Supplementary Documents for: Do Words Have Power? Understanding and Fostering Civility in Code Review Discussion

ACM Reference Format:
. 2024. Supplementary Documents for: Do Words Have Power? Understanding and Fostering Civility in Code Review Discussion. 1, 1 (March 2024), 7 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 TRANSLATION RUBRIC

This rubric presented in Table 1 serves as a systematic guide for transforming uncivil comments encountered during code review discussions into more respectful and constructive expressions. It comprises eight distinct rules, each addressing various aspects of incivility, along with accompanying rationale, examples, and suggested translations. These rules encompass avoiding profanity, acronyms denoting offensive language, derogatory comments targeting individuals or entities, inappropriate behavior based on identities, aggressive behavior or threats, references to sexual activities, flirtations or inappropriate dynamics, and intended teasing or provocation.

2 SUMMARY KNOWLEDGE OF PRIOR INCIVILITY STUDY

Table 2 shows the potential causes and consequences of incivility in code review discussion from prior studies.

3 SURVEY OVERVIEW

The survey in this study aims to gather insights from software developers about incivility in company discussion forums. It encompasses various topics such as developers' demographic information, their methods of collaboration and communication, and their level of contentment with software development practices. Furthermore, it is worth noting that the survey and questionnaires have received approval from the Ethics Behavioral Board of the University of *Anonymous*. The complete survey and all accompanying materials are available in the replication package. The survey has several parts:

- **Demographics:** The survey guarantees complete anonymity, and no personally identifiable information is requested except for participants' software development experience (in years) and current job location (country). Participants are also asked for their consent to participate in the survey.
- **Experience of incivility:** As our study focuses on incivility within code review discussions, we inquire about software developers' review practices. For those whose projects or companies engage in code review, we proceed to ask questions about instances of incivility in code review

Author's address:

XXXX-XXXX/2024/3-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Table 1. Rubric to translate uncivil review comments into civil ones.

Sl	Category	Rule	Example	Translation
TR1	Profane or curse words in comments.	All profanity, including frustration, anxiety, or other emotional words, would be removed. Reframe the situation using positive or neutral language.	*Fuck! Consider it done!*	Alright! Consider it done!
TR2	Acronyms that typically denote offensive or vulgar language.	Acronyms for profanities or other vulgarity would be removed. Focus only on the context in a collaborative manner.	*WTF are you doing!*	What are you doing recently?
TR3	Derogatory comments targeting individuals or entities.	Eliminate the insulting, derogatory and aggressive words and offer constructive criticisms with appreciation and supportive attitudes.	*Shut up! This is the worst implementation I've ever seen.*	There is room for improvement in this implementation.
TR4	Inappropriate behavior based on identities, such as race, religion, nationality, gender, or sexual orientation.	Focus solely on developmental aspects, avoiding words related to identity or personal attacks. Transform any remarks with such tones into constructive feedback, emphasizing a collaborative mindset.	*Your code is terrible, just like all women programmers!*	There are some areas in the code that could be improved.
TR5	Engaging in aggressive behavior or making threats towards an individual or a community.	Discard instances of inappropriate and threatening words. The translated version should encourage open communication and teamwork rather than resorting to harmful tactics.	*I will ruin your reputation if you don't accept my code changes.*	If my code changes are not accepted, I would appreciate discussing the reasons behind the decision.
TR6	Referring to sexual activities either implicitly or explicitly	Remove inappropriate personal feelings and appreciation that could make others uncomfortable. Focus on the technical aspects of the code and express appreciation professionally.	*This code makes me so horny. It's beautiful.*	I find this code to be impressive and well-written.
TR7	Flirtations or inappropriate dynamics among developers	Delete explicit or implicit expressions of romantic attraction and inappropriate remarks about someones' appearance or talents. Keep interactions respectful and centred on development and collaboration.	*You're not just talented; you're a total babe!*	You're not just talented; your skills and expertise are exceptional.
TR8	Intended to tease. mocking or provoke others	Remove words that are intended to blame or belittle, sarcasm and offensive remarks. Translate the comments in a more collaborative and supportive manner with constructive feedback.	*Nice job breaking the build again!*	Let's work together to address the build issues and find a solution.

discussion they may have encountered and the frequency of such occurrences. Furthermore, we inquire about their emotional responses during and after engaging in uncivil discussions.

- **Potential causes of incivility:** This part of the survey is designed to gather valuable insights from software developers regarding the potential causes of incivility within their teams. We recognize the significance of understanding the underlying factors that contribute to uncivil behaviors in the software development environment, which can inform the development of strategies and interventions to foster a more positive and respectful team culture.

Table 2. Exploring the potential causes and consequences of incivility in code review discussion from prior studies.

