced productivity. It may also affect cognitive skills, leading to frustration and sloppy work. Moreover, incivility can manifest in extended code review times, as well as the quality of relationships among peers [19, 27].

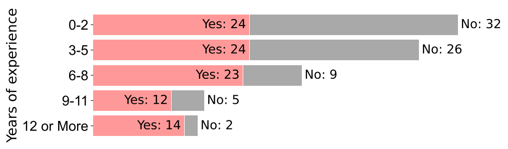
This overview highlights the nature of incivility, addressing its causes and diverse impacts on individuals and development teams. However, prior research has limitations, focusing mainly on open-source development [30, 57], hypothetical scenario-based surveys [41], and providing limited assessments of uncivil interactions [19, 27, 39]. To gain deeper insights into closed-source dynamics, we conducted an online survey with 171 respondents from diverse companies worldwide.

4.1.2 Developers' Perception on Incivility: In our survey of 171 participants, the majority were from Bangladesh (49.7%), followed by the USA (12.86%), Canada (12.28%), and nine other countries (in Figure 1a). Notably, 56.72% of respondents reported experiencing incivility during their software development careers. This prevalence varied based on participants' years of experience (Figure 1b). As shown in Figure 1c, 83.7% encountered incivility at least monthly, with 14.3% facing it weekly and an additional 16.3% mentioning irregular encounters, often early in their careers. For example, a developer with over 12 years of experience noted experiencing incivility "*more frequently when I joined as a junior developer*." Examining the participants experiencing weekly instances of uncivil conversations reveals a noteworthy distribution. The majority, constituting 43.75% and 25%, possess 0-2 and 3-5 years of experience, respectively, while a smaller yet significant proportion consists of senior developers—12.50% with 6-8 years and 18.75% with nine or more years of experience. For those participants who reported monthly encounters of incivility (69.4%), have 0-2 (24.28%) and 3-5 years (28.57%) of experience, followed by 6-8 years (20%), and nine or more years (27.14%). Geographically, incivility is more prevalent on a monthly basis across Bangladesh, USA, Canada, India, and Germany, with 70.45%, 90%, 64.29%, 77.78%, and 60% of participants, respectively. Additionally, about 10-35% of participants from Bangladesh, Canada, USA, and India experienced incivility every week. Notably, only participants from Bangladesh (11.36%) reported facing uncivil review comments daily. Meanwhile, irregular encounters predominantly involve developers with nine or more years of experience (56.25%), mentioned by 40%, 60%, and 6.81% of participants from Germany, Japan, and Bangladesh, respectively. These findings highlight varying levels of incivility exposure at different career stages.

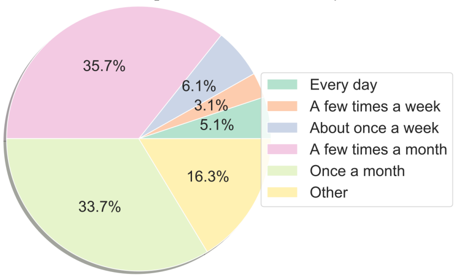
Discoursal causes: In Figure 1d, we observe key factors contributing to incivility in online software development discussions. We received a total of 271 responses potentially triggering uncivil conversations, primarily from Bangladesh (46.86%), followed by Canada (11.44%), Germany (9.59%), USA (9.23%), India (6.64%), Japan (5.9%), and others (10.7%). In terms of development experience, the majority falls within 3-5 years (27.62%), followed by 6-8 years (22.96%), 0-2 years (21.40%), 9-11 years (15.95%), and 12 or more years (12.06%), respectively. The most prominent factor is the excessive workload of developers, with 71 responses, primarily from experienced developers with six or more years of experience. This increased workload results from their critical responsibilities, including mentoring junior developers, making crucial decisions, and ensuring project success, leading to heightened stress and incivility. Two other significant factors involve design issues, code quality, and the submission of untested buggy code (65 responses). Senior developers, typically shouldering more extensive responsibilities due to their experience, may submit code with design and quality issues. Junior developers reviewing this code often face the challenge of identifying and addressing these problems, leading to incivility. Similarly, junior developers submitting buggy code or not following instructions place an additional burden on senior developers to review (55 responses).



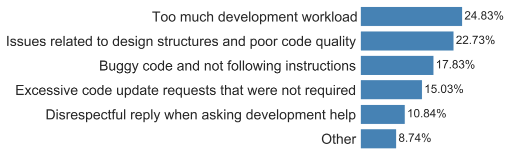
(a) Responses based on country.



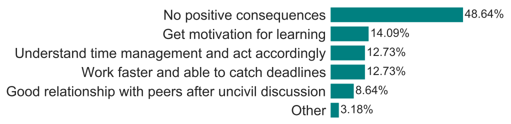
(b) Responses based on years of development experience.



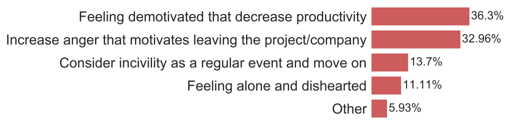
(c) Frequency of occurrence of uncivil incidents.



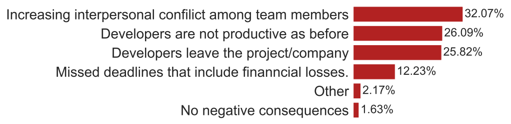
(d) Potential reasons of uncivil incident.



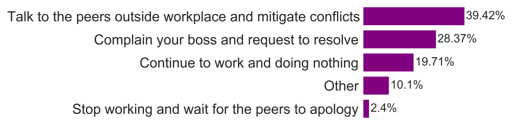
(e) Potential positive aspect of incivility.



(f) Developers' feelings after uncivil discussions.



(g) Potential negative aspect of incivility.



(h) Potential course of actions taken to resolve conflicts.

Fig. 1. Comprehensive summary of our conducted online developer survey.

Furthermore, other causes of incivility stem mainly from developers with 0-2 and 3-5 years of experience. These developers often face excessive workloads or code refinement tasks assigned by seniors or project leads (43 responses). Disrespectful responses can occur when seeking clarifications or posting development queries, especially when one lacks knowledge of development stacks early in their career (31 responses). Additionally, incivility sources include a lack of appreciation, insufficient guidance after task assignments, absence of logical arguments, and undervaluing years of experience. Regarding gender-related discrimination, we received only six responses (3.5%) indicating that female developers are sometimes less promoted and appreciated. One developer expressed, *"Sometimes I feel like, despite outperforming my male colleagues, my contributions are not adequately appreciated."* Conversely, some responses mentioned that female developers face less criticism, and their performance may be perceived as poorer compared to males.

Discoursal impacts: After investigating incivility causes, our research sought to uncover its potential consequences in software development. Our data collection included insights from participants on both positive (Figure 1e) and negative outcomes (Figure 1g), along with their emotional experiences (Figure 1f) during these incidents.

- **Positive impacts:** About half of the respondents (48.64%) did not perceive any positive outcomes, highlighting the predominantly negative nature of incivility in software development discussions. However, we received 113 responses from Bangladesh (48.67%), USA (16.81%), Canada (10.62%), Germany (9.73%), and India (3.54%) that reveal the positive impacts of incivility. According to the result, developers with 0-2 years (34.38%) and 3-5 years (38.54%) of experience reported more positive impacts from uncivil interactions compared to their more experienced counterparts with over six years of experience (27.08%). This suggests that less experienced developers perceive incivility as a form of constructive criticism or feedback, leading to improvement, while more experienced developers are less receptive to such interactions. Specifically, developers (31 responses) get inspiration to improve their skills or understanding of certain topics in response to uncivil feedback. Despite the negativity, this feedback prompts self-reflection and self-improvement, leading to new knowledge acquisition, skill enhancement, or seeking additional learning resources. It also underscores the determination of these individuals to turn negative experiences into opportunities for personal and professional growth. One developers' perspective emphasizes this positivity, as they stated, *"Try to learn fast and get experienced and leave the company."* This shift in mindset can significantly impact team dynamics and future projects, emphasizing the need for supportive work environments aligning with company goals. Moreover, our survey revealed that 28 respondents reported improved time management, enhanced efficiency in completing work, and meeting deadlines. Additionally, some developers (19 responses) found ways to resolve conflicts through better communication and mutual understanding with their peers. Overall, experiencing uncivil or destructive criticism can catalyze exploring new learning areas and refining team-building strategies.

- **Negative impacts:** We thoroughly analyzed the adverse consequences of incivility in software development, considering both immediate emotional impacts on developers and broader implications for project teams. We analyzed a total of 622 responses, encompassing immediate feelings (42.60%) and overall consequences (57.40%). The impact of uncivil interactions is notably pronounced in less experienced developers, particularly those with 0-2 years (31.32%) and 3-5 years (29.43%) of experience. While more experienced developers (6-8 years, 9-11 years, and 12+ years) also experience negativity, the rates are significantly lower (18.11%, 11.70%, and 9.43%, respectively). This suggests that developers may adapt and develop resilience to incivility over time through learning coping mechanisms. In contrast, almost all developers (91.81% of total participants) expressed potential threats to the overall project, indicating the long-term impacts of incivility among team members. Therefore, it is crucial to ensure civility in discussions to provide a supportive environment for early-career professionals and uphold the overall well-being of projects and teams.

  Developers' feelings during uncivil discussions: Our analysis revealed that incivility during discussions significantly impacted the emotional well-being of developers. A significant portion of respondents expressed feeling demotivated, which led to decreased productivity (98 responses). This demotivation hindered their engagement with tasks, resulting in lower work quality. Additionally, a substantial number of developers reported experiencing heightened anger, and some even contemplated leaving their current project or company (89 responses). This anger could have negative effects on team dynamics and retention rates. A smaller subset of developers felt isolated and disheartened (30 responses), while others viewed incivility as a common occurrence and chose to move forward (37 responses). Nevertheless, these emotional responses underscore the adverse effects of incivility on developers, impacting their well-being and commitment to work. Developers also reported feeling insecure, distressed, and distracted during uncivil discussions.

  Overall negative impacts: Incivility during code discussions had significant and widespread consequences, both emotionally and operationally. It resulted in heightened interpersonal conflicts among team members (118 responses), disrupting project progress and straining team cohesion, thus negatively affecting project continuity and effectiveness. Furthermore, a significant number of developers (96 responses) experienced reduced productivity following incidents of incivility. Notably, 95 respondents disclosed that they left their projects or companies due to negative incivility-related experiences. This turnover disrupted ongoing projects, causing missed deadlines and incurring additional costs for recruiting and training new team members, resulting in financial losses to the company (45 responses). Developers also reported mental health issues, negative impressions of their peers, and a distressing work atmosphere, impacting their personal lives. While a minority of respondents (6 responses) did not report experiencing any negative consequences, the majority highlighted the substantial impacts of incivility on both their personal and professional lives, underscoring the significance of addressing this issue.

4.2 RQ2: Responding to Incivility and Strategies for Fostering Civility

RQ2 draws insights from the survey into the usual courses of action employed by developers when confronted with incivility during online team discussions and extracts potential strategies for fostering civility.

4.2.1 Courses of Action Responding Incivility: Figure 1h illustrates the diverse courses of action (a total of 205 responses) commonly taken by developers when faced with incivility during online team discussions. Notably, a substantial majority of respondents (82 responses and mainly from less experienced developers (61.76%)) opt for proactive conflict resolution by engaging in direct conversations with peers involved in uncivil discourse. This approach reflects a concerted effort to address the conflict constructively and restore a harmonious working relationship. Another significant action chosen by 59 participants (mostly from Canada, USA and Germany and developers with six or more years of experience (64.41%)) involves reporting the issue to their superiors and seeking formal intervention and assistance in conflict resolution. This underscores the acknowledgment of the seriousness of incivility and the willingness to involve higher authorities when necessary. In contrast, a relatively smaller proportion (41 responses) indicates a preference for continuing to work without taking specific actions, adopting a *"take the punch and move on"* mentality. Conversely, a minority of respondents (5 responses) opt to pause their work and await apologies or further discussion from their peers, suggesting that this passive approach is less common among developers. One respondent mentioned, *"I try to identify any issues within myself, and if the problem isn't on my end, I wait for the next discussion."* Additionally, participants (mostly experienced developers with more than nine years of experience) mentioned several other actions taken in response to incivility, including asserting boundaries by informing the other party that such incidents are unacceptable, initiating initial conversations with respective peers, escalating the matter to authorities if similar events recur, temporarily ceasing responses until uncivil peers regain composure, resolving conflicts with a focus on technical context, and applying emotional intelligence to navigate the situation effectively. These various courses of action demonstrate developers' multifaceted approaches to addressing incivility within their professional interactions.

4.2.2 Courses of Strategies Fostering Civility: In our efforts to tackle incivility, we thoroughly explored prior research for valuable insights and potential solutions. Moreover, via the survey, we solicited developers' input on effective strategies for fostering civility in online team discussions.