Serial	Study	Type	Potential Causes	Potential Consequences
1	Ferreira et al. [3]	Survey+ Qualitative	* Propose non-optimal solutions that can have bad impacts. * Reject patches after devoted effort asked by reviewers. * Violate community conventions and workflows. * Misinterpret or unable to understand reviewers' instructions. * Receive early rejection without sufficient explanations. * Get inappropriate suggestions from reviewers. * Present different opinions to solve problems. * Submit patches that lack coding standards. * Provide insufficient explanation or documentation for proposed patches.	* Discontinue further discussion. * Abandon patches without replying * Attack reviewers for patch rejection * Change reviewers and add new ones. * Ban developers for further contributions.
2	Miller et al. [6]	Qualitative	* Get vague descriptions or error messages for a problem. * Unable to use tools, complains about wasted time. * Do not get immediate responses from reviewers. * Show disagreements over technical aspects, politics, ideology, culture, beliefs, and involvements. * Has previous bad experiences with the same peers. * Submit patches with wrong language versions.	* Increase disengagement of newcomers. * Report stress and burnout. * Invest substantial time to respond to toxic comments. * Block user accounts from further interactions. * Reject or lock patches if too heated.
3	Gunawardena et al. [5]	Survey	* Do not take into account developers' efforts. * Give harsh and insensitive criticisms. * Focus primarily on finding faults rather than providing constructive feedback. * Imbalance power between developers and reviewers. * Experience high levels of stress and time pressure. * Ignore previous feedback from reviewers. * Test the submitted code change inadequately.	* Decrease motivation to continue work. * Feel lower self-efficacy and inferior in future tasks. * Increase the turnover intentions. * Decrease team unity, teamwork, and teams' ability. * Create an unsafe place where people are not confident enough to ask for help or how they can improve. * Question how others value individuals' contributions and skills. * Discriminate minors, especially Women. * Impact in moods, uncomfortable and less productivity.
4	Graziotin et al. [4]	Survey	* Do not follow organizational culture and values. * Go unnoticed or unappreciated by peers. * Get insufficient support. * Submit low-quality code changes without maintaining coding conventions.	* Drop cognitive skills, frustrated and sloppy. * Experience inadequate performance and motivation. * Increase stress and project turnover.
5	Chouchen et al. [1]	Qualitative	* Get controversial comments unrelated to the submitted code. * Receive harsh and rage-expressive review comments.	* Increase code review time * Influence the quality of relationship among peers.
6	Egelman et al. [2]	Survey	* Hold the decision longer than necessary. * Request more changes than required. * Attack on work quality. * Get unnecessary pressure to make changes. * Receive late unjustified criticisms.	* Create interpersonal conflicts. * Increase review rounds.

- **Potential impacts of incivility:** In this section of the survey, our focus is to gain insights into the potential impacts or consequences, both positive and negative, of incivility within software development teams. By engaging software developers, we aim to comprehensively understand how uncivil behavior affects various aspects of team dynamics and individual experiences. This exploration will enable us to grasp the wide-ranging implications of incivility, shedding light on its effects on collaboration, communication, productivity, job satisfaction, and overall team performance.
- **Promote civility in discussion:** This section aims to gather insights from software developers on strategies to foster civility in team discussions. Participants are also encouraged to

provide feedback on a proposed solution. The survey findings will inform evidence-based approaches to improve communication dynamics and enhance productivity within software development teams.

- **Interest in follow-up:** Developers interested in participating in follow-up surveys or interviews to discuss incivility are invited to provide their email addresses to delve deeper into the subject matter and explore potential solutions.

4 PARTICIPANTS RECRUITMENT

Our survey aims to engage professional software developers with expertise in software development who are part of development teams with communication channels for discussing development-related matters. To recruit participants, we will reach out through LinkedIn and use 217 available email addresses obtained from 17 company websites across Canada and Bangladesh. Notably, we refrained from filtering email addresses to minimize sampling bias and sent invitations to all publicly available email addresses. Participation is voluntary, allowing individuals to decide whether to participate in the study. We also encouraged participants to forward the invitation to those who meet the selection criteria, ensuring broader coverage. Strict confidentiality measures were implemented to ensure participant privacy and data protection.

5 EVALUATION METRICS

In this section, we discuss the evaluation metrics employed to assess the efficacy of our translation model in converting uncivil comments into civil expressions. The selection of these metrics is integral to gauging the models' performance across various dimensions, ranging from linguistic nuances to the overall impact on fostering civility within code review conversations. The metrics are described as follows:

- **Incivility Decrease:** This metric quantifies the reduction in incivility observed in the translated comments generated by the models. This metric is fundamental to assessing the models' proficiency in transforming uncivil review comments into more civil counterparts. The decrement is calculated by comparing the number of initial uncivil comments with the number of remaining uncivil comments after translation, measured as a percentage decrease. A higher Incivility Decrease value indicates a more efficient model performance.

$$\text{Incivility\_Decrease} = \left(\frac{\text{initial\_uncivil\_comments} - \text{final\_uncivil\_comments}}{\text{initial\_uncivil\_comments}} \right) \times 100 \qquad (1)$$

- **Semantic Similarity:** Semantic similarity, in the context of our translation model evaluation, refers to the degree of likeness in meaning between the original uncivil comments and their translated civil versions. Utilizing the Sentence Transformer pretrained models[1], we calculate the semantic similarity to quantify how well the translated comments capture the essence and intent of the original uncivil statements. This metric provides insights into the models' ability to preserve the underlying meaning during the translation process. A higher Semantic Similarity value indicates a more efficient model performance. The code is available in the replication package.
- **Sentence Similarity:** This metric evaluates the resemblance between the original uncivil comments and their translated civil versions. This measurement considers both the number and order of tokens in the sentences. The expectation is that an effective translation would result in minimal changes to the token structure (i.e., higher similarity), preserving the

[1]https://huggingface.co/sentence-transformers

overall sentence composition. We utilized the Levenshtein distance [7], also known as the edit distance, which quantifies the minimum number of single-character edits (insertions, deletions, or substitutions) required to transform one string into another. In the context of sentence similarity, it serves to measure the dissimilarity between the original and translated sentences at the token level.