Strategies from prior research: Three of the six studies we analyzed (Table 2 in the supplementary document) offered comprehensive recommendations. The study by Ferreira et al. [30] emphasized fostering a culture of respect, politeness, and humility in online discussions. It highlighted the importance of avoiding personal attacks, differentiating between the target and the problem, and gracefully accepting mistakes. Additionally, both the studies by Gunawardena et al. [41] and Ferreira et al. [30] emphasized the significance of delivering constructive feedback to maintain civility. They stressed the importance of providing feedback in a manner that does not trigger adverse reactions from code authors. To bridge the gap between newcomers and established developers, the studies by Ferreira et al. [30] and Miller et al. [57] recommended comprehensive training programs. These programs familiarize individuals with community conventions and ensure adherence to expected code quality standards. Furthermore, Miller et al. [57] introduced the concept of implementing a platform-wide system to report toxic behavior. This system, equipped with warnings, strikes, and potential for user bans, effectively addresses uncivil conduct. In summary, these studies collectively advocate for clear and considerate language in interactions and recommend open communication between developers and reviewers as the most effective means of resolving conflicts.

Strategies from conducted survey: Ensuring civility in collaborative settings is essential yet it comes with challenges. We compiled insights and strategies from 127 software developers who shared thoughts on developing a culture of civility in online team discussions. All the provided responses are categorized into six contexts and rigorously reviewed by two evaluators with strong agreement (Cohens' Kappa coefficient: +0.98). We have briefly outlined each category as follows.

- **Building positive relationships (51 responses):** Developers recognize the importance of positive team relationships in promoting civil communication. They emphasize the value of respect, mutual learning, and recognizing individual contributions. They also highlight setting aside personal differences, acknowledging skills, and promoting positive behavior.
- **Communication guidelines and accountability (43 responses):** Developers underscored the importance of clear and transparent communication in upholding civility. They advocated for well-defined codes of conduct, clear communication standards, proper guidelines, and expectations for professional interactions to reduce misunderstandings and conflicts. This category also emphasizes effective feedback mechanisms, boundaries, and consequences for incivility, promoting workplace well-being and fostering open and inclusive discussions. Developers believed that accountability for behavior and interactions deters incivility. Encouraging reporting and thorough investigations are vital steps in ensuring accountability.
- **Training and skill development (15 responses):** Developers suggested training programs and skill development initiatives to familiarize all team members with community conventions, code quality standards, and communication expectations. These programs aim to provide guidance on effective and respectful communication and equip individuals with the interpersonal skills needed for civil interactions in professional settings.
- **Work-Life balance and flexibility (6 responses):** Developers emphasized the significance of work-life balance and flexibility in work to foster civility. They suggested creating an environment that respects the personal lives of team members and offers flexibility to meet project deadlines without undue pressure. Developers believe that reducing unnecessary work-related stress and accommodating individual needs can foster a more harmonious and civil professional atmosphere.
- **Conflict resolution (8 responses):** In this category, developers emphasized the importance of addressing conflicts promptly and directly through open and respectful discussions involving relevant parties as needed. Mediation, proactive approaches, and in-person meetings are viewed as effective methods for managing conflicts and preserving civil communication.
- **Leadership behavior (4 responses):** Developers highlighted leaderships' role in promoting civil communication, suggesting that managers and team leaders model civil behavior by demonstrating active listening, empathy, and patience. This leadership example can profoundly impact team dynamics, inspiring respectful and professional interactions among team members.

4.2.3 Our Proposed Strategy to Foster Civility in Discussion. After examining various strategies proposed in prior research and by industry developers to promote civility in code review discussions, we have identified challenges in applying these strategies to both open-source and closed-source projects, considering team members' presence or absence in physical and virtual spaces. Notably, i) participation differs between open-source (voluntary) and industry (compulsory) projects; ii) both project types have distinct codes of conduct, yet incivility issues persist, suggesting predefined rules

Table 1. Result of comparative analysis classifying civil and uncivil code review comments

Model	**Accuracy**	**Precision**	**Recall**	**F1-score**	**# of False Negative**
ToxiCR	73.89%	86.26%	63.75%	73.31%	207
Stanford PD [25]	52.11%	62.54%	37.13%	46.59%	359
Polite ChatBot [52, 61]	58.52%	66.23%	53.59%	59.24%	265
Detoxification Classifier [24, 51, 67]	50.34%	100%	11.73%	21.01%	504
GPT4.0	55.47%	**98.37%**	21.19%	34.87%	450
Our Refined Model	**90.73%**	85.86%	**100%**	**92.39%**	**0**

may not suffice; iii) relying on senior team members for mediation can be hindered by their own uncivil behavior. One respondent inquired, *"What steps can be taken when the boss is involved in instances of incivility in the workplace?"* and iv) training could be practical for industry professionals but faces unique challenges in open-source due to its volunteer-driven nature.

Therefore, we propose an automated approach to transform uncivil comments into civil ones during communication. This involves using a detection model to identify incivility and recommend one or more civil alternatives to the comment authors. This approach has two key benefits: i) it alerts comment authors to potential code of conduct violations, discourages uncivil behavior, and promotes a respectful atmosphere; ii) it saves authors time by providing civil alternatives, improving the overall quality of their comments, and helping them develop constructive communication skills. The following section explores the effectiveness of our models in fostering civility in discussion.

4.3 RQ3: Effectiveness and Feasibility of Civil Discourse Strategies

In this section, we assess the performance of our proposed civil discourse strategy, using the refined incivility detector and five models to transform uncivil comments into civil ones. Moreover, we performed a comparative analysis with cutting-edge politeness transfer and detoxification models.

4.3.1 Evaluation of Detection Model: We comprehensively evaluated our detection model using three approaches. Firstly, we used labelled testing data to assess its accuracy in distinguishing between civil and uncivil comments. Secondly, we applied our model to a dataset of over 200K unlabeled comments from 40 GitHub projects, with manual verification on a random sample of 1K comments. Finally, we compared our models' performance against ToxiCR, the renowned Stanford PD [25], state-of-the-art Polite ChatBot [52, 61], the RoBERTa toxicity classifier[8] employed in cutting-edge detoxification studies [24, 51, 67], designed to identify texts deviations from politeness norms, such as rudeness, sarcasm, harmful or offensive language, hate speech, threats, and various explicit forms of impoliteness and toxicity. Additionally, we incorporated OpenAI GPT4.0[9], aiming to assess its default efficiency and practicality in detecting uncivil characteristics within comments.

We trained our detection model with a dataset of over 22K labeled review comments for training and validation, and more than 2.4K review comments for testing. The results show that our model excelled in classifying comments as civil or uncivil, achieving an accuracy of 96.22%, a precision of 92.77%, a recall of 91.84%, and an F1-score of 92.30%. We then applied the model to a GitHub dataset containing over 200K comments, identifying 6,860 comments (3.43%) as uncivil. To ensure the models' reliability, we manually evaluated random 1K of the collected GitHub comments, comprising 571 uncivil and 439 civil comments. This evaluation resulted in an accuracy of 90.73%, a precision of 85.86%, a perfect recall of 100% (no uncivil comments classified as civil), and an F1-score of 92.39%. Finally, we compared our detection model to ToxiCR, Stanford PD, Polite ChatBot, RoBERTa toxicity classifier and GPT4.0, using the same sample of 1K comments. The comprehensive findings presented in Table 1 demonstrate the superior performance of our detection model in categorizing civil and uncivil code review comments compared to all other models.

4.3.2 Evaluation of Translation Model: We evaluated the effectiveness of our translation model in three ways. First, we assessed how well it transformed uncivil comments into more civil ones using eight evaluation metrics (as shown in Table 2) on a random sample of 2K uncivil comments from

Table 2. Efficiency of proposed models on translating uncivil comments into civil counterparts.

Models	Incivility Decrease	Semantic Similarity	Sentence Similarity	Length Dissimilarity	SentiCR	SentiStrength-SE	RoBERTa	J	Perplexity
T5	**94.90%**	88.98%	53.44%	11.93% ↑	32.17% ↑	65.04% ↑	37.09% ↑	55.88%	5.73
BART	93.92%	**92.40%**	**62.40%**	**3.06%** ↑	21.91% ↑	**65.90%** ↑	32.27% ↑	50.14%	6.47
NLLB	91.59%	90.63%	59.08%	5.98% ↑	27.97% ↑	50.29% ↑	25.57% ↑	48.52%	**5.64**
MarianMT	79.81%	88.09%	59.54%	15.76% ↑	21.21% ↑	37.39% ↑	31.21% ↑	44.17%	6.64
GPT3.5 (FT)	90.09%	85.79%	41.49%	29.98% ↑	**51.05%** ↑	64.45% ↑	**39.37%** ↑	**57.54%**	6.58

Here, ↑ presents increasing trends.

Table 3. Performance comparison on translation efficiency among politeness and detoxification models.

Analyzer	ChatBot [61]	Polite Transfer [52]	Marco [42]	ParaDetox [51]	ParaGedi [24]	CondaBert [24]	COUNT [67]	DiffuDetox [35]	GPT4.0	Proposed Model
Incivility Decrease	76.71%	71.10%	65.63%	76.58%	89.14%	76.44%	75.58%	76.41%	90.95%	**94.90%**
Semantic Similarity	77.70%	89.31%	92.18%	83.14%	90.34%	**98.45%**	82.83%	81.25%	86.22%	92.34%
Sentence Similarity	39.53%	64.85%	80.05%	46.20%	51.76%	**85.59%**	46.15%	44.47%	38.40%	62.40%
Length Dissimilarity	↓ 47.19%	↓ 1.94%	↓ 1.37%	↓ 53.79%	↓ 33.72%	↑ 6.33%	↓ 53.71%	↓ 55.64%	↑ 56.45%	↑ **3.06%**
SentiCR	↑ **56.44%**	↑ 31.87%	↑ 13.38%	↑ 47.93%	↑ 16.30%	↑ 41.36%	↑ 48.66%	↑ 45.01%	↑ 42.89%	↑ 51.05%
SentiStrength-SE	↑ 19.63%	↑ *11.46%	↑ *0.29%	↓ *3.01%	↓ *1.58%	↓ *1.86%	↑ *0.14%	↓ *3.72%	↑ 43.41%	↑ **65.90%**
RoBERTa	↑ 30.66%	↑ 12.67%	↑ 13.90%	↑ 25.84%	↑ 25.76%	↑ *6.05%	↑ 27.31%	↑ 25.35%	↑ 27.47%	↑ **46.08%**
J	40.84%	28.18%	1.92%	15.21%	8.86%	9.06%	0.96%	17.46%	45.35%	**67.20%**
Perplexity	8.31	8.25	10.43	8.93	8.71	5.17	9.18	7.59	**4.67**	5.64

Here, ↑, ↓, and ∗ present increasing, decreasing and statically not significant (i.e., $p > 0.05$) trends, respectively.

our GitHub dataset. Among the metrics, we applied two software engineering domain-specific (i.e., SentiCR [2] and SentiStrength-SE [45]), and one general (i.e., Roberta [16, 49]) sentiment analyzers to determine whether the model successfully elicited more positive sentiments in the translated comments. We used the Wilcoxon signed rank test [91] with a 95% confidence level (i.e., p-value<0.05) to test for significance, aiming to detect any improvement in sentiment where "$\mathcal{H}_0$: There is no significant difference in sentiment scores between uncivil comments and their translated civil alternatives." and "$\mathcal{H}_1$: There is significant difference in sentiments after translation." Second, we thoroughly analyzed the scores achieved by our top-performing model compared to two state-of-the-art politeness transfer models, six detoxification models, and the latest GPT4.0 model (as illustrated in Table 3). Third, we evaluated the quality of the generated civil alternatives by randomly selecting 100 uncivil comments and their corresponding translation from each sub-model. Two evaluators manually assessed these comments, ranking the translation quality of the six distinct sub-models (T5, BART, NLLB, MarianMT, Fine-tuned GPT3.5 and GPT4.0).

While negative sentiments do not always indicate incivility, incivility is often associated with negative sentiments, as uncivil comments tend to convey hostility, disrespect, or offensive language. We aimed to assess how effectively these sub-models transformed uncivil comments into more positive and civil expressions. Table 2 summarizes the individual sub-model performance across various metrics. A higher value indicates better results for all metrics except Length Dissimilarity and Perplexity, where lower values are preferred. Additionally, we propose a joint metric J calculated as the harmonic mean of the first seven metrics to establish an overall ranking of the sub-models.

The translated version of 2K uncivil comments by our models revealed a significant improvement in civility, surpassing 90% in most cases—exceeding MarianMTs' improvement of 79.81%. Moreover, context similarity remained high, ranging between 85% and 92%, while sentence similarity (i.e., token order) was preserved at above 53% by all models except GPT3.5, which exhibited more significant modifications (41.49%). Sentence length at the character level also generally increased slightly, except for GPT3.5, causing a higher increase (around 30%). Importantly, across all sentiment analyzers and sub-models, p-values consistently fell below the 0.05 threshold, demonstrating statistically significant results and rejecting the null hypothesis. This confirms that our translation models effectively integrated more positive sentiment into the translated civil comments. Overall, considering the joint metrics J and Perplexity, we find that the performance of T5 (55.88%) and fine-tuned GPT3.5 (57.54%) is closely matched among other models. Notably, GPT3.5 exhibits a slightly higher J score, while T5 achieves a lower perplexity (5.73). Other models also demonstrate commendable perplexity values, surpassing the original uncivil comments (9.77).