- **Length Dissimilarity:** It gauges the variance in length between the original uncivil comments and their translated civil counterparts. This metric aims for the generated civil alternatives to maintain a length close to or equal to that of the original comments. A lower value is preferable, suggesting minimal information loss and maintaining a communication style aligned with normal online interactions. The calculation is performed using the provided equation, which computes the normalized length difference as a percentage of the original sentence length.

$$\text{Length Dissimilarity} = \left(\frac{\text{Original Length} - \text{Modified Length}}{\text{Original Length}} \right) \times 100 \tag{2}$$

- **Sentiment Analyzers:** In assessing sentiment analysis results, our focus was on discerning the increase in positive sentiment scores within the translated versions. The sentiment analyzers employed yielded ternary outputs, encompassing positive, negative, and neutral sentiments, denoted as +1, -1, and 0, respectively. Aggregating the sentiment scores for both uncivil and translated civil comments, we observed that if the cumulative score for translated civil comments surpassed that of uncivil comments, it indicated a transformation of negative sentiments into neutral or positive, and neutral sentiments into positive ones. Utilizing the following equation, we calculated the percentage score reflecting the improvement in sentiments.

$$\text{Sentiment Improvement} = \left(\frac{\sum \text{sentiments in civil comments} - \sum \text{sentiments in uncivil comments}}{\sum \text{sentiments in uncivil comments}} \right) \times 100 \tag{3}$$

- **Perplexity:** Perplexity is a statistical measure used to evaluate the effectiveness of a language model. It assesses the models' ability to predict a given sequence of words or tokens in a text. Lower perplexity values indicate better performance, suggesting that the model can more accurately predict the next word in a sequence, reflecting a clearer understanding of the language and context. In the context of our evaluation, perplexity is employed to gauge the coherence and fluency of the translated comments, with lower perplexity values indicating more linguistically coherent and contextually appropriate translations.
- **Joint Metric (J):** The Joint Metric, denoted as J, provides a comprehensive evaluation by combining the performance of seven individual metrics. It is calculated as the harmonic mean of these metrics, offering a balanced measure that considers multiple aspects of the translation model's effectiveness. A higher J value indicates a well-rounded and robust performance across various dimensions, emphasizing the importance of achieving good scores in all evaluation criteria. The equation for calculating the Joint Metric J is as follows:

$$J = \frac{n}{\frac{1}{X_1} + \frac{1}{X_2} + \ldots + \frac{1}{X_n}} \tag{4}$$

where $X_1, X_2, ..., X_n$ represent the individual metric scores, and n is the total number of metrics considered. In our context, where $n = 7$, the individual metrics encompass Incivility Decrease, Semantic Similarity, Sentence Similarity, (100 - Length Dissimilarity), and scores

from three sentiment analyzers. To maintain uniformity in metric goals (i.e., higher is better), we subtracted Length Dissimilarity from 100.

6 RANKING TRANSLATION

Here, we present a methodology for ranking models that translate uncivil comments into civil alternatives within the domain of code review discussions.

6.1 Data Collection

We randomly took 100 uncivil comments and generated six civil alternatives for each uncivil comment using translation models T5, BART, NLLB, MarianMt, GPT3.5, and GPT4.0. Therefore, we got a total of 600 civil alternatives.

6.2 Scoring Criteria

A structured scoring system is established to evaluate the civil alternatives. Three criteria are considered:

(1) Minimal Change: Assessing the degree of alteration to the original comment.
(2) Context Preservation: Evaluating whether the alternative maintains the sole context of the original comment.
(3) Communication Style: Analyzing the choice of words and sentence structure to ensure alignment with the usual online communication style, avoiding excessive formality or complexity.

6.3 Score Calculation

Individual scores (ranging from 1 to 5) are assigned to each civil alternative based on the three criteria. The total score for each alternative is calculated by summing these scores (resulting in scores out of 15).

(1) Minimal Change: Higher number of changes, lower score (i.e., 1 to 5)
(2) Context Preservation: Higher preservation, higher score (i.e., 5 to 1)
(3) Communication Style: Higher complexity, lower score (i.e., 1 to 5)

6.4 Model Ranking

Models are ranked for each uncivil comment based on the average total score of their generated civil alternatives across all 100 uncivil comments. The model with the highest average total score ranks highest, indicating its efficacy in producing civil alternatives that align with the criteria of minimal change, context preservation, and appropriate language style.

REFERENCES

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

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