Table 3 offers a comprehensive comparison of two politeness transfer, six detoxification and GPT4.0 models in their ability to translate uncivil comments with the scores given by our prepared models. Notably, our proposed model outperforms all other models in reducing incivility, achieving a remarkable 94.90% decrease. This signifies its superior ability to transform uncivil code review comments into civil counterparts. While GPT4.0 (90.95%) and ParaGedi (89.14%) also exhibit commendable performance, ParaGedis' tendency to shorten sentences (33.72%) results in significant information loss, and GPT4.0 tends to lengthen sentences (56.45%) during translation excessively. Similar decreasing trends are observed in ChatBot (47.19%), Paradetox (53.79%), COUNT (53.71%), and DiffuDetox (55.64%), while Polite Transfer (1.94%) and Marco (1.37%) show lower decreasing trends and CondaBert (6.33%) and our model (3.06%) exhibit lower increasing trends.

In the case of semantic similarity and sentence similarity, CondaBert stands out with the highest scores at 98.45% and 85.59%, followed by Marco with scores of 92.18% and 80.05%. Upon conducting an in-depth analysis of the translated dataset, we observed that these models faced challenges translating uncivil comments that lacked explicit toxic or rude words. In several instances, the returned sentences were incomplete, and the models struggled to find alternatives for toxic words, indicated by tokens being masked with '##,' resulting in the persistence of incivility. Moreover, in the context of the domain-specific sentiment analyzer SentiCR and the general RoBERTa sentiment analyzer, all models demonstrated an increase in positive sentiments in translated comments, ranging from six to over 56%. However, for SentiStrength-SE, all the politeness transfer and detoxification models showed negligible improvements, which were not statistically significant. On the contrary, our proposed model exhibited a significant enhancement in positive sentiments in the translated comments. Notably, our proposed model outperformed all others comprehensively, achieving a J score of 67.20%, with the closest competitor being GPT4.0 (45.35%). While GPT4.0 excelled in incivility reduction (90.95%), it struggled with an increased sentence length of 56.45% and lower sentence similarity of 38.40% (i.e., a higher number of token changes). Additionally, our model demonstrated lower token-level perplexity (5.64) compared to most models, indicating higher confidence and accuracy in predicting the next word not only contributes to making the sentences more civil but also maintains a substantial level of semantic similarity, reaching 92.34%.

Although our models achieved higher J scores compared to GPT4.0 (except for MarianMT), we manually evaluated 100 uncivil comments and their corresponding 600 civil alternatives produced by our five models and GPT4.0 (due to its second-best J score) to assess the generated comments' quality beyond metrics. Two evaluators conducted the analysis, achieving a high level of agreement (Kappa coefficient +0.83). Details are provided in Section 6 of the supplementary document.

In the evaluation of six translation models aimed at transforming uncivil comments into civil expressions, T5 emerged as the standout performer. It consistently maintained minimal changes in the comments, preserved context, and adhered to the usual communication style. On the other hand, BART, GPT3.5, and NLLB demonstrated similar rankings in their ability to meet these criteria, with some exceptions. GPT3.5 occasionally exhibited a tendency to generate overly lengthy sentences (as shown in Table 2) for relatively short uncivil comments. Moreover, GPT3.5 displayed an inclination to alter pronouns and even provide answers to questions mentioned in comments, as seen in the case of *"still looking for answers? holy crap,"* where GPT3.5 generated the response *"Yes, I am still looking for answers. I appreciate your patience."* While MarianMT delivered satisfactory performance, it suffered from lower accuracy in generating civil alternatives. GPT4.0, while performing comparably to other models, exhibited a tendency to employ overly formal language and complex sentence structures. Additionally, GPT4.0 often struggled to maintain context for long input comments, and made the content more complex. In addition, we identified five comments for which all sub-models could not produce civil alternatives. Such comments include *"Mobi I miss you â™¥ I even tattooed your name on my buttock,"* and *"Keep doing this, we'll ban you from the organization,"* among others.

As the finetuned GPT3.5 performed similarly to T5 and surpassed other models but had limited availability, our focus shifted to pre-trained translation models to achieve our goals. Hence, we conducted a survey involving 36 active developers who used our models, obtaining valuable real-world feedback on their practicality and effectiveness in fostering civility.

4.3.3 Developers' Perception on Model Performance: The second survey involved 36 actively engaged developers (13 from Bangladesh, 11 from Canada, six from the USA, and six from five other locations). They provided hands-on usability feedback, evaluating the detection models' performance and assessing the quality of comments generated by the translation model when testing uncivil comments. The survey received participants (who agreed to follow-up in the first survey), diverse in development experience, including a majority with 3-5 years (45%) and 0-2 years (30%) of experience and nine respondents with 6 to 12 years or more. Over 88% of those with significant experience reported regular or occasional involvement in online development discussions, particularly in code review, system design, and testing domains.

We tasked each participant to generate five uncivil and two civil review comments, contributing a total of 252 comments. Our detection model correctly classified 216 comments, achieving an accuracy rate of 85.71%. Regarding the translation, over 77% of participants found them very or moderately accurate. About 20% found the translations somewhat accurate, with only one participant expressing dissatisfaction. Additionally, participants found suitable civil alternatives for 75.56% (i.e., 136 out of 180) of uncivil comments where they could use the suggestion without further modification. However, for the remaining cases, we sought open-ended reasons for not accepting the generated suggestions. A few reasons highlighted that, in some instances, the generated comments lacked important context from the original comments. For example, one uncivil comment, *"Nice job, genius! Your code just crashed the whole system again!"* was noted by a participant where the suggestion did not include the context *"crashed."* This underscores that the alternative suggestions sometimes omit crucial context. In other instances, participants observed inappropriate words in the suggestions. For example, the suggestion, *"The code and the coder are both quite hot. Let's collaborate on refining them."* contained the inappropriate word *"hot"*. This highlighted the need for greater sensitivity to professionalism in the generated alternatives.

Furthermore, in subsequent inquiries, more than 94% of participants expressed their belief in the potential of the proposed civility model to foster a respectful and constructive environment. This, in turn, could lead to long-term enhancements in online communication within the software development community. Additionally, over 88% of respondents indicated their willingness to continue using the model in future team interactions, confirming its practicality in online discussions.

5 THREATS TO VALIDITY

In our research contributions, we have endeavored to ensure the validity of our findings and the robustness of our conclusions. Here, we discuss the major threats to the validity [90] of our study.

5.1 Construct Validity

While relying on prior studies to understand incivility and adhering to their operational definitions, our efforts to use comprehensive search criteria may have unintentionally missed some relevant papers on incivility or other toxicity in code discussions. Moreover, our assumption that all identified uncivil comments harm team dynamics guided the development of our upgraded detection and translation models. However, our training dataset might not cover the full spectrum of incivility practices. Despite thorough analysis and rigorous manual evaluation in prior research [21, 30, 77], there could still be nuances and variations in incivility not fully represented in our dataset.

5.2 Internal Validity

Although our detection model performs better, it may struggle with subtle forms of uncivil behavior or potentially misinterpreting certain nuances. However, the detection models' performance on our manual evaluation of 1K comments yielded a high inter-rater agreement, mitigating the subjectivity associated with human assessment. Moreover, the effectiveness of our translation model is closely tied to the quality of the training dataset, which is carefully crafted through manual analysis. Although the evaluation survey indicated a higher acceptance rate for the generated civil alternatives, these alternatives may not always be the optimal replacements in diverse contexts, such as varying countries, cultures, project types, and team dynamics. Another potential threat lies in the selection of data sources used to evaluate the proposed models drawn from 40 open-source projects. However, these projects were selected based on specific criteria, including the number of pull requests, commit activity, and project activity levels, ensuring a certain level of relevance and engagement. Finally, we fine-tuned the pre-trained models and GPT with default settings, however, alternative choices of hyperparameters may yield different performance outcomes.

5.3 Conclusion Validity

Our study, which surveyed 171 developers to explore uncivil communication, may not cover all possible reasons and consequences of incivility. However, we introduced the incivility theme comprehensively in the survey introduction, citing prior research [21, 30, 41]. While sentiment analysis results positively impact software development discussions, we did not quantify how uncivil comments affect project productivity, abandonment, or relationships. To support our claim, we manually analyzed civil alternatives generated by our model and validated them through developers' perceptions (discussed in Section 4.3.2).

5.4 External Validity

Firstly, our findings are primarily relevant to online discussions within software development teams, making them less applicable to different team types, workplaces, or online communities. Secondly, the survey respondents are predominantly developers from various countries and companies. They may not fully represent the entire developer community, and we did not account for potential variations based on country-specific cultures, company sizes, or individual perspectives in our analysis. Thirdly, the study heavily depends on the accuracy of our dataset and the limitations of the ToxiCR [77] model, which was solely validated on GitHub projects, possibly limiting its applicability to other repositories. Finally, our translation dataset is designed explicitly for review discussions conducted in English, so the models' translation ability may not be generalizable to other languages. Additionally, the way we converted the uncivil comments into civil ones employing ChatGPT may not be generalized to other domains rather than code review discussion. Moreover, there could be multiple ways to do the conversion. However, we mainly focused on collaborative mindsets and supportive attitudes between code authors and reviewers [30, 57]. This focus, coupled with a manual analysis, ensured both the civility and semantic meaning of the translated versions, achieving a high level of agreement. We provide all the materials[3] for replication purpose.

6 CONCLUSION

This study contributes to understanding and managing incivility in code review discussions. We summarised prior studies and surveyed developers about incivility, highlighting its insight and adverse effects on collaboration, productivity, and team dynamics. We improved the incivility detection model, particularly in identifying subtle incivility. Notably, we pioneered the creation of a dataset comprising uncivil comments and their civil counterparts, rigorously evaluated for translation quality. Furthermore, our translation model transforms uncivil comments into polite, respectful, and civil alternatives while preserving the original semantics and communication style. The evaluation of our models using GitHub projects, comparative analysis with state-of-the-art politeness transfer and detoxification models, and developer feedback confirmed their effectiveness in fostering civility, enhancing relationships, and expediting healthier communication patterns among developers. Considering all the analysis and results, it is evident that uncivil words indeed possess significant power to have adverse effects on both individuals and teams. Conversely, civil interaction and constructive words create healthier collaborative environments. In the future, we will make the tool open-source as a plugin and publicly available for the community to utilize in online discussions. In addition, our research will focus on cross-platform analysis, exploring multilingual and cross-cultural aspects of incivility in software development.

ACKNOWLEDGEMENTS

This research is supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery, Collaborative Research and Training Experience (CREATE), and graduate program on Software Analytics Research (SOAR) grants.

REFERENCES

[1] Sonam Adinolf and Selen Turkay. 2018. Toxic behaviors in Esports games: player perceptions and coping strategies. In *Proceedings of the 2018 Annual Symposium on computer-human interaction in play companion extended abstracts*. 365–372.

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

[3] 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.

[4] Ashley A Anderson, Dominique Brossard, Dietram A Scheufele, Michael A Xenos, and Peter Ladwig. 2014. The "nasty effect:" Online incivility and risk perceptions of emerging technologies. *Journal of computer-mediated communication* 19, 3 (2014), 373–387.

[5] Ashley A Anderson, Sara K Yeo, Dominique Brossard, Dietram A Scheufele, and Michael A Xenos. 2018. Toxic talk: How online incivility can undermine perceptions of media. *International Journal of Public Opinion Research* 30, 1 (2018), 156–168.

[6] Alberto Bacchelli and Christian Bird. 2013. Expectations, outcomes, and challenges of modern code review. In *2013 35th International Conference on Software Engineering (ICSE)*. IEEE, 712–721.

[7] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[8] Mike Barnett, Christian Bird, João Brunet, and Shuvendu K Lahiri. 2015. Helping developers help themselves: Automatic decomposition of code review changesets. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, Vol. 1. IEEE, 134–144.

[9] Gabriele Bavota and Barbara Russo. 2015. Four eyes are better than two: On the impact of code reviews on software quality. In *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 81–90.

[10] Teresa M Bejan. 2017. *Mere civility*. Harvard University Press.

[11] John D Blischak, Emily R Davenport, and Greg Wilson. 2016. A quick introduction to version control with Git and GitHub. *PLoS computational biology* 12, 1 (2016), e1004668.

[12] Amiangshu Bosu and Jeffrey C Carver. 2012. Peer code review in open source communities using reviewboard. In *Proceedings of the ACM 4th annual workshop on Evaluation and usability of programming languages and tools*. 17–24.

[13] Amiangshu Bosu and Jeffrey C Carver. 2013. Impact of peer code review on peer impression formation: A survey. In *2013 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. IEEE, 133–142.

[14] Amiangshu Bosu, Jeffrey C Carver, Christian Bird, Jonathan Orbeck, and Christopher Chockley. 2016. Process aspects and social dynamics of contemporary code review: Insights from open source development and industrial practice at microsoft. *IEEE Transactions on Software Engineering* 43, 1 (2016), 56–75.

[15] Deborah Jordan Brooks and John G Geer. 2007. Beyond negativity: The effects of incivility on the electorate. *American Journal of Political Science* 51, 1 (2007), 1–16.

[16] Jose Camacho-collados, Kiamehr Rezaee, Talayeh Riahi, Asahi Ushio, Daniel Loureiro, Dimosthenis Antypas, Joanne Boisson, Luis Espinosa Anke, Fangyu Liu, and Eugenio Martínez Cámara. [n. d.]. TweetNLP: Cutting-Edge Natural Language Processing for Social Media.

[17] Kevin Daniel André Carillo, Josianne Marsan, and Bogdan Negoita. 2016. Towards developing a theory of toxicity in the context of free/open source software & peer production communities. *SIGOPEN 2016* (2016).

[18] Stanley F Chen, Douglas Beeferman, and Roni Rosenfeld. 1998. Evaluation metrics for language models. (1998).

[19] 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.

[20] Jamie Cleland. 2014. Racism, football fans, and online message boards: How social media has added a new dimension to racist discourse in English football. *Journal of Sport and Social Issues* 38, 5 (2014), 415–431.

[21] Kevin Coe, Kate Kenski, and Stephen A Rains. 2014. Online and uncivil? Patterns and determinants of incivility in newspaper website comments. *Journal of communication* 64, 4 (2014), 658–679.

[22] Jason Cohen, Steven Teleki, and Eric Brown. 2006. *Best kept secrets of peer code review.* Smart Bear Incorporated.

[23] Marta R Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, et al. 2022. No language left behind: Scaling human-centered machine translation. *arXiv preprint arXiv:2207.04672* (2022).

[24] David Dale, Anton Voronov, Daryna Dementieva, Varvara Logacheva, Olga Kozlova, Nikita Semenov, and Alexander Panchenko. 2021. Text detoxification using large pre-trained neural models. *arXiv preprint arXiv:2109.08914* (2021).

[25] 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).

[26] Maeve Duggan. 2017. Online harassment 2017. (2017).

[27] 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 *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. 174–185.

[28] 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.

[29] Michael Fagan. 2002. Design and code inspections to reduce errors in program development. In *Software pioneers*. Springer, 575–607.

[30] Isabella Ferreira, Jinghui Cheng, and Bram Adams. 2021. The"" Shut the f** k up"" Phenomenon: Characterizing Incivility in Open Source Code Review Discussions. *Proceedings of the ACM on Human-Computer Interaction* 5, CSCW2 (2021), 1–35.

[31] 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.

[32] Anna Filippova and Hichang Cho. 2015. Mudslinging and manners: Unpacking conflict in free and open source software. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*. 1393–1403.

[33] Anna Filippova and Hichang Cho. 2016. The effects and antecedents of conflict in free and open source software development. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing*. 705–716.

[34] Joseph L Fleiss, Bruce Levin, Myunghee Cho Paik, et al. 1981. The measurement of interrater agreement. *Statistical methods for rates and proportions* 2, 212-236 (1981), 22–23.

[35] Griffin Floto, Mohammad Mahdi Abdollah Pour, Parsa Farinneya, Zhenwei Tang, Ali Pesaranghader, Manasa Bharadwaj, and Scott Sanner. 2023. DiffuDetox: A Mixed Diffusion Model for Text Detoxification. *arXiv preprint arXiv:2306.08505* (2023).

[36] Corrado Fumagalli. 2021. Counterspeech and ordinary citizens: how? when? *Political Theory* 49, 6 (2021), 1021–1047.

[37] Daniela Girardi, Filippo Lanubile, Nicole Novielli, and Alexander Serebrenik. 2021. Emotions and perceived productivity of software developers at the workplace. *IEEE Transactions on Software Engineering* 48, 9 (2021), 3326–3341.

[38] Sandra González-Bailón and Yphtach Lelkes. 2023. Do social media undermine social cohesion? A critical review. *Social Issues and Policy Review* 17, 1 (2023), 155–180.

[39] Daniel Graziotin, Fabian Fagerholm, Xiaofeng Wang, and Pekka Abrahamsson. 2018. What happens when software developers are (un) happy. *Journal of Systems and Software* 140 (2018), 32–47.

[40] Sanuri Gunawardena, Ewan Tempero, and Kelly Blincoe. 2023. Concerns identified in code review: A fine-grained, faceted classification. *Information and Software Technology* 153 (2023), 107054.

[41] Sanuri Dananja Gunawardena, Peter Devine, Isabelle Beaumont, Lola Piper Garden, Emerson Murphy-Hill, and Kelly Blincoe. 2022. Destructive criticism in software code review impacts inclusion. *Proceedings of the ACM on Human-Computer Interaction* 6, CSCW2 (2022), 1–29.

[42] Skyler Hallinan, Alisa Liu, Yejin Choi, and Maarten Sap. 2022. Detoxifying text with marco: Controllable revision with experts and anti-experts. *arXiv preprint arXiv:2212.10543* (2022).

[43] Wenjian Huang, Tun Lu, Haiyi Zhu, Guo Li, and Ning Gu. 2016. Effectiveness of conflict management strategies in peer review process of online collaboration projects. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing*. 717–728.

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

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

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

[47] Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. 2018. Marian: Fast Neural Machine Translation in C++. , 116–121 pages. http://www.aclweb.org/anthology/P18-4020

[48] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461* (2019).

[49] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).

[50] Jon Loeliger and Matthew McCullough. 2012. *Version Control with Git: Powerful tools and techniques for collaborative software development.* " O'Reilly Media, Inc.".

[51] Varvara Logacheva, Daryna Dementieva, Sergey Ustyantsev, Daniil Moskovskiy, David Dale, Irina Krotova, Nikita Semenov, and Alexander Panchenko. 2022. Paradetox: Detoxification with parallel data. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 6804–6818.

[52] Aman Madaan, Amrith Setlur, Tanmay Parekh, Barnabas Poczos, Graham Neubig, Yiming Yang, Ruslan Salakhutdinov, Alan W Black, and Shrimai Prabhumoye. 2020. Politeness transfer: A tag and generate approach. *arXiv preprint arXiv:2004.14257* (2020).

[53] Suman Kalyan Maity, Aishik Chakraborty, Pawan Goyal, and Animesh Mukherjee. 2018. Opinion conflicts: An effective route to detect incivility in Twitter. *Proceedings of the ACM on Human-Computer Interaction* 2, CSCW (2018), 1–27.

[54] Marcelo Marinho, Luís Amorim, Rafael Camara, Brigitte Renata Oliveira, Marcos Sobral, and Suzana Sampaio. 2021. Happier and further by going together: The importance of software team behaviour during the COVID-19 pandemic. *Technology in society* 67 (2021), 101799.

[55] Mary L McHugh. 2012. Interrater reliability: the kappa statistic. *Biochemia medica* 22, 3 (2012), 276–282.

[56] Shane McIntosh, Yasutaka Kamei, Bram Adams, and Ahmed E Hassan. 2016. An empirical study of the impact of modern code review practices on software quality. *Empirical Software Engineering* 21, 5 (2016), 2146–2189.

[57] Courtney Miller, Sophie Cohen, Daniel Klug, Bogdan Vasilescu, and Christian KaUstner. 2022. " Did you miss my comment or what?" understanding toxicity in open source discussions. In *Proceedings of the 44th International Conference on Software Engineering*. 710–722.

[58] Rocío Galarza Molina and Freddie J Jennings. 2018. The role of civility and metacommunication in Facebook discussions. *Communication studies* 69, 1 (2018), 42–66.

[59] Rodrigo Morales, Shane McIntosh, and Foutse Khomh. 2015. Do code review practices impact design quality? a case study of the qt, vtk, and itk projects. In *2015 IEEE 22nd international conference on software analysis, evolution, and reengineering (SANER)*. IEEE, 171–180.

[60] Murtuza Mukadam, Christian Bird, and Peter C Rigby. 2013. Gerrit software code review data from android. In *2013 10th Working Conference on Mining Software Repositories (MSR)*. IEEE, 45–48.

[61] Sourabrata Mukherjee, Vojtěch Hudeček, and Ondřej Dušek. 2023. Polite Chatbot: A Text Style Transfer Application. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop*. 87–93.

[62] Dawn Nafus, James Leach, and Bernhard Krieger. 2006. Gender: Integrated report of findings. *FLOSSPOLS, Deliverable D* 16 (2006).

[63] BM İnsan Hakları Ofisi. 2021. Report: online hate increasing against minorities, says expert.

[64] Gwenn Schurgin O'Keeffe, Kathleen Clarke-Pearson, et al. 2011. The impact of social media on children, adolescents, and families. *Pediatrics* 127, 4 (2011), 800–804.

[65] 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.

[66] Carla Pérez-Almendros, Luis Espinosa Anke, and Steven Schockaert. 2020. Don't Patronize Me! An Annotated Dataset with Patronizing and Condescending Language towards Vulnerable Communities. In *Proceedings of the 28th International Conference on Computational Linguistics*. 5891–5902.

[67] Mohammad Mahdi Abdollah Pour, Parsa Farinneya, Manasa Bharadwaj, Nikhil Verma, Ali Pesaranghader, and Scott Sanner. 2023. COUNT: COntrastive UNlikelihood Text Style Transfer for Text Detoxification. In *Findings of the Association for Computational Linguistics: EMNLP 2023*. 8658–8666.

[68] Huilian Sophie Qiu, Bogdan Vasilescu, Christian Kästner, Carolyn Egelman, Ciera Jaspan, and Emerson Murphy-Hill. 2022. Detecting interpersonal conflict in issues and code review: cross pollinating open-and closed-source approaches. In *Proceedings of the 2022 ACM/IEEE 44th International Conference on Software Engineering: Software Engineering in Society*. 41–55.

[69] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research* 21, 140 (2020), 1–67. http://jmlr.org/papers/v21/20-074.html

[70] Md Shamimur Rahman, Debajyoti Mondal, Zadia Codabux, and Chanchal K Roy. 2023. Integrating Visual Aids to Enhance the Code Reviewer Selection Process. In *2023 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 293–305.

[71] Naveen Raman, Minxuan Cao, Yulia Tsvetkov, Christian Kästner, and Bogdan Vasilescu. 2020. Stress and burnout in open source: Toward finding, understanding, and mitigating unhealthy interactions. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: New Ideas and Emerging Results*. 57–60.

[72] 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.

[73] Punyajoy Saha, Kanishk Singh, Adarsh Kumar, Binny Mathew, and Animesh Mukherjee. 2022. CounterGeDi: A controllable approach to generate polite, detoxified and emotional counterspeech. *arXiv preprint arXiv:2205.04304* (2022).

[74] Jaydeb Sarker. 2022. Identification and Mitigation of Toxic Communications Among Open Source Software Developers. In *37th IEEE/ACM International Conference on Automated Software Engineering*. 1–5.

[75] Jaydeb Sarker. 2022. 'Who built this crap?'Developing a Software Engineering Domain Specific Toxicity Detector. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*. 1–3.

[76] 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.

[77] Jaydeb Sarker, Asif Kamal Turzo, Ming Dong, and Amiangshu Bosu. 2023. Automated Identification of Toxic Code Reviews Using ToxiCR. *ACM Transactions on Software Engineering and Methodology* (2023).

[78] 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.

[79] Igor Steinmacher and Marco Aurélio Gerosa. 2014. How to support newcomers onboarding to open source software projects. In *Open Source Software: Mobile Open Source Technologies: 10th IFIP WG 2.13 International Conference on Open Source Systems, OSS 2014, San José, Costa Rica, May 6-9, 2014. Proceedings 10*. Springer, 199–201.

[80] Igor Steinmacher, Igor Wiese, Ana Paula Chaves, and Marco Aurélio Gerosa. 2013. Why do newcomers abandon open source software projects?. In *2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. IEEE, 25–32.

[81] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *Advances in neural information processing systems* 27 (2014).

[82] Yida Tao and Sunghun Kim. 2015. Partitioning composite code changes to facilitate code review. In *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*. IEEE, 180–190.

[83] Kurt Thomas, Devdatta Akhawe, Michael Bailey, Dan Boneh, Elie Bursztein, Sunny Consolvo, Nicola Dell, Zakir Durumeric, Patrick Gage Kelley, Deepak Kumar, et al. 2021. Sok: Hate, harassment, and the changing landscape of online abuse. In *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 247–267.

[84] 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.

[85] Alexia Tsotsis. 2011. Meet phabricator, the witty code review tool built inside facebook. *Retrieved November* 30 (2011), 2021.

[86] Russell M Viner, Aswathikutty Gireesh, Neza Stiglic, Lee D Hudson, Anne-Lise Goddings, Joseph L Ward, and Dasha E Nicholls. 2019. Roles of cyberbullying, sleep, and physical activity in mediating the effects of social media use on mental health and wellbeing among young people in England: a secondary analysis of longitudinal data. *The Lancet Child & Adolescent Health* 3, 10 (2019), 685–696.

[87] Jing Wang, Patrick C Shih, Yu Wu, and John M Carroll. 2015. Comparative case studies of open source software peer review practices. *Information and Software Technology* 67 (2015), 1–12.

[88] John Wieting, Taylor Berg-Kirkpatrick, Kevin Gimpel, and Graham Neubig. 2019. Beyond BLEU: Training Neural Machine Translation with Semantic Similarity. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 4344–4355.

[89] Thilini Wijesiriwardene, Hale Inan, Ugur Kursuncu, Manas Gaur, Valerie L Shalin, Krishnaprasad Thirunarayan, Amit Sheth, and I Budak Arpinar. 2020. Alone: A dataset for toxic behavior among adolescents on twitter. In *Social Informatics: 12th International Conference, SocInfo 2020, Pisa, Italy, October 6–9, 2020, Proceedings 12*. Springer, 427–439.

[90] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in software engineering*. Springer Science & Business Media.

[91] Robert F Woolson. 2007. Wilcoxon signed-rank test. *Wiley encyclopedia of clinical trials* (2007), 1–3.

[92] Li Yujian and Liu Bo. 2007. A normalized Levenshtein distance metric. *IEEE transactions on pattern analysis and machine intelligence* 29, 6 (2007), 1091–1095.